J. Inst. Eng. India Ser. B
https://doi.org/10.1007/s40031-024-01067-2

ORIGINAL CONTRIBUTION

# Design, Implementation and Analysis of HIBRI Cipher on IoT Platforms

**H. R. Nagesh**[1] · **Asmita Poojari**[2] · **V. G. Kiran Kumar**[3]

Received: 24 February 2023 / Accepted: 15 April 2024


**Abstract** With the exponential growth of the Internet of Things through time there has been an enormous increase in the usage of tiny devices and information being exchanged, between low resource devices like sensors, PDA's and the high-end devices like servers or clouds. This opens up several challenges in securing the information being exchanged. Consequently, there is a greater demand for designing encryption algorithms, particularly for devices with limited resources. This paper introduces the HIBRI cipher, a combination of the HIGHT and the BRIGHT cipher implemented in software as well as hardware. The proposed cipher is evaluated for randomness test, strict avalanche criteria, key sensitivity tests to assess the cipher's security while memory utilization, energy consumption, execution time and throughput are analyzed by implementing in Arduino Nano based on the Atmega328 microcontroller. The NIST tests for key generation yielded $P \geq 0.01$ which proves that the proposed key generation scheme generated truly random numbers while the hardware implementation shows that the comparison results for code size, RAM and execution size yielded better results than the implementations that are cutting-edge.

**Keywords** Lightweight cryptography · BRIGHT · HIGHT · NIST tests · Avalanche Criteria

✉ Asmita Poojari
asmitapoojari@nitte.edu.in

H. R. Nagesh
nageshhrcs@rediffmail.com

V. G. Kiran Kumar
kiranvgk@gmail.com

[1] Canara Engineering College, Benjanapadavu, Bantwal Taluk, Mangaluru, Karnataka 574219, India

[2] NITTE (Deemed to Be University), Department of Computer Science and Engineering, NMAM Institute of Technology, Nitte, Karnataka 574110, India

[3] Department of Electronics and Communication Engineering, A J Institute Engineering and Technology, Mangaluru, Karnataka 575006, India

## Introduction

With the IoT (Internet of Things) gaining significant popularity, wherein billions of heterogeneous devices communicating in tandem, with constantly exchanging wide range of information from benign to highly sensitive one, between the LRDs (Devices with low resources) [10] like the sensors, RFID etc. and the high performing devices like the supercomputers or the server or cloud platforms. The devices that have low performance having limitations in computational power, size, area and storage by Juels and Weis [17]. With an exponential growth in IoT, the security of the information is also increasing exponentially. Thus, the need for implementations of block ciphers that can encrypt faster with limited code size and memory for target applications is increasing. The need for lightweight cryptography is becoming more and more significant. More security mechanisms need to be introduced to the software and hardware of the device so that the IoT device security can be improved and guarantee the preservation of confidential data [25]. However, implementing the conventional cryptographic algorithms in IoT devices is ineffective due to memory, processing speed and energy usage limitations. use of ciphers in resource limited devices is not convenient [33–36]. Therefore, the NIST came up with the idea of lightweight block ciphers for secure implementation. Cryptographic algorithms can be classified as either symmetric or asymmetric. Further, the symmetric key ciphers can be block ciphers and stream ciphers [40]. When

Published online: 11 May 2024

compared with stream ciphers, block ciphers are more effective as seen from the survey Chaitra et al. [9]. Furthermore, block ciphers are categorized based on structure as Feistel structures and substitution permutation network. Since Feistel structures only process half of the data every round, they often need more rounds than SPNs.

The same Feistel structure is used for ciphering and deciphering so as to save memory. As a result, compared to SPN implementations, the Feistel structure's decryption function does not require expensive implementations, which reduces the amount of hardware, memory, and power needed for computations.

This paper presents a novel light-weight optimised security algorithm that has been developed by combining HIGHT and the BRIGHT ciphers is implemented in ATmega328 (Arduino Nano 3.0). The proposed algorithm uses the structure of HIGHT cipher Kim et al. [7] and uses the ARX operations of BRIGHT Family of ciphers as in Seo et al. [31]. The performance and the efficacy of the proposed cipher is also evaluated using various tests to demonstrate the algorithm's security strength such as randomness test, the key-sensitivity test, the strict-avalanche criteria and correlation coefficient test. To evaluate the effectiveness based on key, the attributes such as memory utilization, energy consumption, execution time and throughput Asmita et al. [29], the proposed key generation is implemented using of the concept of LFSR and Fibonacci scrambling algorithm to provide better confusion and diffusion when compared with other existing algorithms. The key generation scheme is tested for Randomness using the NIST statistical test suites developed by NIST (National Institute of Science and Technology) Bassham et al. [3, 4].

## Background

Kerry et al. [27] published a report on lightweight cryptography project, The report emphasizes the need for lightweight encryption algorithms for resource-constrained devices. Conventional ciphers are unsuitable due to their resource-intensive nature, raising concerns about algorithm performance. NIST has launched the Lightweight Cipher (LWC) program to address these challenges, develop strategies for implementing lightweight ciphers, and standardize the project. The LWC project aims to ensure encryption algorithms are compatible with the evolving technology landscape, especially in resource-constrained devices. A detailed presentation about the target devices for the algorithm design and the performance metrics for both hardware and software are discussed in this paper. This paper also mentions about the desired properties required by the NIST to evaluate the ciphers design. Also, the NIST approved cryptographic primitives, design considerations and evaluation parameters are mentioned. Recently held "FELICS Triathlon," a lightweight block cipher competition for lightweight devices like IoT environments, the HIGHT encryption scheme cipher implementation won the most effective block cipher for lightweight devices [23]. The competition measured three parameters: memory (RAM), code-size (ROM) and the time for execution. Some of the features from 64/128-bit HIGHT encryption algorithm that utilizes less power and memory by Hong et al. [15], are incorporated into the proposed algorithm where fiestel structure with 32 rounds is used. HIGHT makes use of simple ARX operations. Beaulieu et al. [5] developed two highly optimized family of lightweight block named SIMON and SPECK suitable for both hardware and software. Both the cipher family follow Feistel structure. SIMON was optimized for hardware implementation. The SIMON is a 2n-block long cipher using Feistel structure. The most effective published attacks on Simon are those that use differential cryptanalysis. An ARX-design is used in the SPECK. It employs XOR and rotations of linear mixing in addition to having a modular addition to the nonlinearity. The proposed SIMON/SPECK 128/128 [24] is implemented in Atmel 8-bit microcontroller has a better throughput and needs just 388 bytes of flash memory and 256 bytes of SRAM while the compared implementations like AES-128 which needs 125 cycles/byte, and uses 1912 bytes of flash and 432 bytes of SRAM. In the paper Kim et al. [18], the HIGHT and PRESENT block cipher's optimal hardware and software implementations for resource constrained IoT platforms are evaluated. The paper illustrates an implementation on hardware of the HIGHT block cipher, a special type of bit- and digit-serial hardware that functions in units of a few or one bit. The computation of a delta variable in a key scheduling algorithm is the intensive unit, as it requires sequence of bitwise operations thus the implementation in software or bytewise machine is inefficient. Bitwise operations can be easily replaced in the Look-Up Table (LUT) for high-speed implementation. There was a reduction in code size by 51% and improvement in time for execution by 1 cycle/ byte and thus the performance improved by 23 and 18% respectively by utilizing the general-purpose registers to handle the data efficiently. Kwon et al. [21] implemented compact PRESENT encryption algorithm on embedded processors, The ECB(ELECTRONIC_CODE_BOOK) and CTR(COUNTER) is supported by the proposed PPRESENT Algorithm[16]. Seo et al. [31] implemented ARX based LEA and HIGHT cipher on AVR (8-bit), MSP (16-bit), ARM-NEON (32-bit) and ARM (32-bit) processors. Optimising the 32-/8-bitwise ARX-operations for the block-ciphers like HIGHT and LEA involved adjusting parameters including differences in the size of plaintext, the numberof general-purpose registers, and the target IoT device's instruction set. Eisenbarth et al. [12] HIGHT implementation illustrated smaller code sizes than the proposed balanced

