Received: January 22, 2019

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Performance Evaluation of Newly Proposed Lightweight Cipher, BRIGHT

Deepti Sehrawat[1]* **Nasib Singh Gill[1]**

[1]*Department of Computer Science & Applications,*
Maharshi Dayanand University, Rohtak, Haryana, India
* Corresponding author's Email: dips.scorpio@gmail.com

Abstract: Lightweight security algorithms are tailored for resource-constrained environment. To improve the efficiency of an algorithm, usually, a tradeoff is involved in lightweight cryptography in terms of its memory requirements and speed. By adopting several performance enhancement techniques, a security framework for IoT enabled applications is presented in this paper. Proposed BRIGHT family of ciphers is comparably better than existing lightweight ciphers and support a range of block and key sizes for constraint environment. It enables users to match their security needs with application requirements by supporting a range of cryptographic solutions. The BRIGHT family of ciphers is a software-oriented design. The performance of BRIGHT family of lightweight ciphers is evaluated on different parameters. All versions of BRIGHT family ciphers fulfill Strict Avalanche Criteria, key sensitivity test, and randomness test. BRIGHT family ciphers show better performance in terms of memory requirements, cost and speed as compared to existing lightweight ciphers.

Keywords: Performance evaluation, BRIGHT, Cryptographic solutions, Lightweight block cipher, ARX, GFN, Feistel block ciphers.

1. Introduction

In IoT field, various resource constraints devices communicate in the network using RFID (Radio Frequency Identification Devices) which is a fast-growing technology that allows automated identification of items having RFID tags. These RFID tags are integrated circuits having an antenna. RFID reader, through radio interface, communicate with these RFID tags [1]. WSN have some form of design limitations, different communication and deployment patterns that pose a number of security problems to it. Moreover, making a good security solution requires analyzing security requirements of WSN [2]. A number of low cost and low energy sensor nodes are involved in Wireless Sensor Networks (WSNs). These nodes communicate via wireless links at a short distance. For such a platform, regular algorithms may incur very high power consumption. Like sensor networks which connects numerous sensors to a central hub. Batteries or solar panels run these sensors. For all of these connected devices information security is evidently necessary [3]. To provide high security and privacy, cryptographic solutions must be used. However, due to very low available energy, the limited size of ROM and RAM consumption and high-security demand in a resource-constrained environment, lightweight cryptographic security solutions are required [4]. Lightweight ciphers vary from traditional cryptographic algorithms. These can be implemented either through software implementation or through hardware implementations. Software oriented cipher designs provide more flexibility at lower costs on manufacturing and maintenance as compared to hardware implementations. Even providing strong resistance against mathematical attacks could not protect hardware-oriented block ciphers from side channel attacks thereby losing its keys. So a good software design is required to provide enough security guard against attacks.

This paper proposes a software-oriented family of highly optimized lightweight block cipher, BRIGHT. The structure of BRIGHT family of cipher is so designed that makes it more efficient than existing

ones. It uses ARX operations, pre-key whitening and round permutation which are so designed that it not only leads to smaller code size it also provides fast diffusion. For resource constraint devices in IoT environment, proposed cipher supports a range of block and key sizes. There is a total of 6 variants of BRIGHT cipher with prevalent block sizes 64-bit and 128-bit. This enables users to match their security needs with application requirements. All 6 versions of BRIGHT family ciphers fulfill Strict Avalanche Criteria (SAC) along with key sensitivity test and randomness test. In this paper performance of BRIGHT cipher is evaluated on a 64-bit processor and is compared with benchmarked ARX-based lightweight block ciphers. All variants of BRIGHT cipher have a comparably lower cost, low memory utilization, and high speed.

The rest of the paper is structured as follows: section 2 summarized related work followed by section 3 describing performance enhancement implementation techniques and ideas. Section 4 explains BRIGHT cipher design, and the performance evaluation of the proposed family of lightweight cipher BRIGHT is presented in section 5.

2. Related work

Some of the benchmarked lightweight block ciphers with ARX structure are LEA (2015) [5], HIGHT [6], SIMON [7], SPECK [7], Chaskey [8], RoadRunneR [9] and SPARX [10].

LEA has a block size of 128-bits and supports three different key sizes viz. 128, 192 and 256-bits long. LEA uses a generalized FN with four branches of 32-bit each. It is optimized for 32-bit processor than a 64-bit processor [5]. There exists a 15 round boomerang attack on this cipher [11].

HIGHT is another ARX based lightweight block cipher optimized for 8-bit operations [6]. Designers studied a 26 round impossible differential attack on HIGHT [12]. Some other attacks implemented on HIGHT are multidimensional zero-correlation attack [13], biclique attack [14], and related-key attack [15].

SPECK and SIMON family of lightweight ciphers have 10 instances of each. Speck applies simple round functions due to which its code size is very small. There exist differential attacks targeting 19 rounds out of 26 in SPECK-64/96 and 20 rounds out of 27 in Speck-64/128 [16]. SIMON cipher with a block size of 128-bit is comparable with LEA, LEA's performance exceeds that of SIMON in both 32-bit and 64-bit processors. Whereas performance of SPECK exceeds LEA'S performance in 64-bit processor because of SPECK's 64-bit addition in a 64-bit processor.

