Towards Designing Efficient Lightweight Ciphers for Internet of Things

Muhammad Tausif[1], Javed Ferzund[2], Sohail Jabbar[3*] and Raheela Shahzadi[4]
[1]Department of Computer Science, COMSATS Institute of Information Technology
Vehari, Pakistan
[email: raotausif@ciitvehari.edu.pk]
[2] Department of Computer Science, COMSATS Institute of Information Technology
Sahiwal, Pakistan
[email: jferzund@ciitsahiwal.edu.pk]
[3]Department of Computer Science, National Textile University,
Faisalabad, Pakistan
[email : sjabbar.research@gmail.com]
[4]Department of Computer Science, COMSATS Institute of Information Technology
Sahiwal, Pakistan
[email: raheela@ciitsahiwal.edu.pk]
*Corresponding author: Sohail Jabbar

Received December 28, 2016; revised March 20, 2017; accepted April 23, 2017; published August 31, 2017

Abstract

Internet of Things (IoT) will transform our daily life by making different aspects of life smart like smart home, smart workplace, smart health and smart city etc. IoT is based on network of physical objects equipped with sensors and actuators that can gather and share data with other objects or humans. Secure communication is required for successful working of IoT. In this paper, a total of 13 lightweight cryptographic algorithms are evaluated based on their implementation results on 8-bit, 16-bit, and 32-bit microcontrollers and their appropriateness is examined for resource-constrained scenarios like IoT. These algorithms are analysed by dissecting them into their logical and structural elements. This paper tries to investigate the relationships between the structural elements of an algorithm and its performance. Association rule mining is used to find association patterns among the constituent elements of the selected ciphers and their performance. Interesting results are found on the type of element used to improve the cipher in terms of code size, RAM requirement and execution time. This paper will serve as a guideline for cryptographic designers to design improved ciphers for resource constrained environments like IoT.

Keywords: Internet of Things, Smart Objects, Information Security, Lightweight Cryptography, Ciphers, Association Rule Mining

1. Introduction

The Internet of Things (IoT) is a concept of universally identifiable physical things (or objects), their integration with the Internet, and their demonstration in the digital or simulated world. In order to construct the Internet of Things, a comprehensive range of technologies are elaborated for example, Radio Frequency Identification (RFID) for device and location recognition and Wireless Sensor Networks (WSN) for freely connecting with intelligent systems and among each other. With the assistance of these technologies, we can construct an environment where things talk to each other. Because of sensitivity of applications, security in physical deployments of the Internet of Things is the key constraint [1,2]. In the Internet of Things, the subsequent security facilities like Confidentiality, Data Integrity, Source Integrity or Authentication, and Availability are needed [3]. Smart things may be small computing devices, containing constrained resources such as low computation capabilities, small size RAM and limited battery power. Communication with smart things in resource constrained situation need consideration with these harsh limitations.

LightWeight Cryptography (LWC) is a very active research domain by targeting at the plan of novel ciphers whose strong point is to fulfill the requirements set by the use of constrained objects . The word “lightweight” talks about a family of cryptographic ciphers with smaller code size, low computational power and low energy consumption. Because of these hard resource limitations there is a growing need for security solutions based on lightweight cryptography that are designed according to IoT requirements. Lightweight cryptography emphasizes on efficient implementations of cryptographic algorithms and it is a comparatively young scientific sub-field that is positioned at the intersection of computer science, electrical engineering, and cryptography. All people working at the research area of lightweight cryptography has to manage with the compromise between performance, security and cost. Commonly, two out of the three design aims, can be easily improved, however at the same time it is very difficult to boost all three design objectives, as shown in **Fig. 1** taken from [4].

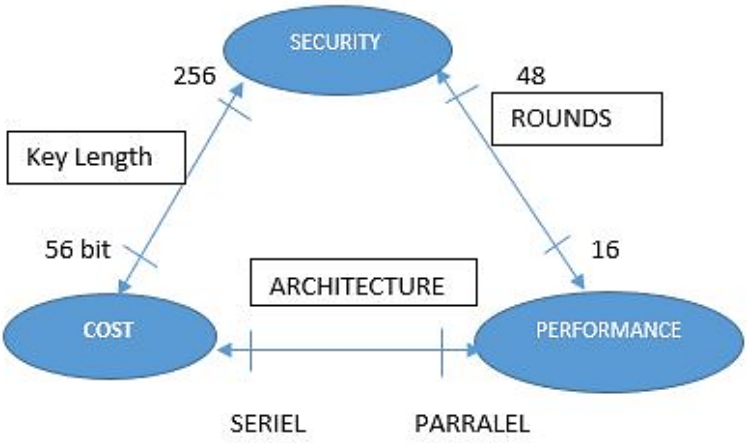

Fig. 1. Trade-off among Security, Cost and Performance [4]

For resource constrained objects, the selection of the cryptographic algorithm is a key part that can disturb performance[5]. When efficient energy consumption and low cost are harsh requirements, computational power must essentially be reduced consequently [6, 7]. Using 8

bit microcontrollers (such as AVR microcontrollers, which have restricted abilities in terms of storage and computing power), it is needed that implemented algorithms must be kept simple, having low footprint. This could result in lower energy consumption and faster execution which might be important for battery powered objects [8, 9]. Even though maximum symmetric ciphers have been established by concentrating on good software executions, the placement of smart objects will lead to growing attention to those cryptographic algorithms that will have efficient implementations for hardware in terms of energy consumption and speed [10]. During this study we have tabulated the thorough benchmarking results of 13 lightweight cryptographic algorithms, namely PRINCE, RC5 AES, Fantomas, Speck Piccolo, PRESENT HIGHT, L Block, LED, Robin, Simon, , and TWINE. Our motivations for choosing these cryptographic algorithms are first, each of these ciphers has a distinct property that makes it motivating for IoT applications. Secondly, they cover extensive range of approaches and different design strategies. Our evaluation considers one use case that is a simple challenge handshake authentication which covers the need of authentication for applications such as access control or object identification in IoT.

