https://doi.org/10.48047/AFJBS.6.12.2024.6203-6221

African Journal of Biological Sciences

Journal homepage: http://www.afjbs.com

Research Paper Open Access

NLBCIT: A Novel Approach for Securing data in IoT devices based on a Lightweight Block Cipher Design Algorithms

Abdullah Alahdal[1*], Sai Kumar D[2],P. V. Sudha[3], Abdulrazzaq H. A.Al-ahdal[4], and Adnan Alkarmadi[5]

[1]Department of Computer science Engineering, University college of Engineering, Osmania University, Hyderabad 500007, India; ab.alahdal@gmail.com

[2]Department of Computer science Engineering, University college of Engineering, Osmania University, Hyderabad 500007, India;sai.daripalli@gmail.com

[3]Professor, Department of Computer science Engineering, University college of Engineering, Osmania University, Hyderabad 500007, India; sudha.p@uceou.edu

[4]Department of Computer Science, Faculty of CSE, Hodeidah University, Hodaidah, Yemen; alahdal201211@gmail.com

[5]Department of Computer science Engineering, University college of Engineering, Osmania University, Hyderabad 500007, India;

Corresponding Author : *Abdullah Alahdalab.alahdal@gmail.com

Article History
Volume 6, Issue 12, 2024
Received: 6 June 2024
Accepted: 6 July 2024
doi:
10.48047/AFJBS.6.12.2024.6203-6221

Abstract—Modern applications incorporate various control devices and sensors that connect to the Internet, collectively known as the Internet of Things (IoT). These innovative technologies have garnered significant research attention due to their extensive applications and rapid development. The transmission of private and sensitive data between IoT devices demands stringent data confidentiality measures. Traditional cryptographic algorithms, which involve complex mathematical operations and numerous computational cycles, result in high memory usage and energy consumption, rendering them unsuitable for resource-constrained devices. To address this issue, a lightweight and efficient algorithm called NLBCIT has been proposed. This algorithm is designed to provide robust data protection while meeting the limitations of IoT devices.Simple math techniques like XNOR, swapping, shifting, and XOR are used by NLBCIT to encrypt 64-bit data with a 64-bit key.It incorporates features from both SP Network and Feistel architectures to enhance diffusion and confusion, thereby strengthening data security.The NLBCIT algorithm is simulated using FELICS and MATLAB. This approach uses a variety of data types, including text and images. Simulations show that the proposed technique is better in many aspects, including memory usage, security, efficiency, and fewer cycles (both for encryption and decryption).

Keywords : IoT Security, FELICS, RFID tags, Lightweight Cryptography LWC,

I. Introduction

In the field of technology, the Internet of Things is a relatively new idea. It has greatly improved human lives in many ways. Sophisticated computational resources are widely available. As a result, academics have been interested in it and have focused on it. A network of physical "things" that are linked to the Internet via sensors, software, and other technologies and have the ability to share data and interact with other devices and systems is referred to as the "Internet of Things" (IoT) [1]. In the recent past, there has been a notable global increase in the number of IoT applications. The Internet of Things (IoT) is driving an increase in intelligence in many challenging areas of operation, including smart traffic [4, 5], smart energy grids, smart healthcare, and home and building automation [2, 3]. IoT is growing quickly in this new information era. Its controllable course of events is eventually affected by insufficient security and protection assessments. IoT devices often collect various types of observed data through their numerous sensors, and some of this data may be considered sensitive from the user's perspective, requiring protection from unauthorized access. Research indicates that security remains a major concern for IoT users.It is important to add advanced security features to both the software and hardware structure of IoT devices to make them safer and protect sensitive information.

It is preferable that this design embed the cryptography principle into the functional weaknesses of IoT devices. Data in Internet of Things devices will be integrated and encrypted using cryptographic techniques before being published on back-end networks, including cloud servers. Cryptography is growing more and more dependent on security. However, IoT hardware and software pose a significant security risk due to their vast area and exorbitant power consumption. Additionally, a dependable and efficient public key management system is required for Internet of Things encryption operations. Security and data integration are both safeguarded by efficient key management. Given the lightweight nature of most IoT devices, a minimum stockpiling restriction ought to be in place. Moreover, the majority of IoT gadgets use less electricity because they are battery-operated. The current trend toward lightweight encryption replaces classic encryption because the former involves numerous mathematical operations and depletes the device's battery [7]. Symmetric key encryption algorithms are therefore frequently used in the design of Internet of Things devices since they need less memory, computing power, and storage space than asymmetric key cryptography techniques [6].

Block ciphers are a type of secret key cryptography that was developed in the last ten years for the development of computer devices with low resources since they are more widely disseminated and have easier software and hardware implementation. Unlike the stream figure, it requires a significant amount of equipment resources [6]. As opposed to traditional Feistel ciphers, replacement-permutation networks (SPNs) require a significantly fewer number of circular functions in order to generate block ciphers. As a result, a system that combines the SPN and Feistel systems has been proposed. Circular functionalities that are tiny and light, store minimal energy, and accelerate diffusion are provided by the synthesis structure [8, 9]. This paper presents a LW symmetric key cryptography strategy that improves all of the susceptibilities that have been demonstrated in previous studies, addressing all of the difficulties and constraints that small IoT devices face. Moreover, the Feistel and SPN configurations are combined to create the suggested cryptographic technique, which facilitates quicker spread. Conversely, a circular key management strategy has also been used in encryption.

The following structure was used to arrange the remaining portions of the paper: The relevant works in the required field are thoroughly discussed in Section II. In Area III, the suggested lightweight cryptographic algorithm was displayed top to bottom. Parts IV, V, and VI present the exploration environment together with results and other kinds of analysis. Section VII marked the conclusion of the report.

I. Literature survey

It is evident that LWC must minimize the prices of small computer devices, processing power, and data size. Thus, the primary goal of any tiny computer device should be to make a cryptographic approach LW in every way, including memory and power usage [10]. Algorithms likeLEA [11], TEA [12], PICCOLO[13], PRESENT [14],PRIDE [15], CLEFIA [16], SIMON/Spot [17], Sovereign [18], KATAN/KTANTAN [19], and Drove [20], are a few instances of ones that are still in use in several programming disciplines. The most well-known lightweight block ciphers that have been released recently are described in this section.

Feistel and SP networks are combined in [21], to provide the security needed to develop lightweight encryption techniques. This algorithm design provides a low-complexity architecture with 80 bit keys for 64 bit data that can be implemented in the Internet of Things. IoT is essentially a collection of limited devices that communicate with one another and transfer secret data without human intervention [22]. Symmetric cryptography must be used for IoT devices to provide end-to-end security over the data transported over the internet is vulnerable to unauthorized access [23].

A group of researchers introduced the CHAM family of lightweight cryptographic algorithms, based on a four-branch Feistel structure operating as an ARX. These algorithms employ three distinct metrics. Despite varying operational characteristics, all CHAM computations are suitable for resource-constrained environments. CHAM-128/128 is optimized for general-purpose use on 32-bit microcontrollers, while CHAM-128/256 offers enhanced security [24]. You should utilize the CHAM-64/128 primarily for low-end applications.

In light of SPN, proposed the BORON LWC approach in [25]. 64-bit plaintext and 128/80 key bits are supported. 25 rounds of round shift, permutations, and XOR operations are performed using 4-bit to 4-bit S-boxes. It also resists both differential and linear attacks.

The LiCi lightweight (LW) block cipher algorithm, as described in [26], involves thirty-one successive transformations combined with 4x4 lightweight S-boxes. With a Feistel structure, LiCi works as a balanced network. Its architecture can use a 64-bit key to handle 128-bit plaintext keys.[27] developed the SPN-based "RECTANGLE" block cipher technique, which could produce 64-bit ciphertext with 80-bit or 128-bit keys. The round count is 25 in a row. Three steps make up each of the twenty-five rounds: the Roundkey, the SubColumn, and the ShiftRow. The extremely hardware-friendly design and the improved competitive performance of apps are two benefits of this strategy.

ITUbee was introduced as software based on LW block ciphers in [28]. Using twenty rounds and principal brightness levels, more noteworthy assurance is guaranteed. It accepts 80 bits for the key lengthandplaintext block. Because it uses AES S-box, it uses less power, memory, and time. It was believed that the associated key attack would not have an effect on ITUBEE.

The best design features of Speck and Simon are combined to form a new family of LW block ciphers called Simeck, which makes block ciphers considerably smaller and more effective. In [29], it is suggested. For key lengths of 64, 96, or 128 bits, it can handle 32, 48, or 64 bits of data. Moreover, it has been demonstrated that SIMECK is robust against random-byte failure scenarios and bit-flip scenarios.