CHASKEY is a permutation-based Message Authentication Code (MAC) lightweight cipher optimized for 32-bit processor [8]. Because of its Even-Mansour structure, key generation does not follow any key schedule, it simply consists of two shifts and two conditional XORs with the state for two sub-keys. Differential-linear attack targeting 7 out of 8 rounds is the best-known attack against Chaskey [17].

RoadRunneR is a 64-bit lightweight block cipher with key sizes 80-bits or 128-bits. It is optimized for an 8-bit processor. The Feistel function is an SPN composed of four 4-bit S-box layers, three linear layers, and three key additions [9]. The best-known attack is high-probability truncated trail targeting 5 out of 7 rounds in RoadRunneR-128 [18].

SPARX cipher follows LTS (Long Trail design Strategy) and supports two block sizes viz. 64-bits and 128-bits. Three versions are supported by SPARX, these are Sparx-64/128 having 8 steps with 3 rounds in each step, Sparx-128/128 having 8 steps with 4 rounds in each step, and Sparx-128/256 having 10 steps with 4 rounds in each step [10]. There exists an integral attack covering 15 and 22 rounds out of 24 and 32 rounds of SPARX-64/128 and SPARX-128/128 respectively [10].

3. Performance enhancement ideas

It is really a challenging task to choose the best algorithm in terms of its memory requirements and energy efficiency [19]. This section presents some useful performance enhancement implementation ideas to improve the software based primitives.

3.1 Key schedule

A complex key schedule results in the increase in RAM size or gate area. For lightweight algorithms, an increase in memory size is not acceptable so it is common to not consider related key attacks in lightweight cryptography and allows to use simple key scheduling [10]. Furthermore, if keys can be chosen independently and randomly then it provides resistance against related-key attacks [20]. Small key sizes usually 80 bits or smaller offers slight security margin against brute-force search. Time-memory-data tradeoff can turn out to be an issue if in multi-key setting key size is very small [21]. To limit the data available to the attacker, changing key frequently provides resistance to the related key attacks [22]. XORing actual key prior to its use thwarts the weak key attacks in a cipher. Same was realized by the authors in [23] in which a lightweight algorithm SIT was proposed in which actual key is not used, the key is first XORed before its use.

3.2 Internal state

To minimize the constraints posed by IoT, most of the lightweight block algorithms considered to use smaller internal states in the form of block sizes. As a result, most of the block ciphers in IoT use a block size of 64-bits. This results in low memory footprints in software implementation as well as in hardware implementation of the security algorithm. This reduced block size, as a result, creates a problem as the security of some modes of operation like CBC (Cipher Block Chaining) erodes rapidly when the number of n-bit blocks encrypted reaches $2n/2$ [24].

XOR function is mostly used in ARX-based ciphers. Using addition mod 2b is more useful in place of bitwise XOR because for several reasons. Using a mod 2b key addition improves the diffusion layer by introducing sufficient non-linearity. This is done at the same cost and same speed as by bitwise XOR. It also helps to provide enough resistance against structural attacks [25]. Rotations are not considered good operations, especially rotations of same amount on neighboring large operands. Furthermore, different rotation amounts offer a reasonable amount of different trade-offs between efficiency and security (especially linear and differential probabilities) [26]. So choosing carefully best rotation amounts is considered as a good decision in a cipher design [27].

For most efficient implementation choosing the word size is an important decision, it is advisable to choose the word size equal to the register size of the microcontroller. Furthermore, if the register size of the microcontroller is greater than that of word/ operand size, the worst results are obtained. Consequently, normal implementation efficiency is achieved when the word size is a multiple of register size [26]. For optimization, using same size bitwise operations as the platform is, provides good efficiency because these can be directly mapped to the ARM instructions. For example, using 32-bitwise ARX operations for a 32-bit platform can be mapped without additional effort to 32-bit ARM instruction [28].

It is good to define the integer type length explicitly which are available in "inttypes.h" like *int8_t* in place of *char*, *int32_t* in place of *int*. This gives consistent behavior and when defined for local variables, it improves register usage.

4. Proposed BRIGHT cipher

By adopting several performance enhancement techniques presented in section 3, a security framework for IoT enabled applications is presented in this section. The main features of the proposed design are:

1) Supports two prevailing block sizes viz. 64-bits and 128-bits.
2) Minimum key size chosen is 80-bits.
3) Pre-key whitening is used to thwart the weak key attacks.
4) Use of addition mod 2b for better diffusion.
5) Different amount of rotations to improve security.
6) Round permutation for fast diffusion.
7) Chosen word size is equal to register size i.e. 64-bitwise ARX operations for 64-bit platform.
8) Defined integer type length explicitly for better register usage.

4.1 Notation

Throughout the paper the following notations are used:

$\boldsymbol{V_i}$	Word/ Branch
+	Addition modulo 2^n
^	n-bit exclusive OR
$\boldsymbol{x}$<<<$\boldsymbol{m}$	Left circular shifts by m-bits
$\boldsymbol{x}$>>>$\boldsymbol{m}$	Right circular shifts by m-bits
$\boldsymbol{Mk}$	Master-key
$\boldsymbol{Rk_i}$	Round key for i^{th} round
$\boldsymbol{\&}$	Bitwise AND

