

Enhancement process of AES: a lightweight cryptography algorithm-AES for constrained devices

Hussein M. Mohammad, Alharith A. Abdullah

Network Departement, College of Information Technology, University of Babylon, Babil, Iraq

Article Info

Article history:

Received Apr 22, 2021

Revised Apr 03, 2022

Accepted Apr 11, 2022

Keywords:

AES Algorithm

Continued fraction

Lightweight AES algorithm

Security analysis

ABSTRACT

The restricted devices have a small memory, simple processor, and limited power. To secure them, we need lightweight cryptography algorithms, taking into account the limited specifications. Lightweight cryptography (LWC) algorithms provide confidentiality and maintain information integrity for devices with limited resources. This paper improves and enhances advanced encryption standard (AES) algorithm by reducing algorithm computation power and improving cryptography performance from the point of resource constraint devices. The proposed algorithm is fast and lightweight, which is essential for securing all kinds of data. Besides, the use of mix column overhead is dispensing with, and the ciphertext is processed by the mathematical function (continued fraction) to compress the ciphertext and make it more confusing and also to increasing the data transfer speed. The proposed lightweight cryptography-AES (LWC-AES) algorithm highly suitable for the timely execution of encryption and decryption (such as when encrypt text has (45.1 KB) encryption execution time for AES was (294 ms), while in LWC-AES was (280 ms), as well as suitable for the memory size of the resource-constrained devices for all types of data, than the AES algorithm. The proposed algorithm tested for security analysis using the avalanche effect parameter, and this test showed acceptable and within required security results.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Alharith A. Abdullah

Network Departement, College of Information Technology

University of Babylon, Babil, Iraq

Email: alharith@itnet.uobabylon.edu.iq

1. INTRODUCTION

The design of traditional cryptographic algorithms is suitable for conventional devices but not compatible with restricted devices [1], and classical cryptography methods require a significant allocation of resources. To resolve these problems, or rather challenges, the National Institute of Standards and Technology (NIST) recommended favoring lightweight cryptography (LWC) algorithms which provide the same level of security as traditional algorithms, and their performance is also acceptable on these resource-limited devices [2]. LWC is essential for securing resource-limited devices like smart cards, radio-frequency identification (RFID) tags, sensor networks, and embedded systems [3], [4]. They are also used for hastily rising resource constraints applications. The applications include wireless sensor networks, smartcards [5], internet of things (IoT) [6], RFID tags [7], wireless body area networks, healthcare devices [8], [9], and other applications.

Lightweight cryptography's main objective is to minimize the overall implementation cost in software and hardware, where software has the implementation size, random access memory (RAM) consumption, and throughput (bytes per cycle). In contrast, the hardware has code size, memory consumption

(RAM), energy consumption, and gate equivalence (GE) [10]. Lightweight cryptography is intended for hastily increasing applications that highly employ smart and limited-resource devices.

There are many LWC in security for constraint devices like (advanced encryption standard (AES), rivest's cipher 5 (RC5), PRESENT, Simon, Speck, high security and lightweight (HIGHT), lightweight encryption algorithm (LEA), tiny encryption algorithm (TEA), and KATAN) [11]. However, it differs in terms of memory and power consumption, and security, which is the most crucial factor. Due to the unique specification of restricted devices, the obtaining of a high level of security is difficult and needs developing the lightweight encryption algorithm always to be more compatible and to make a balance between the security and the work nature of restricted devices [1] which is the biggest challenge for lightweight encryption algorithms. In other words, the perfect algorithm should preserve the appropriate balance between cost, performance, and security [12]. From this standpoint, and to achieve the aforementioned balance and because traditional algorithms do not fulfill these conditions in the restricted devices environment and because the traditional AES encryption algorithm, which will be addressed later, carries near-perfect specifications to ensure confidentiality, we have improved its specifications and made it work in a manner that achieves the required balance between cost, performance and security. Therefore, a lightweight encryption algorithm has been proposed that carries strong specifications compared to the rest of the lightweight encryption algorithms that were previously employed.