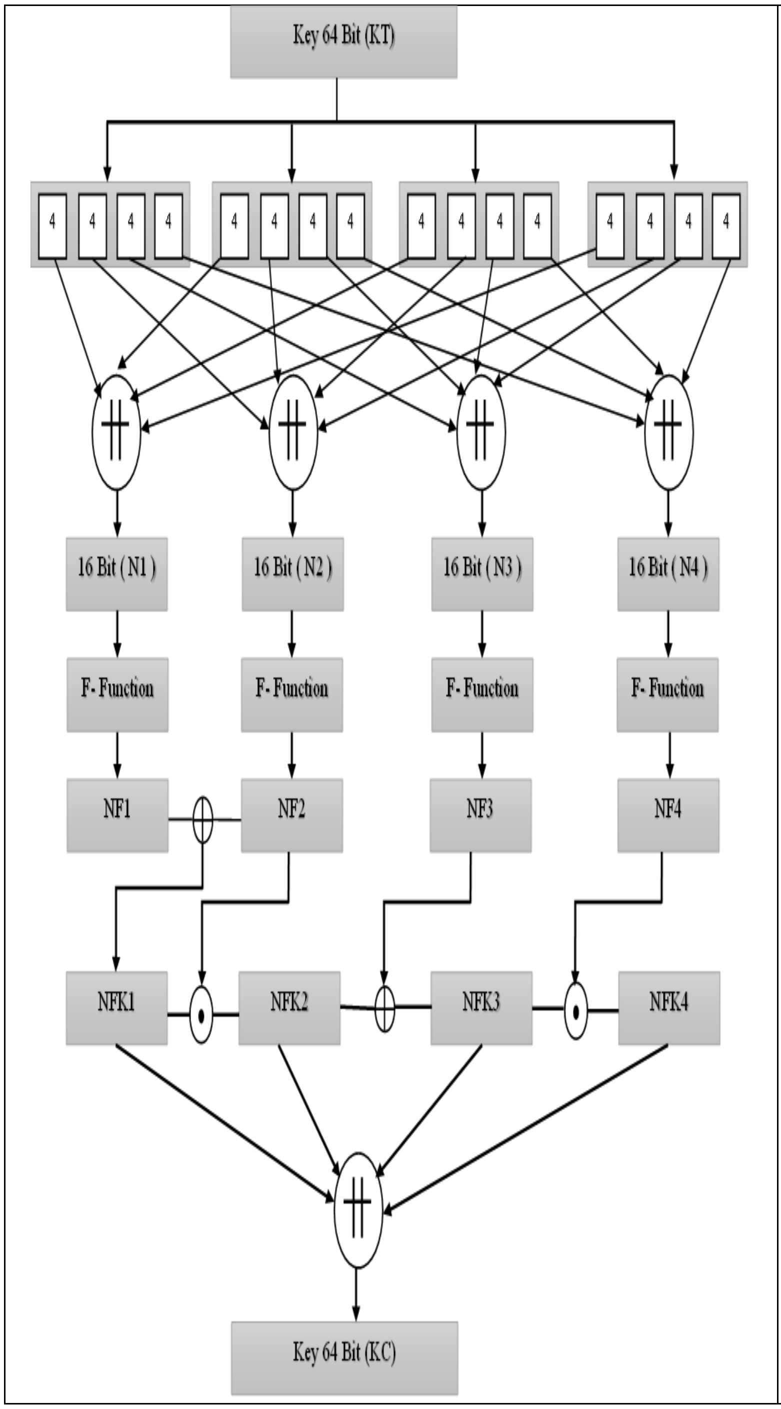
The "Lightweight Encryption Device," or LED, was proposed by Guo et al. (2011) [30]. It's among the most recent light codes. It could even be able to repel the most sophisticated threats. A four-bit matrix represents an ordered specification in the cipher state. The Drove's hash capabilities are comparable to those of the lightweight PHOTON.For Internet of Things operations, a lightweight six-round block cipher algorithm is utilized in [8]. Every round uses fundamental mathematical techniques. This increases the ambiguity for attackers and increases the difficulty of the encryption process. The encryption can handle 64-bit data and has an 80-bit key size. Strong security is provided by its interplay with the Feistel and SPN architecture [31].

II. Overview of NLBCIT

The recommended technique was developed using concatenation, XOR, and other fundamental logical operations. Encryption is applied to the data or information during these procedures. Thus, only the customer or someone with authorization can open the information/data encryption without any problems. The proposed algorithm considers a 64-bit block of data in exchange for a 64-bit key, since the goal of the key is to create the least possible complicated encryption scheme. The NLBCIT approach provides a high level of security since the SP and Feistel models were used in its development. The algorithm makes advantage of the SP and Feistel designs and performs simple mathematical operations with less rounds. For devices with low resources, this means that the algorithm is rapid, very safe, and consumes less RAM and power.

- **Expansion of The Key**

Essential Progression A key is the essential component of an encryption and decryption algorithm. The size of the encryption key is important for security reasons. An attacker won't be able to identify you as a result of the key's size. Stronger key creation is produced by the propagation and confusion stages of the suggested technique, which raises encryption complexity, decreases attacker keyknowledge, and boosts security. A 64-bit extension key is required. Key expansion is seen in Figure 1. Algorithm 1.



Algorithm 1 Key Expansion

Input: 64 bits key (KT)
Output: 64 bits key (KC)

1.Divided into 4 group, each group 16 bits
N1= KT[5] ‖KT[1] ‖ KT[13] ‖KT[9]
N2= KT[6] ‖KT[2]‖ KT[14] ‖ KT[10]
N3= KT[7‖] KT[3] ‖ KT[15] ‖ KT[11]
N4= KT[8] ‖KT[4] ‖ KT[16] ‖ KT[12]

2.Get F-Function For each N1...N4

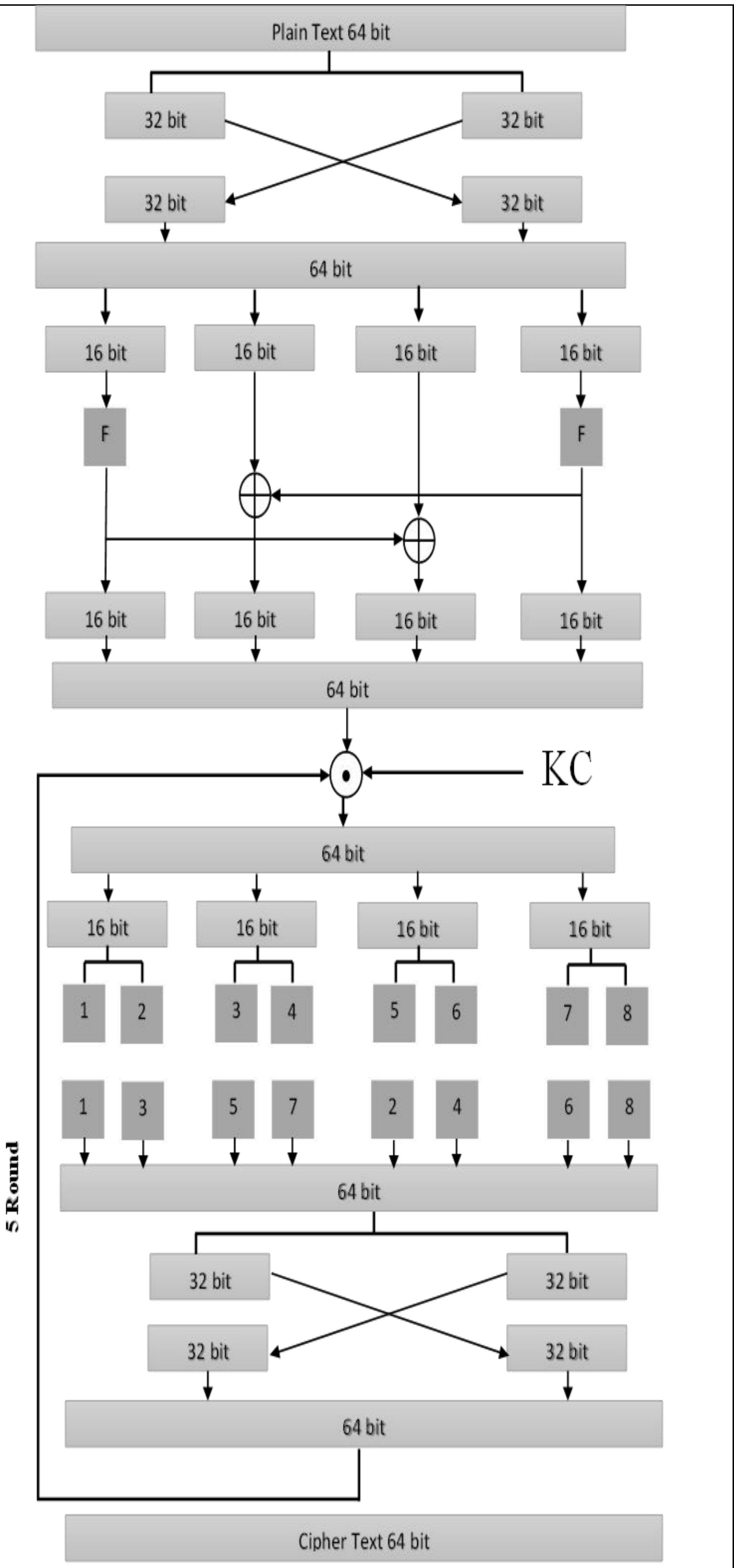
NF1= F-Function(N1)
NF2= F-Function(N2)
NF3= F-Function(N3)
NF4= F-Function(N4)

