LSA: A LIGHTWEIGHT SYMMETRIC ENCRYPTION ALGORITHM FOR RESOURCE-CONSTRAINED IOT SYSTEMS

Amita Shah[1], Sanjay Shah[2], Hiren Patel[3], Namit Shah[4]

•

[1]Ph.D Scholar, Computer/IT Engineering, Gujarat Technological University, Gujarat, India
[2]Professor & Head, Computer Engineering Dept., Government Engineering College, Rajkot, Gujarat, India
[3]Principal, Vidush Somany Institute of Technology and Research, Sarva Vidyalaya Kelavani Mandal, Kadi, Gujarat, India.
[4]Student, Computer Engineering Dept., L D College of Engineering, Ahmedabad, Gujarat, India

Abstract

Today, Internet of Things (IoT) systems are being employed in a wide variety of domains, such as education, healthcare, industrial equipment automation, etc. With gigabytes of data being generated and processed by even the average IoT system, securing this generated data is crucial task. It requires a low-cost, high-performance encryption system for constrained IoT systems. The Advanced Encryption Standard (AES) is widely used for many cryptographic domains because of its strong security characteristics. AES is designed for general-purpose symmetric encryption algorithm but there is a need for a lighter algorithm that is specifically tuned for the needs of IoT devices with limited computation capabilities. Aim: This paper is proposing Lightweight Symmetric Algorithm (LSA) as a faster and lighter alternative to the standard AES-128 for IoT applications. The primary objective of its design was to minimize the time and memory usage required for encryption and decryption processes while retaining the strong security characteristics. Method: The research also demonstrates the comparative analysis of LSA and AES based on efficiency and resource usage. It also proves the difficulty of performing a successful brute force attack, confusion and diffusion properties, and avalanche criterion satisfiability are identical for AES and LSA algorithms. Findings: The comparison analysis of LSA and AES suggests a 14.68% lower memory usage for encryption and decryption as well as more than a 50% decrease, on average, in the required time for encryption or decryption of differently-sized files consisting of the same 128 bit data blocks. The comparisons and empirical observations show that AES and LSA are both almost identical in terms of their security characteristics such as the difficulty of performing a successful brute force attack, confusion and diffusion properties, and avalanche criterion satisfiability. Conclusion: The proposed LSA algorithm is compared with various available lightweight cipher technologies with respect to time, memory, and security properties suggests the suitability of LSA for resource constrained IoT devices with strong security requirements.

Keywords: IoT, Data security, Cryptography, Lightweight Algorithms, Security Encryption.

1. INTRODUCTION

The Internet of Things (IoT) is becoming an increasingly significant aspect of daily life as more and more devices with digital identities are connected to the Internet. The IoT paradigm is based on the connection between widely used and extremely diverse networked "things" like sensors, actuators, smartphones, etc., whose widespread use is due to recent advances in communication, sensor technologies, networking capabilities, mobile devices, cloud computing, etc. Data security is now a major necessity for many organisations. Security and privacy needs must be met because the IoT devise are integrated in users' daily lives [1-2].

However, due to the multiple standards and communication technologies involved, the IoT does not directly support conventional security measures. The term "lightweight cryptography" refers to a branch of cryptography that aims to create algorithms for use on hardware without the necessary resources like memory, power, and operational capacity to carry out the operation [3]. Only a few reliable hybrid cryptosystems are available to protect IoT smart devices. The objective is to create hybrid cryptosystems that can match the high performance demands of these constrained environments while possessing similar encryption capabilities. Even though many other new lightweight algorithms have been developed, there is always potential for development in terms of security and overhead reduction. [4-6].

The encryption of IoT data can be achieved by two ways, Symmetric and Asymmetric Cryptography. Symmetric encryption techniques are effective at protecting data, but communicating a secret key requires a separate mechanism. The key distribution issue is solved by asymmetric encryption techniques, although they are slower and consume far more resources than symmetric encryption. According to NIST, information security, like any other information technology management system, relies on three fundamental aspects: confidentiality, availability, and integrity. [8-10].

In this research, LSA aims to enhance the time and memory efficiency of AES without compromising the security features. The remaining sections of this paper are organized as follows: Section 2 provides an overview of current lightweight symmetric algorithms suitable for IoT environments. Section 3 outlines the design of the LSA algorithm being proposed. Section 4 showcases the performance and security analysis of the implemented algorithms, along with a comparison based on specific key parameters. At last, in Section 5, the paper is concluded by proving the security requirements of IoT system along with communication efficiency, resource utilization and strong encryption methodology.

2. RELATED WORKS

This section presents an overview of the existing lightweight symmetric encryption algorithms and offers a comparative analysis among them. Additionally, it investigates the suitability of the AES algorithm for enhancement, specifically to cater to the requirements of resource-constrained devices.

2.1 Lightweight Cryptography

Embedded systems, Internet of Things, and mobile computing devices are used across many industries [13], which is lacking the security mechanism for resource-constrained network. Lightweight cryptography is a trade-off between communication efficiency and data security. Due to its applicability to IoT systems with limited battery life, space, and memory size, lightweight cryptography has gained popularity [11-12]. Different approaches can be taken to implement lightweight cryptographic techniques, with some relying on software while others on hardware. Hardware-based lightweight cryptography aims to address performance limitations such as device size and power consumption. On the other hand, software-based lightweight cryptography focuses

on reducing CPU/memory usage, calculation complexity, and energy/power consumption [14]. Lightweight cryptography can be achieved through various methods, such as modifying or enhancing existing algorithms or creating new algorithms with lightweight characteristics. [15].