Advanced encryption standard [13] is considered one of the best encryption algorithms for several reasons, among which is its classification among the symmetric encryption algorithms that use a single key for both encryption and decryption of the data, and its key is extraordinarily secure and relatively fast, as well as its guarantee integrity and confidentiality of data [14]. Therefore it is suitable for resource-constrained devices such as IoT devices and embedded systems. It can effectively defend against many well-known attacks [15]. One of the advantages of this algorithm is the variation of block size and key size. In other words, the block size and key size can vary, making the algorithm more flexible and versatile. AES is initially designed for unclassified US government information, but AES-256 can be used to obtain top-secret government information [16]. Among the many advantages of the AES algorithm is implementing LWC-AES encryption in the .net application as an aspect of a programming algorithm based on light encryption, which has a significant role in providing the necessary security for many devices and resource-limited applications.

There are many much-related researchs on the modification of AES as it is the standard for encryption, such as:

- Kawle *et al.* [17] proposed a method to overcome the computational overhead, the AES is modified after analyzing it through eliminating the use of mixcolumn overheads on data and the use of the permutation step instead. The modified version of this algorithm provides a faster encryption technique for the protection of multimedia data. This lightweight encryption algorithm transmits data and it is compatible with large plaintext.
- Sari *et al.* [18] has been proposed improved security and message capacity using AES and Huffman coding to reduce the total of the message's bit and increase the capacity on image steganography. The results of this paper shown a secure message image and conceal into a cover image, where provided a higher capacity in discrete wavelet transform (DWT) for steganography by reducing the total of message's bit from the original message's bit.
- Kumar *et al.* [19] proposed an idea of encrypting voice signals over peer to peer communication with the help of a modified and lightweight AES algorithm. This algorithm is similar to the traditional AES algorithm in most of the aspects but does not imply the use of a mix column that is used in the traditional AES algorithm. The results of the algorithm are analyzed on Artix-7 and Kintex-7 field programmable gate arrays (FPGAs).
- Hazzaa *et al.* [20] proposes a lightweight and low energy encryption algorithm to secure voice traffic over wireless networks. It is capable to reduce the execution time and power consumption of the encryption process compared with the state of the art standard algorithm and at the same time maintains the desired security (confidentiality) level. The proposed algorithm employs similar methods with those used in the AES algorithm.

In this paper, an enhancement of lightweight AES algorithm is proposed to reduce delay and computation power in the context of the traditional AES algorithm, making it more flexible and compatible with the characteristics of IoT devices and military services with C# programming .NET Framework implementation in Windows 10 and Microsoft Visual Studio. The used framework enhanced the computation power constraints, and it is analyzed with the avalanche effect (AE) security parameter. This paper's outline falls into the following sections: section 1 introduced the fundamental concepts of lightweight cryptography algorithms with the most related papers, section 2 presents the proposed lightweight AES algorithms architecture with all details of design where it presents the used system and illustrates the working steps of the system as the used architecture for designing and implementing lightweight AES algorithms with the

main relevant topics, section 3 discusses the implementation of the proposed algorithm with the results and comparisons where it shows the achievement of the suggested lightweight AES algorithm with the result, and compares the proposed AES lightweight with others lightweighted algorithms, section 4 is devoted to explain security analysis. Finally, section 5 sums up the findings and conclusions of the research.

2. ADVANCED ENCRYPTION STANDARD

In 2000, NIST developed the data encryption standard (DES) to create a new algorithm known as AES [21], the new algorithm was developed through mathematical computational properties and simple execution for both in hardware and software implementations as well as using key size with different lengths 128 bits, 192 bits, and 256 bits [15]. The AES algorithm, as a block cipher algorithm, consists of several rounds N , the number of the round in AES is ten rounds based on length of the encryption key, key length 128-bit, 12 rounds, key length 192-bit, and 14 round, key length 256-bit. AES usage with a key size of 128 bits (AES-128 bits) is the most common in the current period. In AES, a block of data (128 bit) is divided into four blocks. They are handled as a bytes array and arranged in a 4×4 matrix called state. This algorithm starts with add-round-key for encrypting and decrypting data in add round processes; the plaintext is exclusive-ORed with a key [22]. All the AES algorithm rounds involve four operations (sub-bytes, shift-rows, mix-columns, add round key) except the final round which involves only three operations (sub-bytes, shift-rows, add round key). The decryption process uses the inverse functions, substitute-bytes inverse, shift-rows inverse, and mix-columns inverse, as shown in Figure 1.