In this article, lightweight cryptographic algorithms are evaluated with an analysis of their software implementation results on 8-bit, 16-bit, and 32-bit microcontroller. This study focusses on three parameters: binary code size, execution time and runtime RAM usage. These parameters are investigated with respect to the structural elements of different lightweight ciphers. Results of this study give novel understandings to the query of which cipher is more suited to the IoT scenario. Association Rule Mining have been used to find the associations among the constituent elements of different ciphers and the performance parameters. Based on the results, interesting information has been inferred about cipher behavior on different platforms under the same scenario. Finally, guidelines are formulated for designing efficient lightweight ciphers.

The rest of the paper is organized as follows. Related work is presented in Section 2. Performane ealuation and comparison of cryptographic algorithms is presented in Setion 3. Results are discussed in Section 4 and finally paper is concluded in Setion 5.

2. Related Work

For wireless sensor networks (WSN) Law *et al.* present a survey on cryptographic algorithms [11]. They consider properties like energy-efficiency and storage capacity of different cryptographic algorithms including Twofish, MISTY1 Skipjack [12], RC5, RC6, AES, KASUMI, MISTY1 and Camellia. In this work the consequence of examination delivers us standard of picking cryptographic algorithm appropriate for wireless sensor networks. Memory efficient cryptographic algorithms are necessary in a situation where security is significant and energy efficient cryptographic algorithm has to be used in a situation where availability of network is vital, since sensor hops whose consumption of battery is more are no longer available in the network.

Karlof *et al.* [13], considered de-facto standard of security design for WSN, concluded that RC5 and Skipjack are suggested cryptographic algorithms in a particular scenario of WSN. Each candidate has their own characteristics security, memory and energy efficiency. Consequently, if several nominees of cryptographic algorithm are practically applied, user can select easily for according to the condition for wireless sensor networks.

Woo *et al.* consider another candidate HIGHT on Mica2 [9], designed to be suitable to ubiquitous 8 bit devices for wireless sensor networks. They examine the performance between Skipjack, RC5 and HIGHT cryptographic algorithm on TinySec. Finally, author

show performance evaluation on the basis of memory efficiency and power usage. The author concluded that as compared with traditional ciphers on TinySec, HIGHT is suitable candidate for ubiquitous devices.

Swernendu *et al.* [14] has provided a survey of a number of current light weight cryptographic algorithms. The author described with the fast developments in wireless networks and low end devices such as Radio frequency identification tags, WSN nodes are positioned in growing numbers every day. Such devices are used in several situations and applications important to a constantly increasing requirement to deliver security. When picking Cryptographic algorithms for resource constrained devices the implementation cost should be considered. In order to fulfill these requirements, efficient and secure authentication and encryption arrangements have to be established. In the resource constrained environment symmetric key ciphers, particularly lightweight block ciphers still play a significant part to deliver confidentiality.

Parbhat *et al.* [15] did a comparative examination of unlike symmetric-key lightweight cryptographic algorithms such as PRINT, EPCBC, DESL PRESENT, KATAN, LED Puffin, KLEIN, RECTANGLE, LBLOCK and TWINE. It focuses on the tradeoffs between throughput, area and cycle per block of unlike algorithms. Even though the cost is little, the symmetric-key lightweight algorithms are required to be better in numerous directions like Gate Equivalents (throughput and area) and number of cycles per block.

Katagi *et al.* [16] described that lightweight cryptography is the back bone to the security of smart objects networks because of its smaller footprint and efficiency. Authors believe that lightweight cryptography should be deliberated to be executed in the networks. Specifically, lightweight block ciphers are used now days. They presented a summary of the state of the art technology and normalization position of lightweight cryptography, which can be executed efficiently in resource constrained devices. This technology allows protected and efficient communication among smart objects.

3. Light Weight Cryptographic Algorithms

Our objective is to know the link between the cryptographic algorithm structure and the performance result on the particular platforms and devices in the Internet of Things scenario. We have carefully chosen lightweight cryptographic algorithms demonstrating an enormous variety of design results from the two big families of Substitution Permutation Networks (SPN) and Feistel Networks (FN).

In the following sub sections, we shortly describe the selected lightweight ciphers. A summary of the selected ciphers is presented in **Table 1**.

