Enhancement of Tiny Encryption Algorithm for Resource-Constrained WSNs Using Four Connected Additive Fibonacci Generators

Atheer Hussein Zyara[1], Hakeem Imad Mhaibes[2,*] and Qahtan Makki Shallal[3]

[1]*College of Health & Medical Technology, Middle Technical University, Baghdad, Iraq*
[2]*Technical Institute of Kut, Middle Technical University, Baghdad, Iraq*
[3]*Management Technical College of Basra, Southern Technical University, Basrah, Iraq*
E-mail: hakeem.emade@mtu.edu.iq
[*]*Corresponding Author*

Received 05 September 2023; Accepted 15 December 2023;
Publication 09 April 2024

Abstract

The usage of wireless sensor networks (WSN) is widespread in industries where data security is crucial. Due to the energy and computational limits of WSN, the cryptographic protocols designed for it must be as computationally cheap as feasible. The components of these protocols, such as the random number generator, are subject to the same constraints. For such resource-constrained devices, several lightweight encryption techniques have been created. One of the most efficient lightweight ciphers is Tiny Encryption Algorithm (TEA). TEA uses a few lines of source code that are based on Feistel. It is however susceptible to attacks utilizing equivalent and related key attacks. In order to address TEA's key vulnerabilities, a modification is suggested in this paper that focuses on key creation. Four connected Additive

Journal of Cyber Security and Mobility, Vol. 13_3, 349–368.
doi: 10.13052/jcsm2245-1439.1331

Fibonacci Generators (AFGs) make up the structure, which addresses the security vulnerability by using a unique key each round. Performance evaluation was assessed using three statistical tests: avalanche effect, randomness analysis, and completeness testing. Through experimental results, ATEA outperforms TEA by an average of 51.68 % to 47.51 % for the avalanche effect, and 51.95 % to 48.36 % for the completeness test, and satisfies all the NIST requirements. Results of advanced security measurements indicate that, ATEA can be used to secure WSN devices.

Keywords: Wireless sensor network, constraint devices, lagged fibonacci generator, lightweight cryptography, random number generator.

1 Introduction

In recent years, the increased computing power of smart devices has facilitated the development of an ecosystem of connected physical objects accessible via WSNs [1]. These devices, characterized by their small silicon footprint, limited computational capabilities, and constrained battery life, having a significant challenge to secure them using conventional cryptographic algorithms like AES [2]. Therefore, many lightweight cryptographic algorithms have been proposed to address this pressing need [3, 4].

The major concern about designing a lightweight cryptographic algorithm is to cope with the trade-offs between security, cost, power and speed [5]. Many lightweight ciphers adopt the symmetric key block cipher approach as it involves using a single key for both encryption and decryption, leading to simpler designs and better hardware performance metrics. Several well-known lightweights using symmetric key block ciphers, such as PRESENT [6], AES [4], RC5 [7], HIGHT [8], and TEA [9]. As a result, many symmetric key block ciphers utilize the Feistel structure for encryption. This structure is a symmetric network that is commonly used for creating block ciphers, and it provides the benefit of having identical encryption and decryption operations with only a slight difference in the key scheduling process [10, 11].

TEA is suitable for resource-constraint devices since it is a fast and effective light-weight encryption algorithm. A mere few lines of source code are sufficient to build TEA. However, TEA does have a limitation when it comes to key scheduling [12, 13]. In general, achieving cryptographically secure keys are challenging task in lightweight cryptography due to the limitations

of conducted devices [14, 15]. Consequently, several modifications to TEA have been proposed to overcome this issue [16–18].

A modified version of TEA is proposed in this paper called ATEA. The main motivation is to overcome the vulnerabilities and shortcomings of the original TEA. The proposed structure incorporates four connected AFGs that are mutually scrambled for perturbation, thereby increasing security performance through the use of a unique key each TEA round. To assess the performance of proposed ATEA, three advanced statistical analysis tests were conducted, avalanche effect, random-ness, and completeness test. In addition, results are compared with state-of-the-art algorithms.

2 Structure of Paper

Section 3 examines the literature review. Section 4 presents the theoretical background of AFG and TEA. Section 5 explains the design and implementation of the proposed work. The simulation environment is explained in Section 6. Results and statistical analyses are explained in Section 7. Finally, the paper conclusions are depicted in Section 8.

3 Literature Review

TEA has been a subject for study and modification within the field of cryptography, this literature review explores the most relevant key findings related to TEA.

In [19], the authors proposed a modified version of the TEA encryption algorithm (MTEA) with a stronger key scheduling algorithm using a S-Box-based approach. Four 32-bit words make up the 128-bit cipher key, each of which is rotated left by 11 bits and subject to a 4x4 S-Box twice. The MTEA uses 32 rounds of encryption, with each round using a 128-bit round key generated by the key scheduling algorithm. The paper offers a clear explanation of the proposed modification and its implementation, but its effectiveness remains unclear without further evaluation.

In 2019, the paper [20] proposes a lightweight encryption scheme based on TEA to protect data transfer between IoT devices. The scheme includes a pseudorandom number generator, key exchange protocol, and ciphertext integrity check. The study demonstrates how secure and effective the recommended plan is, and has lower computational overhead than other lightweight encryption schemes. However, the scheme is limited to text file encryption

and decryption, and it may not be suitable for securing data with varying formats and sizes.