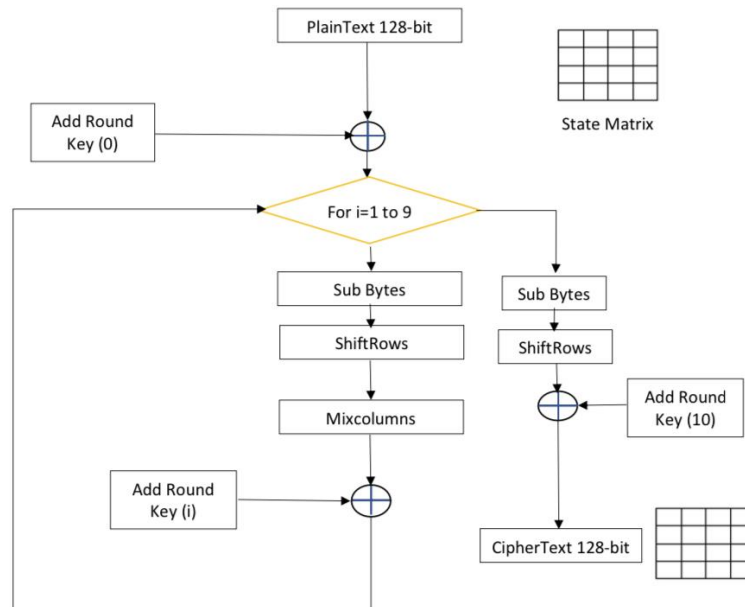


Figure 1. Block diagram for AES algorithm

3. CONTINUED FRACTION

In this paper the proposed enhancement procedure is based on the continued fraction. It is a way of expressing a number as an integer plus a series of nested functions continued fraction data back from around about 300 BC around about the time of Euclid. It has many applications, one of them is proposed in this paper [23], [24]. The mathematical expression of a continued fraction as follows:

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\dots + \frac{1}{a_n}}}} \tag{1}$$

Where n is a non-negative integer, a_0 is an integer, and a_i a positive integer, for $i = 1, \dots, n$. This mathematical expression is called a generalized continued fraction. Typically, the numbers may be real or complex, and the expansion may be finite or infinite [23].

4. METHOD

As mentioned earlier, this work proposes enhancing the AES algorithm to improve security, reduce high calculation, provide better encryption speed, and reduce computational overhead in constrained devices. The proposed algorithm characteristics are block size equals 128-bit, key size equals 128-bit, and dynamic rounds. There is no difference between the proposed LWC-AES and original AES in the encryption and decryption process, the number of rounds, data, and key size. The primary operations are remaining without a change in encryption and decryption steps (substitution bytes, shift-rows, and add-round-key). The main improvement in the proposed LWC-AES algorithm is mix column operation, where it is reduced in comparison with the original AES that takes a long time in encryption and decryption; now, after reduction, the text of 128-bit is directly taken from the output of shift rows operation. Then, a mathematical function (continued fraction) is used to reduce the number of bit transfer among entities and increase the security of the used method, hence, increasing the difficulty to break by different attacks such as side-channel attack and differential attack because the size of the encrypted text is different from the standard case by the compressed feature of continued fraction method. Its execution has been modified according to the final step of encryption optimizer that is added after the final ciphertext (where the encryption block is reduced to 16 bit) assuming that the encrypted text security is improved. The LWC-AES calculation can be separated into three phases, excluding the mix column process. The flow chart of the process is explained in Figure 2.