2.2. Overview of AES Algorithm

The AES algorithm [20] is built upon the SPN (substitution-permutation network) structure and incorporates key features such as high sensitivity to initial round and control parameters, random-like behaviors, and simplicity [18]. In the SPN structure, even slight modifications in the initial state and parameter configurations within the round function can result in significant and unpredictable changes in the final state [18][20]. AES operates on 128-bit (16-byte) blocks for both encryption and decryption of data. Figure 1 illustrates the basic block diagram of the AES algorithm. AES supports three key sizes: 128, 192, or 256 bits. For 128-bit keys, AES employs 10 rounds, for 192-bit keys it uses 12 rounds, and for 256-bit keys, it utilizes 14 rounds. Each round, except the final one, incorporates the SubBytes, ShiftRows, AddRoundKey, and MixColumns operations [17], [18], and [19]. It is worth noting that in this context, the term AES specifically refers to AES-128.

Key Expansion Routine:

The Key Expansion Routine of the standard AES-128 is used without any modification, which generates 11 keys of 128 bit from one single encryption/decryption key. It generates an array of 11 keys from the original seed key, which now becomes key 0. For our variations with a reduced number of rounds (7, instead of 10), only the first 8 keys are used. As each key consists of 4 words of 4 bytes each, we need 44 words in total for 10 rounds of encryption or decryption.

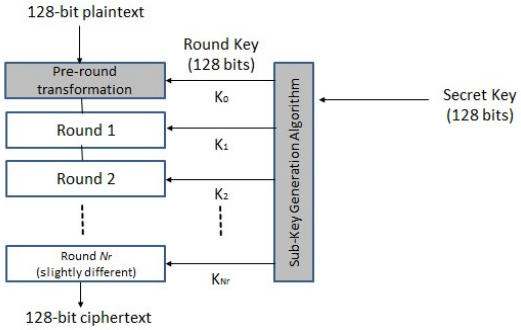

Figure 1: *Block diagram of AES-128 [16]*

AES-128 Key Expansion

```
for (i = 0 ; i < 4 ; i++)
        w[i] = key[i];
for (i = 4; i < 44 ; i++)
temp = w[i – 1];
if (i Mod 4 == 0)
temp = SBox ( RotWord ( temp ) )  Xor  Rcon[i / 4];
w[i] = w[i – 4]  Xor temp
```

RotWord(): Performs right shift on the word by 1 byte (0, 1, 2, 3 => 1, 2, 3, 0).
SBox(): Substitution from the Rijndael S-Box.
Xor: Bitwise Xor.
Rcon: The round constant array consists of successive powers of 2, one value for each round.
1 word = 4 bytes.
4 words -> 16 bytes -> the key for one round.

2.3. Study of other lightweight algorithms

Examples of some lightweight symmetric algorithms include AES [21], CAST-128 [22], PRESENT [23], TEA [24], HIGHT [25], BCC [26], MCBB [27], RC5 [28] etc. With the least amount of resource usage possible, lightweight cryptography strives to provide proper security levels. TEA's initial release was followed by a subsequent version that included additional features aimed at improving its security. Meanwhile, Block TEA was introduced as a complement to XTEA, and it operates on blocks of any size, unlike the original's 64-bit blocks [24] TEA exhibits some vulnerability, primarily its susceptibility to equivalent keys. In other words, each key is interchangeable with three other keys, reducing the effective key size to just 126 bits. Consequently, TEA is not suitable for use as a cryptographic method. Several researchers, including in [28], [29], and [30], focused on reducing the complexity of common algorithms, and based on their findings, these approaches can be applied in an IoT environment. Since the S-Box is crucial to AES and causes confusion during the encryption process, numerous researchers, including those in [31] and [32], have attempted to develop new S-Boxes to replace the old ones to increase the security of the AES algorithm. The IoT often uses a high number of resource-constrained nodes, necessitating the adoption of lightweight cryptographic primitives [33]. PRESENT employs bit-oriented permutations, which make it hardware-oriented and less suitable for software implementations. Bit permutations can be easily accomplished in hardware through straightforward wiring, whereas software implementations struggle to achieve similar performance. The FELICS (Fair Evaluation of Lightweight Cryptographic Systems) benchmarking framework is used to assess the performance of PRESENT when executed on microcontroller software environments. However, the results of software-only implementations may be significantly slower due to the inherent hardware orientation of the algorithm [34]. A hybrid approach was employed in [35] to merge the symmetric cipher AES, asymmetric cipher RSA, and the hashing function MD5 to provide confidentiality, data integrity, and authentication. However, the use of AES in processing occupies a considerable amount of ROM and RAM, while the MD5 algorithm is vulnerable to differential attacks and the RSA key requires a significant amount of memory for processing.

Table 1: *Comparison of Lightweight Algorithms for IoT Devices*

Lightweight Algorithms	Structure	No of Rounds	Key Size	Block Size
AES	SPN	10	128	128
PRESENT	SPN	32	80	64
TEA	Feistel	32	128	64
HEIGHT	GFS	32	128	64
RC5	ARX	20	16	32

To address the limitations of the cryptographic models outlined in Tab 1, the research explores a variety of suggested cryptosystems that incorporate various mathematical calculations. Subsequently, it also proposes a resilient and secure lightweight symmetric algorithm that offers efficient protection for IoT smart devices, as detailed in the upcoming sections.

3. THE PROPOSED ALGORITHM: LSA

In this paper, we propose a lightweight, secure, and fast symmetric encryption algorithm – Lightweight Symmetric Algorithm (LSA), to provide confidentiality in resource constrained IoT Devices. LSA can encrypt and decrypt data more quickly than AES.

In the context of the IoT environment, the importance of time and memory usage is on par with security considerations. This research focuses on reducing the time complexity of the algorithm while maintaining its security measures. The proposed algorithm aims to provide a lower-

complexity encryption method compared to AES, making it suitable for resource-constrained wireless devices. Additionally, it offers enhanced resilience against attacks compared to PRESENT and TEA. The proposed lightweight symmetric encryption algorithm adopts a substitution-permutation structure and builds upon the widely-used AES algorithm. By reducing the number of rounds and replacing the mixcolumn operation with junction jumping in the proposed LSA, performance improvements are achieved without compromising the security properties of the algorithm. Further details of the LSA are discussed in the following subsections.

