BRIGHT: A Small and Fast Lightweight Block Cipher for 32-bit Processor

Deepti Sehrawat, Nasib Singh Gill

Abstract: ***Recently a number of lightweight ciphers are designed for improved hardware or software performance. Designing block ciphers for a resource-constrained 32-bit processor is even more challenging. Usually, for 32-bit CPU the lightweight ciphers are designed with low-security margins. This paper presents, an efficient family of LBC, named BRIGHT. The proposed design is a software-oriented security framework for resource-constrained IoT-enabled applications. It has lowest code size and fastest execution speed on 32-bit processor because of its 32-bit ARX operations. It enables users to match their security needs with application requirements by supporting a range of cryptographic solutions. The proposed design has 6 variants and all variants of BRIGHT family ciphers fulfills Strict Avalanche Criteria and key sensitivity test. BRIGHT family ciphers show better performance in terms of memory requirements, cost and speed. This is because of its simple and efficient internal structure. In this paper, a comparison is made among existing benchmarked lightweight ciphers and proposed design on 32-bit CPU. BRIGHT is competitive among existing lightweight ciphers.***

Index Terms: BRIGHT performance evaluation, lightweight block ciphers, performance enhancement, small and fast cipher.

I. INTRODUCTION

Nowadays, ubiquitous computing, IoT, and smart world are concepts becoming popular day by day. The communication in these is made possible by RFID enabled devices and most commonly 32-bit CPUs are deployed in these smart applications. Security and privacy is a major concern for these resource constraint devices [1]. An attacker can get access to the information shared between devices. For this reason, there is a need of good security solutions in the form of cryptographic algorithms. Furthermore, the participating nodes in IoT are constrained in terms of memory, power, and computational ability [2]. So, a new field of lightweight cryptography came into existence. Besides, block ciphers are considered as workhouses in cryptographic applications. Therefore, in the last 10 years, the design of LBC has attracted the attention of many researchers. Recently, a number of lightweight block ciphers are proposed by various researchers' for IoT enabled smart environment. Most of these algorithms are optimized for hardware implementations by using building blocks and are not considered a good choice for software-oriented applications on a 32-bit processor. 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.

Manuscript published on 30 June 2019.
***** Correspondence Author (s)

Deepti Sehrawat*, Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, Haryana (India)
Nasib Singh Gill, Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, Haryana (India).

Our Contribution

First, the analysis of various existing benchmarked designs is carried out to identify the strong points and flaws of each one which is culminated in the publication in [3, 4]. Then the design goals are formulated to ensure a fair comparison of the proposed design with other benchmarked designs. The framework of the proposed design is then described and the performance is evaluated on different platforms. In this paper, a software-oriented security framework for resource-constrained IoT-enabled applications is presented. The main focus is on reducing memory and achieving high speed without compromising security. BRIGHT design implements 32-bit ARX operations because of its enhanced performance on 32-bit CPU. BRIGHT uses pre key-whitening to thwart weak key attacks and round permutation for faster diffusion. Since the proposed design uses ARX structure, there is low decryption overhead, and decryption is performed by simply reversing the operations of the encryption process. The simple and efficient internal structure of BRIGHT is so designed that it provides fast diffusion, tiny-size code and high-speed. In this paper, the performance of BRIGHT is evaluated on a 32-bit processor.
The rest of the paper is structured as follows: section 2 presents related work relevant to lightweight cryptography. BRIGHT cipher is explained in section 3. Section 4 details performance evaluation of BRIGHT on different parameters.

II. RELATED WORK

Some lightweight block ciphers with ARX structure are TEA [5], XTEA [6], LEA [7], HIGHT [8], SIMON [9], SPECK [9], RoadRunneR [10] Chaskey [11], and SPARX [12]. TEA and XTEA use simple round function and key schedule. Both have lower block length and a larger number of rounds, their encryption speed is not as fast as that of LEA. LEA has three variants with a block size of 128-bits and key sizes 128, 192 and 256-bits long. It is based on 4-branch Generalized Feistel Network (GFN) and is optimized for 32-bit processor than a 64-bit processor [7]. There exists a boomerang attack targeting 15 rounds on this cipher. HIGHT cipher is optimized for 8-bit operations and there are a number of attacks implemented on HIGHT like multidimensional zero-correlation attack [13], biclique attack [14], and related-key attack [15].