3.1 AES-128

The AES is the present day lightweight block cipher [17]. AES was designed by V. Rijmen in 1997 and selected as a standard in 2000. It is the widely used cryptographic algorithm. The AES is based on SPN structure. AES block size is 128 bits under three different key sizes 128, 192, or 256 bits. We focus here on the case of AES 128 bit block size under a key of length 128 bits. This Advance encryption standard version consists of 10 rounds that reiterate four basic steps:

1- Sub Bytes
2- Shift Rows
3- Mix Columns
4- Add Round Key on blocks seen as 4×4 byte matrices

3.2 Fantomas

Fantomas is a 128 bits lightweight cryptographic algorithm. It is similar to Robin. It is based on LS-design. In LS-design linear layer includes in the parallel applications of so-called L boxes. The S box configuration makes simpler the operation of masking. Master key is added at every round [18]. There is no key schedule.

3.3 HIGHT

HIGHT has been good candidate for light weight cryptography by seeing low resource hardware performance [19]. HIGHT practices very simple arithmetic and logic operations such as addition and exclusive OR and bitwise rotation. HIGHT has 64-bit block length and 128-bit key length. HIGHT was considered to be suitable for application in the low resource atmosphere such as Radio Frequency Identification tag or small universal devices. HIGHT comprises of 4 key steps:

1. Key schedule
2. Initial transformation
3. 32 iterative round operations
4. Final transformation

3.4 L Block

L block is based on Feistel Network structure. It consists of 32 rounds. The Feistel function consists of XOR with the round sub key, substitution layer of 8 different S-boxes and a permutation of 8 nibbles. Furthermore, the content of one of the branches is rotated by 8 bits in each round. The design trade-offs between security and performance led not only to hardware efficiency but also software efficiency [20]. The best cryptanalysis of this primitive is an impossible differential attack on 23 out of 32 rounds [21].

3.5 LED (Light Encryption Device)

LED (Light Encryption Device) [22] has provided sound performance background for software aspects. LED has 64 bits block size with four different key sizes 64 bits, 80 bits, 96 bits, and 128 bits. Light Encryption Device algorithm practices PRESENT cipher s-box. It consists of following steps:

1. Add Round Key (Key XOR with cipher.)
2. Add Constants-Round (constants are combined with cipher using bitwise XOR).
3. Sub Cells- (Each nibble is replaced by the generated nibble using PRESENT s-box.)
4. Shift Rows Serial

3.6 Piccolo

Piccolo is a comprehensive Feistel construction with four 16 bit branches. Piccolo uses a byte permutation among rounds to increase diffusion. The Feistel function contains two S-box layers separated by a diffusion matrix [23]. The superlative attack on Piccolo is a Meet in the Middle attack described by its creators in the article in which cipher is introduced.

3.7 PRESENT Cipher

PRESENT cipher focused on the hardware performance [24]. It has been considered to be efficient lightweight cryptographic algorithm in hardware. It functions on 64 bit block size and with the key size of 80 bits. It has 32 rounds of iteration. PRESENT is an example of SPN structure. One round contain following steps:

1. Add Round Key: Key XOR with cipher.
2. Substitution: Uses 4 bits S-box.
3. Permutation: Uses P-layer.

S-box used in PRESENT cipher is:

$$S(x) = \{C, 5, 6, B, 9, 0, A, D, 3, E, F, 8, 4, 7, 1, 2\}$$

Table 1. Complete Description of Building Elements of Ciphers

CIPHER	Key Size	Block Size	Rounds	Structure	S-Box	Round Function	Key Scheduling
Speck	64	96	26	FEISTEL	Not based	XOR, Left, Right, Shift	Based on Round Function
Simon	64	96	42	FEISTEL	Not based	XOR, AND, Circular Shift	Based on Round Function
AES	128	128	10	SPN	4*4 s-box	Shift Rows, Mix Column, add Keys	Based on S-Box
RC5	64	128	20	FEISTEL	8 bit s-box	XOR, Left Rotation, Right Rotation	Use Magic Constant
Fantomas	128	128	12	SPN	Bit slice S-Box	N/A	Depend on Master Key
Robin	128	128	16	FEISTEL	Bit slice S-Box	N/A	Depend on Master Key
L block	64	80	32	FEISTEL	4*4 s-box	XOR, addition Subtraction	Based on Round Function
HIGHT	64	128	32	FEISTEL	not based	XOR, Add, Sub	Key Whitening, Sub Keys
PRESENT	64	80	31	SPN	4*4 s-box	XOR, Add	Key Register
Piccolo	64	80	25	FEISTEL	2 S-box	NOR,XOR,XNOR	key Whitening,
Twine	64	80	36	FEISTEL	4*4 s-box	XOR, modolu2 add	GFS
PRINCE	64	128	12	SPN	4 bit S-box	AND,XOR,XNOR	Key Whitening
LED	64	80	48	SPN	4*4 s-box	Shift Rows, Mix Column, Sub Cells	Based on S-Box

3.8 PRINCE

PRINCE uses an FX construction. It has SPN structure where the key whitening is used in first two sub keys, whereas for the 12 rounds third sub key is the 64 bit key called PRINCE core. PRINCE applies distinctive stuff called α-reflection [25]. On 10 out of 12 rounds the best attack on this cipher is a multiple differential attack [26]. PRINCE is a good candidate for light weight cryptography by seeing low resource hardware performance.

3.9 RC5

RC5 is a Feistel network Structure and it uses data dependent rotations [27]. However RC5 was intended before lightweight cipher strategy became general. It is clearly lightweight as confirmed by its extensive use in WSN. The block, number of rounds and key size can be selected without restrictions, so we study RC5 32/20/16 i.e. a type of RC5 functioning on

two 32 bit words, using 20 rounds and a 16 byte key.