4.2 Design

BRIGHT cipher is a family of new lightweight Feistel type block cipher which is based on 4-branch GFN. Here the term lightweight does not only mean that a security algorithm is suitable for some constraint platform but it should be platform independent. Proposed family of BRIGHT cipher has an application independent design choice that provides good performance. Since devices and the applications vary greatly, proposed cipher provides a wide range of options in form of block sizes and key sizes. Prevailing block sizes are 64-bit and 128-bit and key sizes are related to the desired security level, for instance, a very low-cost device may achieve sufficient security using just 64-bits of a key while on the other hand, more sensitive applications may require 256 bits of key. BRIGHT has low decryption overhead, means it is easy to apply decryption from encryption. BRIGHT operations are fast and are supported in multiple platforms in an efficient and parallel way. A total of six instances of BRIGHT cipher has proposed, these are, BRIGHT 64/80, BRIGHT 64/96, BRIGHT 64/128, BRIGHT 128/128, BRIGHT 128/192 and BRIGHT 128/256. A general description of BRIGHT *n/m* describes BRIGHT cipher with *n*-bit block size and *m*-bit key size.

Table 1. Parameters for all versions of BRIGHT

Block Size *4n*	Key size *mn*	Word Size *n*	Key Words *m*	Rounds	Rotation Amount *a*	*b*
64	80	16	5	32	2	6
	96		6	33	2	6
	128		8	34	2	6
128	128	32	4	35	5	8
	192		6	36	5	8
	256		8	37	5	8

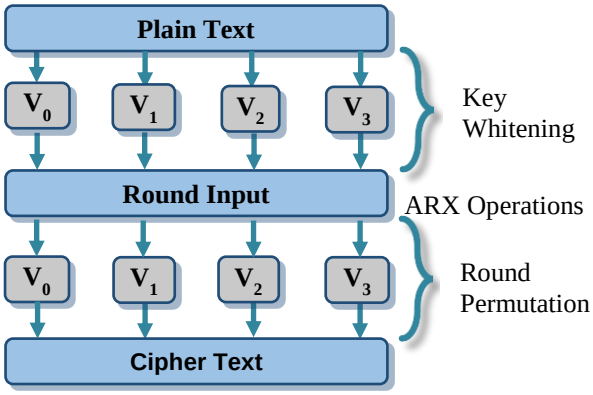

Figure. 1 Layers in the BRIGHT family of ciphers

Performance of BRIGHT is better than existing lightweight ciphers. To thwart most of the attacks on this cipher, first, the minimum number of rounds '*R*', required to reach complete diffusion is found for all the variants of BRIGHT family. Then, the actual number of rounds is given by applying the formula *3R, 3R+1, 3R+2* for BRIGHT 64/80, BRIGHT 64/96 and BRIGHT 64/128.

Table 1 gives the block size, key size, and the number of rounds for BRIGHT family ciphers.

The encryption function of BRIGHT has three layers, these are key whitening, ARX operations, and round permutation. Fig. 1 depicts the structure of proposed cipher. There are three layers, first, the input plain text is divided into 4 words/ branches of equal length on which key whitening is applied in the first layer. After applying pre-key whitening, the second layer performs ARX operations on the round input. The last layer performs round permutation to give the output for the next round. The operations of second and third layers are performed for a specific number of times which is equal to the number of rounds.

a) Key Whitening:

To provide immunity against brute force and MITM attacks, the actual key is first XORed as a pre-whitening affect. This does not provide any immunity to analytical attacks like linear cryptanalysis and differential cryptanalysis. We have used only pre-whitening, post whitening just adds to the code length. Furthermore, it is due to the key whitening that makes it almost impossible to extend the attack by even one round which requires searching for all *n*-keys.

b) ARX Operations:

A software implementation of ARX ciphers gives efficient implementation in a parallel way. In BRIGHT, ARX operations are so arranged that it leads to not only smaller code size, it also improves diffusion in form of fast diffusion. To add non-linearity to the cipher addition modulo 2^n is used. This is done at the same speed as by bitwise XOR. Addition modulo 2^n are non-linear operations and propagate differences indefinitely. Different amount of left and right circular shifts are used to add diffusion in the cipher and XOR operation is used in combination with addition modulo 2^n to improve the diffusion layer. This improves the diffusion layer and offers different trade-offs between security and efficiency.

c) Round Permutation:

Generalized Feistel Network has slow diffusion property due to which some attacks like impossible differential cryptanalysis can be applied easily to these ciphers. To further improve the diffusion speed in BRIGHT cipher, round permutations are added at the end of each round.

Encryption is a composition of round functions given by Eq. (1):

$$R_{kr-1} \circ R_{kr-2} \circ \dots R_{k1} \circ Rk_0 \quad (1)$$

This can be read from right to left. Decryption uses the same round function but here the order of the operations viz. key whitening, round constants, round keys, and ARX operations are reversed. In place of addition modulo 2^n, subtraction modulo 2^n is replaced. All versions are BRIGHT performs fast diffusion and are supported in multiple i.e. 8/16/32/64-bit platforms in an efficient and parallel way. The operations have so arranged that lead to fast encryption and small code size.

4.3 Encryption algorithm

INPUT: Block (*n*-bits long) and master-key (*m*-bits long).

OUTPUT: Block (*n*-bits long)

1) Compute *n*-bits after applying initial key-whitening by partitioning *n*-bits plain text '*P*' into 4 equal parts and XORing with sub-key i.e. $P = P_0 \wedge k_0 \mid P_1 \wedge k_1 \mid P_2 \wedge k_2 \mid P_3 \wedge k_3$.