SPECK and SIMON ciphers are family of lightweight ciphers having 10 instances of each. SIMON cipher with a block size of 128-bit is comparable with LEA. Whereas the performance of SPECK exceeds LEA'S performance in 64-bit processor because of SPECK's 64-bit addition in a 64-bit processor. CHASKEY ciphers is a permutation-based Message Authentication Code (MAC) lightweight block cipher. It does not follow any key schedule it simply consists of two shifts and two conditional XORs with the state for two sub-keys. A differential-linear attack targeting 7 out of 8 rounds is applied on Chaskey [11]. RoadRunneR is a block cipher with block size 64-bits and it supports two key sizes viz. 80-bits and 128-bits. The Feistel function of RoadRunneR is an SPN having four 4-bit S-box layers, 3 linear layers, and 3 key additions [10]. There exists a high-probability truncated trail in RoadRunneR-128 covering 5 out of 7 rounds. SPARX cipher has three variants Sparx-64/128, Sparx-128/128, and Sparx-128/256. Variant Sparx-64/128 has 8 steps and in each step there are 3 rounds, Sparx-128/128 has 8 steps with 4 rounds in each step and variant Sparx-128/256 has 10 steps with 4 rounds in each step [12]. There exists an impossible differential attack on Sparx [16].

III. PROPOSED BRIGHT CIPHER

A. Notation

Through the paper the following notations are used:

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

B. Design

BRIGHT cipher is a family of lightweight GFN (Generalized Feistel Network) block cipher. It has 4 branches/ words and supports 64-bit and 128-bit block sizes. BRIGHT cipher proposed six variants, 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. It implements ARX operations for faster diffusion and its operations are supported in multiple platforms in an efficient and parallel way. The encryption of BRIGHT is performed in layers. There are three layers in BRIGHT design, these are key whitening, ARX operations, and round permutation. Fig. 1 depicts the structure of the proposed cipher. To provide resistance against most of the attacks, first for all the variants of BRIGHT the minimum number of rounds '*R*', required to reach complete diffusion is found. 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 respectively.

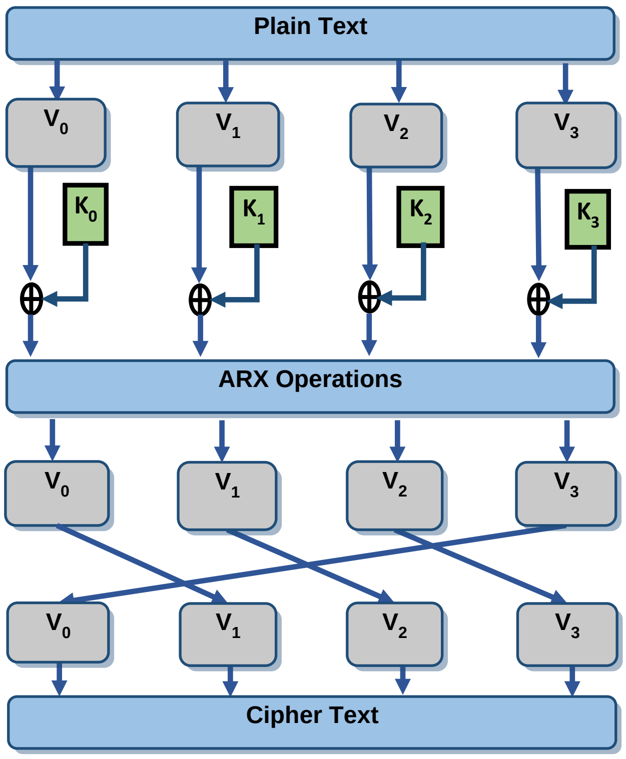


Figure 1: Layers in the BRIGHT family of ciphers

1. **Top Layer (Key Whitening):** To provide resistance against weak key attacks, the actual key is first XORed as a pre-whitening affect. Key whitening does not provide any immunity to analytical attacks like linear cryptanalysis and differential cryptanalysis but provides good resistance to attacks like brute force and MITM attacks. We have used only pre-whitening, post whitening does not add any security it just adds to the code length. Besides, 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.