and speed-optimized implementations by 69.4 and 80.5%, respectively, but execution times where 91.4 and 93.4% slower than the proposed algorithms. Guo et al. [39] have developed ECLBC, a lightweight block cipher that includes error detection and correction mechanisms. It ensures security by achieving a certain level of security in fewer rounds than conventional methods. ECLBC facilitates mode transitions within AND-Rotation-XOR (AND-RX) lightweight block ciphers, transitioning from Feistel to Substitution-Permutation Network (SPN) and half-round key XOR to full-round key XOR. It also detects and corrects erroneous ciphertext from channel interference. The researchers applied linear block code detection and correction mechanisms to IoMT devices. VAYU, a lightweight cipher presented by Bansod [2], uses 80/128-bit key to encrypt 64-bit plaintext in 31 rounds. The cipher is implemented in ARM 7 LPC2129 processor the proposed cipher requires lesser memory, GE compared to PRESENT, CLEFIA, AES and LED cipher.

Ledda et al. [22] presented an improved iteration of the IDEA algorithm, which utilizes the circular shift method along with middle square method. IDEA algorithm- a block cipher which is symmetric designed by Massey et al. [26]. The algorithm uses simple arithmetic functions such as shift, modulo-add and modulo-multiplication, thus the use of S-BOX's and P-BOX's are minimized. This algorithm's expansion process was improved by increasing the rounds, and bit shifting modifications were made while encryption as well as decryption. According to the experimental analysis, the communication took 28.83 ms to encrypt. By achieving 50.53% of the Avalanche effect, the encryption criteria were met [8]. This method has been improved, with the number of rounds increased and the crucial expansion procedure improved. The randomness tests like the frequency (monobit) frequency test and runs tests performed yielded 0.789, 0.997462, and 0.897437 respectively thus passing the randomness tests (significance level $P \geq 0.01$) and thus strengthening security**.** Kiran Kumar et al. [20] presented a novel lightweight block cipher named BRISI, an ARX based operation. The cipher structure is developed by combining SIMON and BRIGHT algorithms. The algorithm uses 64-bit key to encrypt 32-bit plaintext. The key scheduling for this algorithm is inspired from the SIT algorithm implemented by Usman et al. [37]. The algorithm is found to fulfil various tests for software implementations like the Key-sensitivity test, Avalanche criterion, entropy test, correlation coefficient and histogram which portrays that the algorithms efficacy. Usman et al. [37] presented an encryption algorithm SIT (Secure IoT) for the Internet of Things which 64-bit plaintext is encrypted and takes 64-bit key. The algorithm has a hybrid approach and is based on a Fiestel structure and consists of a uniform SPN (Substitution-permutation network. The SIT algorithm consists of five rounds aimed at improving security and energy efficiency. The algorithm uses a unique key generation scheme using complex mathematical operations. Four-bit segments make up the 64-bit key and an F-function is performed on the permuted, concatenated 16-bit and then XOR and shift operation are used to obtain five round keys. The encryption process consists of shift, swap and substitution operations so as to create diffusion and confusion along with a complex F-function. For security analysis all the basic attacks are performed on the proposed algorithm which reveals that the algorithm is better secured. The proposed algorithm was implemented using MATLAB on Intel Core i7-3770@3.40 GHzprocessor. The Avalanche test revealed 49% alteration in encrypted bits when one-bit plaintext or key was changed and image entropy tests revealed that the entropy was nearer to 8 and correlation close to zero resulting in better security. The Hardware implementation of the proposed algorithm on ATmega 328 based Ardinuo-Uniboard showed the time of execution for encryption was 0.188 ms and decryption time was 0.187 ms also SIT requires 10% lesser RAM than TEA or PRINCE algorithms and 30% less than PRESENT algorithm. The clock cycles required is 10–20% less than the existing algorithms. Dridi et al. [11] presented and analysed a Chaos based cipher system that operates in CBC (cipher block chaining) mode. A reliable pseudorandom number generator employing the Chaotic sequences implements the key scheduling for the proposed approach (PRNG-CS). The circular substitution operation is performed by a S-BOX is a chaos based so as to introduce confusion process. Four distinct 1-D chaotic maps are used in the development of the PRNG-CS to counteract the divide and conquer tactic and thus improving the randomness of the numbers that are generated. HAD—Horizontal Addition—Diffusion and a 2-D modified cat map carry out the diffusion process. A range of statistical and cryptographic assaults are determined to be ineffective against the chaos-based encryption system proposed in the paper, according to security analysis and a range of simulation findings of both the complete cryptosystem and its essential components (PRNG-CS and S-box). Despite the availability of abundant state-of-the-art cryptographic implementation techniques, a great amount work is still needed. Each of the available methods has its own limitations, and the issue still poses numerous challenges.

## Proposed Encryption Scheme

The suggested HIBRI cipher's encryption procedure is depicted in Fig. 1. The proposed algorithm is developed by fusing the features of HIGHT algorithm and used the ARX operations from the BRIGHT family of ciphers like TEA proposed by Wheeler and Needham (1994) [28], [38], LEA