2) Apply ARX operations on P_i for i^{th} round i.e.
$P_{i1}=(P_{i1}<<<b)\wedge Rk_i$,
$P_{i0}=((P_{i0}+P_1)\&C)<<<a$,
$P_{i3}=P_{i3}<<<b$,
$P_{i2}=(P_{i2}+P_{i3})\&C$,
$P_{i3}=P_{i3}\wedge P_{i2}$,
$P_{i0}=P_{i0}\wedge P_{i3}$,
$P_{i3}=P_{i0}\wedge P_{i3}$,
$P_{i2}=P_{i2}\wedge P_{i1}$,
$P_{i1}=(P_{i1}<<<b)\wedge P_{i2}$,
$P_{i2}=P_{i2}<<<a$
3) Perform Round Permutation i.e. $P_{i0}|P_{i1}$, $P_{i1}|P_{i2}$, $P_{i2}|P_{i3}$, $P_{i3}|P_{i0}$.
4) Perform steps 2 and 3 for the remaining rounds.

4.4 Key scheduling

Key scheduling is the most fundamental and crucial part in a cipher's design and it can lead to break down the cipher if otherwise not designed cleverly. The entire security of the data is dependent on the data scheduling and key scheduling of a cipher design. Furthermore, GFN based algorithms are composed of several rounds and each round requires a separate key. Classes of weak keys can be found if there is relatively weak key scheduling used by a cipher which ultimately makes the cipher non-resistant to different key scheduling attacks like zero correlation attack, MITM attack, weak key attack, and their variants. BRIGHT key scheduling is inspired from SPECK key scheduling. Key scheduling stores a user-defined master key is in the register key, represented by Eq. (2).

$$\boldsymbol{K = k^n\, k^{n-1}\, k^{n-2} \ldots k^2\, k^1\, k^0} \quad (2)$$

After applying initial key whitening (where the original sub-key from the master key is XORed with the plain text), sub-keys are derived from the master key which are fed to the BRIGHT rounds, i.e. 'n' keys for n-rounds. More confusion-diffusion is introduced with the increase in the number of rounds, providing more security. Furthermore, no attack should be faster than exhaustive key search i.e. brute force attack. To provide resistance against exhaustive search attack, the master key length must be large enough so that it becomes difficult for the attacker to perform 2^{k-1} encryptions for the key searching attacks. Also, longer key sizes require more number of rounds so that every key bit affect the ciphertext bit in a similar way i.e. without measurable differences which would otherwise allow any cryptanalysis.

Key scheduling part uses the round function to generate round keys k_i. Let K be a key for a BRIGHT $4n$ block cipher. We can write $K = (l_{m-2}, \ldots l_0, k_0)$ where $l_i, k_0 \in GF(4)^n$, for a value of m in {2,3,4}. Sequences k_i and l_i are defined by Eq. (3) and Eq. (4) respectively.

$$\boldsymbol{L_{i+m-1} = (k_i + l_i>>>2) \wedge i} \quad (3)$$

$$\boldsymbol{K_{i+1} = k_i <<<5 \wedge l_{i+m-1}} \quad (4)$$

The value k_i is the i^{th} round key, for $0 \le i < T$.

5. Performance evaluation

Most of the IoT devices interact with other similar devices and higher-end backend server. These constraints systems have to perform some functions like aggregating data received from sensors or inventory. Hence, lightweight block cipher should support sound performance on 64-bit processors. There is a need for flexible secure block cipher which is able to perform well on all of these platforms. Confusion and diffusion are two main parameters which test the suitability of a cipher. Confusion means that there must exist a complicated relation of ciphertext with plaintext and key. This is achieved by mixing operations in a complicated way. Diffusion means that every ciphertext bit must be influenced by every plaintext bit and a key bit. Spreading every plaintext bit influence over many ciphertext bits hides the statistical structure of the plaintext. A cipher is considered secure if a cipher proves to be secure against all known cryptanalytic attacks until it is realized otherwise. A number of cryptanalytic attacks can be applied to a cipher. Out of these, two main attacks are linear and differential cryptanalysis. In this paper, our main goal for the security of BRIGHT is to thwart possible attacks and to provide sufficient security margin against unknown attacks.

Implementation results of the proposed BRIGHT family of lightweight ciphers on a processor Intel (R) Core (TM) i5-2430M CPU @ 2.40 GHz are observed.

Evaluation Parameters

Following criteria are taken to evaluate the security of the proposed BRIGHT family of ciphers.

1. Strict Avalanche Criteria (SAC): According to Strict Avalanche Criteria (SAC), the test is considered to be perfect if a single bit change in input (key/ plaintext) results in 50% change in the bits. Cipher fulfilling SAC has a higher probability to thwart all possible attacks [29]. Contrary, if SAC is not satisfied, it is considered that poor randomization occurs and cipher is not considered good. Table 2 summarizes the results of diffusion for BRIGHT 64/128, when there is a single-bit change in plaintext.