2. **Middle Layer-ARX Operations:** ARX operations in software implementation provides an efficient parallel implementation. XOR function is mostly used in ARX-based ciphers. Most effective operations are modular arithmetic operations like addition-modulo and subtraction-modulo. Using addition modulo 2^n 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 [17]. Furthermore, addition modulo 2^n add non-linearity to the cipher. Addition modulo is preferred over XOR and when these are used in combination on an alternate basis then it provides faster diffusion. Many of the existing ciphers have used the same ARX concepts like HIGHT [8], SPECK [9], RoadRunneR [10], CHASKEY [11], and SPARX [12] with which we have compared our proposed design. Rotations are not considered good operations, especially rotations of the 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). So choosing carefully best rotation amounts is considered as a good decision in a cipher design which is followed in the design of BRIGHT cipher. So, BRIGHT cipher uses a different amount of rotations which are carefully chosen.

3. **Bottom Layer-Round Permutation:** Generally, GFN 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 after each round. The round permutation is performed after applying ARX operations on the ciphers. Round permutation provides faster diffusion.

Encryption is a composition of round functions i.e. $R_{kr-1} \circ R_{kr-2} \circ \ldots R_{k1} \circ R_{k0}$ that can be read from right to left. In decryption, the same round function is used but here the order of the operations viz. key whitening, round constants, round keys, and ARX operations are reversed. Subtraction modulo 2^n is used in place of addition modulo 2^n. All variants of 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. Table 2 gives the block size, key size, and the number of rounds for all versions of BRIGHT family ciphers.

Table 2: 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

C. Encryption Algorithm

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

OUTPUT: Block (n-bits long)

1) First partition the *n*-bits plain text 'P' into for branches/words (P_0, P_1, P_2, P_3) of equal length and perform key-whitening by XORing with four sub-keys i.e.
   $P = P_0 \oplus k_0 \mid P_1 \oplus k_1 \mid P_2 \oplus k_2 \mid P_3 \oplus k_3$.
2) Apply ARX operations on P_i for i^{th} round i.e.
   $P_{i1} = (P_{i1} <<< b) \oplus Rk_i$,
   $P_{i0} = ((P_{i0} \boxplus P_1) \& C) <<< a$,
   $P_{i3} = P_{i3} <<< b$,
   $P_{i2} = (P_{i2} \boxplus P_{i3}) \& C$,
   $P_{i3} = P_{i3} \oplus P_{i2}$,
   $P_{i0} = P_{i0} \oplus P_{i3}$,
   $P_{i3} = P_{i0} \oplus P_{i3}$,
   $P_{i2} = P_{i2} \oplus P_{i1}$,
   $P_{i1} = (P_{i1} <<< b) \oplus P_{i2}$,
   $P_{i2} = P_{i2} <<< a$
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.

D. Key Scheduling

There are two phases in a cipher design, first is data scheduling and second is 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. Furthermore, if weak key scheduling is used by a cipher then it is easy to find weak keys which ultimately makes the cipher non-resistant to different key scheduling attacks like zero correlation attack, MITM attack, weak key attack, and their variants. Besides, each round in GFN based ciphers require a separate key.

To provide resistance against exhaustive search attacks, the length of the master key must be sufficiently large enough which makes difficult for the attacker to perform 2^{k-1} encryptions for the key searching attacks. Also, longer key sizes require more number of rounds to facilitate the affect of every key bit on the ciphertext equally i.e. without measurable differences which would otherwise allow any cryptanalysis. Also, no attack should be faster than exhaustive key search i.e. brute force attack. Keeping this in view, the key scheduling of BRIGHT is inspired from SPECK key scheduling which is explained below:

A user-defined master key is stored in the register key, represented by (1):

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

First, from this master-key 4 sub-keys are retrieved for initial key whitening which are XORed with the four words of the plain text. Then the key scheduling generates '*n*' keys for *n*-rounds to further introduce confusion-diffusion.

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 € $GF(4)^n$, for a value of m in {2,3,4}. Sequences k_i and l_i are defined by (2) and (3):

$$L_{i+m-1} = (k_i + l_i >>> 2)\ \hat{}\ i$$
(2)

$$K_{i+1} = (k_i <<< 5)\ \hat{}\ l_{i+m-1}$$
(3)

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

IV. 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. There is a need for flexible and secure block cipher which is able to perform well on all resource constraint platforms. Earlier, the performance of BRIGHT was evaluated on a 64-bit processor and results show that all members of the BRIGHT family have better performance on a 64-bit processor [18].

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)
© Copyright: All rights reserved.