In 2019, the authors in [21] proposed another modified version of TEA to enhance data security for IoT. The proposed modification includes the introduction of key rotation, which changes the encryption key at regular intervals to prevent attacks that may exploit weaknesses in a static key. The paper presents simulation results that demonstrate the effectiveness of the proposed modification in enhancing data security for IoT devices. However, the proposed modification is not compared against the latest encryption algorithms used in IoT devices, which may limit its generalizability. Additionally, the paper does not provide a comprehensive security analysis of the proposed modification, including potential vulnerabilities and attack scenarios.

In 2020, the authors in [22] proposing a method for securing image files. This algorithm was based on cryptography and steganography. Authors used steganography by Least Significant Bit (LSB) method and the Linear Congruential Generator (LCG). The proposed method provides a means of securing image files by encrypting them with TEA and hiding them within other image files using LSB steganography with the LCG. It provides a more secure means of transmitting sensitive image files in various settings, such as medical imaging or military applications.

In 2021, the authors in [23] proposes a new modification of TEA for securing data in IoT devices. The proposed algorithm is based on block ciphers and hash functions, which enables efficient encryption and decryption of data while also ensuring data integrity. The experimental results showed high performance of encryption speed, memory usage, and energy consumption. However, it does not highlight the statistical analysis of the security properties of the proposed algorithm, which may limit its applicability to certain IoT security scenarios.

4 Theoretical Background

4.1 Lagged Fibonacci Generator (LFG)

Pseudorandom Random Generators (PRG) play a vital role in the cryptographic algorithm. A special place among such generators is occupied by LFG. LFGs have gained popularity for their ease of implementation and relatively low cost and complexity. These generators are defined by their recurrence relation;

$$x_n = x_{n-r} \otimes x_{n-s} \; mod \; m \tag{1}$$

Table 1 Maximum attainable periods p of LFG $x_n = x_{n-r} \otimes x_{n-s}\ mod\ 2^m$ [24]

Operations	Maximum Attainable Period
Addition, mod	$p = 2^{N-1}(2^r - 1)$
Subtraction, mod	$p = 2^{N-1}(2^r - 1)$
Multiplication mod	$p = 2^{N-3}(2^r - 1)$
Exclusive-or	$p = 2^r - 1$

The symbol $\otimes$ represents an operation which could be any of the following: addition ($+$), subtraction ($-$), multiplication ($\times$), or exclusive OR ($\oplus$). To generate *R* bits random number, $m = 2^R$. r and s are called the lags of the generator where $r > s > 0$ [24]. The output sequence is represented by x_n, and the time is denoted by n. See [25] for more detailed analysis on this generator.

The maximum period that can be achieved by LFG depends on the specific operation employed, as illustrated in Table 1.

The term Additive Fibonacci generators (AFG) is used to group generators that utilize the $+$ or $-$ operator. As shown in the above table, $+$ and $-$ operators obtain large periods, therefore, most stringent statistical tests show that AFG produce satisfactory outcomes even if the lags have small values. As such, the equation of an AFG can be expressed as:

$$x_n = x_{n-r} \pm x_{n-s}\ mod\ m \tag{2}$$

Where m denotes the base, r and s represent the lags of the past samples, and $\{x_0, \ldots, x_{r-1}\}$, constitutes the seeds values [24].

To reach the maximum period of p, that is $p = 2^{N-1}(2^r - 1)$. The following conditions must satisfy; (1) $m = 2^N$, where N, represents the word length, (2) $x^{31} + x^3 + 1$, is the trinomial irreducible and primitive over GF(2), (3) One of the seed values must be at least odd.

Despite their advantages, LFG is subject to limitations regarding randomness and security. Specifically, they are known to fail certain randomness tests. Therefore, in order to have sufficiently large periods and exhibit ideal random behavior, it requires large lags. However, large lags mean large amount of memory, since the state of an LFG is proportional to its lags [26].

Recently, various researches have been done to modify LFG, due to its easy implementation and simplicity [27]. To overcome the aforementioned issues, a more sophisticated architecture to generate keys has been introduced in this work.

4.2 TEA Overview

TEA is a lightweight encryption algorithm presented in 1994, which was the most efficient and fastest encryption algorithm ever devised at that time [28]. D. Wheeler and R. Roger created TEA at the Computer Laboratory at the University of Cambridge. It is one of the block cipher algorithms that encrypts a 64-bits plaintext with a 128-bits key. The key length is adequate, as required by contemporary encryption standards. The process of encryption starts by dividing a 64-bit plaintext into two 32-bit blocks, which we will refer to as $r[0]$ and $l[1]$. Additionally, the encryption key K is divided into four blocks (K[0], K[1], K[2], and K[3]).

The diagram in Figure 1 displays the block layout of TEA. It consists of 64 Feistel Rounds. Each TEA round includes two Feistel Rounds, resulting in a total of 32 rounds. Although TEA can achieve full diffusion in just six iterations, it is set to 32 iterations for heightened security. In cases where encryption time is limited, the number of iterations can be reduced.