Table 2. Summarizes the results of diffusion and randomness test for BRIGHT 64/128, when there is a single-bit change in plaintext (Key (Hexadecimal) = 07 04 02 03 08 29 2a 0b 10 11 4f ae)

Plaintext (Hexadecimal)	**Ciphertext (Binary)**								**Number of**	
									Zero's	One's
00 00 00 00 00 00 00 00	10010011	11001100	01110111	01101100	11100000	11100001	01100111	00110010	31	33
00 00 00 00 00 00 00 01	01010100	01110100	11001111	10011011	01000000	10010100	00101110	10000111	34	30
00 00 00 00 00 00 00 02	00101100	11111111	11010101	10100100	11101000	10100000	01010001	10011010	32	32
00 00 00 00 00 00 00 03	11101111	11110100	00100000	01000010	01111000	00101001	11000110	01100000	36	28
00 00 00 00 00 00 00 04	11011010	00010100	11010000	11000010	01111111	10111110	01100001	01001100	32	32
00 00 00 00 00 00 00 05	11100110	11001100	10010000	10001010	00111010	01001110	00010001	10011011	35	29
00 00 00 00 00 00 00 06	00101110	10110011	00010000	01001000	01001011	11001110	10110000	01000111	36	28
00 00 00 00 00 00 00 07	11000010	00010111	10010111	11111111	11001111	00111111	00000110	00111101	25	39
Number of bits changed	**(0, 1) = 35 (0, 2) = 27 (0, 4) = 37 (1, 3) = 38 (2, 3) = 34 (3, 7) = 35 (4, 6) = 28 (6, 7) = 35 (5, 7) = 34 Average diffusion = 33.66 (52.60%)**								**32.62**	**31.38**

Table 3. Summarizes the diffusion range and average diffusion between plaintext block and ciphertext block when there is a single-bit change in plaintext

Block Length	**Key Length**	**Diffusion Range**	**Average diffusion**	**Diffusion Percentage**
64	80	29 - 35	32.67 bits	51.04 %
64	96	26 - 37	31.89 bits	49.82 %
64	128	27 - 38	33.66 bits	52.60 %
128	128	48 - 71	62.78 bits	49.05 %
128	192	57 - 68	65.33 bits	51.04 %
128	256	56 - 72	63.56 bits	49.65 %

Table 4. Summarizes the results of randomness test and diffusion for BRIGHT 64/128, when there is a single-bit change in key (Plaintext (Hexadecimal) = 00 00 00 00 00 00 00 00 00 00)

S.No	**Key (Hexadecimal)**	**Ciphertext (Binary)**								**Number of**	
										Zero's	**One's**
1	00 04 02 03 08 29 2a 0b 10 11	10011111	01111001	01100100	10111101	00000100	00101111	00110011	00111011	29	35
2	01 04 02 03 08 29 2a 0b 10 11	10110111	00011110	01010011	00110100	01101011	00111010	10000101	10110000	32	32
3	02 04 02 03 08 29 2a 0b 10 11	00010001	00100001	11011000	11011001	11000011	10111011	11000100	10101011	33	31
4	03 04 02 03 08 29 2a 0b 10 11	10011011	00001100	00000010	01010100	01001100	10011011	11000011	10001000	39	25
5	04 04 02 03 08 29 2a 0b 10 11	01001001	01111001	00101110	11110011	11100011	11000000	11111001	10100010	30	34
6	05 04 02 03 08 29 2a 0b 10 11	10000000	00010101	01110011	00110010	00010101	00101010	11111011	01101110	34	30
7	06 04 02 03 08 29 2a 0b 10 11	10001001	00000001	10010111	00011000	00010000	01101010	11111110	01000110	39	25
8	07 04 02 03 08 29 2a 0b 10 11	00100101	11111000	11111100	10110101	10011100	00011010	10100001	01001111	34	34
9	**Number of bits changed**	**(1, 2) = 33 (1, 3) = 32 (1, 5) = 33 (2, 4) = 23 (3, 4) = 28 (4, 8) = 35 (5, 7) = 34 (7, 8) = 34 (6, 8) = 30 Average diffusion = 31.33 (48.96%)**								**33.75**	**30.25**

It also gives the average diffusion and diffusion percentage for BRIGHT 64/128. Similar results of Avalanche test are obtained for all versions of BRIGHT family when there is a single-bit change in plaintext which is summarized in Table 3. The diffusion range given in Table 3 shows that a similar amount of diffusion is not obtained for all cases. Achieving different amount of diffusion is significant and it shows the strength of the cipher's properties. A total of nine diffusion values are taken for each variant for which Table 3 gives the diffusion range and average diffusion. It is proved that all versions of BRIGHT family fulfill SAC criteria by achieving approximately 50% diffusion for all versions.

Tale 5. Summarizes the diffusion range and average diffusion between plaintext block and ciphertext block when there is a single-bit change in key.

Block Length	Key Length	Diffusion Range	Average diffusion	Diffusion Percentage
64	80	23-35	31.33 bits	48.96
64	96	29-40	31.89 bits	49.82
64	128	27-41	33 bits	51.56
128	128	59-68	64.56 bits	50.43
128	192	56-74	65.11 bits	50.87
128	256	54-74	63.22 bits	49.39

2. Key sensitivity: An algorithm is said to be key sensitive if retrieving original data is not possible when the key has even a minute difference from the original key. For this, Avalanche test is used to evaluate the amount of changes in the resulting ciphertext. Table 4 summarizes the results of key sensitivity for BRIGHT 64/128, when there is a single-bit change in key. It also gives the average diffusion and diffusion percentage for BRIGHT 64/128. Similar results of key sensitivity are obtained against all versions of BRIGHT family when there is a single-bit change in key which is summarized in Table 5. Diffusion range in Table 5 shows that different amount of diffusion is obtained and approximately there is a 50% average diffusion for all versions. Results of key sensitivity test shown in Table 5 are obtained for nine diffusion values for each variant of BRIGHT family for which it gives the diffusion range and average diffusion. It shows that all members of BRIGHT family fulfill key sensitive criteria.