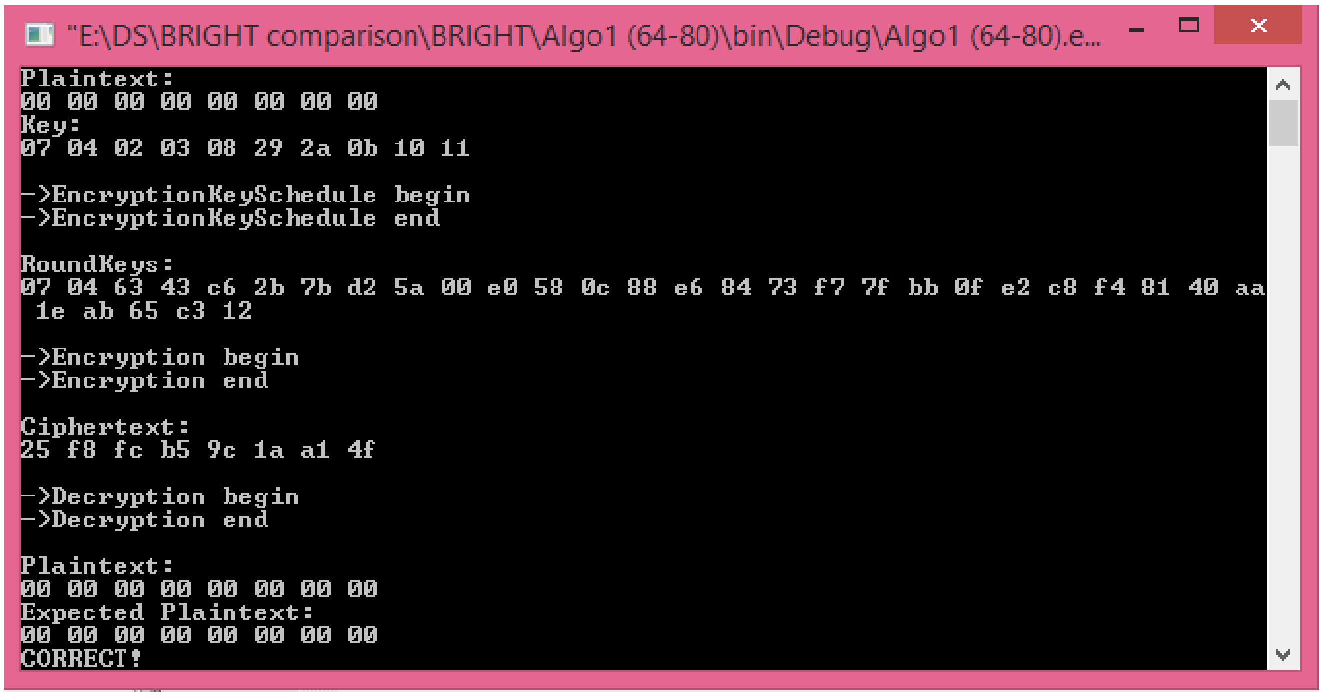
Besides, most commonly 32-bit CPUs are deployed in IoT based smart applications. Hence, lightweight block cipher should support sound performance on 32-bit processors. In this paper performance of BRIGHT is evaluated on a 32-bit CPU.

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.

Implementation results of the proposed BRIGHT family of lightweight ciphers on a processor Intel (R) Core (TM) 2 Duo E4500 @ 2.20 GHz are observed. The results of the proposed BRIGHT family of ciphers and some other benchmarked ciphers with which comparison is made are obtained on a 32-bit processor in C-language to compare their parameters on the same platform so that no artifacts can be made regarding the implementation platform. Following criteria are taken for the performance evaluation of the proposed BRIGHT family of ciphers:

A. Strict Avalanche Criteria (SAC)

According to SAC, if input bit (key/ plaintext) is changed by a single bit then there should be 50% change in the ciphertext bits. Cipher fulfilling SAC has a higher probability to thwart all possible attacks [19]. Contrary, a cipher is not considered good if Avalanche criteria is not satisfied. Fig. 2 represents the encryption and decryption process of BRIGHT 64/80. Table 2 to Table 7 summarizes the results of diffusion when there is a single-bit change in plaintext for all variants of BRIGHT. Whenever a single bit is changed in the plaintext of BRIGHT family ciphers it brought approximately half of the bits changed in the ciphertext. It is found that all variants of BRIGHT family fulfill SAC criteria by achieving approximately 50% diffusion.