Considering the constraints and requirements of IoT, there is a need to improve the AES algorithm in terms of time and energy consumption. With this objective in mind, we conducted tests and evaluations to identify the most time-consuming parts of the AES algorithm, which could be potential areas for optimization.

Each round of the AES algorithm involves four operation calls: Substitution, Shift Rows, Mix Columns, and Add Round Key. While AES can be implemented efficiently and cost-effectively in hardware [36], its software implementation tends to be more computationally intensive in terms of processing time.

Analysis of Modification in AES Algorithm:

To improve the performance of the algorithm, we developed three different versions of each operation of AES with the following variations. The research shows modified compute-intensive operations to make them lighter and examined nine more versions of modified AES.

Three different versions of SubBytes (Disabling ShiftRows and MixColumns).

- SubBytes_v0> Substitution bytes with 100% Substitution (Original)
- SubBytes_v1> Substitution bytes with 50% Substitution (checkerboard pattern)
- SubBytes_v2> Substitution bytes with 25% Substitution (only 1 in every four elements in the block)

depicts execution time analysis of different variants of SubBytes operation from which SubBytes_v2 is a relatively lightweight operation as per the performance.

Three different versions of ShiftRows (Disabling SubBytes and MixColumns)

- ShiftRows_v0> keep the 1st row unchanged and shift 2nd, 3rd and 4th row by 1,2 and 3 bytes subsequently (Original)
- ShiftRows_v1> keep the 1st and 3rd row unchanged and shift the 2nd and 4th row by 1 byte
- ShiftRows_v2> Let us keep the 1st, 2nd and 3rd row unchanged and shift the 4th row by 1 byte

The experiment shows the execution time analysis of different variants of ShiftRows operation from which SubBytes_v2 is a relatively lightweight operation.

Three different versions of MixColumns (Disabling ShiftRows and SubBytes).

- MixColumns_v0> Matrix Multiplication with the constant matrix (Original)

MixColumns_v1> Matrix Addition with the constant

- matrix
- MixColumns_v2> Matrix Subtraction with the constant matrix

This demonstrates that MixColumns_v1 is a lightweight component as per the experimental analysis of MixColumns variants. The constant matrix, here, refers to the AES Multiplication Matrix [19]. For V0, the Inverse Multiplication Matrix [19] is also required, while for V1 and V2, the same matrix is used during decryption as well.

Based on the various performed variations we have developed AES with combination of the fastest versions of all the 3 operations (SubBytes, ShiftRows & MixColumns) along with AddRoundKey) which incorporates V2_SubBytes (Substitution bytes with 25% Substitution), V1_ShiftRows (keeps the 1st and 3rd row unchanged and shift the 2nd and 4th row by 1 byte), V1_MixColumn (Matrix Addition with the pre-defined constant and simple XORing with the key in the AddRoundKey operation. It definitely reduces execution time, but at the same time, compromises certain level of

security.

Optimization Operations: Lightweight Security Algorithm for IoT

Reducing the complexity of operations in the proposed algorithm can potentially compromise its security level. To address this, the next improvement focuses on enhancing the security measures. The analysis of this test primarily serves the purpose of benchmarking and facilitating future experimentation. In the context of this research, the proposed algorithm is denoted as LSA-v1, which integrates the fastest versions of all the stages in AES002E

To investigate the effect of various operations on the encryption time, the experiment removes the operations one by one from the encryption process in AES.

First of all, to find heavy components, AES is modified and created the following variations:

- Only keeping ShiftRows and MixColumns and disabling SubBytes
- Only keeping SubBytes and MixColumns and disabling ShiftRows
- Only keeping SubBytes and ShiftRows and disabling MixColumns

The experiment depicts that execution time will be decreased if we remove MixColumns operation from AES.

The Mix Columns operation is generally the most resource-intensive operation in AES, and its removal leads to an overall improvement in the algorithm's execution time. The results demonstrate a significant reduction in encryption time for 1024-byte data, decreasing from 70 milliseconds to 15 milliseconds. Additionally, the Shift Rows operation is identified as the second most time-consuming operation after Mix Columns. Consequently, it becomes necessary to either remove or optimize the Shift Rows operation to make the algorithm more lightweight.

For this reason, the research extended experiment by further removing two operations.

- Only keeping SubBytes and disabling ShiftRows and MixColumns
- Only keeping ShiftRows and disabling SubBytes and MixColumns
- Only keeping MixColumns and disabling ShiftRows and SubBytes

In the optimization mentioned earlier, if all the operations are executed in isolation, it becomes apparent that MixColumns consumes the most time compared to other operations, reaffirming its heavyweight nature. Therefore, to enhance execution time performance, the most effective solution is to exclude MixColumns from the main core of the encryption operation. To generate lighter versions of the AES algorithm and for the replacement of the mixcolumn operation, we have introduced one more operation - Junction Jumping, which plays an important role to make the algorithm lightweight in terms of processing needs and to achieve a certain level of security.

Junction Jumping

This stage's main objective is propagating change from one byte to the next, thus introducing interdependence and linkage. Unlike the Mix Columns Operation, which is inherently exponential, this operation is linear, and primarily uses one of the most cost-effective CPU operations, bitwise XOR [29]. Fig. 2 shows the overall functioning of the newly introduced Junction Jumping operation on the 128 bits of input data. 'U' and 'L' refer to the upper and lower nibbles (4 bit groups) of all the bytes. For this stage, we consider the current state as an array of 16 bytes rather than a 4x4 matrix. Its main objective is propagating change from one byte to the next, thus introducing interdependence and linkage. Unlike the MixColumns Operation, which is inherently exponential, this operation is linear. It primarily uses one of the most cost-effective CPU operations, bitwise XOR. The proposed algorithm incorporates all the standard operations except MixColumns. It replaces MixColumns operation with Junction Jumping. The research achieved an improvement in the security characteristics, whereas the time complexity of the algorithm increases. Fig. 2 exhibits the process of AES with JJ, which for the research is referred as LSA-v2.