3. Randomness Test: Randomness test is based on diffusion characteristics of the cipher. It is the ability of the cipher's round function to produce random output. The process of randomness test consists of two steps; the first step takes the sample sequence from the algorithm and in the second step analysis of a sample is done by performing statistical randomness tests. Potentially, a randomness test could be used as a distinguisher based on diffusion. If a cipher passes the randomness test, then there does not exist some form of input/output relation. Following criteria should meet to pass the randomness test for Ψ matrices which deals with the similarities of the diffusion instances, Ψ:

- The number of ones and zeros should be equal.
- A random distribution of ones and zeros.
- Ψ_i & Ψ_j should be dissimilar for $i \neq j$.

Tables 2 and 4 show the results of randomness test for BRIGHT 64/128 and gives an average number of one's and zero's. It is clearly seen from the tables that BRIGHT cipher has an almost equal number of ones and zeros which are randomly distributed and also for $i \neq j$, Ψ_i & Ψ_j are dissimilar. Tables 6 and 7 summarize the results of randomness test for all versions of BRIGHT family when there is a single-bit change in plaintext and key respectively. So BRIGHT family passes the randomness test as well.

Table 6. Average number of zero's and one's in ciphertext block when there is a single-bit change in plaintext

Block Length	Key Length	Average number of Zero's	Average number of One's
64	80	30.13	33.87
64	96	30.25	33.75
64	128	32.62	31.38
128	128	65.12	62.88
128	192	63.75	64.25
128	256	60.75	67.25

Table 7. Average number of zero's and one's in ciphertext block when there is a single-bit change in key

Block Length	Key Length	Average number of Zero's	Average number of One's
64	80	33.75	30.25
64	96	32.75	31.25
64	128	29.63	34.37
128	128	62.75	65.25
128	192	62.13	65.87
128	256	64.25	63.75

4. Execution Time: Amount of time an algorithm takes to encode and decode a particular data is known as execution time. For IoT, lower execution time is demanded and lower the execution time, better the algorithm is. Table 7 summarizes the execution time for each variant of BRIGHT family and a comparison of ARX based lightweight ciphers is made. All the algorithms compared in Table 8 and Fig. 2 are implemented on a 64-bit processor in C-language to compare their parameters on the same platform. This was done so that no artifacts can be made due to platform issues, either in results of BRIGHT cipher or in the comparisons with other ciphers. It is clearly

Table 8. Memory and execution time comparison of standard ARX-block ciphers with BRIGHT family implemented on a 64-bit platform

Cipher (Block size/ Key size)	Rounds	Memory (Bytes)				Cost (Cycles/ byte)	Speed (Mbytes/ sec)
		Encryption	Decryption	Key-schedule	Encryption+ Decryption+ Key-schedule		
BRIGHT (64/80)	**32**	**633**	**637**	**640**	**1804**	**1578**	**1.45**
RoadRunneR (64/80)	10	1642	1643	202	2334	1458	1.61
BRIGHT (64/96)	**33**	**634**	**638**	**640**	**1806**	**1505**	**1.52**
SPECK (64/96)	26	678	685	837	1802	2724	0.84
BRIGHT (64/128)	**34**	**635**	**638**	**640**	**1807**	**1589**	**1.44**
SPECK (64/128)	27	680	690	837	1820	2008	1.14
HIGHT (64/128)	32	1884	1864	1044	4280	2409	0.95
SPARX (64/128)	8	1501	1521	1453	3516	2630	0.87
BRIGHT (128/128)	**35**	**641**	**639**	**639**	**1819**	**1774**	**1.29**
SPARX (128/128)	8	3334	3353	1246	5059	5201	0.44
BRIGHT (128/192)	**36**	**642**	**640**	**639**	**1821**	**2062**	**1.11**
BRIGHT (128/256)	**37**	**643**	**641**	**639**	**1823**	**2179**	**1.05**

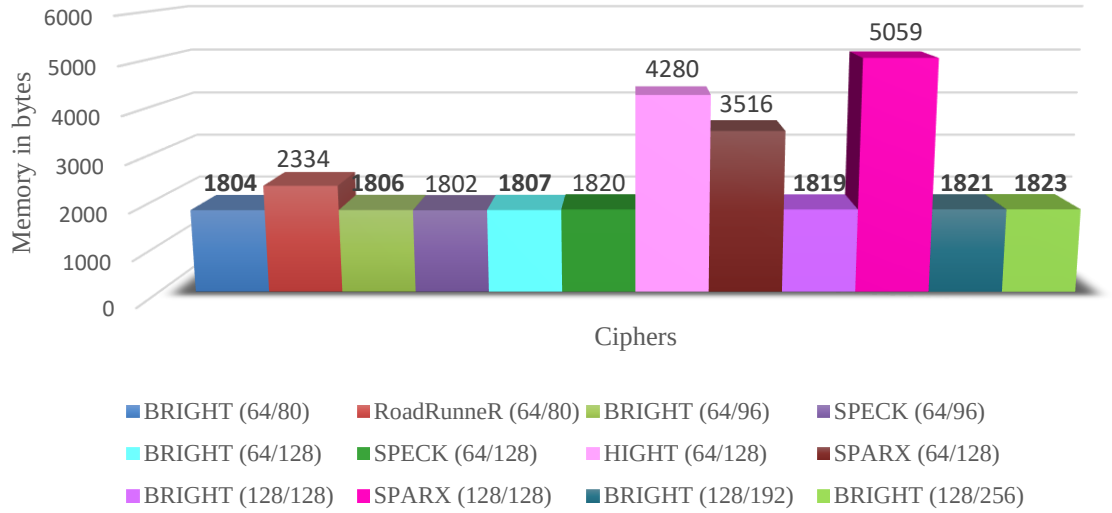

Fig. 2 represents the memory (Encryption + Decryption + Key-Scheduling) comparison of the existing lightweight ciphers with BRIGHT cipher