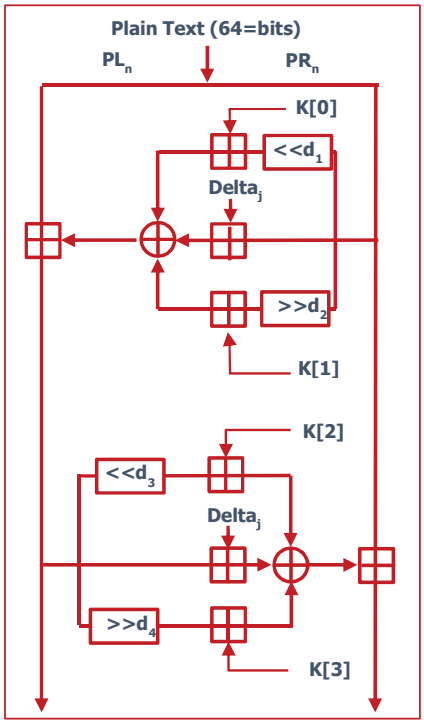

Figure 1 TEA block diagram for one round.

As seen in Algorithm 1, TEA requires very few resources and has a small footprint. It can be implemented in any programming language. The encryption algorithm employs a number called "Delta", which is derived from a golden ratio [13].

Algorithm 1

TEA Encryption
Input: (*l, r*) Plain Text, (*K*) Key, Delta
Output: 64-bits Ciphertext

$delta = (\sqrt{5} - 1) * 2^{31} = 9E3779B9h$
for n in Range(0, 32, 1) # 32 *Rounds*
$sum+ = delta$
$l+ = (r \ll 4) + K[0]\ XOR\ r + sum\ XOR\ (r \gg 5) + K[1]$
$r+ = (l \ll 4) + K[2]\ XOR\ l + sum\ XOR\ (l \gg 5) + K[3]$
$return(l, r)$

A 64-bits plaintext is encrypted in 32 rounds using basic operations, in a way that each half of the plaintext is used to encrypt another half, then these halves are merged to create a 64-bit ciphertext. To mix the input bits and provide non-linearity, has two bitwise shits, XOR, and ADD operations. The purpose of Delta (golden ratio) is to ensure that, the encryption k is not repeated.

In the decryption, same process is inverted. Please refer to Algorithm 2 for more details.

Algorithm 2

TEA Decryption
Input: (*l, r*) Ciphertext, (*K*) Key, Delta
Output: 64-bits Plaintext

$delta = (\sqrt{5} - 1) * 2^{31} = 9E3779B9h$
for n in Range (32, 0, -1) # 32 *Rounds*
$sum+ = delta$
$r+ = (l \ll 4) + K[2]\ XOR\ l + sum\ XOR\ (l \gg 5) + K[3]$
$l+ = (r \ll 4) + K[0]\ XOR\ r + sum\ XOR\ (r \gg 5) + K[1]$
$return(l, r)$

5 Proposed Method ATEA

In this section, we present the new modification for the original TEA called ATEA, which is based on four connected AFGs to overcome the vulnerabilities of the original TEA. The proposed model replaces the original TEA keys schedule by a combination of four AFGs, mutually scrambled. The general structure is depicted in Figure 2.

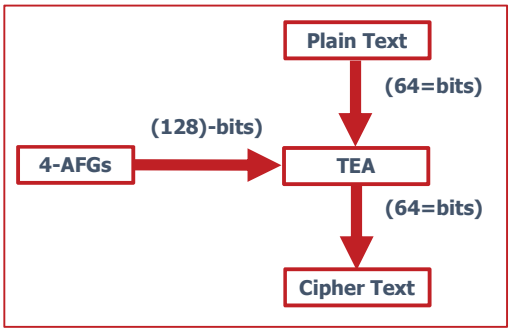

Figure 2 General structure of proposed model ATEA.

To generate keys, the algorithm employs the addition and bitwise XOR of the outputs from four basic AFGs. These AFGs are mutually scrambled by perturbing the most and least significant bits of their lags. The outputs of the AFGs are then summed and mod by *m*, such as;

$$A_n = A_{n-r1} \oplus (B_{n-s2} \ll d_1) + B_{n-s2} \oplus (A_{n-r1} \gg d_4) \ mod \ m$$

$$B_n = B_{n-r2} \oplus (C_{n-s3} \ll d_2) + C_{n-s3} \oplus (B_{n-r2} \gg d_3) \ mod \ m$$

$$C_n = C_{n-r3} \oplus (D_{n-s4} \ll d_1) + D_{n-s4} \oplus (C_{n-r3} \gg d_4) \ mod \ m$$

$$D_n = D_{n-r4} \oplus (A_{n-s1} \ll d_2) + A_{n-s1} \oplus (D_{n-r4} \gg d_3) \ mod \ m$$

d_1, d_2, d_3, d_4 are four constants, where $0 < d_i < N$. The symbol $\oplus$ is a bitwise XOR. ($\geq$) and ($\leq$) are the bitwise shits. It is worth noting that, d_i can be represented as the result of multiplying by 2^{-d_i}, then performing a floor operation. Similarly, $\leq di$ can be represented as the result of multiplying by 2^{d_i} forming a mod.