"E:\DS\BRIGHT comparison\BRIGHT\Algo1 (64-80)\bin\Debug\Algo1 (64-80).e...

```
Plaintext:
00 00 00 00 00 00 00 00
Key:
07 04 02 03 08 29 2a 0b 10 11

->EncryptionKeySchedule begin
->EncryptionKeySchedule end

RoundKeys:
07 04 63 43 c6 2b 7b d2 5a 00 e0 58 0c 88 e6 84 73 f7 7f bb 0f e2 c8 f4 81 40 aa
 1e ab 65 c3 12

->Encryption begin
->Encryption end

Ciphertext:
25 f8 fc b5 9c 1a a1 4f

->Decryption begin
->Decryption end

Plaintext:
00 00 00 00 00 00 00 00
Expected Plaintext:
00 00 00 00 00 00 00 00
CORRECT!
```

Figure 2: Represent encryption and decryption of BRIGHT 64/80.

Table 2: Diffusion for BRIGHT 64/80, when there is a single-bit change in plaintext.

BRIGHT 64/80, Key (Hexadecimal)= 07 04 02 03 08 29 2a 0b 10 11		
Plain Text (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
00 00 00 00 00 00 00 00	25 f8 fc b5 9c 1a a1 4f	34
00 00 00 00 00 00 00 01	9d 9d 5c 32 43 e6 30 97	

Table 3: Diffusion for BRIGHT 64/96, when there is a single-bit change in plaintext.

BRIGHT 64/96, Key (Hexadecimal)= 00 04 02 03 08 29 2a 0b 10 11 4f ae		
Plain Text (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
00 00 00 00 00 00 00 00	e7 8c 6f 7a a1 e7 bd 04	31
00 00 00 00 00 00 00 00	aa 39 d2 f0 79 f9 50 4d	

Table 4: Diffusion for BRIGHT 64/128, when there is a single-bit change in plaintext.

BRIGHT 64/128, Key (Hexadecimal)= 07 04 02 03 08 29 2a 0b 10 11 57 09 56 0f 29 56		
Plain Text (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
00 00 00 00 00 00 00 00	93 cc 77 6c e0 e1 67 32	35
00 00 00 00 00 00 00 00	54 74 cf 9b 40 94 2e 87	

Table 5: Diffusion for BRIGHT 128/128, when there is a single-bit change in plaintext.

BRIGHT 128/128, Key (Hexadecimal)= 07 04 02 03 08 29 2a 0b 10 11 12 13 f0 1e 45 c3		
Plain Text (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	cf 63 48 00 ca 83 93 91 78 fa df 23 27 9d 63 7f	62
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	06 86 5a ff 1c 9a 00 00 c6 c4 00 00 1d 00 00	

Table 6: Diffusion for BRIGHT 128/192, when there is a single-bit change in plaintext.

BRIGHT 128/192, Key (Hexadecimal)= 07 04 02 03 08 29 2a 0b 10 11 12 13 f0 1e 45 c3 1a 52 55 ad 62 3c b4 7b		
Plain Text (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	be 75 9a a0 bc ef ff 13 20 8b 0c 09 a8 56 4c 49	66
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01	e3 35 11 27 56 c4 cc a7 99 b7 48 c2 4e d9 5a a4	

Table 7: Diffusion for BRIGHT 128/256, when there is a single-bit change in plaintext.

BRIGHT 128/256, Key (Hexadecimal)= 07 04 02 03 08 29 2a 0b 10 11 12 13 f0 1e 45 c3 1a 52 55 ad 23 7a dd 0c 18 83 ad ca 62 3c b4 7b		
Plain Text (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	7c 1e 11 07 d9 14 0a 30 22 08 1f 7e 9e 7d b3 79	72
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01	7c 11 5a 31 b8 4b 8a 7a 78 4c b3 0e 1c ad b8 5b	

B. 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 find the extent of changes in the resulting ciphertext. Table 8 to Table 13 summarizes the results of key sensitivity when there is a single-bit change in the key for all variants of BRIGHT. In these tables, the values of plaintext, ciphertext, and key are given in hexadecimal. To find the extent of change in ciphertext first these values are converted to binary values and then amount of changes is evaluated by finding the number of bits changed in ciphertext whenever there is a single bit change in key. All variants of BRIGHT family ciphers fulfill SAC by changing approximately 50% of ciphertext bits whenever there is a single bit change in the key. Hence, it is proved that all members of the BRIGHT family fulfill key sensitive criteria.

Table 8: Diffusion for BRIGHT 64/80, when there is a single-bit change in key.