seen from Table 8 that all variants of the BRIGHT family member show better execution speed than other existing lightweight ARX ciphers except the RoadRunneR (64/80). RoadRunneR cipher uses on-the-fly key scheduling which results in fast speed. Also, the use of simple key scheduling in RoadRunneR may lead to weak key attacks. Further increase in the speed of BRIGHT cipher is possible but only at the cost of increased memory. So keeping this in mind a speed memory tradeoff is followed in the design of proposed BRIGHT cipher.

5. Memory Utilization: In IoT based devices, there are some constraints like low computation power, limited memory, limited power consumption and etc., In most of the IoT devices, there is a limited amount of memory available to the devices which is a major concern in resource constraint IoT applications. Proposed BRIGHT cipher is evaluated in terms of memory utilization. The BRIGHT family of ciphers consumes a smaller amount of memory as compared to other ciphers which is favorable for its deployment in IoT. Additional code size savings are possible but it lowers the throughput. Contrary to this, loop unrolling can be used to improve register usage and this speed up the process but at the cost of increased memory size. So an intermediate concept of loop unrolling can be used for balanced performance. It all depends on the need of a particular application. Table 8 summarizes the memory utilization of BRIGHT family ciphers and compare them with other existing ARX based lightweight block ciphers on the same platform (64-bit processor). Memory consumption of all variants of the BRIGHT

family ciphers, in terms of flash memory, is lowest except the variant with block size 64 and key size 96. SPECK (64/96) has lower memory consumption than BRIGHT (64/96). SPECK family has lowest flash memory but SPECK has no security proof and because of its simplest structure, there are a number of attacks which are successfully applied on SPECK. A few of these attacks are linear and differential attacks [30 - 32].

6. Conclusion

Design and implementation of a lightweight cipher go simultaneously and this has revealed some significant limits and inherent conditions. Designing a security algorithm for IoT enabled devices must consider the criteria provided by standard organizations from time to time. We have evaluated the performance of newly proposed BRIGHT cipher, a family of lightweight block ciphers with 6 instances, supporting block sizes of 64-bit and 128-bit on a 64-bit processor. BRIGHT ciphers fulfill Strict Avalanche Criteria, passes key sensitivity and randomness test. Results for execution time and memory utilization of BRIGHT family ciphers are better than existing ones. All the variants of BRIGHT cipher have a comparably lower cost. So, due to its fast execution speed and low memory utilization, BRIGHT cipher proves to be a better security algorithm than the existing benchmarked lightweight block ciphers. This paper set a base for further research work and in the near future, we will evaluate the performance of the proposed ciphers on different platforms (8-bit, 16-bit and 32-bit processors). This work helps the researchers in the area of IoT security. We invite researchers for the cryptanalysis of the newly proposed family of BRIGHT cipher.

References

[1] R. Baashirah, A. Kommareddy, S. K. Batchu, V. Sunku, R. S. Ginjupalli, and S. Abuzneid, "Security implementation using present-puffin protocol in RFID devices", In: *Proc. of 2018 IEEE Long Island Systems, Applications and Technology Conference*, pp. 1-5, 2018.

[2] 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 Journal on Wireless Communications and Networking,* No. 1, Article No. 117, 2018.

[3] B. J. Mohd, T. Hayajneh, K. M. Ahmad Yousef, Z. A. Khalaf, and M. Z. A. Bhuiyan, "Hardware design and modeling of lightweight block ciphers for secure communications", *Future Generation Computer Systems, Vol.* 83, pp. 510-521, 2018.

[4] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, and C. Manifavas, "A review of lightweight block ciphers", *Journal of Cryptographic Engineering*, Vol. 8, No. 2, pp. 141-184, 2018.

[5] H. Seo, Z. Liu, J. Choi, T. Park, and H. Kim, "Compact implementations of LEA block cipher for low-end microprocessors", In: *Proc. of International Workshop on Information Security Applications*, Springer, Cham, pp. 28-40, 2015.

[6] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee, "HIGHT: A new block cipher suitable for low-resource device", In: *Proc. of International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 46-59, 2006.

[7] R. Beaulieu, S. T. Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers, "The SIMON and SPECK lightweight block ciphers", In: *Proc. of Design Automation Conference (DAC), 52nd ACM/EDAC/IEEE*, pp. 1-6, 2015.

[8] N. Mouha, B. Mennink, A. V. Herrewege, D. Watanabe, B. Preneel, and I. Verbauwhede, "Chaskey: an efficient MAC algorithm for 32-bit microcontrollers", In: *Proc. of International Workshop on Selected Areas in Cryptography*, pp. 306-323, 2014.