Note that, to ensure the resulting sequence of each generator have a unique length, the lags r_1, r_2, r_3, and r_4 of the generators must be chosen by different values. In addition, it is essential to choose the lags r_1, r_2, r_3, r_4, s_1, s_2, s_3, and s_4, in a way that satisfies the trinomials $A^{r1} + A^{s1} + 1$, $B^{r2} + B^{s2} + 1$, $C^{r3} + C^{s3} + 1$, and $D^{r4} + D^{s4} + 1$, which should be both irreducible and primitive over GF(2). Additional information on the suggested modification can be found Figure 3.

The modified structure enhances the security of the generator by increasing the number of system states and increasing the period, entropy, and key space. The use of a combination of arithmetic and bit-oriented operations ensures protection against algebraic and related key attacks. Although,

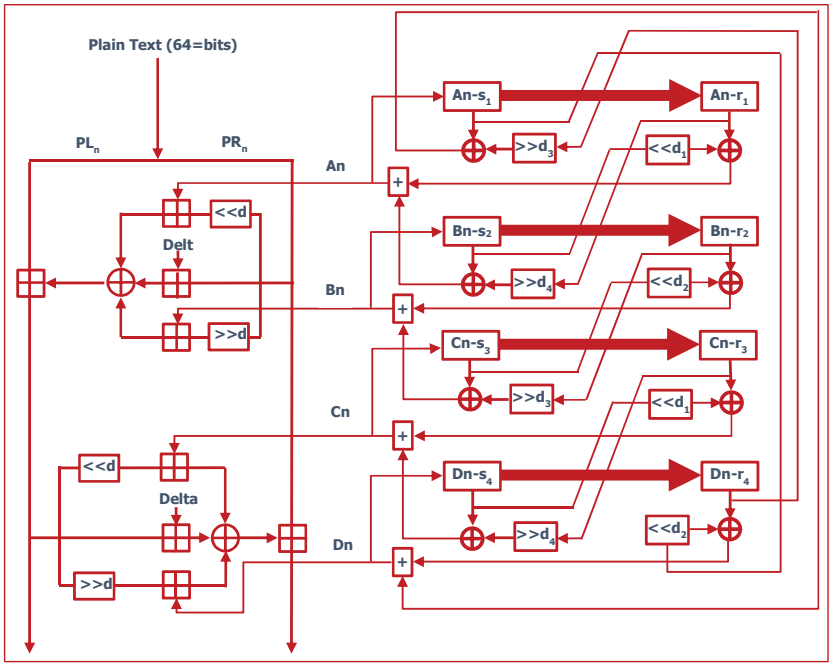

Figure 3 The general diagram of proposed ATEA.

when combining multiple streams to generate one output key with a smaller word size, it becomes challenging to analyze, making it more difficult for a cryptanalysis attack. Since it is subjected to resource constraint devices, the proposed structure employs efficient operations that are easy to implement in hardware or software, such as addition, XORing, bitwise shift, and mod.

Through the experiment, the generator was able to pass previously failed randomness tests of original AFG by perturbing the LSBs of A_{n-r1}, B_{n-r2}, C_{n-r3} and D_{n-r4}, and the MSBs of A_{n-s1}, B_{n-s2}, C_{n-s3} and D_{n-s4} before their addition. To achieve this, the samples A_{n-r1} B_{n-r2}, C_{n-r3} and D_{n-r4} were right-shifted by d_4, d_3, d_4, and d_3 bits respectively, and then XORing with B_{n-s2}, C_{n-s3}, D_{n-s4} and A_{n-s1}. Then, the samples A_{n-s1}, B_{n-s2}, C_{n-s3} and D_{n-s4}, were left-shifted by d_2, d_1, d_2, and d_1 bits respectively, then XORing with D_{n-r4}, A_{n-r1}, B_{n-r2}, and C_{n-r3}, respectively. All samples were perturbed in the same manner. The opposite approach can also be used.

To start the generator, a set of initial values $\{A_0, \ldots, A_{n_1-1}\}$, $\{B_0, \ldots, B_{n_1-1}\}$, $\{C_0, \ldots, C_{n_1-1}\}$, and $\{D_0, \ldots, D_{n_1-1}\}$, must be

provided. It's important to note that a null seed should not be used as it produces a null sequence.

An experiment was conducted on several trinomials to search exhaustively for the perfect generator periods, this work discovered that, the period of the keys was significantly increased by mutual cross perturbation. The period of the generated numbers is much longer than that of an equivalent generator created without scrambling by c XORing four conventional AFGs. In the latter case, four generators constitute the resulting periods A_n, B_n, C_n and D_n.

The maximum possible period of the generated keys is $p = 2^{N-1}(2^{r_1+r_2+r_3+r_4} - 1)$, which is equal to the total number of possible generator states, except for the all-zeros state. The period of keys depends on the parameters r_1, r_2, r_3, r_4, s_1, s_2, s_3, s_4 d_1, d_2, d_3, d_4, and N and also on the initial values.