BRIGHT 64/80, Plaintext (Hexadecimal)= 00 00 00 00 00 00 00 00		
Key (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
07 04 02 03 08 29 2a 0b 10 11	25 f8 fc b5 9c 1a a1 4f	34
06 04 02 03 08 29 2a 0b 10 11	89 01 97 18 10 6a fe 46	

Table 9: Diffusion for BRIGHT 64/96, when there is a single-bit change in key.

BRIGHT 64/96, Plaintext (Hexadecimal)= 00 00 00 00 00 00 00 00		
Key (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
00 04 02 03 08 29 2a 0b 10 11 4f ae	e7 8c 6f 7a a1 e7 bd 04	32
01 04 02 03 08 29 2a 0b 10 11 4f ae	56 63 05 9b 34 7b 8f 60	

Table 10: Diffusion for BRIGHT 64/128, when there is a single-bit change in key.

BRIGHT 64/128, Plaintext (Hexadecimal)=00 00 00 00 00 00 00 00		
Key (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
00 04 02 03 08 29 2a 0b 10 11 57 09 56 0f 29 56	94 df 47 cd 6a 36 c9 fe	34
01 04 02 03 08 29 2a 0b 10 11 57 09 56 0f 29 56	78 e6 0a 57 91 90 45 cf	

Table 11: Diffusion for BRIGHT 128/128, when there is a single-bit change in key.

BRIGHT 128/128, Plaintext (Hexadecimal)= 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
Key (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
06 04 02 03 08 29 2a 0b 10 11 12 13 f0 1e 45 c3	4b 51 7e 17 87 8f 4e 4b 9e eb 68 48 3d 80 3f e1	64
07 04 02 03 08 29 2a 0b 10 11 12 13 f0 1e 45 c3	cf 63 48 00 ca 83 93 91 78 fa df 23 27 9d 63 7f	

Table 12: Diffusion for BRIGHT 128/192, when there is a single-bit change in key.

BRIGHT 128/192, Plaintext (Hexadecimal)= 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
Key (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
00 04 02 03 08 29 2a 0b 10 11 12 13 f0 1e 45 c3 1a 52 55 ad 62 3c b4 7b	ac 7a ec 0f 9f ea ad 5d ed 50 f2 61 f3 2b 6b e5	63
01 04 02 03 08 29 2a 0b 10 11 12 13 f0 1e 45 c3 1a 52 55 ad 62 3c b4 7b	19 b3 29 33 d9 25 30 ff a5 2f a0 2d d6 cd d6 e5	

Table 13: Diffusion for BRIGHT 128/256, when there is a single-bit change in key.

BRIGHT 128/256, **Plaintext (Hexadecimal)=** 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
Key (Hexadecimal)	**Cipher Text (Hexadecimal)**	**Number of bits changed**
06 04 02 03 08 29 2a 0b 10 11 12 13 f0 1e 45 c3 1a 52 55 ad 23 7a dd 0c 18 83 ad ca 62 3c b4 7b	11 6a a5 65 72 e5 96 28 18 f5 4f f5 88 11 52 b2	65
07 04 02 03 08 29 2a 0b 10 11 12 13 f0 1e 45 c3 1a 52 55 ad 23 7a dd 0c 18 83 ad ca 62 3c b4 7b	7c 1e 11 07 d9 14 0a 30 22 08 1f 7e 9e 7d b3 79	

C. Execution Time

Amount of time an algorithm takes for encryption and decryption of a particular data is known as execution time. Execution time is reflected from the speed of the cipher. For IoT based resource-constrained devices, lower execution time is demanded and that algorithm is preferred over others which have lower execution time. Table 14 summarizes the execution time for each variant of BRIGHT family ciphers. Besides, a comparison with some benchmarked ARX based lightweight ciphers is also made with the proposed family of ciphers. All the algorithms compared in table 14 are implemented on a 32-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 seen from the table that all variants of the BRIGHT family members show better execution speed than other existing lightweight ARX ciphers. So, BRIGHT family ciphers are fastest on 32-bit CPU than existing ARX based lightweight block ciphers.

Besides, cost is also considered an important parameter when considering a cipher for resource constrained IoT applications. Cost is given in cycles/byte. Table 14 shows the cost of all the ciphers mentioned in this table on a 32-bit CPU. It is clear from the table that all 6 members of BRIGHT family ciphers have lower cost than their competitors. So proposed cipher is considered better over others.