3.10 Robin

Robin is a 128-bits block cipher. Robin is comparable to Fantomas. The look-up table created diffusion layers and the construction of the S boxes makes the robin lightweight cryptographic algorithm good nominee for software applications [18].

3.11 Simon

Simon uses a Feistel structure. It consists of simple arithmetic and logic operations with a simple round function left circular shifts, bitwise XOR and bitwise AND. It has good performance in hardware implementations, but accomplishes decent consequences in software as well [28].

3.12 Speck

Speck is planned to deliver admirable outcomes in both software and hardware, but is adjusted for software execution on embedded devices. Its design structure is Feistel Network. It consists of simple arithmetic and logic operations with a simple round function left circular shift bitwise XOR and bitwise AND [28].

3.13 TWINE

With 16 branches twine is a comprehensive Feistel Network structure. The major step contains key adding and a 4 bit S box. With considerable advanced diffusion, the linear layer is a nibble permutation. It has good performance in hardware in terms of small foot print implementations, but accomplishes decent consequences in software as well in terms of RAM consumption [29].

4. Experimental Results and Discussion

In this section, first we present the performance analysis of implementing the lightweight cryptographic algorithm on three platforms: AVR microcontroller, MSP microcontroller and ARM microcontroller. Specification of these microcontrollers are given in **Table 2**. The analysis is based on three factors: code size, RAM foot print, and execution time. Secondly, we present the results of Association Rule Mining applied on constituent elements of lightweight cryptographic algorithms.

Table 2. Specification of Targeted Devices

Device	Flash Memory (KB)	SRAM (KB)
8-bit AVR	128	4
16-bit MSP	48	10
32-bit ARM	512	96

4.1 Scenario and Performance Metrics

Test handshake authentication covers the requirement of confirmation in the Internet of Things. The scenario considers an authentication protocol, where the lightweight cipher is

used in CTR mode of operation to encode 128 bits of information. Cipher round keys are kept in Flash memory while the master key is kept into the device. The information that has to be encoded is kept in random access memory along with the counter value. To decrease the random access memory usage, the process to encode the information is done in place. This situation is appropriate for actual constrained situations where random access memory usage and binary code size have to be very low, although the execution time should be sufficiently fast to avoid reducing the device's battery. A detailed performance comparison of the selected ciphers on three different platforms is presented in **Table 3**.

4.1.1 Code Size

The code size is measured in bytes and corresponds to the program footprint which is stored in the flash memory of the target device. The code size for each cipher implementation is computed using the size tool on object files generated by the compiler.

Table 3. Implementation Results of Performance Metrics on All Three Platforms

Cipher	Code Size			RAM			Execution Time		
	ARM [Bytes]	MSP [Bytes]	AVR [Bytes]	ARM [Bytes]	MSP [Bytes]	AVR [Bytes]	ARM [Cycles]	MSP [Cycles]	AVR [Cycles]
Speck	1628	618	666	196	58	54	3763	6054	3251
Simon	2156	732	772	216	72	62	4564	10930	5341
AES	1056	1438	1410	152	80	79	11623	4190	3175
RC5	1240	700	1712	172	54	58	10236	20543	8449
Fantomas	2260	1920	2496	216	78	108	41758	3646	5919
Robin	920	1942	2530	168	80	108	175092	4935	7813
L block	4124	976	1440	248	58	64	14365	18988	11183
HIGHT	988	982	1202	184	60	59	18418	23016	11335
PRESENT	676	1244	1416	128	58	54	1751	12226	15245
Piccolo	2160	966	1298	216	70	70	6195	21448	25745
Twine	636	1922	1528	128	136	64	1930	23938	21701
PRINCE	560	3418	4420	120	70	68	925	25340	17271
LED	1228	4422	2602	164	104	91	20531	148334	143317

4.1.2 RAM

The RAM consumption is divided into stack consumption and data consumption. The size of the data stored in the RAM is computed using the implementation information file and the size tool. It includes scenario specific RAM data such as data to encrypt keys, round keys or initialization vectors. The stack consumption is measured using gdb.

4.1.3 Execution Time

The execution time is expressed in 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. The metric is extracted for the four basic operations performed by a block cipher. To measure the execution time on AVR, cycle accurate simulator Avrora is used [30]. For MSP, the cycle accurate simulator MSP Debug is used.

4.2 Association Rule Mining

After having a performance analysis, we tried to investigate the relationships between the performance parameters and the constituent elements of the lightweight ciphers. For this purpose, we used the association rule mining. It is a popular and well researched method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness. We used Weka tool for extracting the association rules. To apply the association rule mining, we divided the data into two groups: constituent elements of ciphers (key size, block, S box, round function, rounds, key scheduling) and performance parameters (code size, RAM size, execution time). We labeled the data before applying association rule mining. The labels used are presented in **Table 4**.

The values of key size block size, number of rounds, round function, S-box table and key scheduling vary in different lightweight cryptographic algorithms. So, it would be interesting to know which value of these parameters results in good performance of lightweight ciphers.