In this work, and to let TEA works properly, every two AFGs are scrambling to generate 32-bits sub-key, resulting in a total of 128-bits each round. Since the original TEA has 32 rounds, 32 keys are generated for the encryption process, that is one key for each round. To provide more confusion, each key is perturbed based on another key in the structure.

In the decryption process, the keys are utilized in the opposite sequence, meaning that K_{31} is employed in the initial round, K_{30} in the second round, and so on.

6 Simulation Environment

The structure of the suggested model can be constructed using uncomplicated hardware or software. To confirm its proper functioning, the modification algorithm was executed using Python, and the pyFirmata package was installed to employ the Firmata sketch protocol. The implementation was programmed on a PC with Core i7 CPU.

A microcontroller was selected to design as it offers clear benefits such as reduced cost and size, high flexibility, and the added advantage of not limiting the system to a specific platform [29]. Among similar alternatives, Arduino platforms were preferred due to their lower cost and network support. In addition, Arduino platforms are easy to build and update, and have extensive usage in IoT.

The 8-bit ATmega328P microcontroller was used. To power the board, a USB connection is utilized. We used Arduino IDE 1.8.16 for implementation. The resulting programs were then directly flashed into the device. Because

the boards have limited RAM, we chose four irreducible primitive trinomials of the generator, namely $\{x^{17} + x^3 + 1\}$, $\{x^{31} + x^3 + 1\}$, $\{x^{17} + x^3 + 1\}$, and $\{x^{31} + x^3 + 1\}$. The lag values are of $r_1 = 17$, $r_2 = 31$, $r_3 = 17$, and $r_4 = 31$. N = 32 bits was selected, and $0 < d_i < 32$.

It is clear that, the period of combining four AFGs using only XORing operation yield a repetition period of the least common multiple, which is approximately $p = 2^{31}(2^{96} - 1)$. However, calculating the actual repetition period is impossible. Therefore, the generated sequence length is more than sufficient for any cryptographic task. A brute force key attack seems to be infeasible to attack the generated keys. The generator's structure also makes it immune to algebraic attacks since the internal state cannot be learned.

7 Result and Security Analysis

7.1 Avalanche Effect Test

The quality of a cryptographic algorithm can be evaluated by avalanche effect test. It describes how sensitive the algorithm is to small changes in the input, that is a slight change (even one bit) in the plaintext during encryption can cause a significant alteration in the output [30].

The presence of the avalanche effect is crucial in determining the quality of a block cipher. If the degree of the avalanche effect is not significant, it indicates a weak randomization process, making it easier for an attacker to predict the plaintext. This flaw partially or completely compromises the block cipher algorithm. Therefore, the avalanche effect is a desirable characteristic that designers should consider when assessing the cryptographic algorithm's effectiveness. This can be calculated using an equation;

$$AE = \frac{\textit{No. of filled bit in ciphertext}}{\textit{No. of bit in cipherdtext}} \tag{3}$$

To assess precision, the output of encryption and decryption was evaluated by testing text files, and the results are presented in the following figures; Figure 4(a), depicts the content of the file selected for encryption, while Figure 4(b), shows the decrypted file's content using the original TEA using a fixed key for each block during encryption. In contrast, Figure 4(c), displays the encrypted file's content using ATEA, where each block is encrypted using different generated keys. The avalanche effect is prominently visible in Figure 4(d), where changing one bit in the input plaintext results in a considerable difference compared to Figure 4(c). Additionally, Figure 4(e),

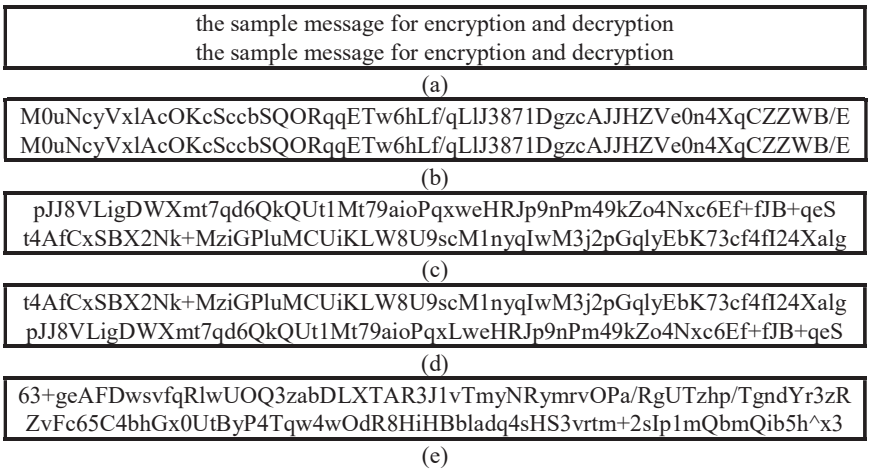

Figure 4 (a) A message considered for encryption using original TEA. (b) The encrypted message using TEA. (c) The same message encrypted using ATEA. (d) The decrypted message after changing one number in initial sequence. (e) The decrypted message after changing one irreducible primitive trinomial.

illustrates the avalanche effect evaluation of the decrypted message after modifying an irreducible primitive trinomial, the encrypted results shows completely different cipher comparing to to Figure 4(c), therefore, the proposed method achieved high performance regarding this test.