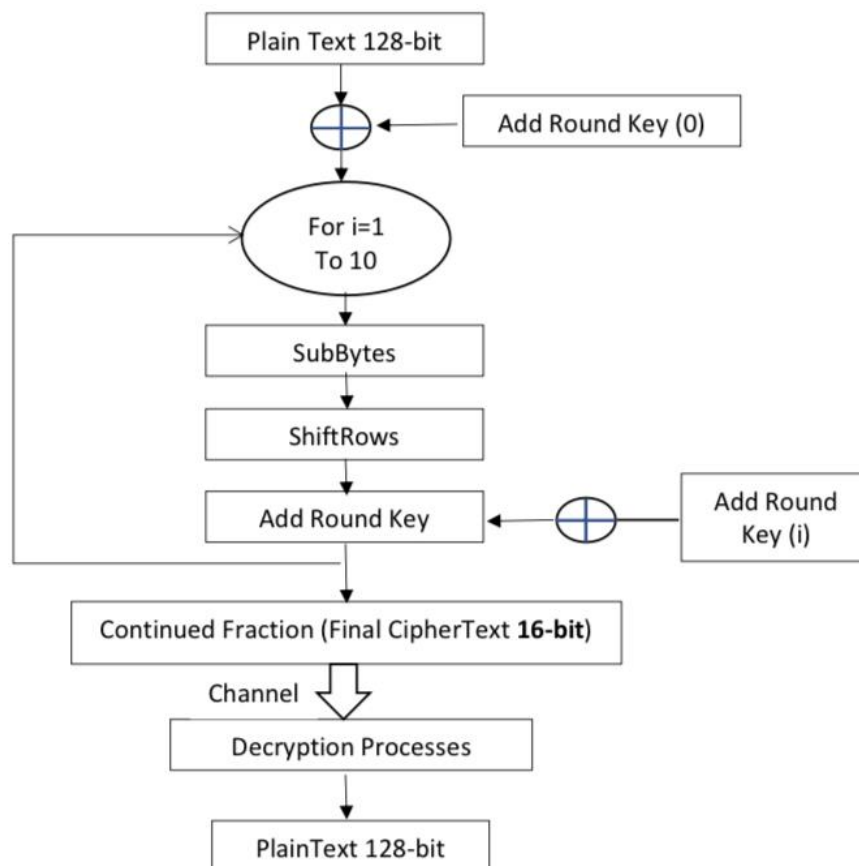


Figure 2. The flowchart of LWC-AES algorithm process

5. THE PROPOSED COMPONENTS OF LWC-AES

The main component of the proposed method is the initial round (add-round-key), then (10) rounds of three functions on the encryption side. In Figure 2, we notice there is no conditional function so that all rounds work together, taking into account the reduction of the steps of the conditional sentences. Steps are composed of: non-linear byte substitution (sub-byte). Besides, we removed the MixColumn to reduce computation power process, shift row transformation (shift-row), and key addition (add-round-key). After ten rounds are performed, the ciphertext is obtained, and it is entered into the mathematical function called

(continued fraction) which reduces and compresses the encrypted text from 128 bits to 16 bits. Decryption is the reverse encoding process, which is converting the encrypted text into an original plain text. The whole algorithm steps of the proposed LWC-AES encryption are clarified in Algorithm 1.

5.1. The encryption LWC-AES

The proposed development of LWC-AES encryption algorithm to improve security, reduced high calculation, provide better encryption speed and reduced computational overhead in constrained devices. Non-linear byte substitution is one of the steps (SubByte). In addition, to minimize calculation power, we deleted the MixColumn, Shift Row Transformation (ShiftRow), and Key Addition (AddRound-Key). After 10 rounds, the encrypted message is acquired and placed into the (continued fraction) mathematical function, which lowers and compresses the encrypted text from 128 bits to 16 bits. The step of the proposed LWC-AES encryption algorithm is showed in Algorithm 1.

Algorithm 1. Steps of the proposed LWC-AES encryption

-
1. Initializing values for encryption
 - byte[] plainText = new byte[MAX_BLOCK_LENGTH]
 - byte[] cipherText = new byte[MAX_BLOCK_LENGTH]
 - byte[] bzkey = new byte[MAX_KEY_LENGTH]
 2. Getting password
 - bzkey = Encoding.Unicode.GetBytes (cPassword);
 3. Getting bytes from File
 - FileStream fileStream = new FileStream (cOpenFile, FileMode.Open);
 - fileStream.Seek (0, SeekOrigin.Begin);
 4. Getting the file stream for save process
 - FileStream saveStream = new FileStream (cSaveFile, FileMode.Append);
 5. Setting length of the File
 - long IFileLength = fileStream.Length;
 6. Setting position of the File
 - long IPostion = fileStream.Position;
 7. Reading byte and encrypt
 - while (IPostion < IFileLength)
 8. Initializing the buffer
 - initialize (plainText, MAX_BLOCK_LENGTH)
 - long IHasRead = fileStream.Read (plainText, 0, MAX_BLOCK_LENGTH);
 - if (0 >= IHasRead) → break;
 9. Setting current cursor position
 - lpostion = fileStream.Position;
 10. Encryption with AES
 - aes aes = new Aes (ekeySize, bzkey, eblockSize)
 11. Initializing the buffer
 - initialize (cipherText, MAX_BLOCK_LENGTH)
 - aes.Cipher (plainText, cipherText)
 - saveStream.Write (cipherText, 0, MAX_BLOCK_LENGTH)
 - saveStream.Close()
 - fileStream.Close()
 - return true
-