D. 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. For all the algorithms listed in table 14, memory utilization for encryption, decryption, and key-schedule is given. Besides, total amount of memory consumption by an algorithm is also given in table 14. 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 14 summarizes the memory utilization of BRIGHT family ciphers and compare them with other existing ARX based benchmarked lightweight block ciphers on the same platform (32-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 flash memory than BRIGHT 64/96 due to its simple internal structure but this has dropped the speed of the speck to a great amount and as a result, the cost is increased. Besides, there are a number of attacks which are successfully applied on SPECK. A few of these attacks are linear and differential attacks [20-22]. So considering these points, Speck cipher cannot be considered better than the proposed BRIGHT cipher.

Table 14: Memory and execution time comparison of standard ARX-block ciphers with BRIGHT family implemented on a 32-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**	**123**	**17.10**
RoadRunneR (64/80)	10	1642	1643	202	2334	202	10.37
BRIGHT (64/96)	**33**	**634**	**638**	**640**	**1806**	**166**	**12.65**
SPECK (64/96)	26	678	685	837	1802	519	4.04
BRIGHT (64/128)	**34**	**635**	**638**	**640**	**1807**	**206**	**10.22**
SPECK (64/128)	27	680	690	837	1820	328	6.40
HIGHT (64/128)	32	1884	1864	1044	4280	350	6.00
SPARX (64/128)	8	1501	1521	1453	3516	309	6.78
CHASKEY (128/128)	8	869	925	188	1982	290	7.23
BRIGHT (128/128)	**35**	**641**	**639**	**639**	**1819**	**251**	**8.36**
SPARX (128/128)	8	3334	3353	1246	5059	464	4.52
BRIGHT (128/192)	**36**	**642**	**640**	**639**	**1821**	**257**	**8.15**
BRIGHT (128/256)	**37**	**643**	**641**	**639**	**1823**	**298**	**7.03**

V. CONCLUSION

In a lightweight block cipher, both the design part and implementation part go simultaneously which has revealed some significant limits and inherent conditions. Designing a LBC in a resource constraint environment especially for a 32-bit processor is even more challenging. In this paper, a new family of LBCs named BRIGHT with 6 instances, supporting block sizes of 64-bit and 128-bit on a 32-bit processor is presented. We have evaluated the performance of BRIGHT, on a 32-bit processor. BRIGHT ciphers not only fulfill Strict Avalanche Criteria and key sensitivity test, the results for execution time and memory utilization of BRIGHT family ciphers are better than existing benchmarked ciphers. All the variants of BRIGHT cipher have a comparably lower cost, tiny code size and fastest execution speed on a 32-bit processor. BRIGHT is competitive among existing lightweight ciphers on 32-bit CPU. This paper set a base for further research work and in the near future, the performance of the proposed ciphers will be evaluated on different platforms (8-bit, 16-bit and 64-bit processors). This work helps the researchers in the area of IoT security. Besides, we invite researchers for the cryptanalysis of BRIGHT family ciphers.

REFERENCES

1. M. Dabbagh, and A. Rayes, "Internet of things security and privacy." In *Internet of Things from Hype to Reality*, Springer, Cham, 2019, pp. 211-238.
2. P. P. Ray, "A survey on Internet of Things architectures." *Journal of King Saud University-Computer and Information Sciences* 30, no. 3, 2018, pp. 291-319.
3. D. Sehrawat, and N. S. Gill. "Lightweight Block Ciphers for IoT based applications: A Review." *International Journal of Applied Engineering Research* 13, no. 5, 2018, pp. 2258-2270.
4. D. Sehrawat, and N. S. Gill., and M. Devi. "Comparative Analysis of Lightweight Block Ciphers in IoT-Enabled Smart Environment." In: *Proc. 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, IEEE, 2019, pp. 915-920. https://doi.org/10.1109/SPIN.2019.8711697
5. D. J. Wheeler, and Roger M. Needham. "TEA, a tiny encryption algorithm." In: *Proc. International Workshop on Fast Software Encryption*, Springer, Berlin, Heidelberg, 1994, pp. 363-366.
6. R. M. Needham, and David J. Wheeler. "Tea extensions." *Report, Cambridge University, Cambridge, UK,* 1997.
7. D. Hong, Deukjo, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu, and Dong-Geon Lee. "LEA: A 128-bit block cipher for fast encryption on common processors." In: *Proc. International Workshop on Information Security Applications*, Springer, Cham, 2013, pp. 3-27. DOI: https://doi.org/10.1007/978-3-319-05149-9_1
8. 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*, Springer, Cham, 2013, pp. 3-27. https://doi.org/10.1007/978-3-319-05149-9_1
9. R. Beaulieu, S. T. Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers, "The SIMON and SPECK lightweight block ciphers." In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, IEEE, 2015, pp. 1-6. DOI: 10.1145/2744769.2747946
10. A. Baysal, and S. Şahin. "Roadrunner: A small and fast bitslice block cipher for low cost 8-bit processors." In *Lightweight Cryptography for Security and Privacy*, Springer, Cham, 2015, pp. 58-76. DOI: https://doi.org/10.1007/978-3-319-29078-2_4
11. 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, Springer,* 2014, pp. 306-323. https://doi.org/10.1007/978-3-319-13051-4_19
12. 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, Springer, Berlin, Heidelberg,* 2016, pp. 484-513. DOI: https://doi.org/10.1007/978-3-662-53887-6_18
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, 2014, pp. 322-330. https://doi.org/10.1016/j.ipl.2014.01.007
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, 2014, pp. 1641-1652. DOI: 10.1109/TIFS.2014.2344445
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*, Springer, Berlin, Heidelberg, 2010, pp. 49-67. https://doi.org/10.1007/978-3-642-24209-0_4
16. A. Abdelkhalek, M. Tolba, and A.M. Youssef, "Impossible differential attack on reduced round sparx-64/128", In Proc. *International Conference on Cryptology in Africa*, Springer, Cham, 2017, pp. 135-146.