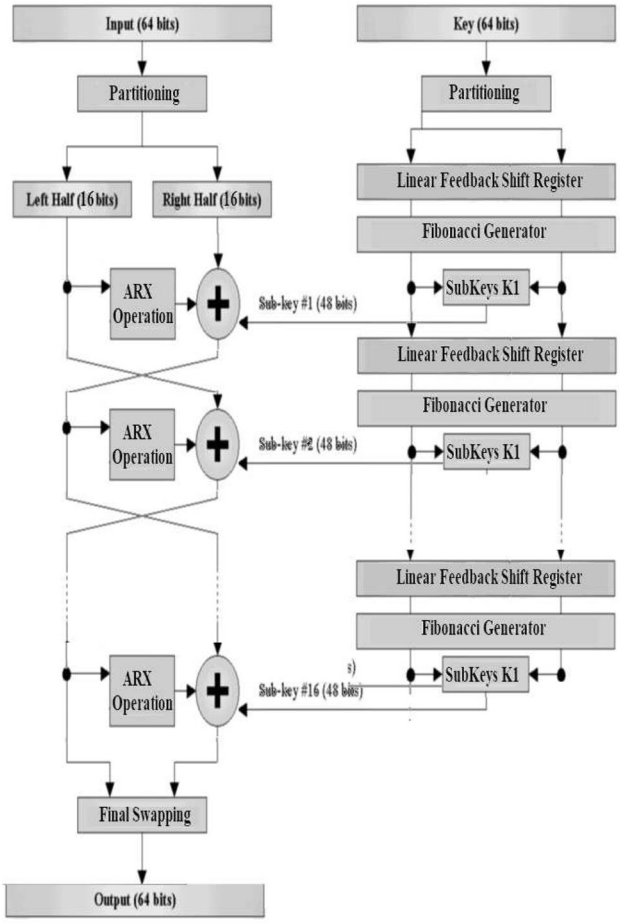


**Fig. 1** HIBRI Encryption Process

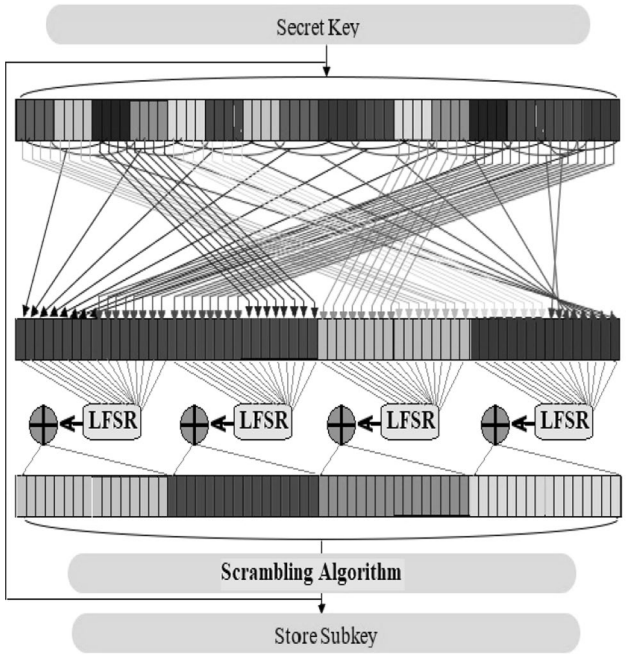


**Fig. 2** Key Generation Process

proposed by Hong et al. [14], SIMON and SPECK proposed by Beaulieu et al. (2015).

### Key Scheduling Algorithm

The algorithm for key scheduling is developed by combining SIT algorithm developed by Usman et al. [37] and Fibonacci scrambling algorithm proposed by Amiruddin et al. [1]. The novel key generation scheme is implemented which uses Rotation and XOR operations along with LFSR and Fibonacci scrambling algorithm. The Permutation, Modulo-addition, Rotate and e-XOR operations in the proposed scheme's architecture contributes to the properties of confusion and diffusion thus improving the strength of the proposed technique in terms of security. The proposed lightweight encryption is based on basic two-step process:

(1) Key scheduling algorithm
(2) Encryption Algorithm.

One of the most vital components of any cryptographic scheme is the key, as the integrity and confidentiality of the encrypted information is at risk if the key is known to an adversary. The information's overall security is dependent upon this key. The proposed algorithm is implemented for 5 rounds so as to reduce power consumption and area so as to suit the need for resource constrained devices and also to boost security, the number of rounds can be increased. The proposed key scheduling algorithm illustrated in Fig. 2 uses the combination of the feistel network and the ARX features. Also, the proposed algorithm makes use of the concept of LFSR and Fibonacci scrambling algorithm to provide better confusion and diffusion as compared to other existing algorithms.

The 64-bit initial key is the input to the key scheduling algorithm. The first key is kept big enough to fend off attacks using an extensive search. The output of this key scheduling algorithm is 5 round keys which becomes the input for the next step of encryption algorithm.

The steps followed are:

1. The key scheduling algorithm receives as input an initial seed 64 bits and is selected by the user
2. The initial input key (64-bits) is permuted into four-bit blocks as IKb1, IKb2, IKb3, …,IKb16.
3. IKb1, IKb2, IKb3, …, IKb16 are permuted and concatenated to form 16-bit blocks as follows.

SK1 = [ IKb1 || IKb5 || IKb9 || IKb13]
SK2 = [ IKb2 || IKb6 || IKb10 || IKb14]
SK3 = [ IKb3 || IKb7 || IKb11 || IKb15]
SK4 = [ IKb4 || IKb8 || IKb12 || IKb16]

4. Each of the 16-bit produced in third step is fed into the Linear feedback shift register (LFSR) which then generates 16-bit (PRNG) pseudorandom number.
5. The pseudo random number generated in step four and the 16-bit output of step three is XORed to produce The 16-bit outputs, o1, o2, o3, and o4, are placed into the Fibonacci scrambling method to obtain the round keys Kr(1)-Kr(5) for encryption process. The steps for the Fibonacci scrambling algorithm are as follows:

i. The key Kr(1) = mod ( o1 + o2, n)
ii. Key Kr(2) = mod ( o3 + o4, n)
iii. The remainder of the round-keys are acquired as follows (n is largest prime)
for i = 3 to n do
Kr(i) = mod(Kr(i-1) +Kr(i-2), n)
end for
iv. end

Hence, the keys Kr(1) through Kr(5) are produced and utilized as round keys for the HIBRI lightweight encryption algorithm's encryption procedure.

Algorithm 1: HIBRI Cipher Algorithm

RoundFunction($P_{T0}$, $P_{T1}$, $P_{T2}$, $P_{T3}$, $P_{T4}$, $P_{T5}$, $P_{T6}$, $P_{T7}$, Kr1}
{
$X_{1,0} \leftarrow P_{T0} \boxplus Kr1;$
$X_{1,1} \leftarrow P_{T1} \oplus F(X_{1,0});$
$X_{1,2} \leftarrow P_{T1} \oplus F(X_{1,3});$
$X_{1,3} \leftarrow P_{T3} \boxplus Kr1;$
$X_{1,4} \leftarrow P_{T4} \boxplus Kr1;$
$X_{1,5} \leftarrow P_{T6} \oplus F(X_{1,4});$
$X_{1,6} \leftarrow P_{T5} \oplus F(X_{1,7});$
$X_{1,7} \leftarrow P_{T7} \boxplus Kr1;$
}
FinalTransformation($T_{1,0}$, $T_{1,1}$, $T_{1,2}$, $T_{1,3}$, $T_{1,4}$, $T_{1,5}$, $T_{1,6}$, $T_{1,7}$)
{ Q0← $T_{1,0;}$ Q1← $T_{1,2;}$ Q2← $T_{1,3;}$ Q3← $T_{1,4;}$ Q4← $T_{1,5;}$ Q5← $T_{1,6;}$ Q6← $T_{1,7;}$ Q7← $T_{1,0;}$ }

## The HIBRI Encryption Scheme

The round function of the proposed encryption algorithm is.

inspired from the HIGHT lightweight encryption algorithm. The algorithm uses the ARX features from the BRIGHT family of ciphers since by using ARX operations the decryption overhead is reduced since the decryption process using ARX reverses the steps of encryption process. The round keys generated in the key scheduling process is used in the encryption process. Figure 3 shows one round encryption process.

## Encryption Process

The notations used in the encryption process are as follows.

$\boxplus$: addition mod $2^{16}$
$\oplus$: XOR (bitwise-EX-OR)
RA<< S: Shift left register RA by S bits

The algorithm 1 depicts the process of Encryption steps for the proposed HIBRI cipher it encrypts 128-bit plaintext block. The 128-bits original plaintext block is first segmented into 8 blocks of 16-bits each as $P_{T0}$, $P_{T1}$, $P_{T2}$, $P_{T3}$, $P_{T4}$, $P_{T5}$, $O_{PT6}$, $P_{T7}$ respectively. The keys for each round generated by the key scheduling process is used in each of the rounds of the HIBRI encryption process. Each of the encryption rounds makes use of a shift function F(x):

$$F(x)=\{[Kro << 7) \oplus Kro << 2] \oplus Lo \boxplus Kri\}$$

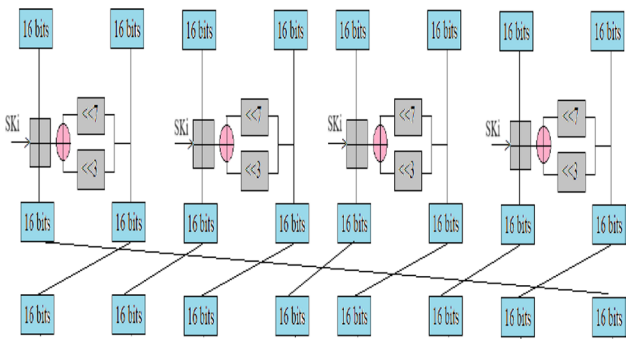


**Fig. 3** One round HIBRI Encryption Process

The proposed algorithm is implemented for 5 rounds, the numberof rounds may be increased in order to increase security. The following functions shows the operations performed in the first round using key R1. The output of the first round Q0–Q7 becomes the input to this fed into the next round. The process is repeated for next rounds. Figure 4 shows the keys.

generated for one round using the proposed key generation scheme.