To implement this test, we encrypted 10 plaintexts in order to obtain 10 ciphertexts. Then, changed one bit in each given plaintext and encrypted again. After that, XORed each pair of generated ciphertext, and calculate no. of 1 s in each result. Eventually, calculating average percentage for the entire ciphertext.

Table 2, presents the results of a test conducted on ATEA and compared with TEA, and our previous work. The probability of bit reoccurrence in each ciphertext is significantly lower in TEA. For example, in block 1, the recorded avalanche effect for TEA was 47.24, 50.35 for our previous work, while 51.7 for the proposed ATEA.

Figure 5, demonstrates averages of the all 10 blocks, providing evidence that the modified ATEA exhibits superior encryption performance compared to other according to above table.

The benchmark was developed to distinguish between the them as shown in Figure 6. It is clear that, the method devised has surpassed the recommended value of avalanche effect.

Table 2 Avalanche effect Test (10 blocks)

No. of Blocks	TEA [22]	Previous Work [13]	ATEA
1	47.24	50.35	51.7
2	45.95	48.78	50.45
3	47.45	49.36	51.36
4	47.44	50.14	51.15
5	46.37	49.11	50.76
6	47.75	50.37	52.67
7	49.44	51.64	52.77
8	47.12	49.23	51.14
9	47.11	49.97	51.92
10	49.21	51.71	52.84
Average	**47.51**	**50.07**	**51.68**

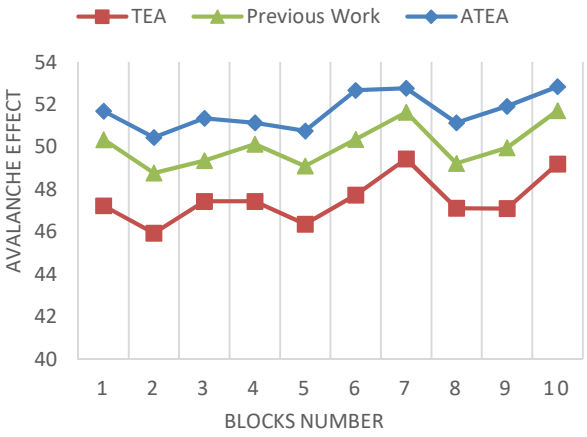

Figure 5 Avalanche effect comparison.

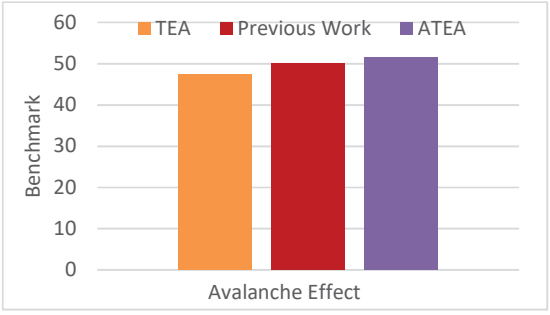

Figure 6 Benchmark results.

7.2 Completed Test

In cryptography, when each bit in ciphertext is sensitive to each bit in plaintext, the ciphertext is considered completed [31]. To implement this test, 65 plaintexts were utilized for testing by the original TEA, our previous work, and proposed work ATEA. Each of which is 64-bits, and varied by one bit from the others. All plaintexts were then encrypted. The ciphertexts then XORed together. The results were subjected to 64 * 64 array. Then, calculating ones in each row. Table 3, shows the results of this test.

Table 3 Evaluation of completeness test

TEA % [8]	Previous Work %	ATEA %
48.36	51.75	51.95

In this case, the completeness percentages are obtained. ATEA exhibits 51.95% which is slightly above the TEA and previous work, which is the goal and objective of this work. Since each round of the proposed ATEA has different key for encryption.

8 Conclusion and Future Works

The goal of this study was to identify an ideal cryptography solution to WSN devices. A novel block cipher algorithm (ATEA) was developed to address the limitations of the original TEA. It was compared to TEA, and our previous work through performance analysis in Python utilizing fast operations such as addition, bitwise XOR, left, and right shifts. To ensure compatibility with the constraint devices, the analysis focused on the most critical factors that need to be considered when employing lightweight ciphers.

An analysis of the modified TEA's encryption and decryption performance was conducted using tests for avalanche effect, randomness, and completeness. The results were compared with those of the original TEA, and our previous work.

- To determine the quality of a cryptographic algorithm. The more significant the avalanche effect, the better the randomization of the block cipher, making it difficult for a cryptanalyst to predict the input from the output ciphertext. In this study, the avalanche effect was evaluated for the original TEA and ATEA by encrypting 10 blocks of plaintexts, changing one bit, and XORing the generated ciphertext. ATEA outperformed the original TEA, and our previous work in terms of avalanche effect, proving that it has better encryption performance.

- Through randomness analysis evaluation, the ATEA algorithm passed all the statistical tests with satisfactory P-values. It is evidence that, the keys of proposed ATEA have high randomness characteristics and more confusion.
- T evaluate the sensitivity of the ciphertext to changes in the plaintext, a completeness test was conducted. The desired probability for this test is 50%. A total of 65 plaintexts were used, each varying by one bit, and encrypted to obtain ciphertexts. The ATEA algorithm was compared with the original TEA, and a previous work. The results show that ATEA achieved the highest completeness test percentage of 51.95%.