3.Generate NfK array
NFk1= NF1 ⊕ NF2
NFk2= NFk1 ʘ NF2
NFk3= NFk2 ⊕ NF3
NFk4= NFk3 ʘ NF4

4. Generate Final Key(KC)
KC= NFk1‖ NFk2 ‖ NFk3 ‖ NFk4

5. end

Figure 1: NLBCIT Key Expansion



Algorithm 2 NLBCIT Encryption

```
Input: 64-bits plaintext (PT) and 64-bits key (KC)
Output: 64 ciphertext (CT)
1.  From the input file called plaintext (PT), read the byte values,
2.  Divided the plaintext (PT) into two halves and replaced them together such as,
    A1=PT [1:32]; A2=PT [33:64]
    CT=A1‖ A2
3.  Divided PT into 4 group, each group 16 bits such as,
    CT1 =CT[ 1:16]; CT2=CT[17:32]
    CT3 =CT[33:48]; CT1=CT[49:64]
4.  Get F-Function for all 16 bits of CT such as,
    CTT1= F-Function (CT1) ; CTT4= F-Function (CT4)
    CTT2= CTT4 ⊕ PT2 ; CTT3= CTT1 ⊕ PT3
5.  merge CTT1 , CTT2 , CTT3 , and CTT4 into CT 64 bits such as,
    CT= CTT1 ‖ CTT2 ‖ CTT3 ‖ CTT4
6.  Initializer R with value 1
7.  EX-NOR between ciphertext and key
    CT= CT ʘ KC
8.  Split the block, each 8-bit partition, then switch between them such as,
    CT1 =CT[1:8] ‖ CT[17:24] ‖ CT[33:40] ‖ CT[49:56]
    CT2 =CT[9:16] ‖ CT[25:32] ‖ CT[41:48] ‖ CT[57:64]
9.  merge CT1 And CT2 into CT64 bits such as ,
    CT=CT2 ‖ CT1
    R = R +1
10. If (R <5)
    Go to Step 6
11. Else
    Go to step 12
12. Generate Final Cipher
        CT
```

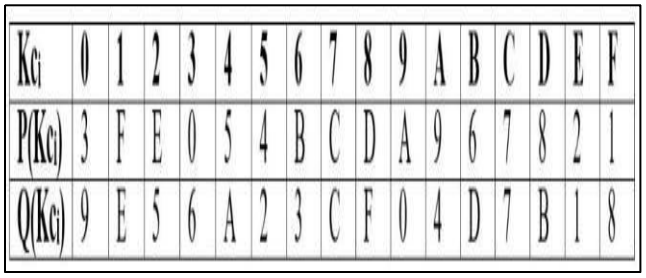
Figure 2: NLBCIT Encryption Process

- **Strategy of encryption**

Encryption is the process of converting data or images from an understandable form into an incomprehensible form using techniques of diffusion and confusion. Conversely, decryption is the process of transforming encrypted data or images back into a comprehensible form for authorized users. The NLBCIT suggested algorithm, described in Figure 2 and detailed in Algorithm 2, performing five rounds of processes to lower the device's energy use. The ciphertext (CT) is made up of the results of several rounds of easy math problems. The next round takes the result from the previous round as its input.

- **F-function**

The F-function algorithm comprises several linear and non-linear processes to ensure that the output bits dynamically depend on the input bits [8, 32, and 33]. This cycle, known as confusion and diffusion, is based on the computations of P ar explained in Figure 4, and detailed in Algorithm 3.



Kci	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
P(Kci)	3	F	E	0	5	4	B	C	D	A	9	6	7	8	2	1
Q(Kci)	9	E	5	6	A	2	3	C	F	0	4	D	7	B	1	8

Figure3 P and Q Table.

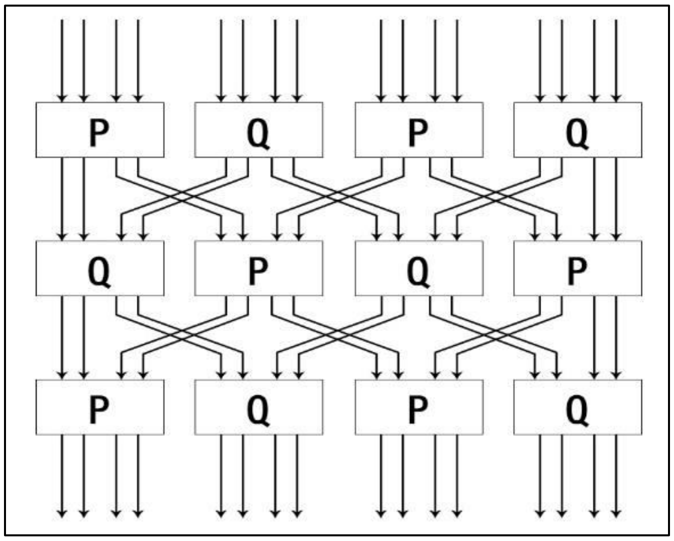


Figure 4: F-Function [5].

Algorithm 3. F-Function

```
Input: 16 bits key (KFT), Four noes of 4 bit
       (C1,C2,C3,C4)
Output: 16 bits key (KFC)
    1. Get Frist Level From F-Function
       SC1=  SBOXP[C1]
       SC2= SBOXQ[C2]
       SC3=  SBOXP[C3]
       SC4= SBOXQ[C4]
    2. Get Second Level F-Function
       SC1=  SBOXQ[SC1]
       SC2= SBOXP[SC2]
       SC3=  SBOXQ[SC3]
       SC4= SBOXP[SC4]
    3. Get Last Level F-Function
       SC1=  SBOXP[SC1]
       SC2=  SBOXQ[SC2]
       SC3=  SBOXP[SC3]
       SC4=  SBOXQ[SC2]
    4. Generate (KFC)
       KFC= SC1|| SC2 || SC3 || SC4
    5. end
```

- **Strategy of decryption**

Decryption Technique It should go without saying that developing an effective encryption strategy requires having a robust decryption procedure. The decryption procedure for the proposed algorithm in this study was constructed in the opposite order as the encryption procedure

VI Simulation and implementation of algorithm

Lightweight encryption and decryption for Internet of Things devices is provided by the proposed NLBSIT algorithm. The software was first created and put into operation using the C programming language and the Dev-C++ integrated development environment, following the guidelines of the GNU General Public License. During this time, it performed its intended functions successfully. Subsequently, the technique was modified to fit the FELICS (Fair Evaluation of Lightweight Cryptographic Systems) framework,ensuring that it produced consistent plaintext and ciphertext results in both C and FELICS evaluations.
FELICS is a tool designed to measure the efficiency of lightweight cryptographic systems (LWC) based on parameters like RAM usage, execution cycles (for both encryption and decryption), and program size. Running on the Ubuntu Linux operating system, FELICS includes a range of standard and widely used lightweight cryptographic algorithms such as TWINE, AES, PICCOLO, and HIGHT. It also enables comparisons between new algorithms and these established standards using various performance evaluation tools, including ARM, MSP, and AVR.

V. SECURITY ANALYSIS

Three distinct algorithms of analysis are used to evaluate the security quality of the suggested solution: Sections A, B, and C include the statistical analysis, evaluation parameters, and attack analysis.
A. Attack Analysis

Linear and Differential Cryptanalysis

Linear and differential cryptanalysis are ineffective against full encryption. The f-function in [8], [32], and [33] provides robust resistance to these attacks. Analyzing two rounds reveals a strong correlation between input and output, making linear approximation impractical. Since the circular transformation is frequently uniform, differential attacks are avoided and each bit is handled consistently.

Weak Key Attacks

If an encryption system generates strong encryption keys, it is deemed strong. The actual key is used to generate the encryption key. The true key is XORed to stop this attack before it starts. The same algorithm is employed by cipher [8] and SIT [33], both of which have been demonstrated to be secure against weak keys. The algorithm makes use of the F-Function in this way; we then add the operations XOR and XNOR to further complicate the process (diffusionand confusion). Thus, the approach stops this kind of attack.