## Experimental Setup and Performance Evaluation

The Proposed HIBRI encryption algorithm software implementation is carried out C programming observed on a Intel (R) Core (TM) 2 Duo E4500 @ 2.20 GHz processor which is 32-bit and is evaluated for randomness test, strict avalanche criteria, key sensitivity test. While the hardware implementation is carried out on Arduino Nano based on the Atmega328 microcontroller and the key parameters such memory utilization, energy consumption, execution time and throughput are analysed.

### Evaluation of the Key Generation Algorithm

The Proposed key scheduling algorithm generated $10^6$ bit streams, to test the randomness of the keys NIST randomness test (Bassham et al. [4] ) with level of significance $P \geq 0.01$.

(Barker et al. 2020) was run and it is evident from the findings in Table 1 that the produced key is, in fact, random.

### Evaluation of the Cipher

The proposed HIBRI cipher is also evaluated for performance using the following criteria:

#### *Strict Avalanche Criteria (SAC)*

The avalanche requirement is that, if the input bit (key bit or the plaintext bit) is altered by a single bit, the ciphertext bits should change by 50% (Gookyi et al. [13], Biswas et al. [7]). Table 2, summarizes the diffusion results. A single one-bit change in the proposed cipher plaintext causes around half of the bits in the ciphertext to change, yielding a 50% diffusion rate.

#### *Key Sensitivity*

When the key varies even little from the original key(one-bit), it is impossible to retrieve the original plaintext and thus the key scheduling algorithm is said to be key sensitive. The Avalanche test evaluates the extent to which modifications in the plaintext result in alterations to the ciphertext. Table 3 lists the key sensitivity findings for a key change of one bit. When a one-bit change in the key is affected, the result is initially transformed to binary vales, and the number of bits altered in the cipher-text is then found to determine the degree of alterations.

#### *Correlation Coefficient*

The calculation of the correlation coefficient is used to determine the dependency of groups of values. It serves as a reliable method for assessing the robustness of encryption