Table 4. Labels used for Different Parameters

Parameter	Value	Label
Key Size	64 bit	A
	128 bit	B
Block Size	96 bit	A
	128 bit	B
	80 bit	C
S box	Not Based	A
	4*4	B
	8 bit	C
	Bit Slice	D
	2 S box	E

Round Function	XOR, LEFT, RIGHT SHIFT	A
	SHIFT ROW, MIX COLOUMS, ADD CONS	B
	XOR, ADD, SUB	C
	XOR, XNOR	D
Number of Rounds	10 to 15	A
	15 to 20	B
	20 to 30	C
	31 to 36	D
	36 to 40	E
	Above=F	F
Structure	Fiestal	A
	SPN	B
Key Scheduling	Based on Round Function	A
	Based on S Box	B
	Use Magic Constant	C
	Depend on Master Key	D
	Key Whiting	E
	Key Register	F
	GFS	G
Code Size	500 to 1000 bytes	S
	1000 to 1500 bytes	M
	1500 to 2200 bytes	L
	Above	VL
RAM Size	120 to 160 bytes	S
	160 to 200 bytes	M
	200 to 300 bytes	L
	300 above bytes	VL
Execution Time	900 to 2500 cycles	S
	2500 to 5000 cycles	M
	5000 to 10000 cycles	L
	10000 above cycles	VL

4.2.1 Rules for Code Size

For code size the extracted rules are presented below:

1. S-Box=A Round Function=A ==> CODE SIZE=S
2. Block Size=A Round Function=A ==> CODE SIZE=S
3. Round Function=A Key Scheduling=A ==> CODE SIZE=S
4. Block Size=A Key Scheduling=A ==> CODE SIZE=S
5. S-Box=A Round Function=A ==> CODE SIZE=S

Rules to keep the code size small on AVR

1. Key Size=B Structure=A ==> CODE SIZE=S
2. Key Size=B Round Function=A ==> CODE SIZE=S
3. Block Size=A Structure=A ==> CODE SIZE=S
4. .Block Size=A S-Box=A ==> CODE SIZE=S
5. Structure=A Round Function=A ==> CODE SIZE=S

Rules to keep the code size small on MSP

1. Key Size=B Structure=A ==> CODE SIZE=S
2. Key Size=B Round Function=A ==> CODE SIZE=S
3. Block Size=A Structure=A ==> CODE SIZE=S
4. Block Size=A S-Box=A ==> CODE SIZE=S
5. Structure=A Round Function=A ==> CODE SIZE=S

Rules to keep the code size small on ARM

We have found that to keep the code size of a cipher small, four elements are important including block size, S-box, round function and key scheduling. Although the elements are used in different associations but their type is almost same. For example, on all the three platforms block size is common that is “96 bits” and round function also common that is using simple arithmetic functions “XOR, Left Shift, Right Shift”. The similarities are more evident in **Table 5**, where the extracted rules are presented in tabular form.

Table 5. Summary of Rules to keep the code size small

Platform	Key Size	Block Size	Structure	S-Box	Round Function	Key Scheduling	Code Size
AVR CODE				A	A		S
		A			A		S
					A	A	S
		A				A	S
				A	A		S
MSP CODE	B		A				S
	B				A		S
		A	A				S
		A		A			S
			A		A		S
ARM CODE	B		A				S
	B				A		S
		A	A				S
		A		A			S
			A		A		S

S BOX and Key Scheduling also have same type by using S box “Not based” and Key Scheduling is “based on round function”. On AVR platform key size is not found in the extracted rules for code size. However, for MSP and ARM key size is same for keeping code size small that is “128 bits”. So, it is concluded from the extracted association rules that when block size is “96 bits”, round function is “XOR, Left Shift, Right Shift”, S box is “Not based”, key Scheduling is “based on round function”, and block size is “96”, the resultant cipher will have small code size that can be used for resource constrained environment.

4.2.2 Rules for RAM Size

For RAM foot print, the extracted rules are presented below:

1. Rounds=D ==> RAM=S
2. Round Function=C ==> RAM=S
3. Key Size=A Rounds=D ==> RAM=S
4. Key Size=A Round Function=C ==> RAM=S
5. Rounds=D Round Function=C ==> RAM=S

Rules to keep the RAM size small on AVR

1. Block Size=B ==> RAM=S
2. Key Size=A Block Size=B ==> RAM=S

3. Block Size=B Structure=A ==> RAM=S
4. Key Size=B 3 ==> RAM=S
5. Rounds=A ==> RAM=S

Rules to keep the RAM size small on MSP

1. S-Box=A ==> RAM=S
2. Round Function=A ==> RAM=S
3. Structure=A S-Box=A ==> RAM=S
4. Structure=A Round Function=A ==> RAM=S
5. Block Size=A ==> RAM=S

Rules to keep the RAM size small on ARM

There are some variations with respect to different platforms. However, key size, block size, no of rounds and round function are important elements for keeping RAM foot print small When we look at the Association Rule Mining results, we find that to keep the Ram size of a cipher small, four elements are important including key size, Block size, no of rounds and round function. Although the elements are used in different associations but their type is almost same as evident from **Table 6**, where the extracted rules are presented in tabular form.