Related Keys Attacks

To decrypt the encrypted text, the attacker makes a series of guesses at several keys. We call these attacks "related attacks" [35]. The attacker wants to uncover the actual concealed keys. Two reasons for this type of attack are the encryption mechanism's latency and the symmetry in the key expansion process. Due to its high diffusion and non-linearity, the suggested algorithm for the main expansion mechanism is built to resist this kind of attack.

Square Attack
Square assault is first described in [8], [32].In the last round of the key, the gatecrasher managed to obtain the last byte. The attack is also conducted eight times in order to obtain the final key. 216 S-box lookups are required in order for an attacker to guess the key and obtain one byte, or 28 by 28 selected plaintexts.

B. Evaluation Parameters

key-sensitive
A strong encryption algorithm is highly sensitive to key changes. Even a single bit difference in the key should drastically alter the decrypted data. Ideally, altering one key bit should result in approximately half of the decrypted data bits changing, as defined by the Strict Avalanche Criterion (SAC)[35]. To visually assess this property, we can decrypt an encrypted image using a key that differs from the correct key by only one bit.

Cycle of Execution
The time taken to encrypt and decrypt the given data is the most important factor in determining how well the algorithm performs. In the smallest period of time, the proposed algorithm seeks to fully secure IoT environments.

Memory Usage
Memory capacity is one of the primary issues restricting resources in Internet of Things devices. The most crucial elements in developing a lightweight encryption algorithm for Internet of Things devices are complex mathematical operations and a higher number of iterations. This is a result of the high memory and hardware resource requirements of these operations. As a result, the recommended algorithm makes use of fewer rounds and basic arithmetic operations. As a result, processing uses little memory and energy.

C. Statistical Analysis

Histogram's analysis
A picture histogram analysis includes a realistic display of pixel power levels and outlines the apparent dispersion of the picture [37].In summary, it improves measurable components of the picture that facilitate perception [38]. Demonstrating the dispersions and disarrays of the encoded data is the main objective of the histogram analysis. In any event, anomalies during document encoding can be found using a histogram. A cryptographic method does not specify sufficient security unless the computed encrypted histogram is in agreement.

Information entropy analysis
To enhance security, encryption methods often introduce redundancy by adding extra data. This additional information, which might include details about the original data or the encryption process itself, increases the complexity for attackers. The amount of this added information is quantified by image entropy. Higher entropy generally correlates with stronger encryption. For instance, an 8-bit grayscale image can have a maximum entropy of 8 bits.Entropy can be calculated using the following formula:

$$Entropy(H) = \sum_{i=0}^{255} P(x_i) \log_2 P(x_i) \qquad (1)$$

where:

- H is the entropy

- P(xi) is the probability that the difference between two adjacent pixels is equal to i

Correlation Analysis

When evaluating a cryptographic algorithm's strength, correlation analysis is a helpful technique. However, "correlation" describes the dependence of two quantities. There is very little connection between the encoded imageand the original image. As a result, the coded image has no relation to the source and is random [39]. The following formulas are used to find the correlation coefficients for both the original and encrypted messages.

$$CorrCoef = \frac{cov(x,y)}{\sqrt{Var}(x) \times \sqrt{Var(y)}} \qquad (2)$$

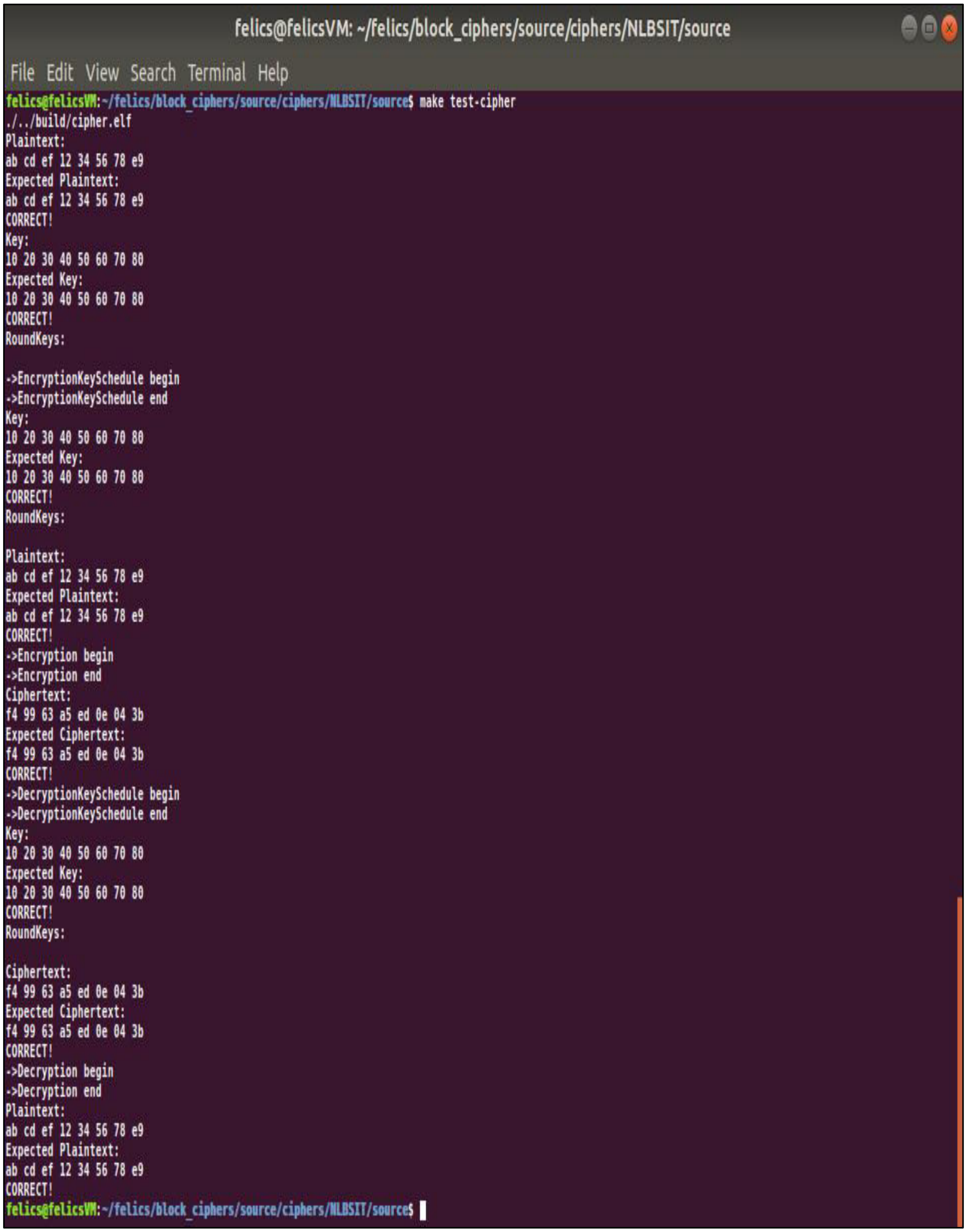
$$Var(x) = \frac{1}{N}\sum_{i=1}^{N}[(x_i - E(x))^2] \qquad (3)$$

$$Cov = \frac{1}{N}[\,(x_i - E(x)) \times (y_i - E(y))](4)$$

where the correlation coefficient CorrCoef and the covariance Cov(x,y) are represented by the pixels x and y. The anticipated value operator is represented by E(x), the variance of the image's pixel value is represented by Var(x), and N is the total number of pixels in the matrix.

VI Functionality analysis

Figure 5 presents the testing and validation process of the recommended computations within the FELICS tool. Table 2 offers a comparison of popular lightweight cryptographic algorithms, including their performance metrics and key features.AVR is used by the tools and devices of the FELICS platform.The proposed NLBCIT was used to conduct functional benchmarking of alternative algorithms, evaluating parameters such as encryption/decryption execution time, RAM footprint, and code size. The histogram in Figure 6 shows how the recommended method uses less memory and requires fewer encoding and decoding cycles. In Figures 7, 8, and 9, the recommended algorithm is shown in red to show the memory comparison and the cycles of encoding and decoding. An avalanche test is used to evaluate the resilience of the suggested algorithm. We selected three images from the database, p635_1_s3(Hand1), p719_1_s3(Hand2) and p643_1_s3(Hand3) [40]. As in Figure 10, a computational trial shows that a single shift of the plaintext key fragment accounts for about 49% of the shape bit changes. The decryption is undetectable if the original keys are changed by even one bit.Furthermore, the number of pixels represented by vertical and horizontal lines on each histogram test. Once encryption is implemented, the best level of protection is represented by a uniform intensity distribution. Finally, as describe in Figure 11, the connection test in Figure 12 shows how encrypted and unencrypted data differ from one another. The encrypted image appears to have a somewhat correlated value, whereas the original image shows a significant correlation of values. Thus, protection increases as correlation decreases.