Since the work is intended for WSNs with limited resources, the proposed modification is straightforward and does not impose any additional complexity overhead. The use of AFG is appropriate due to its simple computation. As a result, this approach is well-suited for modern WSNs to facilitate data transfer over the network. For future works, different cryptanalytic techniques can be applied by researchers to break it and identify its vulnerabilities and design a secure algorithm. Standardizing this algorithm to ensure its interoperability, security, and efficiency across different systems and applications.

References

[1] S. Abdollahzadeh and N. J. Navimipour, "Deployment strategies in the wireless sensor network: A comprehensive review," *Computer Communications*. vol. 91, no. 92, pp. 1–6, 2016. https://doi.org/10.1016/j.comcom.2016.06.003.

[2] V. Hassija, et al., "A survey on IoT security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82721–82743, 2019. https://doi.org/10.1109/ACCESS.2019.2924045.

[3] M. Swan, "Sensor Mania, The Internet of Things, Wearable Computing, Objective Metrics, and the Quantified Self 2.0", *Journal of Sensor and Actuator Network*, vol. 1, no. 3, 1, pp. 217–253, 2012. https://doi.org/10.3390/jsan1030217.

[4] Z. K. Zhang, et al. "IoT security: ongoing challenges and research opportunities," *in 2014 IEEE 7th international conference on service-oriented computing and applications, Matsue, Japan, 17-19 November 2014*, *IEEE*. https://doi.org/10.1109/SOCA.2014.58.

[5] Juels, A., "RFID security and privacy: A research survey," *IEEE journal on selected areas in communications*, vol. 24, no. 2, pp. 381–394, 2006. https://doi.org/10.1109/JSAC.2005.861395.

[6] Bogdanov, et al. "PRESENT: An ultra-lightweight block cipher, " *In Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10–13,*. Proceedings 9, pp. 450–466, Springer Berlin Heidelberg, 2007. https://doi.org/10.1007/978-3-540-74735-2_31.

[7] Rivest, Ronald L, "The RC5 encryption algorithm." *Fast Software Encryption: Second International Workshop Leuven, Belgium*, December 14–16, 1994 Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. https://doi.org/10.1007/3-540-60590-8_7.

[8] Hong, D., et al. "HIGHT: A new block cipher suitable for low-resource device," *in International workshop on cryptographic hardware and embedded systems*. 2006. Springer. https://doi.org/10.1007/11894063_4.

[9] Hardi, S., et al. "Security of Image File with Tiny Encryption Algorithm and Modified Significant Bit Pseudo Random Number Generator," *in Journal of Physics: Conference Series*. 2020. IOP Publishing. https://doi.org/10.1088/1742-6596/1566/1/012108.

[10] Feistel, H., "Cryptography and computer privacy," S*cientific american*, 1973. 228(5): p. 15–23. https://www.jstor.org/stable/24923044.

[11] Kadhim, A. and H.I. Mhaibes, "A new initial authentication scheme for kerberos 5 based on biometric data and virtual password," *in 2018 International Conference on Advanced Sc ence and Engineering (ICOASE)*, 2018, IEEE. https://doi.org/10.1109/ICOASE.2018.8548852.

[12] Ahmad, R. and I. Alsmadi, "Machine learning approaches to IoT security: A systematic literature review," Internet of Things, 2021. 14: p. 100365. https://doi.org/10.1016/j.iot.2021.100365.

[13] Mhaibes, Hakeem Imad, May Hattim Abood, and Alaa Kadhim Farhan, "Simple Lightweight Cryptographic Algorithm to Secure Imbedded IoT Devices," *International Journal of Interactive Mobile Technologies*, 16.20 (2022). https://doi.org/10.3991/ijim.v16i20.34505.

[14] Simon J. Shepherd, "The Tiny Encryption Algorithm", *Cryptologia*, 31:3, 233-245, DOI: 10.1080/01611190601090606.

[15] Mhaibes, Hakeem Imad, and S. Qadir, "A Lightweight Authentication Framework for Wireless Sensor Networks," *International journal of electrical and computer engineering systems, 2022.* 13(1): pp. 19–27. https://doi.org/10.32985/ijeces.13.1.3.

[16] Suresh, S.A. and J. Priyadarsini, "ETSET: Enhanced Tiny Symmetric Encryption Techniques to Secure Data Transmission among IoT Devices," *Turkish Journal of Computer and Mathematics Education*,

2021. 12(10): pp. 1094–1099. https://doi.org/10.17762/turcomat.v12i10.4294.

[17] Aradhyamath, S. and J. Paulose, "Multi-key Modified Tiny Encryption Algorithm for HealthCare," *International Journal of Engineering & Technology*, 2018. 7(2.14). https://doi.org/10.14419/ijet.v7i2.9894.

[18] De Leon, R.M., A.M. Sison, and R.P. Medina, "A Modified Tiny Encryption Algorithm Using Key Rotation to Enhance Data Security for Internet of Things," *in 2019 International Conference on Information and Communications Technology (ICOIACT).* 2019. IEEE. https://doi.org/10.1109/ICOIACT46704.2019.8938456.