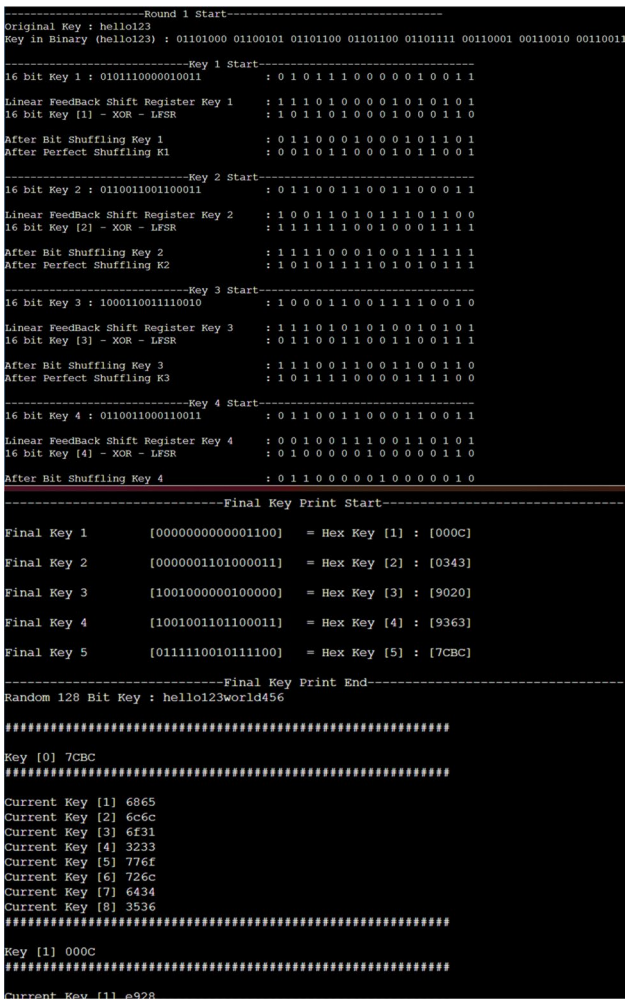


**Fig. 4** Results of Key generation

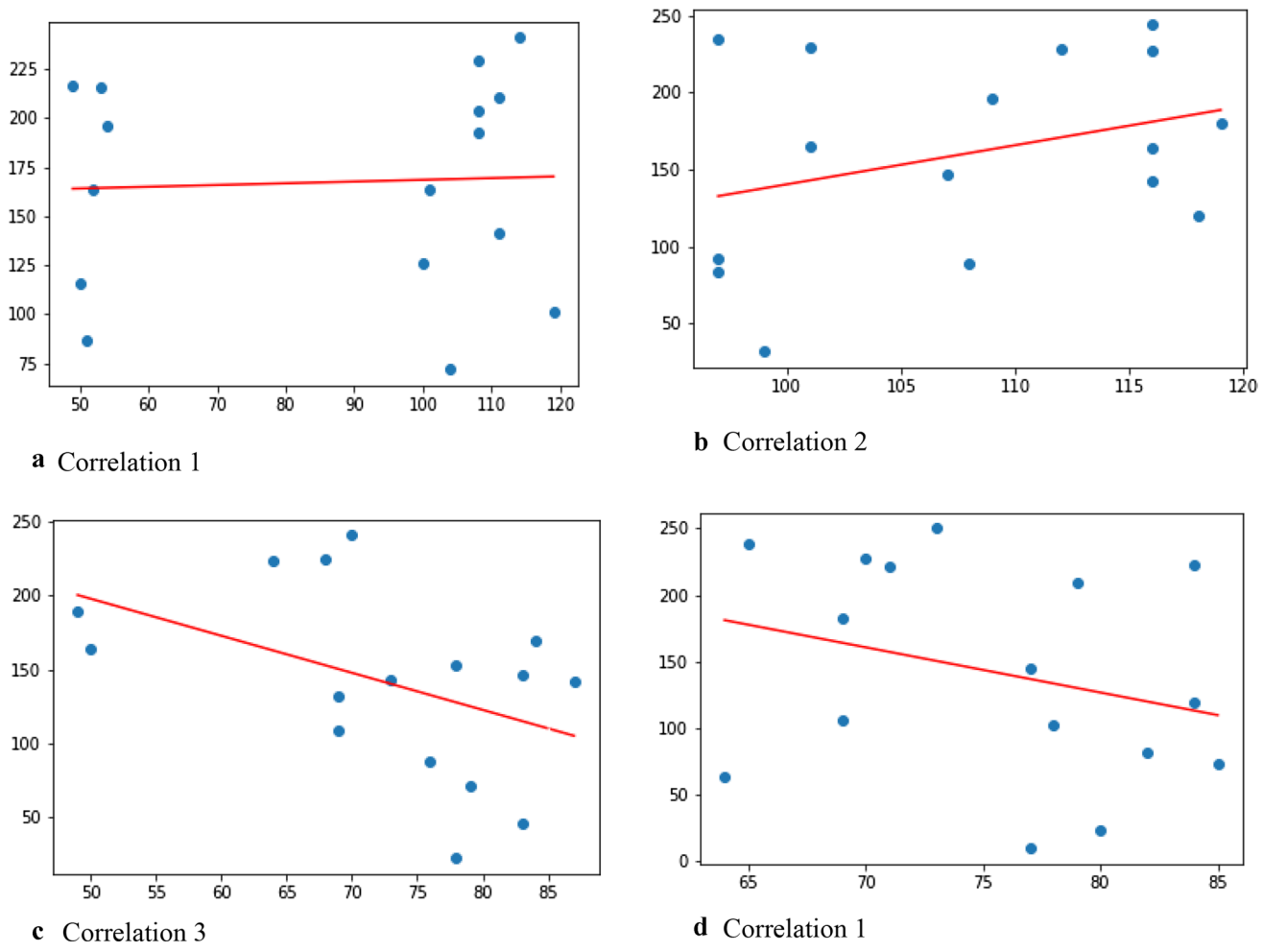


**Fig. 5** Correlation coefficients for different plaintext message

algorithms. This test is intended to determine whether or if there is a relationship between the corresponding characters in plaintext and ciphertext messages. Values near to |1| imply there is a high correlation and dependence between the values. A reading between 0 and −1 indicate that both the variables move in opposite direction and hence not correlated to each other and hence this ensures better security. A coefficient of correlation between plaintext and ciphertext can be determined as follows::

$$\tau_{xy} = \frac{cov(x,y)}{\sqrt{\partial(x)}\sqrt{\partial(y)}}$$

$$\text{Covariance } \partial(\alpha) = \frac{1}{N}\sum_{i=1}^{K}\left(\alpha_i - \varepsilon(\alpha)\right)^2 =$$

$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{K}\left(x_i - \varepsilon(x)\right)\left(y_i - \varepsilon(y)\right)$$

$$E(p) = \frac{1}{N}\sum_{i=1}^{N} p_i$$

The Correlation-Coefficient between the original and encrypted text is shown in Table 4, and Fig. 5 illustrates the same.

## Hardware Implementation

Figure 6 displays the suggested cipher's block diagram. The cipher's conceptualization was translated into a tangible form through successful integration into Arduino Nano 3.0 hardware Arduino Nano 3.0 based on Atmel

**Table 1** NIST test outcomes for the suggested key generation scheme

| Test | Significance level ($p$ value) | Result |
|---|---|---|
| Frequency-test | 0.739918 | A |
| Block-frequency-test | 0.179120 | A |
| Cumulative-sums-test (forward) | 0.534146 | A |
| Cumulative-sums(inverse) -test | 0.739918 | A |
| Runs-test | 0.350485 | A |
| Longest-run-test | 0.179120 | A |
| Rank-test | 0.035174 | A |
| Fft-test | 0.213309 | A |
| Non-overlapping-template-test | 0.122325 | A |
| Overlapping-template-test | 0.430102 | A |
| Universal-test | 0.122325 | A |
| Approximate-entropy-test | 0.350485 | A |
| Random-excursions-test | 0.911413 | A |
| Random-excursions-variant-test | 0.534146 | A |
| Serial-test | 0.035174 | A |
| Linear-complexity | 0.739918 | A |

A – ACCEPT (test Passed)

**Table 2** The diffusion for HIBRI, for a single-bit change in plaintext

| KEY (HEXADECIMAL) = 69 64 6D 6D 6F 31 33 33 | | |
|---|---|---|
| **PLAINTEXT (HEXADECIMAL)** | **CIPHERTEXT (HEXADECIMAL)** | **NUMBEROF BITSCHANGED** |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | F2 84 D3 57 DD 49 FC 52 8C 6D 2B 26 B4 5C F2 84 | 35 |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 | 7E CE F2 84 16 D9 57 A7 FC 52 8C6D 62 7C 7E CE | |

**Table 3** Key Sensitivity A single bit change in the key

| PLAIN TEXT (HEXADECIMAL) = 01 11 00 00 01 00 00 01 00 00 00 01 00 11 00 00 | | |
|---|---|---|
| **KEY (Hexadecimal)** | **CIPHER TEXT (Hexadecimal)** | **Total no ofbitschanged** |
| 07 05 03 04 08 29 2a 0b 10 11 | F2 84 D3 57 DD 49 FC 52 8C 6D 2B 26 B4 5C F2 84 | 64 |
| 06 04 02 03 08 29 2a 0b 10 11 | FC 36 84 00 AC 38 39 19 87 AF FD 32 72 D9 36 F7 | |

ATmega328 microcontroller as the IoT testbed to determine the performance parameters such as throughput, power consumption, average execution time, memory consumption for encryption and decryption process. Atmel ATmega328-based Arduino Nano boasts a 16 MHz 8-bit microcontroller, accompanied by 32 KB of Flash memory, 2 KB of SRAM, and an additional 1 KB of EEPROM. The Arduino Nano's recommended input voltage range is 7 to 12 V, and all simulations have been done in C. The proposed algorithm is first tested using a normal text data on the client–server node IoT system shown in Fig. 7. The encryption algorithm is used on the data being transmitted from client. Once the data reaches the server node it is decrypted to get the original data depicted in the Fig. 8. Then the proposed algorithm is applied on sensor data. For this we use DHT11(Humidity and Temperature and LM35(Temperature sensor) to test the algorithm on sensor data. The sensors individually and independently collect data which is then transmitted to the microcontroller.

## Results and Discussion

The suggested cipher implementations are tested against the most recent state-of-the-art implementations using the ATmega328 (Arduino Nano 3.0). The selection of the ATmega328 stemmed from its widespread adoption within wireless sensor networks. The evaluation of the proposed HIBRI cipher against state-of-the-art implementations is presented in Table 5, with the cipher itself executed on an 8-bit AVR platform requires more RAM than SIMON and lesser than HIGHT ciphers and has lesser RAM than other state-of-the art lightweight ciphers. The execution time required is also nearer to the SIMON and HIGHT ciphers hence the proposed HIBRI cipher is possible candidate for the lightweight cryptographic applications. The comparison chart is shown in Fig. 9.

In order to compare other parameters such as area consumed, execution time and power consumption the proposed method is implemented in FPGA using Verilog. XILINX FPGA chips were used to synthesis, position, and route all of the internal design components. Using XPower Analyzer, power and execution time estimates for a completely sub-pipelined architecture with one round unit and a bit length of 32 have been completed. The results are displayed. Table 6 presents a comparative analysis of the HIBRI cipher's performance evaluation, focusing on area metrics such as LUT (Look-Up Tables), IOB (Input/Output Blocks), and Slice Registers in comparison with the other state of art

**Table 4** Correlation Coefficients

| KEY (HEXADECIMAL) = 48454c4c4f313233 | | | |
|---|---|---|---|
| Examples | PLAIN TEXT (HEXADECIMAL) | CIPHER TEXT (HEXADECIMAL) | CORRELATION COEEFICIENT |
| Correlation1 | 68656C6C6F313233776F726C64343536 | 48A4CCE58DD9745765D3F1C17EA4D8C4 | 0.047156 |
| Correlation2 | 61747461636B61747477656C7665706D | 54A48F5C2093EBF4E3B4A55978E5E4C4 | 0.0328462 |
| Correlation3 | 53454E444E4F57495453454C46403132 | 2D8499E116478E8FA9926C58F1E0BDA4 | −0.442953 |
| Correlation 4 | 4D454554494E47415440464F5552504D | 916AB7DFFA66DDEF7740E3D1495218A | −0.289888 |