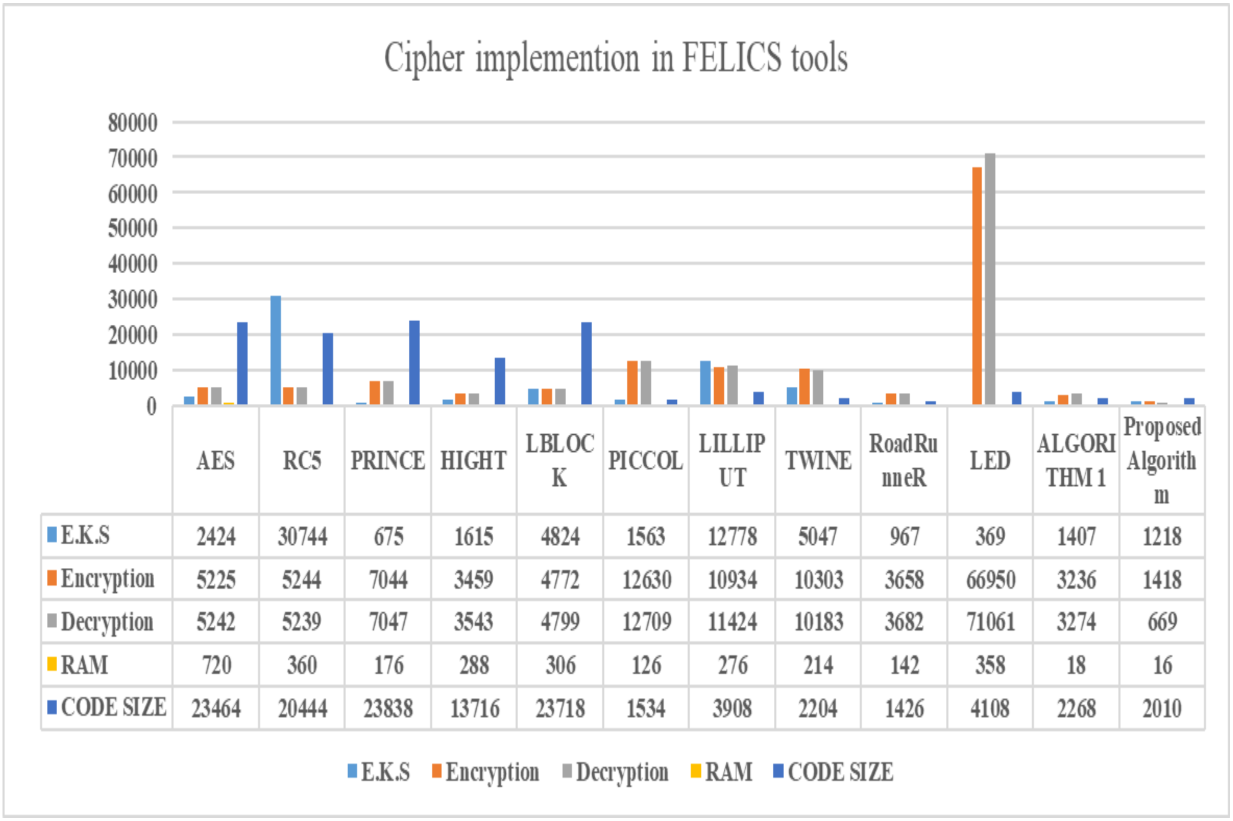

felics@felicsVM: ~/felics/block_ciphers/source/ciphers/NLBSIT/source
File Edit View Search Terminal Help

```
felics@felicsVM:~/felics/block_ciphers/source/ciphers/NLBSIT/source$ make test-cipher
./../build/cipher.elf
Plaintext:
ab cd ef 12 34 56 78 e9
Expected Plaintext:
ab cd ef 12 34 56 78 e9
CORRECT!
Key:
10 20 30 40 50 60 70 80
Expected Key:
10 20 30 40 50 60 70 80
CORRECT!
RoundKeys:

->EncryptionKeySchedule begin
->EncryptionKeySchedule end
Key:
10 20 30 40 50 60 70 80
Expected Key:
10 20 30 40 50 60 70 80
CORRECT!
RoundKeys:

Plaintext:
ab cd ef 12 34 56 78 e9
Expected Plaintext:
ab cd ef 12 34 56 78 e9
CORRECT!
->Encryption begin
->Encryption end
Ciphertext:
f4 99 63 a5 ed 0e 04 3b
Expected Ciphertext:
f4 99 63 a5 ed 0e 04 3b
CORRECT!
->DecryptionKeySchedule begin
->DecryptionKeySchedule end
Key:
10 20 30 40 50 60 70 80
Expected Key:
10 20 30 40 50 60 70 80
CORRECT!
RoundKeys:

Ciphertext:
f4 99 63 a5 ed 0e 04 3b
Expected Ciphertext:
f4 99 63 a5 ed 0e 04 3b
CORRECT!
->Decryption begin
->Decryption end
Plaintext:
ab cd ef 12 34 56 78 e9
Expected Plaintext:
ab cd ef 12 34 56 78 e9
CORRECT!
felics@felicsVM:~/felics/block_ciphers/source/ciphers/NLBSIT/source$
```

Figure 5: Implementation Test of NLBCIT Algorithm on FELICS.

TABLE 2: Results for NLBCIT Cipher with common cipher Implementations on AVR Architecture in FELICS tools.

Cipher	**Device**	**Key Size (bit)**	**Block Size (bit)**	**RAM (byte)**	**Code Size (byte)**	**Encrypted-Key Schedule**	**Decryption (cycles)**	**Encryption (cycles)**
AES	AVR	128	128	720	23464	2424	5242	5225
RC5	AVR	128	64	360	20444	30744	5239	5244
PRINCE	AVR	128	64	176	23838	675	7047	7044
HIGHT	AVR	128	64	288	13716	1615	3543	3459
LBLOCK	AVR	80	64	306	23718	4824	4799	4772
PICCOL	AVR	80	64	126	1534	1563	12709	12630
LILLIPUT	AVR	80	64	276	3908	12778	11424	10934
TWINE	AVR	80	64	214	2204	5047	10183	10303
RoadRunneR	AVR	80	64	142	1426	967	3682	3658
LED	AVR	80	64	358	4108	369	71061	66950
ALGORITHM 1	AVR	64	80	18	2268	1407	3274	3236
PROPOSED ALGORITHM	**AVR**	**64**	**64**	**16**	**2010**	**1218**	**669**	**1418**



	AES	RC5	PRINCE	HIGHT	LBLOCK	PICCOL	LILLIPUT	TWINE	RoadRunneR	LED	ALGORITHM 1	Proposed Algorithm
E.K.S	2424	30744	675	1615	4824	1563	12778	5047	967	369	1407	1218
Encryption	5225	5244	7044	3459	4772	12630	10934	10303	3658	66950	3236	1418
Decryption	5242	5239	7047	3543	4799	12709	11424	10183	3682	71061	3274	669
RAM	720	360	176	288	306	126	276	214	142	358	18	16
CODE SIZE	23464	20444	23838	13716	23718	1534	3908	2204	1426	4108	2268	2010

Figure 6: Comparison analysis in terms of Encryption/Decryption (Cycles), RAM, and Code Size,