5.2. The decryption LWC-AES

Through the decryption side explained in Algorithm 2, we can see the algorithm steps of the proposed LWC-AES decryption. The primary job of the decryption state of the proposed method is based on the value generated from the continued fraction method which is equivalent to the ciphertext, and then within decryption process the key is added to the ciphertext to generate plaintext again, which is less overhead from the meaningful content generated in AES traditional state (without continued fraction method). After 10 rounds are performed, the ciphertext is obtained, and start this step as it is enter into the mathematical function called continuo fraction, which reduces the compresses the encrypted text from 128 bits to 16 bits. Decryption is the reverse encoding process, which is converting the encrypted text into an original plaintext. The steps of using continued fraction in C# are shown in Algorithm 3.

Algorithm 2. Steps of the proposed LWC-AES decryption

1. Getting password
 - bzkey = Encoding.Unicode.GetBytes (cPassword);
 2. Getting bytes from file
 - FileStream fileStream = new FileStream (cOpenFile, FileMode.Open);
 - fileStream.Seek (0, SeekOrigin.Begin);
 3. Getting the file stream for save
 - FileStream saveStream = new FileStream (cSaveFile, FileMode.Append);
 4. Setting length of the file
 - long lFileLength = fileStream.Length;
 5. Setting position of the file
 - long lPostion = fileStream.Position;
 6. Reading byte and decrypt
 - while (lPostion < lFileLength)
 7. Initializing the buffer
 - initialize(plainText, MAX_BLOCK_LENGTH);
 - long lHasRead = fileStream.Read (plainText, 0, MAX_BLOCK_LENGTH);
 - if (0 >= lHasRead)
 - break;
 8. Setting current cursor position
 - lpostion = fileStream.Position;
 9. Encrypt call method
 - aes aes = new Aes (ekeySize, bzkey, eblockSize);
 10. Initializing the buffer
 - Initialize (cipherText, MAX_BLOCK_LENGTH);
 11. Decryption process
 - aes.InvCipher (plainText, cipherText);
 - saveStream.Write (cipherText, 0, MAX_BLOCK_LENGTH);
-

Algorithm 3. Steps of continued fraction algorithm

1. Writing namespaces libraries
 - using System;
 - using System.Collections.Generic;
 2. Building main class program
 3. Creating the continued fraction function
 - static double Calculating (Func<int, int[]> f, int n)
 - double temp = 0.0;
 - looping for (int ni = n; ni >= 1; ni--)
 - int[] p = f(ni);
 - temp = p[1] / (p[0] + temp);
 - return f(0)[0] + temp;
 4. Building main code list conversation with :
 - List<Func<int, int[]>> List = new List<Func<int, int[]>>();
 - List.Add(n => new int[] { n > 0, 2 : 1, 1 });
 - List.Add(n => new int[] { n > 0, n : 2, n > 1 ? (n - 1) : 1 });
 - List.Add(n => new int[] { n > 0, 6 : 3, (int) Math.Pow(2 * n - 1, 2) });
 - foreach (var f in List) {Label1.text=(Calculating (f, 200)).ToString();
-

6. RESULTS AND DISCUSSIONS

The proposed LWC-AES algorithm is implemented in the ASP.Net framework with c# programming language, based on the original text (plaintext of selected keywords of the ministry of the interior words). The results showed the encryption-decryption of the proposed LWC-AES and traditional AES algorithm for many criteria such as (encrypt/decrypt execution time, encryption characters number, code size, and RAM size in bytes, and throughput in Mbps). For example, the word 'freshness', which is usually used in police stations was encrypted. Taking into account the same used encryption key, the results obtained were as follows: the number of characters encryption in the traditional AES was 44 characters and

the time to execute the encryption was 1374 ms and the time to execute the decryption was 951 ms, while in our proposed algorithm LWC-AES, we noticed that the number of characters encryption was 24. The time for executing the encryption was 712 ms, while the time to execute the decryption was 535 ms. Before comparing the two algorithms, we mention in Table 1, which shows the details of the system specifications in which the results were programmed and executed.