[19] Rajak, Chandradeo Kumar, and Arun Mishra, "Implementation of modified TEA to enhance security," *Information and Communication Technology for Intelligent Systems*, vol. 1 No. 2. Springer International Publishing, 2018. https://doi.org/10.1007/978-3-319-63673-3_46.

[20] Rajesh, S., Paul, V., Menon, V.G., Khosravi, M.R. "A Secure and Efficient Lightweight Symmetric Encryption Scheme for Transfer of Text Files between Embedded IoT Devices", *Symmetry* 2019, 11, 293. https://doi.org/10.3390/sym11020293.

[21] De Leon, R.M., A.M. Sison, and R.P. Medina, "A Modified Tiny Encryption Algorithm Using Key Rotation to Enhance Data Security for Internet of Things," *in 2019 International Conference on Information and Communications Technology (ICOIACT).* 2019. IEEE. https://doi.org/10.1109/ICOIACT46704.2019.8938456.

[22] Hardi, S. M., et al. "Security of Image File with Tiny Encryption Algorithm and Modified Significant Bit Pseudo Random Number Generator." *Journal of Physics: Conference Series*. Vol. 1566. No. 1. IOP Publishing, 2020. https://doi.org/10.1088/1742-6596/1566/1/012108.

[23] Abdulraheem, Muyideen, et al. "An efficient lightweight cryptographic algorithm for IoT security." *Information and Communication Technology and Applications: Third International Conference, ICTA 2020, Minna, Nigeria, November 24–27, 2020, Revised Selected Papers 3*. Springer International Publishing, 2021. https://doi.org/10.1007/978-3-030-69143-1_34.

[24] Aluru, Srinivas, "Lagged Fibonacci random number generators for distributed memory parallel computers." *Journal of Parallel and Distributed Computing,* vol. 45, No. 1, pp. 1–12, 1997. https://doi.org/10.1006/jpdc.1997.1363.

[25] López, Amalia Beatriz Orúe, et al., "A lightweight pseudorandom number generator for securing the Internet of Things." *IEEE access*,

vol. 5, pp. 27800–27806, 2017. https://doi.org/10.1109/ACCESS.2017.2774105.

[26] Maksymovych, V., et al., "A New Approach to the Development of Additive Fibonacci Generators Based on Prime Numbers," *Electronics,* 2021, 10, 2912. https://doi.org/10.3390/electronics10232912.

[27] Maksymovych V, Shabatura M, Harasymchuk O, Karpinski M, Jancarczyk D, Sawicki P. "Development of Additive Fibonacci Generators with Improved Characteristics for Cybersecurity Needs", *Applied Sciences*, 12(3):1519. https://doi.org/10.3390/app12031519.

[28] Wheeler, D.J. and R.M. Needham, "TEA, a tiny encryption algorithm," *in international workshop on fast software encryption, Springer*, 1994. https://doi.org/10.1007/3-540-60590-8_29.

[29] A. Mejías, RS. Herrera, MA. Márquez, AJ. Calderón, I. González, and JM. Andújar, "Easy Handling of Sensors and Actuators over TCP/IP Networks by Open Source Hardware/Software," *Sensors*, vol. 17, no. 1, pp. 94, 2017. https://doi.org/10.3390/s17010094.

[30] J. Kaur and K. R. R. Kumar, "Analysis of Avalanche effect in Cryptographic Algorithms," *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, 2022, pp. 1–4. https://doi.org/10.1109/ICRITO56286.2022.9965127.

[31] Rajesh, S., et al., "A Secure and Efficient Lightweight Symmetric Encryption Scheme for Transfer of Text Files between Embedded IoT Devices," *Symmetry,* 2019. 11(2). https://doi.org/10.3390/sym11020293.

Biographies

Atheer Hussein Zyara is a teacher at the computer center at College of Health and Medical Technology/ Baghdad. Middle technical University – Baghdad – Iraq. He completed his bachelor degree in Computer Science at Al-Mustansiriya University, Iraq in 2008. He completed his MSC degree in

computer science at Hamdard University, India in 2011 the teacher Atheer Hussein Zyara is the author of numerous technical papers since 2013, his research interests include: Cryptography, programming languages, Chaos theory, cloud computing.

Hakeem Imad Mhaibes is a member of the Middle Technical University, Baghdad, Iraq. He received the BSc degree in computer science from the Al-Mustansiriyah University, Baghdad, and the Msc degree in computer science (computer programming) from Jamia Hamdard University, New Delhi, India in 2011 and the Ph.D. degree in information security from the University of Technology, in 2019. His research interests include Cryptography, Information Security, Wireless Sensor Network, Biometric Techniques, Image Processing, and Pattern Recognition, Programming.

Qahtan Makki Shallal is presently working as a head of department in the Information Technologies department, technical management college, southern technical university, Basra, (Iraq). He has a vast teaching experience of more than 12 years. Dr. Qahtan obtained his doctorate in the field of network security from Aligarh Muslim University (AMU), and master's degree in computer science from Hamdard University in New Delhi. His research interests include cryptography, cloud computing, smart grid, wireless sensor networks, and machine learning.