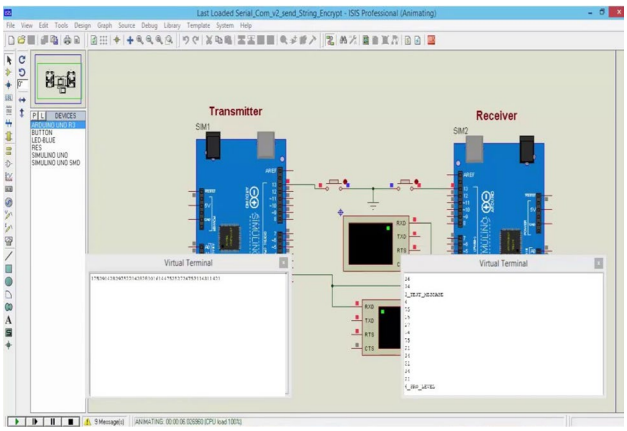


**Fig. 6** Block diagram of the proposed cipher implementation

implementations whereas Table 7 makes the comparison in terms of power and execution time. The comparison chart is shown in Figs. 10 and 11 respectively.

## Conclusion and Future Work

Using a unique key generation method, the suggested encryption algorithm HIBRI is implemented in hardware as well as software. On a 32-bit processor, HIBRI's performance is assessed. HIBRI cipher fulfills the Strict Avalanche Criteria and key sensitivity test which demonstrates how secure the suggested cipher is. The keyscheduling algorithm and its randomness is compared with the existing methods which proves the randomness of the round keys generated. The NIST tests for key generation yielded $P \geq 0.01$ which proves that the proposed key generation scheme generated truly random numbers, while the hardware implementation shows that the comparison results for Code size, RAM and execution size yielded better results, thus concluded the proposed algorithm is possible candidate for lightweight cryptography. IoT makes the life of user comfortable by offering a network that senses, collects, and makes decision effectively. Everything that is connected to a network is more vulnerable to assaults. As a result, adding security to an IoT network will boost consumer trust. As the Internet of Things (IoT) enters its nascent phase with the new cryptographic standards emerging over the period of time, the security and implementation efficiency must fulfil

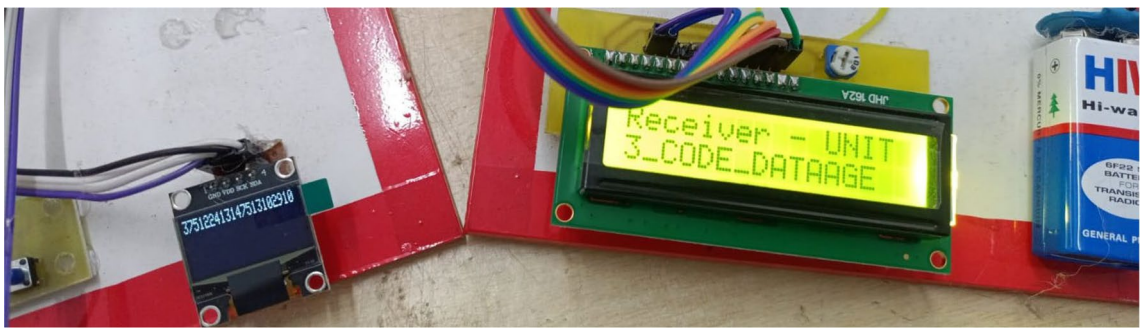


**Fig. 7** Hardware Implementation

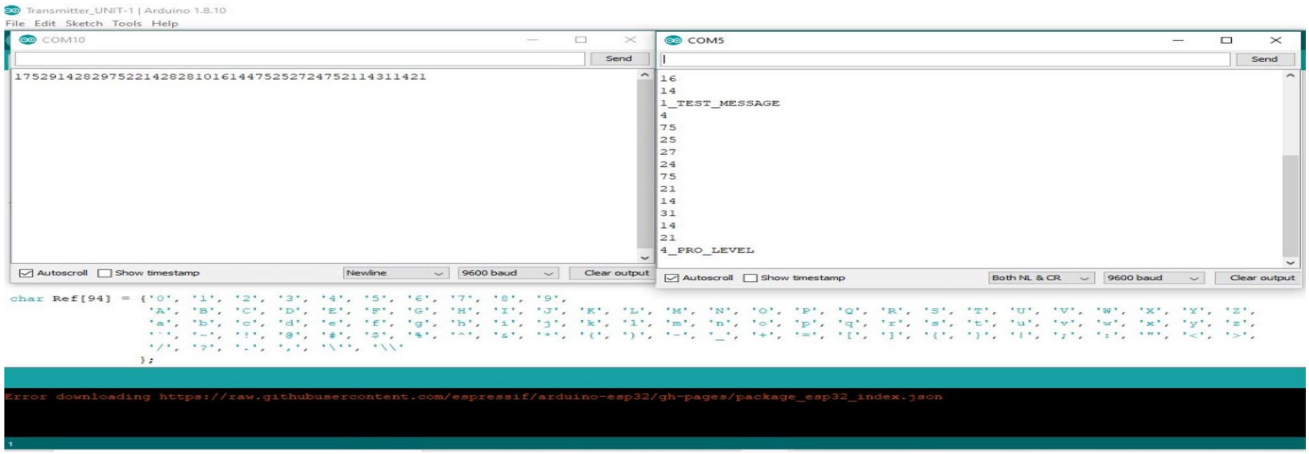


**Fig. 8** Encryption and decryption on COM ports

**Table 5** Comparative anlaysis of Proposed HIBRI cipher against state-of-the-art implementation ciphers on target embedded processors

| Algorithm | Year | Plaintext | KeySize | Code Size | RAM | Timing |
|---|---|---|---|---|---|---|
| SIMON [5] | 2013 | 64 | 128 | 290 | 24 | 253 |
| PRESENT [21] | 2021 | | | 660 | 280 | 729 |
| LEA [32] | 2015 | | | 592 | 596 | 969 |
| CRAFT [6] | 2019 | | | 894 | 243 | 1504 |
| ARIA [30] | 2020 | | | 2352 | 242 | 198 |
| AES [19] | 2020 | | | 1116 | 268 | 222 |
| HIGHT [18] | 2019 | | | 352 | 180 | 197 |
| Proposed HIBRI | 2022 | | | 316 | 264 | 287 |