Table 6. Summary of rules on all platforms to keep the RAM size small

Platform	Key Size	Block Size	Rounds	Structure	S-Box	Round Function	RAM Size
AVR RAM			D				S
						C	S
	A		D				S
	A					C	S
			D			C	S
MSP RAM		B					S
	A	B					S
		B		A			S
	B						S
			A				S
ARM RAM					A		S
						A	S
				A		A	S
				A		A	S
	A						S

4.2.3 Rules for Execution Time

For Execution time, the extracted rules are presented below:

1. Key Scheduling=D ==> EXE TIME=M
2. Block Size=B S-Box=D ==> EXE TIME=M
3. S-Box=D ==> EXE TIME=M

Rules to keep the Execution Time Medium on AVR

1. Block Size=B Structure=B ==> EXE TIME=S
2. Key Scheduling=D ==> EXE TIME=S
3. S-Box=D ==> EXE TIME=S

Rules to keep the Execution Time Small on MSP

1. Structure=A S-Box=A Round Function=A Key Scheduling=A ==> Execution Time=S
2. Block Size=A Structure=A S-Box=A Round Function=A Key Scheduling=A ==> Execution Time=S
3. Block Size=A Structure=A Key Scheduling=A ==> Execution Time=S
4. Block Size=A S-Box=A Round Function=A ==> Execution Time=S
5. S-Box=A Round Function=A Key Scheduling=A ==> Execution Time=S

Rules to keep the Execution Time Small on ARM

There are similarities between the AVR and MSP platform. However, on ARM platform different values are identified. To keep the execution time low, important elements are: block size, S box, round function, structure and key scheduling.In case of AVR and MSP S box “Bit Slice” is important while in case of ARM S box “not based” is important, as can be seen above. Similarly, for AVR and MSP block size “128 bits” is important whereas for ARM block size “64 bits” is important to keep the execution time low. On all the three platforms key size “64 Bits”, no of rounds “31 to 36”, block size “128 bits” and structure “Fiestal” are common constituent elements. Although round function may vary a little bit when platform has changed from AVR 8 bit to ARM 32 bit. In case of AVR round function “XOR, ADD, SUB” is important while in case of ARM “XOR, LEFT, RIGHT shift” is vital as evident from **Table 7**.

Table 7. Summary of rules on all platforms to keep the EXE TIME small

Platform	Block Size	Structure	S-Box	Round Function	Key Scheduling	EXE Time
AVR EXE TIME					D	M
	B		D			M
			D			M
MSP EXE TIME	B	B				S
					D	S
			D			S
ARM EXE TIME		A	A	A	A	S
	A	A	A	A	A	S
	A	A			A	S
	A		A	A		S
			A	A	A	S

4.3 Recommendations

Keeping in view all the rules and findings for each element, following recommendations are made:

- ***Key Size:*** not important for execution time. 128 bits can be used to keep code size small. 64 bits can be used to keep RAM foot print small.
- ***Block Size:*** 96 bits can be used to keep code size small and keep execution time low on ARM. 128 bits can be used to keep RAM foot print small and keep execution time low on AVR and MSP.
- ***Rounds:*** it is not found significant for execution time and code size. 31-36 can be used to keep RAM foot print small.
- ***Structure:*** Fiestal can be used to keep code size, RAM foot print small and execution time low.
- ***S-Box:*** Not based can be used to keep code size, RAM foot print small and execution time low. For AVR and MSP, Bit Slice can be used to keep execution time low.
- ***Round Function:*** "XOR, LEFT, RIGHT SHIFT" can be used to keep code size, RAM foot print small and execution time low. "XOR, ADD, SUB" can be used on AVR to keep RAM foot print small.
- ***Key Scheduling:*** not important for RAM foot print. "Based on round function" can be used for keeping code size small. Same value can be used to keep execution time low on ARM, whereas "depend on master key" can be used to keep execution time low on AVR and MSP.

5. Conclusion

In this paper, we presented an evaluation of 13 light weight block ciphers used for secure communication in Internet of Things. We compared and ranked the ciphers based on three metrics: code size, RAM foot print and execution time. We analyzed the performance of these ciphers on three different platforms: 8 bit, 16 bit and 32 bit. We further dissected these

ciphers into their constituent elements and investigated the role of these elements in the performance of ciphers. We used association rule mining to find associations among the constituent elements. Based on the results, we come up with few guidelines regarding the design of lightweight ciphers. Designer must always remember the algorithm prerequisites to be implemented into the devices. So, intention must be to consume less device resource e.g. memory (RAM), code size, execution time etc. The S-box have to be small generally (4 × 4) bits for compact operation. Simultaneously, it must deliver compulsory non-linearity to the algorithms. Key schedule have to be easy so that it take small space, hence the recently planned cipher keep the keys fixed. As the algorithms are straightly implemented into the device, therefor no need for re-keying. The permutation has to be designed in such a way that it attains optimum stability among mixing of bits and areas. The designer must attempt to accomplish an optimum balance amid the different parameters of cost, security and performance. In short, this research work aimed to provide basement to improve the cipher in several ways like code size, size of memory (RAM Requirement), and execution time. This paper will serve as a guideline for cryptographic designers to design improved ciphers for resource constrained environment like Internet of Things.

References

[1] Yu, Hong, Jingsha He, Ting Zhang, Peng Xiao, and Yuqiang Zhang, "Enabling end-to-end secure communication between wireless sensor networks and the Internet," *World Wide Web*, vol. 16, no. 4, pp.515-540, 2013 Article (CrossRef Link).