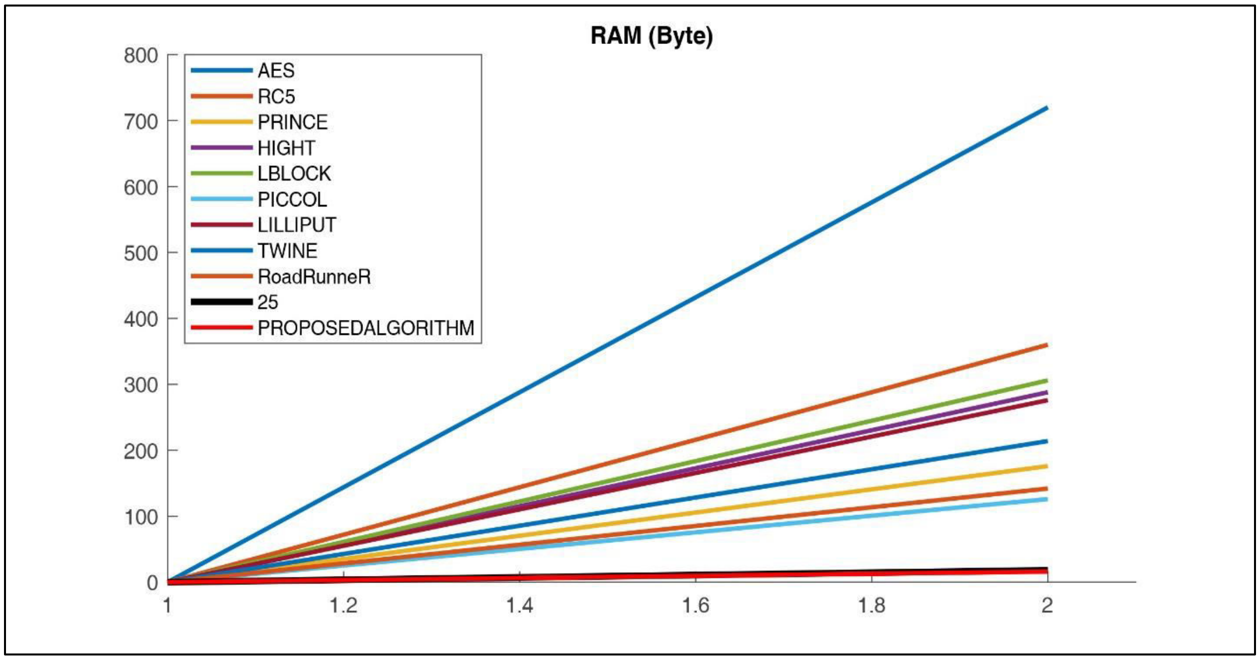


Figure 7: Curve of RAM in byte for different block implementation.

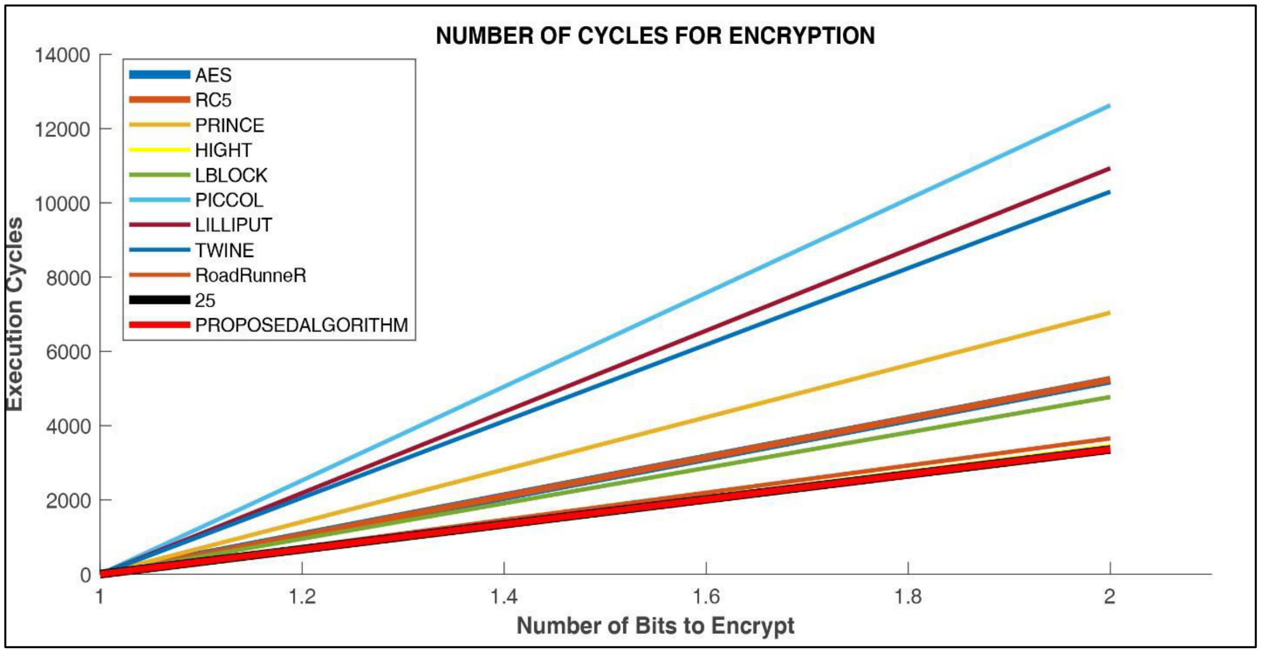


Figure 8: Curve of execution time(cycle) for multiple ciphers in varying block for encryption.

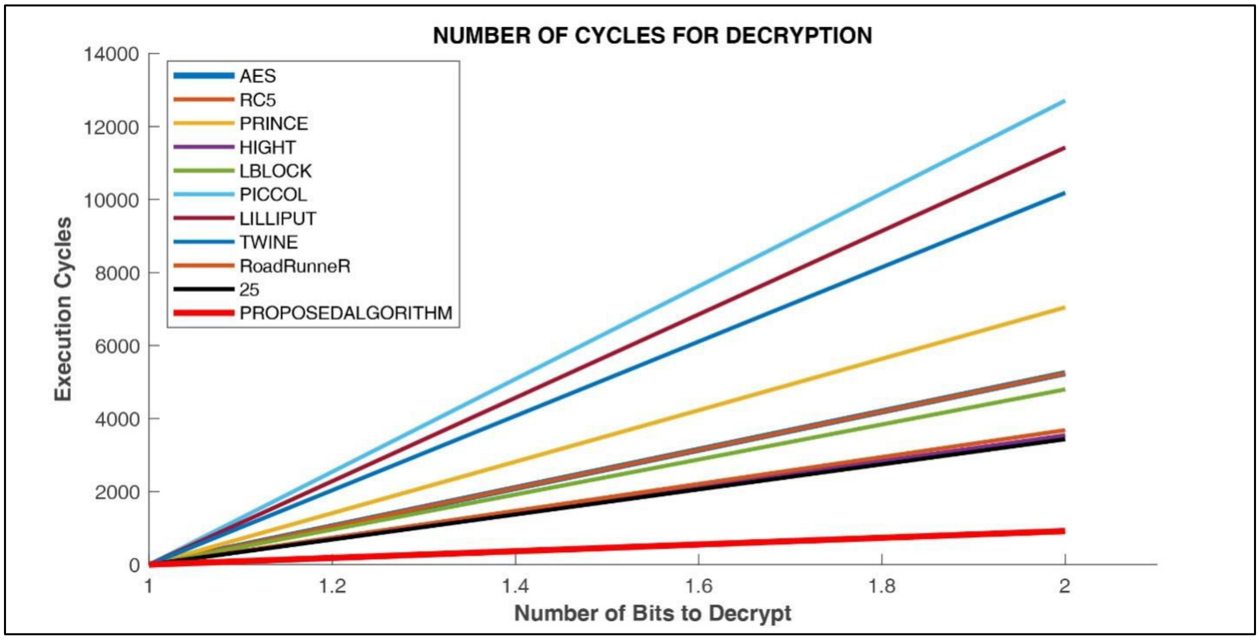


Figure 9: Curve of execution time(cycle) for multiple ciphers in varying block for