17. F. X. Standaert, G. Piret, N. Gershenfeld, J. J. Quisquater, "SEA: A scalable encryption algorithm for small embedded applications," In: *Proc. International Conference on Smart Card Research and Advanced Applications, Springer*, Berlin, Heidelberg, 2006, pp. 222-236. https://doi.org/10.1007/11733447_16
18. D. Sehrawat and N. S. Gill, "Performance Evaluation of Newly Proposed Lightweight Cipher, BRIGHT," *International Journal of Intelligent Engineering and Systems*, Vol. 12, No. 4, 2019, pp. 71-81. DOI: 10.22266/ijies2019.0831.08
19. 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, 2016, pp. 259-265. https://doi.org/10.14429/dsj.66.9276
20. 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, 2015, pp. 546–570. https://doi.org/10.1007/978-3-662-46706-0_28
21. 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*, Springer, Berlin, Heidelberg, 2016, pp. 289-310. https://doi.org/10.1007/978-3-662-52993-5_15
22. I. Dinur, "Improved Differential Cryptanalysis of Round-Reduced Speck", In: *Proc. of International Conference on Computational Intelligence and Security, CIS*, 2010, pp. 367–371. https://doi.org/10.1007/978-3-319-13051-4_9

AUTHORS PROFILE

Ms. Deepti Sehrawat has passed Master of Science and Applications from Department of Computer Science & Applications, M. D. University, Rohtak, India in 2008 and Master of Philosophy from C. D. L. University, Sirsa in 2009. She is currently pursuing Ph. D under the supervision of renowned academician and researcher – Professor Nasib Singh Gill of M. D. University. She has published more than 25 research papers in reputed National and International Journals and Conference Proceedings including IEEE. Her main research work focuses on IoT, Lightweight Cryptography Algorithms, Network Security and Privacy, Big Data Analytics and Data Mining. She has 8 years of teaching experience.

Dr. Nasib Singh Gill is at present senior most Professor of Department of Computer Science & Applications, M. D. University, Rohtak, India and is working in the Department since 1990. He has earned his Doctorate in Computer Science in the year 1996 and carried out his Post-Doctoral research at Brunel University, West London during 2001-2002. He is a recipient of Commonwealth Fellowship Award of British Government for the Year 2001. Besides, he also has earned his MBA degree. He has published more than 245 research papers in reputed National & International Journals, Conference Proceedings, Bulletins, Edited Books, and Newspapers. He has authored seven books. He is a Senior Member of IACSIT as well as a fellow of several professional bodies including IETE and CSI. He has been serving as Editorial Board Member, Guest Editor, Reviewer of International/National Journals and a Member of Technical Committee of several International/National Conferences. He has guided so far 8 Ph.D. scholars as well as guiding about 7 more scholars presently in the areas – IoT, Information and Network Security, Computer Networks, Measurement of Component-based Systems, Complexity of Software Systems, Decision Trees, Component-based Testing, Data mining & Data warehousing, and NLP.