The Jumping Junction Algorithm

Prev ← Initial State of the 16 byte Word.
Next ← The Resultant State.
For 'i' in range(16):
Next[i] = Lower_Nibble(Prev[i]) + Xor(Upper_Nibble(Prev[i]), Lower_Nibble(Prev[i – 1]));
Since, only one half of every byte changes, there are no actual additional memory requirements as Prev is just a temporary state.
The process is also cyclic (1→2, ..., n – 1→n, n→1).

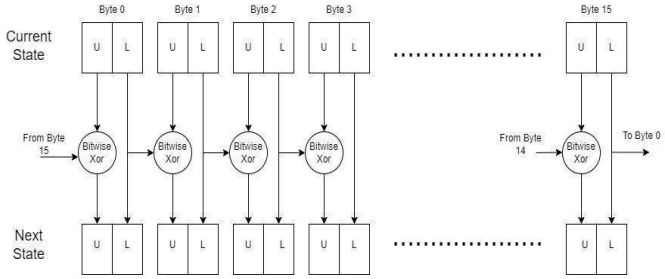

Figure 2: *Junction Jumping Operation*

The evaluations indicate that by replacing mixcolumn with JJ, time consumption has only decreased by 15%. To further explore the experimental possibilities, the proposed algorithm removes the shiftrow operation and reduces the number of rounds from 10 to 7, while maintaining the same block size of 128 bits. This modification improves the time complexity; however, it comes at the cost of compromised confusion and diffusion characteristics. The findings of the investigation into time consumption are depicted in Figure 7. It is important to note that while the round key generation process in the proposed algorithm resembles that of AES, it possesses inferior security properties. As a result, the enhanced version of LSA is denoted as LSA-v3.

In order to broaden the range of the experiments, and to maintain the trade-off between security and performance, a new additional operation - Parity Transformation is introduced which improves confusion and diffusion characteristics and is performed just after Round 0 during encryption, that is, once per the encryption / decryption process for a block. Parity Transformation plays an important role to make the algorithm lightweight in terms of processing needs and to achieve a certain level of security. It adds non-linearity to the system so that we can improve security measures, specifically the Average Avalanche Criteria that improved significantly from 32.5% to 43.33%.

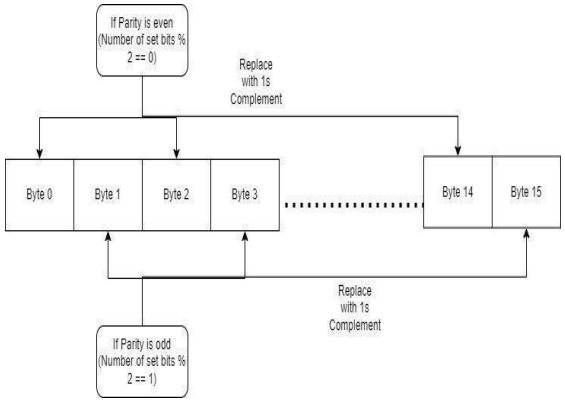

Figure 3: *Parity Transformation Operation*

Parity Transformation

This stage is performed only once and is incorporated in Round 0 to add non-linearity to the system so that we can improve security measures. It works on the principle that if an even number of bits are flipped, the resultant parity remains the same. This ensures that the transformation remains reversible. Fig. 3 shows the functioning of it.

The Steps: Parity Transformation Operation

_____

Run a loop to find the parity of the word.
Run another loop and 1s complement those bytes whose indices Mod 2 = The_Original_Parity.
Mod here refers to the modulo operation (the remainder).
The_Original_Parity will be 1 if the number of bits in the input block were odd; otherwise it would be 0.
Basically, if the parity is 1, the bytes at odd indices will be flipped and if it is even, then the bytes at even indices will be flipped.

_____

The research has advanced with the reduced number of rounds (seven), but because of the Parity Transformation operation there is a huge improvement in the confusion and diffusion characteristics of the algorithm. In this version it reached the required result, and so the paper is proposing it as LSA - Lightweight Symmetric Algorithm. The process of proposed LSA is displayed in Fig. 4 for visualization and clarity purposes. Execution time to process average 20MB data is 1.8s, 1.3s, 1.5s, 0.9s and 0.95s for AES, AES-fastest operations(LSA-v1), AES-JJ(LSA-v2) , AES-JJ-7 rounds(LSA-v3) and LSA, respectively. Fig. 5 and Fig. 6 illustrate the performance comparison of LSA with experimental versions. It has also compared with various lightweight symmetric algorithms like PRESENT and TEA, respectively; which shows that LSA performs better among all in terms of execution time. The proposed encryption process in LSA is designed and executed as shown in Fig. 4.