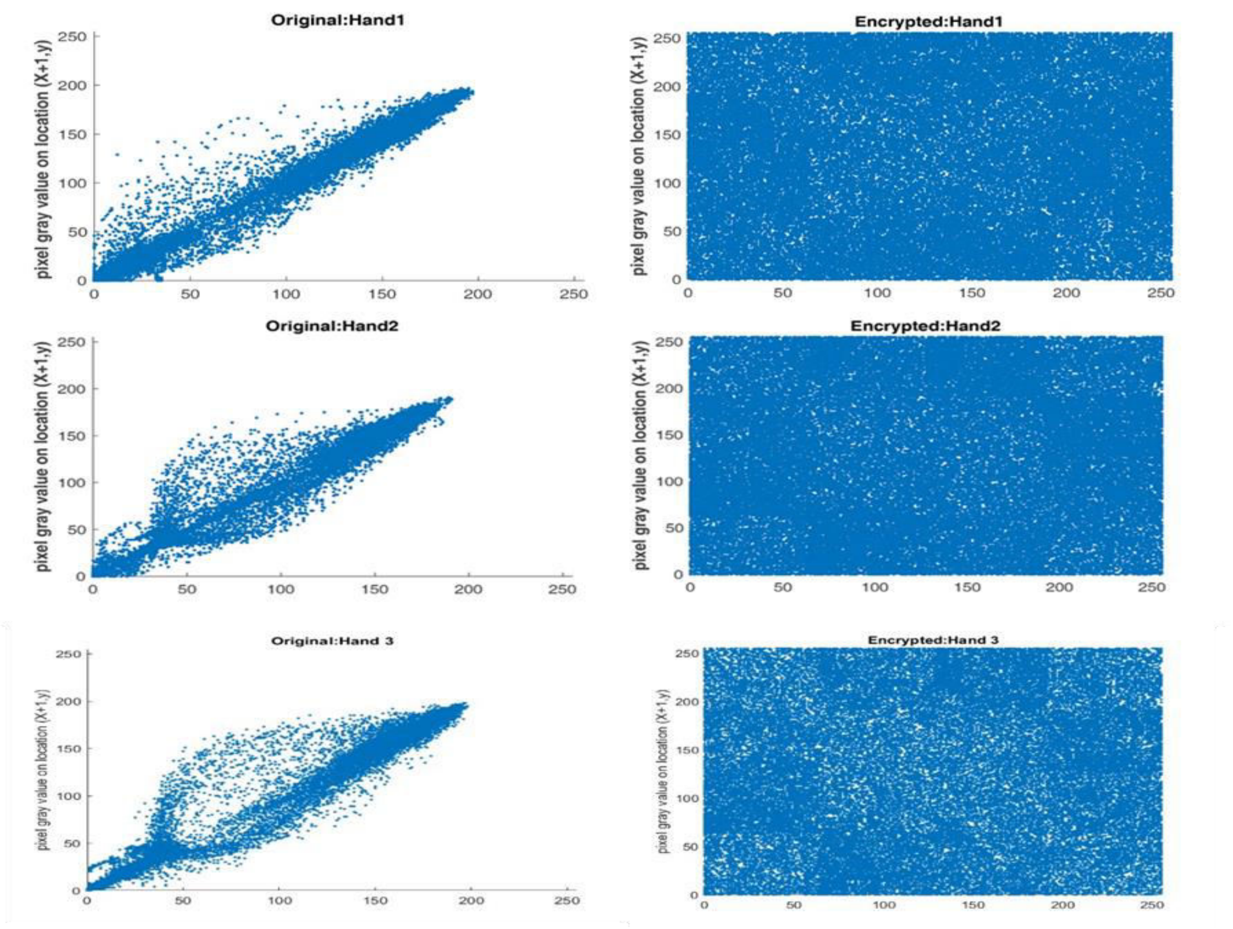


Figure 10: Key Sensitivity of hand image

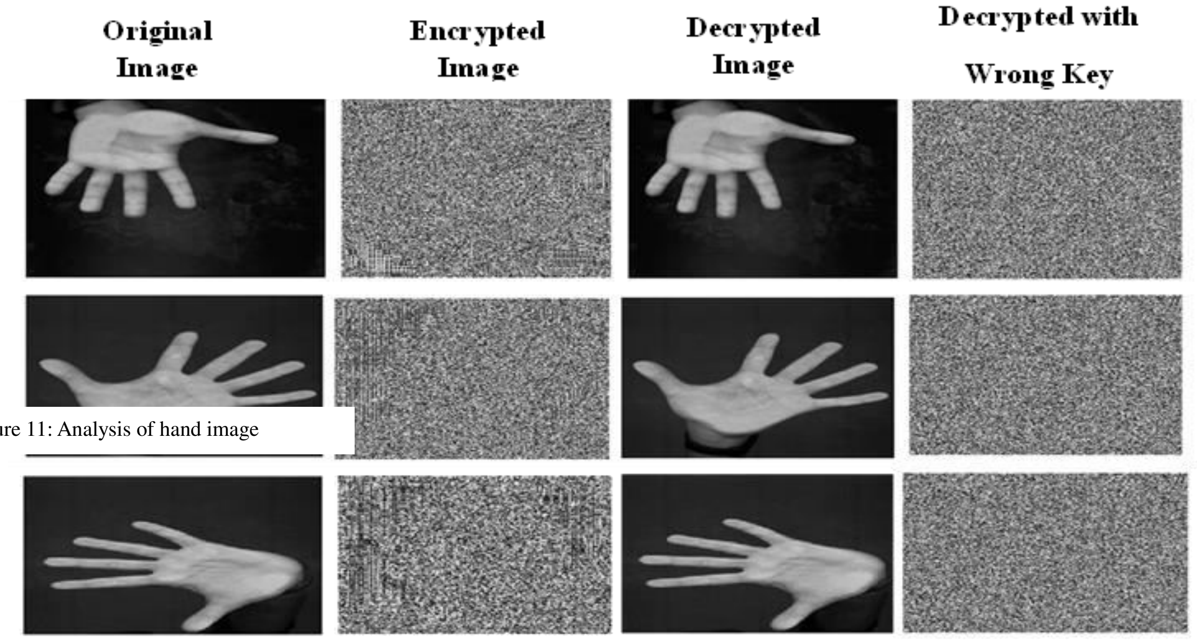


Figure 11: Analysis of hand image

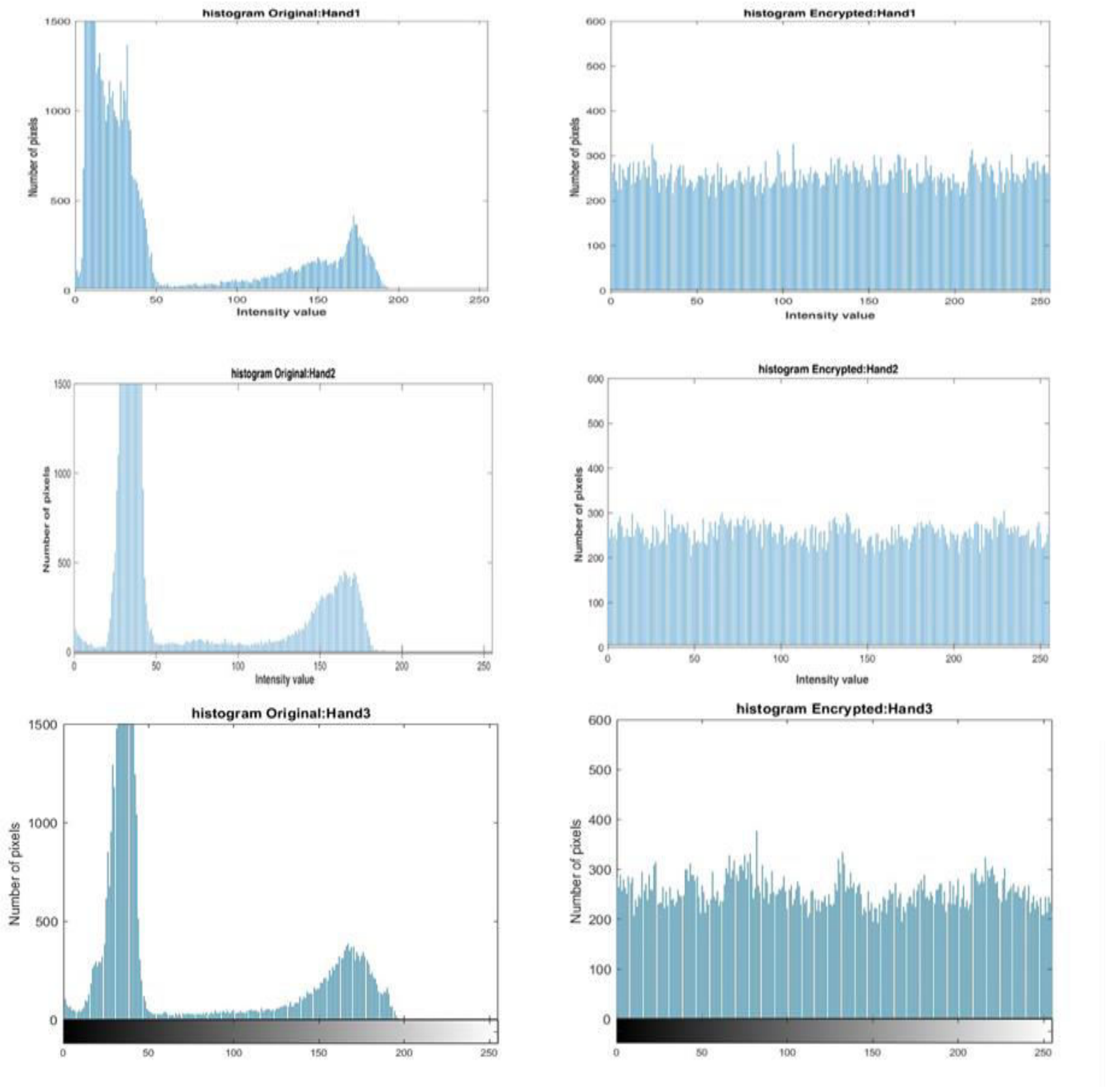


Figure 12: Hand image of correlations for encrypted/decrypted.

CONCLUSION

This study creates and implements a new LW unit for low-resource Internet of Things computers. The recommended encryption improves data security by combining the advantages of the Feistel and SPN architectures with the additional concept of a linear box. Additionally, entropy, histogram, and avalanche tests were used to evaluate the robustness of the approach. The NLBCIT algorithm demonstrates lower energy consumption and reduced memory usage compared to other algorithms during encryption and decryption cycles, making it well-suited for encryption in IoT devices.