[2] Garcia-Morchon, Oscar, Sye Loong Keoh, Sandeep Kumar, Pedro Moreno-Sanchez, Francisco Vidal-Meca, and Jan Henrik Ziegeldorf, "Securing the IP-based internet of things with HIP and DTLS," in *Proc. of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ACM. pp. 119-124. 2013. Article (CrossRef Link).

[3] López, Javier, and Jianying Zhou, "eds," *Wireless sensor network security*, Vol. 1. Ios Press, 2008. Article (CrossRef Link).

[4] Bhattasali, Tapalina, "LICRYPT: Lightweight Cryptography Technique for Securing Smart Objects in Internet of Things Environment," *CSI Communications* 2013. Article (CrossRef Link).

[5] Eisenbarth, Thomas, Sandeep Kumar, Christof Paar, Axel Poschmann, and Leif Uhsadel, "A survey of lightweight-cryptography implementations," *IEEE Design & Test of Computers*, vol. 24, no. 6 pp. 522-533. 2007 Article (CrossRef Link).

[6] Poschmann, Axel York, "Lightweight cryptography: cryptographic engineering for a pervasive world," *PH. D. THESIS*, 2009. Article (CrossRef Link).

[7] Tausif, Muhammad, Javed Ferzund, and Sohail Jabbar, "Emergence of Internet of Things in Current Technological Era," *JOURNAL OF PLATFORM TECHNOLOGY*, vol. 2, no. 3, pp. 19-34, 2014. Article (CrossRef Link).

[8] Zhang, Xueying, Howard M. Heys, and Cheng Li, "Energy efficiency of symmetric key cryptographic algorithms in wireless sensor networks," in *Proc. of Communications (QBSC), 2010 25th Biennial Symposium on*, pp. 168-172. 2010. Article (CrossRef Link).

[9] Koo, Woo Kwon, Hwaseong Lee, Yong Ho Kim, and Dong Hoon Lee, "Implementation and analysis of new lightweight cryptographic algorithm suitable for wireless sensor networks," in *Proc. of Information Security and Assurance*, pp. 73-76., 2008. Article (CrossRef Link).

[10] Kerckhof, Stéphanie, François Durvaux, Cédric Hocquet, David Bol, and François-Xavier Standaert, "Towards green cryptography: a comparison of lightweight ciphers from the energy viewpoint," in *Proc. of International Workshop on Cryptographic Hardware and Embedded*

Systems, Springer Berlin Heidelberg, pp. 390-407, 2012. Article (CrossRef Link).

[11] Law, Yee Wei, Jeroen Doumen, and Pieter Hartel, "Benchmarking block ciphers for wireless sensor networks," in *Proc. of Mobile Ad-hoc and Sensor Systems, IEEE International Conference on*, pp. 447-456, 2004. Article (CrossRef Link).

[12] Skipjack, N. I. S. T., "KEA algorithm specifications," 1998. Article (CrossRef Link).

[13] Karlof, Chris, Naveen Sastry, and David Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proc. of the 2nd International Conference on Embedded networked Sensor Systems*, pp. 162-175, 2004. Article (CrossRef Link).

[14] Jana, Swarnendu, Jaydeb Bhaumik and Manas Kumar Maiti, "Survey on lightweight block cipher," *International Journal of Soft Computing and Engineering*, vol. 3, pp. 183-187, 2013. Article (CrossRef Link).

[15] Kushwaha, Prabhat Kumar, M. P. Singh, and Prabhat Kumar, "A Survey on Lightweight Block Ciphers," *International Journal of Computer Applications*, vol. 96, no. 17 , 2014. Article (CrossRef Link).

[16] Katagi, Masanobu, and Shiho Moriai, "Lightweight cryptography for the internet of things," *Sony Corporation*, pp.7-10, 2008. Article (CrossRef Link).

[17] Daemen, Joan, and Vincent Rijmen, "The design of Rijndael: AES-the advanced encryption standard," *Springer Science & Business Media*, 2013. Article (CrossRef Link).

[18] Grosso, Vincent, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varıcı. "LS-designs: Bitslice encryption for efficient masked software implementations," in *Proc. of International Workshop on Fast Software Encryption*, Springer Berlin Heidelberg, pp. 18-37, 2014. Article (CrossRef Link).

[19] Hong, Deukjo, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee et al., "HIGHT: A new block cipher suitable for low-resource device," in *Proc. of International Workshop on Cryptographic Hardware and Embedded Systems*, Springer Berlin Heidelberg, pp. 46-59, 2006. Article (CrossRef Link).

[20] Wu, Wenling, and Lei Zhang, "LBlock: a lightweight block cipher," in *Proc. of International Conference on Applied Cryptography and Network Security*, Springer Berlin Heidelberg, pp. 327-344., 2011. Article (CrossRef Link).

[21] Boura, Christina, María Naya-Plasencia, and Valentin Suder, "Scrutinizing and improving impossible differential attacks: applications to CLEFIA, Camellia, LBlock and Simon," in *Proc. of International Conference on the Theory and Application of Cryptology and Information Security*, Springer Berlin Heidelberg, pp. 179-199. 2014. Article (CrossRef Link)

[22] Patil, Abhijit, Gaurav Bansod, and Narayan Pisharoty, "Hybrid Lightweight and Robust Encryption Design for Security in IoT," *International Journal of Security and Its Applications*, vol. 9, no. 12 pp. 85-98, 2015. Article (CrossRef Link).