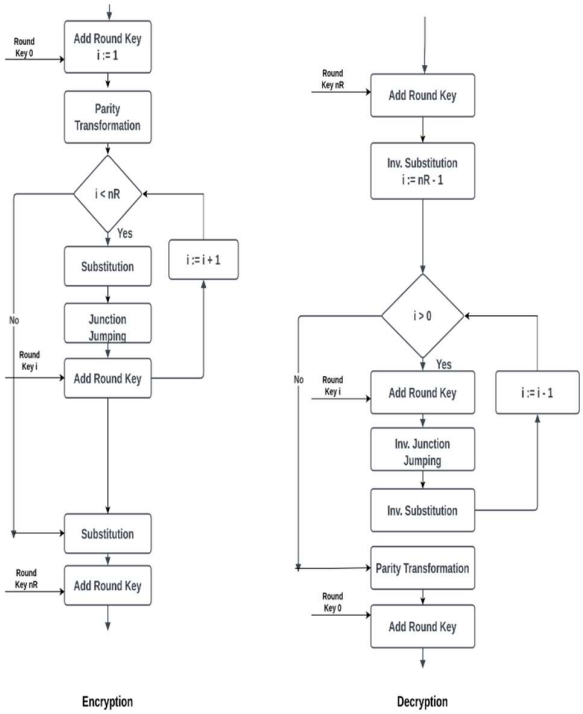

Figure 4: *Process of LSA*

The encryption process in LSA is outlined and illustrated in Figure 4. It closely resembles the encryption process in AES with some notable differences. In LSA, a parity transformation operation is conducted on the initial state during Round-0, and the resulting output becomes the input for the subsequent stages. In each encryption round of LSA, a round key is applied to encrypt a data block, similar to the Add Round Key operation in AES. However, instead of using MixColumns and ShiftRows as in AES, LSA employs the junction jumping operation as replacements.

LSA: Lightweight Security Algorithm

LSA-v1: The research has created a new AES variation with each of the 3 standard stages being replaced by a version of them from above with the 'best' performance characteristics. This version is used mostly for the purposes of benchmarking and further experimentation. While exhibiting impressive time complexity and average runtime characteristics, it had dismal security properties.
LSA-v2: In this version, a single modification has made to the Standard AES algorithm, replacing the high-cost Mix Columns Stage, which happens to be the most time-expensive stage, with the Junction Jumping Stage. This greatly improved the security characteristics but came at the cost of significantly longer runtimes and greater time complexity than LSA-v1.
LSA-v3: In this version, the number of rounds is reduced by 3 (7 instead of 10) because of expected early obfuscation (of a satisfactory level) and to do away with the mixcolumns and shiftrows stages in LSA-v2. Instead, it is replaced with junction jumping. This, as expected, came at the cost of the algorithm's security properties.

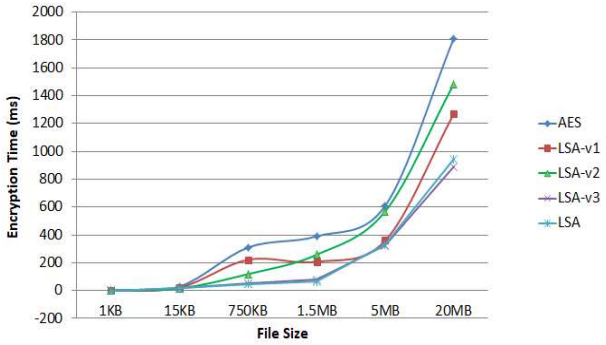

Figure 5: *Performance of different Versions of Lightweight Symmetric Algorithm (LSA)*

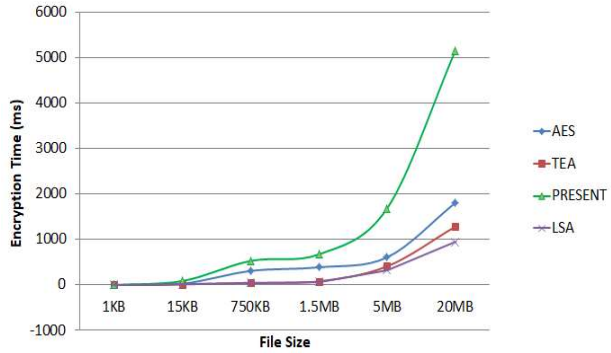

Figure 6: *Comparison of LSA with other lightweight symmetric algorithms*

LSA: This is the final algorithm that is proposed as an optimised solution. It is introduced in one new stage, Parity Transformation, into LSA-v3 for better confusion and diffusion characteristics (specifically the Average Avalanche Criteria that improved significantly from 32.5% to 43.33%).

4. IMPLEMENTATION RESULT ANALYSIS

Cryptographic algorithms are commonly implemented as hardware modules on sensor nodes. However, for off-the-shelf nodes lacking hardware security implementation, software implementation or hardware/software co-design approaches are considered suitable alternatives. It is often impractical to add new hardware circuitry to these nodes, making software implementation and evaluation of encryption algorithms more feasible. In software implementations, the primary design objectives are to minimize memory usage and optimize processor throughput and power efficiency. Consequently, the focus lies on reducing memory occupation while achieving improved performance and power savings. The forthcoming sections will delve into the analysis and results of performance and security metric comparisons between LSA, AES, PRESENT, and TEA algorithms. These discussions will shed light on the achieved performance and security levels of these algorithms.

4.1. Performance Metrics

Performance metrics hold significance in the comparison of various cipher algorithms. Consequently, it is essential to establish a consistent platform and mutually agreed-upon metrics. As part of our study, we have successfully implemented the proposed LSA algorithm and conducted a comparative analysis with existing algorithms, namely AES, PRESENT, and TEA. Given the limitations imposed by memory, power, and processing resources in wireless nodes, our evaluation primarily focuses on measuring time and memory consumption parameters. These metrics allow us to assess the overall performance of the implemented encryption algorithms in the context of wireless node constraints.

1) Encryption Time

The performance of an algorithm improves as its speed increases. Based on the findings presented in Figure 7, it is observed that PRESENT requires the longest time to execute the encryption operation, followed by AES. However, by reducing the number of rounds from 10 to 7 and replacing the ShiftRows and MixColumn operations with Junction Jumping, the LSA algorithm demonstrates a substantial improvement in time complexity. In this section, a more detailed analysis of the LSA algorithm's security properties will be conducted.