References

[1] Prajakta P. Deshpande, "IoT Based Fleet Management Systems : A Review", International Journal of Computer Sciences and Engineering, Vol.7, Issue.5, pp.436-443, 2019.

[2] Hui, T. K., Sherratt, R. S., & Sánchez, D. D. (2017). Major requirements for building Smart Homes in Smart Cities based on Internet of Things technologies. Future Generation Computer Systems, 76, 358-369.

[3] Anjum Sheikh, "IoT Based Security System for Smart Homes", International Journal of Computer Sciences and Engineering, Vol.07, Special Issue.11, pp.35-38, 2019.

[4] Zhou, G., Liu, Z., Shu, W., Bao, T., Mao, L., & Wu, D. (2017). Smart savings on private car pooling based on internet of vehicles. Journal of Intelligent & Fuzzy Systems, 32(5), 3785-3796.

[5] Majumdar, A., Debnath, T., Sood, S. K., & Baishnab, K. L. (2018). Kyasanur forest disease classification framework using novel extremal optimization tuned neural network in fog computing environment. Journal of medical systems, 42(10), 187.

[6] Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., & Uhsadel, L. (2007). A survey of lightweight-cryptography implementations. IEEE Design & Test of Computers, 24(6), 522-533.

[7] Al-ahdal, A. H., & Deshmukh, N. K. A Systematic Technical Survey of Lightweight Cryptography On lot Environment.

[8] Al-ahdal, Abdulrazzaq HA, and Nilesh K. Deshmukh, and Galal A. AL-Rummana. "A Robust Lightweight Algorithm for Securing Data in Internet of Things Networks." .Under Publication.

[9] Biswas, A., Majumdar, A., Nath, S., Dutta, A., & Baishnab, K. L. (2020). LRBC: a lightweight block cipher design for resource constrained IoT devices. Journal of Ambient Intelligence and Humanized Computing, 1-15.

[10] Singh, S., Sharma, P. K., Moon, S. Y., & Park, J. H. (2017). Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions. Journal of Ambient Intelligence and Humanized Computing, 1-18.

[11] Hong, D., Lee, J. K., Kim, D. C., Kwon, D., Ryu, K. H., & Lee, D. G. (2013, August). LEA: A 128-bit block cipher for fast encryption on common processors. In International Workshop on Information Security Applications (pp. 3-27). Springer, Cham.

[12] Wheeler, D. J., & Needham, R. M. (1994, December). TEA, a tiny encryption algorithm. In International Workshop on Fast Software Encryption (pp. 363-366). Springer, Berlin, Heidelberg.

[13] Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., & Shirai, T. (2011, September). Piccolo: an ultra-lightweight blockcipher. In International Workshop on Cryptographic Hardware and Embedded Systems (pp. 342-357). Springer, Berlin, Heidelberg.

[14] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., ... & Vikkelsoe, C. (2007, September). PRESENT: An ultra-lightweight block cipher. In International workshop on cryptographic hardware and embedded systems (pp. 450-466). Springer, Berlin, Heidelberg.

[15] Albrecht, M. R., Driessen, B., Kavun, E. B., Leander, G., Paar, C., & Yalçın, T. (2014, August). Block ciphers–focus on the linear layer (feat. PRIDE). In Annual Cryptology Conference (pp. 57-76). Springer, Berlin, Heidelberg.

[16] Shirai, T., Shibutani, K., Akishita, T., Moriai, S., & Iwata, T. (2007, March). The 128-bit blockcipher CLEFIA. In International workshop on fast software encryption (pp. 181-195). Springer, Berlin, Heidelberg.

[17] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2015, June). The SIMON and SPECK lightweight block ciphers. In Proceedings of the 52nd Annual Design Automation Conference (pp. 1-6).

[18] Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E. B., Knudsen, L. R., Le, G., ... & Rombouts, P. (2012). PRINCE–A Low-latency Block Cipher for Pervasive Computing Applications Full version.

[19] De Canniere, C., Dunkelman, O., & Knežević, M. (2009, September). KATAN and KTANTAN—a family of small and efficient hardware-oriented block ciphers. In International Workshop on Cryptographic Hardware and Embedded Systems (pp. 272-288). Springer, Berlin, Heidelberg.

[20] Guo, J., Peyrin, T., Poschmann, A., & Robshaw, M. (2011, September). The LED block cipher. In International workshop on cryptographic hardware and embedded systems (pp. 326-341). Springer, Berlin, Heidelberg.

[21] A. Alahdal, D. S. Kumar, A. H. A. AlAhdal, M. Krichen, and H. Almansour, "Securing Data Based on Lightweight Algorithm for Internet of Things Networks," Smart Innovation, Systems and Technologies, Jan. 01, 2024. https://link.springer.com/chapter/10.1007/978-981-99-7711-6_12

[22] A. Hassan, "An Effective Lightweight Cryptographic Algorithm to Secure Internet of Things Devices," Lect. Notes Networks Syst., vol. 358 LNNS, no. 11, pp. 403–419, 2022.

[23] P. P., M. M., S. K.P., and M. S. Sayeed, "An Enhanced Energy Efficient Lightweight Cryptography Method for various IoT devices," ICT Express, vol. 7, no. 4, pp. 487–492, Dec. 2021.

[24] Koo, B., Roh, D., Kim, H., Jung, Y., Lee, D. G., & Kwon, D. (2017, November). CHAM: a family of lightweight block ciphers for resource-constrained devices. In International Conference on Information Security and Cryptology (pp. 3-25). Springer, Cham.

[25] Bansod, G., Pisharoty, N., & Patil, A. (2017). BORON: an ultra-lightweight and low power encryption design for pervasive computing. Frontiers of Information Technology & Electronic Engineering, 18(3), 317-331.

[26] Patil, J., Bansod, G., & Kant, K. S. (2017, February). LiCi: A new ultra-lightweight block cipher. In 2017 International Conference on Emerging Trends & Innovation in ICT (ICEI) (pp. 40-45). IEEE.

[27] Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., & Verbauwhede, I. (2015). RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Science China Information Sciences, 58(12), 1-15.

[28] Karakoç, F., Demirci, H., & Harmancı, A. E. (2013, May). ITUbee: a software oriented lightweight block cipher. In International Workshop on Lightweight Cryptography for Security and Privacy (pp. 16-27). Springer, Berlin, Heidelberg.

[29] Yang, G., Zhu, B., Suder, V., Aagaard, M. D., & Gong, G. (2015, September). The simeck family of lightweight block ciphers. In International Workshop on Cryptographic Hardware and Embedded Systems (pp. 307-329). Springer, Berlin, Heidelberg.

[30] Guo, J., Peyrin, T., Poschmann, A., & Robshaw, M. (2011). The LED block cipher. In Cryptographic Hardware and Embedded Systems–CHES 2011 (pp. 326-341). Springer Berlin Heidelberg.

[31] A. H. A. Al-Ahdal, G. A. AL-Rummana, G. N. Shinde, and K. D. Nilesh, "Security Analysis of a Robust Lightweight Algorithm for Securing Data in Internet of Things Networks," .Under Publication.

[32] Barreto, P. S. L. M., & Rijmen, V. (2000). The Khazad legacy-level block cipher. Primitive submitted to NESSIE, 97, 106.

[33] Usman M, Ahmed I, Aslam MI, Khan S, Shah UA. SIT: a lightweight encryption algorithm for secure internet of things. arXiv preprint arXiv:1704.08688. 2017 Apr 27.

[34] D. Dinu, A. Biryukov, J. Großschädl, D. Khovratovich, Y. L. Corre, L. Perrin, "FELICS – Fair Evaluation of Lightweight Cryptographic Systems", University of Luxembourg, July 2015.

[35] Biham, E. (1994). New types of cryptanalytic attacks using related keys. Journal of Cryptology, 7(4), 229-246.

[36] A. Webster and S. E. Tavares, "On the design of s-boxes," in Conference on the Theory and Application of Cryptographic Techniques. Springer, 1985, pp. 523–534.

[37] Kanso, A., & Ghebleh, M. (2018). An efficient lossless secret sharing scheme for medical images. Journal of Visual Communication and Image Representation, 56, 245-255.

[38] Hodeish, M. E., Bukauskas, L., &Humbe, V. T. (2019). A new efficient TKHC-based image sharing scheme over unsecured channel. Journal of King Saud University-Computer and Information Sciences.

[39] Ahmad, M., Doja, M. N., & Beg, M. S. (2018). Security analysis and enhancements of an image cryptosystem based on hyperchaotic system. Journal of King Saud University-Computer and Information Science.

[40] Magalh˜aes, F., Oliveira, H. P., Matos, H., Campilho, A.: HGC 2011 - Hand Geometric Points Detection Competition Database, http://www.fe.up.pt/~hgc2011/.