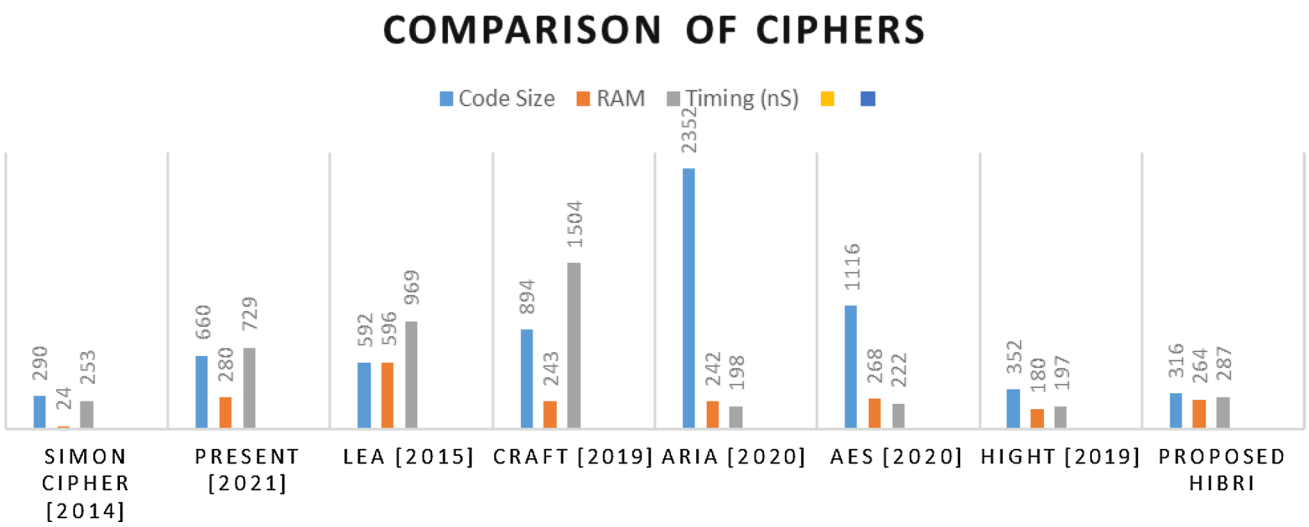


**Fig. 9** The comparison of ciphers

**Table 6** Proposed HIBRI cipher compared with state-of-the-art implementation cipher on FPGA

| Cryptographic algorithms | LUT | IOB | Slice Registers |
|---|---|---|---|
| Proposed HIBRI | 0 | 240 | 168 |
| AES | 2255 | 6 | 1661 |
| HIGHT | 268 | 1 | 2409 |
| SIMON | 45 | 5 | 36 |
| PRESENT | 72 | 35 | 65 |

**Table 7** Performance evaluation based on execution time and power consumed

| Cryptographic Algorithms | Power(W) | Timing(ns) |
|---|---|---|
| Proposed HIBRI | 0.014 | 3.925 |
| AES | 0.441 | 8.975 |
| HIGHT | 0.609 | 8.34 |
| SIMON | 0.072 | 37.8 |
| PRESENT | 0.186 | 4.98 |

the various criteria of novel security applications. The random number generation implemented in this research combining the features of linear feedback shift register and fibonacci scrambling algorithm, can be explored further developing a new key scheduling algorithm with better security aspects. The novel HIBRI cipher implemented consumes lesser power, has better execution speed and lower device utilization, also proves better security algorithm than the existing ciphers. The

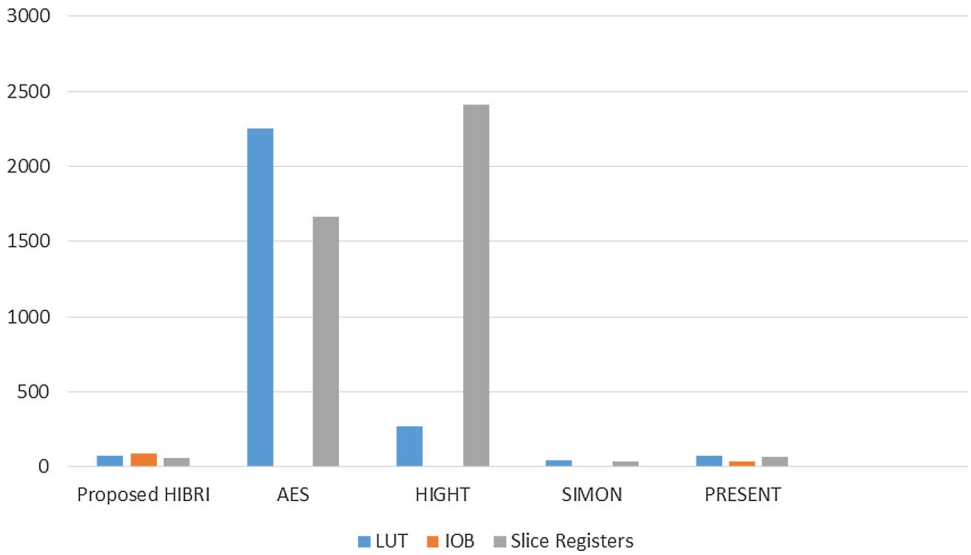


**Fig. 10** Comparison of Performance-metrics for existing ciphers (LUT and IOB)

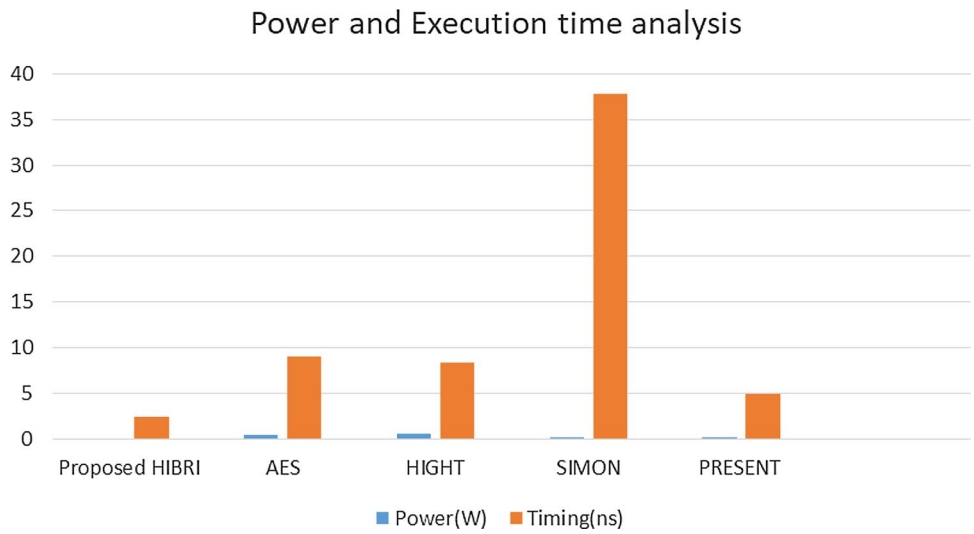


**Fig. 11** Performance-metrics comparison of ciphers (Power and Timing)

cipher can be evaluated and simulated on different platforms and also cryptanalysis can be applied on the proposed method as a future enhancement.