2) Memory Usage

IoT devices, particularly sensors, often have limited storage capacity. This storage space is primarily allocated to the operating system and the data collected by the sensors. Consequently, there is little room available for implementing security algorithms. Due to these constraints, it is not feasible to employ complex encryption algorithms on IoT nodes. In this study, we evaluate the RAM and ROM usage of each of the aforementioned algorithms. ROM usage refers to the space occupied by permanent code on the sensor nodes. On the other hand, RAM usage pertains to the space required for runtime code, including the storage of the stack and variables for intermediate calculation results. Since RAM directly impacts sensor performance during runtime, it holds greater significance than ROM [38].

The set up of a test bed for experiments and implemented these algorithms in raspberry pi to observe the usage of memory. Fig. 7 shows the memory use records for the encryption and decryption techniques. Compared to other algorithms, the PRESENT algorithm uses the most ROM. LSA uses less RAM and ROM than AES, PRESENT, and TEA do but less RAM and ROM than TEA do as well. The implementation of the RAM involves sophisticated technology and is more expensive than the ROM memory [38]. In LSA compared to AES, round-key generation is reduced, resulting in decreases in ROM and RAM utilisation of 13.15% and 14.68%, respectively. According to [39], low-cost implementations call for up to 4 KB ROM and 8 KB RAM, and lightweight implementations call for up to 4 KB ROM and 256 bytes RAM.

4.2. Security Metrics

The energy consumption and latency of the encryption operation are increased when the packet size is increased [37]. As the data packets transmitted by sensor nodes are typically small in size, the performance evaluation metrics focus on these small-sized packets. For the purpose of evaluating performance, we consider 10,000 randomly generated blocks, each sized at 128 bits. The encrypted outputs produced by each algorithm are then used for the analysis of security metrics.

1) Key Size (Length)

The size of the initial key plays a crucial role in determining the security level of encryption algorithms, particularly in their resistance against brute force attacks. The longer the key size, the more secure the encryption algorithm becomes. However, longer key sizes also result in increased key processing time and memory space requirements. Thus, selecting an appropriate key size depends on the desired security levels and the available resources. According to Table 1, PRESENT and TEA have a key size of 64 bits, while AES and LSA employ a key size of 128 bits. Among the algorithms discussed in this paper, LSA has a significantly lower likelihood of a successful brute force attack due to its larger key size compared to the other algorithms.

2) Information Entropy Analysis

Information entropy is a measure of the probability distribution of random events and can be utilized to assess diffusion characteristics. A higher level of system diffusion corresponds to a greater entropy value. In the analysis of entropy, random events can be represented as sequence values in bytes. In our case, the ideal entropy value is 4, which indicates that the values of the encrypted string are fully distributed [40]. To calculate the system entropy, we consider each nibble in the output as a unique symbol, resulting in a total of 24 possible symbols. The Shannon entropy value reflects the prevalence of diffusion, with a maximum possible value of 4 in our setup. Table 2 presents the results of the security are parameters for LSA, which comparable to those of AES. These security parameter results are obtained from the evaluation of 10,000 randomly generated data blocks.

Shannon Entropy equation:

$$H(X) = -\sum_{i=1}^{n}(p_i . log_2(p_i)) \quad (1)$$

Where P_i is the probability of every symbol.

Table 2: *Comparison of LSA and AES in term of Security metrics*

Algorithm	Hamming Distance	Shannon Entropy	Avalanche Effect
AES	50%	3.611	49%
LSA	50%	3.612	46.66%

In the results achieved similarly with the character frequency distribution domain metric, AES and LSA are the two algorithms with almost equivalent entropy.

3) Diffusion and Confusion Analysis

The design of ciphers incorporates two fundamental principles: diffusion and confusion [41]. These principles aim to complicate the statistical relationship between the key and the ciphertext, ensuring that each input bit affects multiple ciphertext bits [42]. Confusion and diffusion serve to prevent the deduction of secret data and secret keys, and their effectiveness disrupts statistical and other cryptanalytic methods. Confusion obscures the connection between the ciphertext and the key, while diffusion conceals the relationship between the plaintext and the ciphertext. Furthermore, the properties of diffusion and confusion in AES and LSA will be investigated in relation to text sensitivity. This investigation will consider metrics such as completeness, the avalanche effect, and the strict avalanche effect. These metrics provide insights into the extent to which AES and LSA exhibit diffusion and confusion properties.

4) Avalanche Effect

The avalanche effect metric measures the extent to which a change in a single input bit influences the output of an encryption algorithm. A secure algorithm is expected to exhibit an avalanche effect where a single bit change in the input causes approximately half of the output bits to change, reflecting the desired confusion and diffusion properties. In the multi-order avalanche effect analysis, LSA demonstrates slightly less growth compared to AES. This decline in metric growth in LSA can be attributed to the reduction in the number of rounds, which results in lower energy consumption. The nth order avalanche criterion quantifies the change in the output when n bits are altered in the input. For AES, the avalanche metric values remain at 49% for the 1st order, 2nd order, and 3rd order avalanche criteria. On the other hand, LSA exhibits metric values of 49% for the 1st order, 42% for the second order, and 49% for the 3rd order avalanche criteria. The average avalanche criterion value for LSA is extremely close to the optimal value of 50%, indicating a high level of diffusion and confusion in the algorithm.

5) Hamming Distance

The Hamming Distance metric is employed to determine the number of bits that change when data is transformed. When more than 50% of the bits are flipped, the complement percentage is considered. This is because any value above 50% (denoted as x%) is equivalent to (100 - x)%. Thus, 50% represents the maximum possible value. The Hamming Distance metric is utilized to assess confusion and observe the degree of obfuscation in the relationship between the input and output. Both AES and LSA exhibit results that align with the optimal expectations for secure algorithms in terms of the Hamming Distance metric.