[9] A. Baysal and S. Şahin, "Roadrunner: A small and fast bitslice block cipher for low cost 8-bit processors", In: *Proc. of International Workshop on Lightweight Cryptography for Security and Privacy*, pp. 58-76, 2015.

[10] D. Dinu, L. Perrin, A. Udovenko, V. Velichkov, J. Großschädl, and A. Biryukov, "Design strategies for ARX with provable bounds: Sparx and LAX", In: *Proc. of International Conference on the Theory and Application of Cryptology and Information Security*, pp. 484-513, 2016.

[11] D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, and D. Lee, "LEA: A 128-bit block cipher for fast encryption on common processors", In: *Proc. of International Workshop on Information Security Applications*, pp. 3-27, 2013.

[12] O. Özen, K. Varici, C. Tezcan, and Ç. Kocair, "Lightweight block ciphers revisited: Cryptanalysis of reduced round PRESENT and HIGHT", In: *Proc. of Australasian Conference on Information Security and Privacy*, pp. 90-107, 2009.

[13] L. Wen, M. Wang, A. Bogdanov, and H. Chen, "Multidimensional zero-correlation attacks on lightweight block cipher HIGHT: improved

cryptanalysis of an ISO standard", *Information Processing Letters, Vol.* 114, No. 6, pp. 322-330, 2014.

[14] S. Ahmadi, Z. Ahmadian, J. Mohajeri, and M. R. Aref, "Low-data complexity biclique cryptanalysis of block ciphers with application to piccolo and hight", *IEEE Transactions on Information Forensics and Security*, Vol. 9, No. 10, pp. 1641-1652, 2014.

[15] B. Koo, D. Hong, and D. Kwon, "Related-key attack on the full HIGHT", In: *Proc. of International Conference on Information Security and Cryptology*, pp. 49-67, 2010.

[16] L. Song, Z. Huang, and Q. Yang, "Automatic differential analysis of ARX block ciphers with application to SPECK and LEA", In: *Proc. of Australasian Conference on Information Security and Privacy*, pp. 379-394, 2016.

[17] G. Leurent, "Improved differential-linear cryptanalysis of 7-round Chaskey with partitioning", In: *Proc. of Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 344-371, 2016.

[18] Q. Yang, L. Hu, S. Sun, and L. Song, "Extension of meet-in-the-middle technique for truncated differential and its application to RoadRunneR", In: *Proc. of International Conference on Network and System Security*, pp. 398-411, 2016.

[19] M. Katagi, and S. Moriai, "Lightweight cryptography for the Internet of Things", *Sony Corp.*, pp. 7-10, 2008.

[20] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni, "Midori: A block cipher for low energy", In: *Proc. of International Conference on the Theory and Application of Cryptology and Information Security*, pp. 411-436, 2014.

[21] A. Biryukov, S. Mukhopadhyay, and P. Sarkar, "Improved time-memory trade-offs with multiple data", In: *International Workshop on Selected Areas in Cryptography*, pp. 110-127, 2005.

[22] N. Mouha, "The Design Space of Lightweight Cryptography", In: *NIST Lightweight Cryptography Workshop, Gaithersburg, United States*, 2015.

[23] M. Usman, I. Ahmed, M. I. Aslam, S. Khan, and U. A. Shah, "Sit: A lightweight encryption algorithm for secure internet of things", *International Journal of Advanced Computer Science and Applications*, Vol. 8, No. 1, pp. 402-411, 2017.

[24] K. Bhargavan, and G. Leurent, "On the practical (in-) security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN", In: *Proc. of 23rd ACM Conference on Computer and Communications Security*, pp. 456-467, 2016.

[25] F. X. Standaert, G. Piret, N. Gershenfeld, and J. J. Quisquater, "SEA: A scalable encryption algorithm for small embedded applications", In: *Proc. of International Conference on Smart Card Research and Advanced Applications*, pp. 222-236, 2006.

[26] D. D. Daniel, "Efficient and secure implementations of lightweight symmetric cryptographic primitives", *Doctoral dissertation, University of Luxembourg, Luxembourg*, 2017.

[27] C. Rajarathnam, S. Bapatla, K. P. Subbalakshmi, and R. N. Uma, "Battery power-aware encryption", *ACM Transactions on Information and System Security (TISSEC)*, Vol. 9, No.2, pp.162-180, 2006.

[28] H. Seo, I. Jeong, J. Lee, and W. H. Kim, "Compact Implementations of ARX-Based Block Ciphers on IoT Processors", *ACM Transactions on Embedded Computing Systems*, Vol. 17, No. 3, Article No. 60, 2018.

[29] G. Bansod, N. Pisharoty, and A. Patil, "PICO: An Ultra Lightweight and Low Power Encryption Design for Ubiquitous Computing", *Defence Science Journal*, Vol. 66, No. 3, pp. 259-265, 2016.

[30] A. Biryukov, A. Roy, and V. Velichkov, "Differential analysis of block ciphers Simon and Speck", *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, Vol. 8540, pp. 546–570, 2015.

[31] A. Biryukov, V. Velichkov, and Y. L. Corre, "Automatic search for the best trails in ARX: Application to block cipher SPECK", In: *Proc. of International Conference on Fast Software Encryption*, pp. 289-310, 2016.

[32] I. Dinur, "Improved Differential Cryptanalysis of Round-Reduced Speck", In: *Proc. of International Conference on Computational Intelligence and Security, CIS*, pp. 367–371, 2010.