[23] Shibutani, Kyoji, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai, "Piccolo: an ultra-lightweight blockcipher," in *Proc. of International Workshop on Cryptographic Hardware and Embedded Systems*, Springer Berlin Heidelberg, pp. 342-357, 2011. Article (CrossRef Link).

[24] Bogdanov, Andrey, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe, "PRESENT: An ultra-lightweight block cipher," in *Proc. of International Workshop on Cryptographic Hardware and Embedded Systems*, Springer Berlin Heidelberg, pp. 450-466, 2007. Article (CrossRef Link).

[25] Borghoff, Julia, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander et al., "PRINCE–a low-latency block cipher for pervasive computing applications," in *Proc. of International Conference on the Theory and Application of Cryptology*

and Information Security, Springer Berlin Heidelberg, pp. 208-225. 2012. Article (CrossRef Link).

[26] Canteaut, Anne, Thomas Fuhr, Henri Gilbert, María Naya-Plasencia, and Jean-René Reinhard, "Multiple differential cryptanalysis of round-reduced PRINCE," in *Proc. of International Workshop on Fast Software Encryption*, Springer Berlin Heidelberg, pp. 591-610, 2014. Article (CrossRef Link).

[27] Rivest, Ronald L, "The RC5 encryption algorithm," in *Proc. of International Workshop on Fast Software Encryption*, Springer Berlin Heidelberg , pp. 86-96, 1994. Article (CrossRef Link).

[28] Beaulieu, Ray, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and LouisWingers, "The SIMON and SPECK lightweight block ciphers," in *Proc. of Proceedings of the 52nd Annual Design Automation Conference*, p. 175. 2015. Article (CrossRef Link).

[29] Suzaki, Tomoyasu, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi, "Twine: A lightweight, versatile block cipher," in *Proc. of ECRYPT Workshop on Lightweight Cryptography*, pp. 146-169. 2011. Article (CrossRef Link).

[30] Dinu, Daniel, Yann Le Corre, Dmitry Khovratovich, Léo Perrin, Johann Großschädl, and Alex Biryukov, "Triathlon of Lightweight Block Ciphers for the Internet of Things," *IACR Cryptology ePrint Archive*, pp. 209, 2015. Article (CrossRef Link).

Muhammad Tausif is currently working as Lecturer with the department of Computer Sciences COMSATS Institute of Information Technology (CIIT), Vehari, Pakistan where he teaches course on Computer Sciences and Software Engineering, He has completed his MS degree in Computer Sciences from COMSATS Institute of Information Technology, Sahiwal, Pakistan in 2016. He received his B.sc degree in Computer Engineering from Bahauddin Zakariya University, Multan, Pakistan in 2011. His main research interests include Internet of Things and sensor networks. . He has published one international journal papers. He is also supervised many Final year projects in different domains of computer sciences. He has led different research and development projects in COMSATS Institute of information Technology Vehari.

Javed Ferzund is an associate professor at Department of Computer Science, COMSATS Institute of Information Technology, Sahiwal, where he served as Head of Department from 2013-2015. He received PhD degree from Graz University of Technology, Austria in 2009. His main research interests include Big Data Analytics, Internet of Things and Machine Learning. Particularly, he is interested in applications of IoT and Big Data in the Agro-Informatics and Bioinformatics fields. Currently, he is leading the Big Data Analytics Research Group at COMSATS Institute Sahiwal.

Sohail Jabbar is an Assistant Professor at Department of Computer Science, National Textile University, Faisalabad, Pakistan. He has been Post-Doctorate Researcher at Network Lab, Kyungpook National University, Daegu, South Korea. He also served as Assistant Professor with the Department of Computer Science, COMSATS Institute of Information Technology (CIIT), Sahiwal and headed Networks and Communication Research Group there. He received many awards and honors from Higher Education Commission of Pakistan, Bahria University, CIIT, and the Korean Government. He received the Research Productivity Award from CIIT in 2014 and 2015. He has been engaged in many National and International Level Projects. His research work is published in various renowned journals and magazines of IEEE, Springer, Elsevier, MDPI, Old City Publication and Hindawi, and conference proceedings of IEEE and ACM. He has been the reviewer for leading journals (ACM TOSN, JoS, MTAP, AHSWN, ATECS, among many) and conferences (C-CODE 2017, ACM SAC 2016, ICACT 2016, among others). He is currently engaged as TPC member chair in many conferences. He is guest editor of Sis in Future Generation Computer Systems (Elsevier), Peer-to-Peer networking and Applications (Springer), Journal of Information and Processing System (KIPS), Cyber Physical System (Taylor & Francis). His research interests include Internet of Things, Wireless Sensor Networks and Software Defined Networking.

Raheela Shahzadi is currently working as Lecturer in department of Computer Science, COMSATS Institute of Information Technology, Sahiwal, Pakistan. She has completed her MS Degree in Computer Science from COMSATS Institute of Information Technology, Sahiwal, Pakistan in 2015 with Distinction. She has completed her bachelor degree COMSATS Institute of Information Technology, Sahiwal, Pakistan in 2013 with Distinction. She has two and half year teaching experience at COMSATS Institute of information Technology, Sahiwal, Pakistan. She has published one international conference article. Her research interests include Data mining, internet of things (IoT), wireless sensor network (WSN), expert system (ES) and digital image processing (DIP).