4.3. Trade-Off Points

The fair trade-off between security and performance is crucial in identifying effective solutions based on specific applications. In the case of the proposed LSA algorithm, modifications have been made to AES to improve time complexity and memory utilization. While there are other encryption algorithms like PRESENT and TEA that are designed for energy-limited systems, they are susceptible to certain types of security attacks. In comparison, LSA aims to provide better security in specific areas compared to PRESENT and TEA. LSA demonstrates a higher level of security against statistical attacks, as indicated by the tested security metrics including entropy, balance analysis, avalanche effect, and Hamming distance, similar to AES. Moreover, LSA's use of a nonlinear structure in the substitution box and the Junction Jumping operation enhances its resistance against differential attacks. Although AES may have slightly stronger security characteristics, LSA's security properties are very close and only marginally weaker. As depicted in Figure 7, LSA exhibits significantly lower time and memory overhead compared to the AES algorithm, while maintaining nearly the same level of security. Furthermore, when compared to low-energy algorithms such as PRESENT and TEA, the security improvements offered by LSA outweigh the associated increase in time and memory consumption. Therefore, LSA can be considered as a suitable encryption algorithm for battery-operated sensors and other resource-constrained IoT nodes, offering robust security properties.

5. CONCLUSION

Encryption techniques play a crucial role in safeguarding data privacy in IoT devices. However, due to the limited resources of IoT nodes, it is essential to use algorithms that are energy and memory efficient. In this paper, we propose LSA, a lightweight symmetric encryption algorithm that is based on the well-known AES algorithm. Our initial observations indicate that LSA exhibits improved resistance against specific differential and statistical attacks compared to algorithms like PRESENT and TEA. This is mainly attributed to the inclusion of nonlinear elements and a larger key space in

LSA. The algorithm leverages junction jumping and parity transformation stages to reduce the overall operation time when compared to AES. In terms of performance, LSA demonstrates a significant decrease in encryption and decryption execution time, averaging over 50% improvement compared to AES, for a variety of file sizes. Considering the resource consumption and performance of sensor network nodes, LSA appears to be a more suitable choice than AES. In our security attack analysis, we evaluated the avalanche effect, which measures the sensitivity of algorithms to changes in plaintext. The results indicate that AES exhibits slightly higher sensitivity than LSA. Additionally, LSA shows a marginally higher vulnerability to differential attacks compared to AES. To further enhance the proposed algorithm, future studies could focus on conducting performance analyses of the security metrics at different rounds and stages within LSA.

References

[1] Hernández-Ramos, J. L., García-Teodoro, P., Díaz-Verdejo, J. E., Luna-Ramírez, F., García-Hernández, Á., & Sandoval Orozco, A. L. (2018). Protecting personal data in IoT platform scenarios through encryption-based selective disclosure. *Computer Communications*, 130:20-37.

[2] Naru, E. R., Saini, H., & Sharma, M. (2017). A recent review on lightweight cryptography in IoT. In 2017 I*nternational Conference on I-SMAC IoT in social, mobile, analytics and cloud) (I-SMAC)* (pp. 1-6)

[3] Xin, M. (2015). A mixed encryption algorithm used in internet of things security transmission system. In *2015 International Conference on Cyber-enabled Distributed Computing and Knowledge Discovery* (pp. 221-225). IEEE.

[4] Goyal, T. K., & Sahula, V. (2016). Lightweight security algorithm for low power IoT devices. In 2016 *International Conference on Advances in Computing, Communications and Informatics* (ICACCI) (pp. 2074-2079). IEEE.

[5] Ragab, A., He, Y., Khan, M. I., Tao, X., & Alghathbar, K. (2019). Robust hybrid lightweight cryptosystem for protecting IoT smart devices. In Y. Pan, J. Chen, T.-H. Kim, X. Li, & R. Niedermeier (Eds.), *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage* (pp. 155-170). Springer.

[6] Singh, S., Sharma, A., Singh, D., Tyagi, S., & Rodrigues, J. J. (2017). Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions. *Journal of Ambient Intelligence and Humanized Computing*, 8(1):1-18.

[7] Pérez, S., Fuentes, E., & Roa, L. M. (2018). A lightweight and flexible encryption scheme to protect sensitive data in smart building scenarios. *IEEE Access*, 6:11738-11750.

[8] Dhanda, S. S., Singh, B., & Jindal, P. (2020). Lightweight cryptography: a solution to secure IoT. Wireless Personal Communications, 112(3):1947-1980.

[9] Yousuf, T., Malik, H., Abdullah, A., Alzahrani, A. I., & Alghathbar, K. (2015). Internet of things (IoT) security: current status, challenges and countermeasures. *International Journal for Information Security Research (IJISR),* 5(4):608-616.

[10] Thabit, F., Alhomdy, S., Al-Ahdal, A. H., & Jagtap, S. (2021). A new lightweight cryptographic algorithm for enhancing data security in cloud computing. Global Transitions Proceedings, 2(1): 91-99.

[11] Rao, V., & Prema, K. V. (2021). A review on lightweight cryptography for Internet-of-Things based applications. *Journal of Ambient Intelligence and Humanized Computing*, 12:8835-8857.

[12] Prakasam, P., Sivaramakrishnan, S., Iqbal, A. T. M., et al. (2021). An enhanced energy efficient lightweight cryptography method for various IoT devices. *ICT Express*, 7(4):487-492.

[13] Eceiza, M., Flores, J. L., & Iturbe, M. (2021). Fuzzing the internet of things: a review on the techniques and challenges for efficient vulnerability discovery in embedded systems. *IEEE Internet of Things Journal*, 8(13):10390-10411.

[14] Prakasam, P., Sivaramakrishnan, S., Iqbal, A. T. M., et al. (2021). An enhanced energy

efficient lightweight cryptography method for various IoT devices. *ICT Express*, 7(4):487-492.

[15] Roy, S., Rawat, U., & Karjee, J. (2019). A lightweight cellular automata based encryption technique for IoT applications. *IEEE Access*, 7:39782-39793.