**Funding** No funding.

**Data Availability** No such data was used.

**Declarations**

**Conflict of interest** The authors declare that they have no confict of interest.

**Consent for Publication** Authors give consent for publication in the journal.

## References

1. A. Amiruddin, A.A.P. Ratna, R. Sari, Construction and analysis of key generation algorithms based on modified fibonacci and scrambling factors for privacy preservation. Int. J. Netw. Sec. **4**, 250–258 (2019)
2. G. Bansod, A new lightweight encryption design at node level. Int. J. Sec. Appl. **10**(12), 111–128 (2016)
3. E. Barker, A. Roginsky, R. Davis, Recommendation for Cryptographic Key Generation, Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, [online] (2020). https://doi.org/10.6028/NIST.SP.800-133r2
4. L. Bassham, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, S. Leigh, M. Levenson, M. Vangel, N. Heckert, D. Banks, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, [online] (2010). https://tsapps.nist.gov/publication/get_pdf.cfm?pub_i d=906762
5. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013)
6. C. Beierle, G. Leander, A. Moradi, S. Rasoolzadeh, CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Trans. Symm. Cryptol. **2019**, 5–45 (2019)
7. A. Biswas, A. Majumdar, S. Nath et al., LRBC: a lightweight block cipher design for resource constrained IoT devices. J. Ambient Intell. Human Comput. (2020). https://doi.org/10.1007/s12652-020-01694-9
8. J. Borghoff, L.R. Knudsen, G. Leander, S.S. Thomsen, Cryptanalysis of PRESENT- Like Ciphers with Secret S-Boxes. In: Joux, A. (eds.) Fast Software Encryption. FSE 2011. Lecture Notes in Computer Science, vol. 6733. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_16
9. B. Chaitra, V.G. Kumar, R.C. Shatharama, A survey on various lightweight cryptographic algorithms on FPGA. IOSR J. Electron. Commun. Eng. **12**(1), 45–59 (2017)
10. D. Coppersmith, Data encryption standard (DES) and its strength against attacks. IBM J. Res. Dev. (1994). https://doi.org/10.1147/rd.383.0243
11. F. Dridi, S.E. Assad, W.E.H. Youssef, M. Machhout, R. Lozi, Design, implementation, and analysis of a block cipher based on a secure chaotic generator. Appl. Sci. **12**(19), 9252 (2022). https://doi.org/10.3390/app12199952.⟨hal-03799417v2⟩
12. Z.G. Eisenbarth, S. Heyse et al., Compact Implementation and Performance Evaluation of Block Ciphers in ATtiny Devices. In: Mitrokotsa, A., Vaudenay, S. (eds) Progress in Cryptology - AFRICACRYPT 2012. AFRICACRYPT 2012.Lecture Notes in Computer Science, vol 7374. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31410-0_11
13. D.A.N. Gookyi, S. Park, K. Ryoo, The efficient hardware design of a new lightweight block cipher. Int. J. Control Autom. **1**(1), 431–440 (2017)
14. D. Hong, J.K. Lee, D.C. Kim, D. Kwon, K.H. Ryu, D.G. Lee, LEA: a 128-bit block cipher for fast encryption on common processors. In International Workshop on Information Security Applications (pp. 3–27). Springer, Cham (2013)
15. D. Hong, J. Sung, S. Hong et al., HIGHT: a new block cipher suitable for low-resource device, in Cryptographic Hardware and Embedded Systems—CHES, pp. 46–59, Springer, Berlin, Germany, 2006

16. K. Jang, G. Song, H. Kim, H. Kwon, H. Kim, H. Seo, Efficient implementation of PRESENT and GIFT on quantum computers. Appl. Sci. **11**, 4776 (2021). https://doi.org/10.3390/app11114776
17. Juels and S.A. Weis, Authenticating pervasive devices with human protocols, in *Proceedings of the Annual International Cryptology Conference*, pp. 293–308, Springer, Santa Barbara, CA, USA, August 2005
18. B. Kim, J. Cho, B. Choi, J. Park, H. Seo, Compact implementations of HIGHT block cipher on IoT platforms. Sec. Commun. Netw. (2019). https://doi.org/10.1155/2019/5323578
19. Y. Kim, S.C. Seo, An efficient implementation of AES on 8-Bit AVR-based sensor nodes. In *Proceedings of the International Conference on Information Security Applications*, Jeju Island, Korea, 26–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 276–290
20. V.G. Kiran Kumar, C. Shantharama Rai, Design and implementation of novel BRISI lightweight cipher for resource constrained devices. Microprocess. Microsyst. **84**, 104267 (2021). https://doi.org/10.1016/j.micpro.2021.104267
21. H. Kwon, Y.B. Kim, S.C. Seo, H. Seo, H. Kwon, Y.B. Kim, S.C. Seo, High-speed implementation of PRESENT on AVRMicrocontroller. Mathematics **9**(4), 374 (2021). https://doi.org/10.3390/math9040374
22. M.K. Ledda, B. Gerardo, and A. Hernandez, Enhancing IDEA Algorithm using Circular Shift and Middle Square Method, pp. 1–6 (2019).https://doi.org/10.1109/ICTKE47035.2019.8966827
23. J. Lee, K. Jeong, H. Kim, J. Kim, S. Chee, HIGHT: A New Block Cipher Suitable for Low Resource Device. In: Goubin L., Matsui M. (eds) Cryptographic Hardware and Embedded Systems - CHES 2006. CHES 2006. Lecture Notes in Computer Science, vol 4249. Springer, Berlin, Heidelberg (2006)
24. J. Lu, Y. Liu, Improved rotational-XOR cryptanalysis of Simon-like block ciphers. IET Inform. Sec. **16**(4), 282–300 (2022). https://doi.org/10.1049/ise2.12061
25. A. Majumdar, N.M. Laskar, A. Biswas, S.K. Sood, K.L. Baishnab, Energy efficient e-healthcare framework using HWPSO-based clustering approach. J. Intell. Fuzzy Syst. **36**(5), 1–13 (2018)
26. J.L. Massey, SAFER K-64: A byte-oriented block-ciphering algorithm. In: Anderson, R. (eds) Fast Software Encryption. FSE 1993. Lecture Notes in Computer Science, vol 809. Springer, Berlin, Heidelberg (1994). https://doi.org/10.1007/3-540-58108-1_1
27. K.A. McKay, M. Bassham, M.S. Turan, N. Mouha, DRAFT NISTIR 8114 Report on Lightweight Cryptography, National Institute of Standards and Technology Internal Report 8114, August 2016
28. R.M. Needham,and D.J. Wheeler, Tea extensions. Report (Cambridge University, Cambridge, UK) (1997)
29. A. Poojary, V.G. Kiran Kumar, H.R. Nagesh, FPGA implementation novel lightweight MBRISI cipher. J. Ambient Intell. Human Comput. **14**, 11625–11637 (2023). https://doi.org/10.1007/s12652-022-03726-y
30. P.P. Ray, Internet of things for smart agriculture: technologies, practices and future direction. J. Ambient Intell. Smart Environ. **9**(4), 395–420 (2017)
31. H. Seo, I. Jeong, J. Lee, W.-H. Kim, Compact implementations of ARX-based block ciphers on IoT processors. ACM Trans. Embed. Comput. Syst. (2018). https://doi.org/10.1145/3173455
32. H. Seo, H. Kwon, H. Kim, J. Park, ACE: ARIA-CTR encryption for low-end embedded processors. Sensors **20**, 3788 (2020)
33. H. Seo, Z. Liu, J. Choi, T. Park, H. Kim, Compact Implementations of LEA Block Cipher for Low-End Microprocessors. In: Kim, Hw., Choi, D. (eds) Information Security Applications. WISA 2015. Lecture Notes in Computer Science(), vol 9503. Springer, Cham (2016). https://doi.org/10.1007/978- 3–319–31875–2_3
34. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, T. Shirai, Piccolo: an ultra-lightweight blockcipher. CHES **6917**, 342–357 (2011)
35. T. Shirai, K. Shibutani, T. Akishita, S. Moriai, T. Iwata, The 128-bit blockcipher CLEFIA. FSE **4593**, 181–195 (2007)
36. T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, E. Twine: A lightweight, versatile block cipher. In ECRYPT Workshop on Lightweight Cryptography (Vol. 2011) (2011)
37. I. Usman, M. Ahmed, S. Imran, U.A. Khan, SIT: a lightweight encryption algorithm for secure internet of things. Int. J. Adv. Comput. Sci. Appl. (2017). https://doi.org/10.14569/ijacsa.2017.080151
38. D.J. Wheeler, R.M. Needham, TEA, a Tiny Encryption Algorithm, in *FSE 1994*, ed. by B. Preneel. LNCS, vol. 1008 (Springer, Heidelberg, 1995), pp.363–366
39. W. Wu, L. Zhang, LBlock: a lightweight block cipher. Appl. Cryptogr. Netw. Sec. (2011). https://doi.org/10.1007/978-3-642-21554-4
40. W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, I. Verbauwhede, RECTANGLE: a bitslice lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. (2015). https://doi.org/10.1007/s11432-015-5459-7