[16] Fadhil, M. S., Khalaf, Z. A., Al-Sultani, Z. M., & Dheyab, W. R. (2020). A New Lightweight AES Using a Combination of Chaotic Systems. In 2020 *1st. Information Technology To Enhance e-learning and Other Applications (IT-ELA)* (pp. 1-6). IEEE.

[17] Naif, J. R., Abdul-Majeed, G. H., & Farhan, A. K. (2019). Secure IOT system based on chaos-modified lightweight AES. In *2019 International Conference on Advanced Science and Engineering (ICOASE) (pp. 1-8)*. IEEE.

[18] Salman, R. S., Farhan, A. K., & Shakir, A. (n.d.). Lightweight modifications in the *Advanced Encryption Standard* (AES) for IoT applications: a comparative survey.

[19] Chatterjee, R., Chakraborty, R., & Mondal, J. K. (2019). Design of lightweight cryptographic model for end-to-end encryption in IoT domain. *IRO Journal on Sustainable Wireless Systems*, 1(4):215-224.

[20] Lee, A. (1999). NIST Special Publication 800-21, Guideline for Implementing Cryptography in the Federal Government. *National Institute of Standards and Technology*.

[21] Sadkhan, S. B., & Salman, A. O. (2018). Fuzzy logic for performance analysis of AES and lightweight AES. In 2018 *International Conference on Advanced Science and Engineering (ICOASE)* (pp. 1-6). IEEE.

[22] Muthavhine, K. D., & Sumbwanyambe, M. (2021). Modifying CAST algorithm in order to Increase Encryption Strength and to Reduce Memory Limitations. In *2021 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)* (pp. 1-6). IEEE.

[23] Chatterjee, R., & Chakraborty, R. (2020). A modified lightweight PRESENT cipher for IoT security. In 2020 *International Conference on Computer Science, Engineering and Applications (ICCSEA)* (pp. 1-6). IEEE.

[24] Shamala, L. M., Varghese, J., Chacko, V., et al. (2021). Lightweight cryptography algorithms for internet of things enabled networks: An overview. *Journal of Physics: Conference Series*, 1717(1):012072.

[25] Hui, Y., Xu, L., Zhang, Z., et al. (2021). BCC: Blockchain-based collaborative crowdsensing in autonomous vehicular networks. *IEEE Internet of Things Journal*, 9(6):4518-4532.

[26] Farajallah, M. (2022). Lightweight chaotic block cipher for IoT applications. *Journal of Theoretical and Applied Information Technology*, 100(15):2879-2889.

[27] Sharafi M, Eslami M, Safkhani M, et al. A low power cryptography solution based on chaos theory in wireless sensor nodes. *IEEE Access*. 2019; 7:8737-8753.

[28] Shahzadi, Romana, et al. "Chaos based enhanced RC5 algorithm for security and integrity of clinical images in remote health monitoring." *IEEE Access 7* (2019): 52858-52870.

[29] Yao X, Chen Z, Tian Y. A lightweight attribute-based encryption scheme for the Internet of Things. *Future Generation Computer Systems*. 2015; 49:104-112.

[30] Sevin A, Mohammed AAO. A survey on software implementation of lightweight block ciphers for IoT devices. *Journal of Ambient Intelligence and Humanized Computing*. 2021:1-15.

[31] Panahi P, Dehghantanha A, Conti M, et al. Performance evaluation of lightweight encryption algorithms for IoT-based applications. *Arabian Journal for Science and Engineering*. 2021;46(4):4015-4037.

[32] Alshammari BM, Alsulaiman FM, Alsulaiman MB, et al. Implementing a symmetric lightweight cryptosystem in highly constrained IoT devices by using a chaotic S-box. *Symmetry. 2021*;13(1):129.

[33] Guo Y, Li L, Liu B. Shadow: A lightweight block cipher for IoT nodes. *IEEE Internet of Things Journal*. 2021;8(16):13014-13023.

[34] Nath S, Som S, Negi MC. Cryptanalysis of a novel bitwise xor rotational algorithm and

security for IoT devices. *International Journal of Knowledge-based and Intelligent Engineering Systems.* 202

[35] Jang K, Lee J, Lee J, Kim K. Grover on GIFT. *Cryptology ePrint Archive*. 2020.

[36] Harini A, Krishnamurthy R, Venkatesan R, Murugan A. A novel security mechanism using hybrid cryptography algorithms. In: *Proceedings of the 2017 IEEE International Conference Electrical Instrumentation and Communication Engineering (ICEICE)*; 2017. p. 1-5.

[37] Zhao W, Ha Y, Alioto M. AES architectures for minimum-energy operation and silicon demonstration in 65nm with lowest energy per encryption. In 2015 *IEEE International Symposium on Circuits and Systems (ISCAS)* 2015 May 24 (pp. 2349-2352). IEEE.

[38] Tech Differences. (2017). Difference Between RAM and ROM Memory (With Comparison Chart)—*Tech Differences*. Accessed: Apr. 28, 2018. [Online]. Available: https://techdifferences.com/difference-between-ram-and-rom-memory.html

[39] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, ''Lightweight cryptography for embedded systems—A comparative analysis,'' in *Data Privacy Management and Autonomous Spontaneous Security. Berlin, Germany: Springer*, 2014, pp. 333–349.

[40]Anand K, Bianconi G. Entropy measures for networks: Toward an information theory of complex topologies. *Physical Review E. 2009* Oct 13;80(4):045102.

[41] X.-Y. Wang and Q. Yu, ''A block encryption algorithm based on dynamic sequences of multiple chaotic systems,'' *Commun. Nonlinear Sci. Numer. Simul*., vol. 14, no. 2, pp. 574–581, 2009.

[42] X.-J. Tong, Z. Wang, Y. Liu, M. Zhang, and L. Xu, ''A novel compound chaotic block cipher for wireless sensor networks,'' *Commun. Nonlinear Sci. Numer. Simul*., vol. 22, nos. 1–3, pp. 120–133, 2015.