Table 1. The used system specifications

Operating system	RAM	CPU	Framework	Programming language
Windows 10	4 GB	Core I 5	Microsoft Visual Studio.net	C#

Table 2 shows the proposed LWC-AES in comparison to other algorithms. The block and key sizes are in bits, while the code and RAM sizes are in bytes. We notice that our proposed algorithm has a less code size than other algorithms and has a less RAM capacity. The used traditional AES has a code size with 23090 bytes and RAM resource allocation with 720 bytes for text data type which requires more computation power and it is considered a challenge to face within restricted devices, while the proposed lightweight works with less code size and RAM allocation as a solve state of the problem faced, as the proposed development algorithm is based on continued fraction algorithm, as well as some algorithms have RAM capacity with slightly better results than our proposed algorithm, we conclude that our results are the best from observing these algorithms' block size.

Table 2. Comparison between the proposed LWC-AES algorithm and other algorithms [1]

The algorithm	Block size in bits	Key size in bits	Rounds	Code size in bytes	RAM in bytes
PRESENT	64	80	32	1738	274
Simon	64	96	42	1370	188
Speck	64	96	26	2552	124
Traditional AES (before lightweight)	128	128	10	23090	720
LEA	128	128	24	3700	432
RC5	64	128	20	20044	360
HIGHT	64	128	32	13476	288
Proposed (text)	128	128	10	587	326
Proposed (files)	128	128	10	1870	433

In Table 3 showed the proposed LW-AES compared with the traditional AES based on the same initial values explained in table columns as plaintext string and we calculated number of cipher text characters, encrypt/decrypt time, input size and proposed fixed values of random key = 128 bits which is generated from random generator method and password (session key) equal 11 bits. Through the table, we note that the less the size of the ciphertext, the less the encryption and decryption time, and this is one of the advantages of our proposed algorithm in improving the execution time of encryption and decryption, which will lead to a reduction in memory size and processor speed, and reduce power consumption as well as response speed in the ideal time and finally reduce the ciphertext helps in the speed of data transmission over the network in the ideal time and finally reduce the ciphertext helps in the speed of data transmission over the network. A comparison between AES and LWC-AES with executions of encryption/ decryption time for different input text sizes and then measuring the average time and throughput value, we noted that the LWC-AES has the minimum execution time for encrypting/decrypting processes to different input text sizes, which means its suitability for constraint devices that have the constraint resources such as RAM, power-consuming, and little battery life. Throughput is measured in bits per second (bit/s or bps/Mbps), and in this paper it is used to measure input data packets (different data file) per execution time for encryption and decryption time slot. As shown in (2) [25].

$$\text{Throughput} = \frac{\text{plain text (Mbps)}}{\text{encryption time (second)}} \quad (2)$$

In Table 4, we note that proposed algorithm was able to improve the execution time of encryption and decryption, as well as it was noted that the throughput was better than the traditional algorithm, depending on the time spent in the encryption and decryption processes. It means that the faster encryption time will produce more throughputs.

Table 3. System comparison between AES and LWC-AES

Plaintext	AES			Proposed LWC-AES		
	Briskness	Fluctuation	Positive information	Briskness	Fluctuation	Positive information
Cipher	luT1Q5ivipTxT +a8VdlfWI BpC2JIRDbf oZF2NylqqSY=	OyX+Ug+G V9Jm2u7rms Ce65wKzPTw5 jtS38n2tVEGicc=	CIVox1R+Ql YE4Fj1CbIB ln1apJcFjieH v8qCo/r4xP+ Q0cgVdb1SI kRTXjGkZimq	6zvMR9wvR w67SzmJGvfn 4A==	Ms3+m48W OmDrbwAG LVSCFA==	aV7vVYN5m rEvvwoKPiIFg ZUiX/EHR8z wQKHrQSkVIqxw=
Number of origin characters	9	11	20	9	11	20
Number of cipher characters	44	44	64	24	24	44
Encrypt. time	1374 ms	859 ms	493 ms	712 ms	386 ms	289 ms
Decrypt. time	951 ms	762 ms	480 ms	535 ms	291 ms	284 ms
Input size in Bytes	9 B	11 B	20 B	9 B	11 B	20 B

Table 4. Comparison for different input data size of AES with LWC-AES

Input size in B/KE	Executions encryption time in millisecond's		Executions Ddcryption time in millisecond's	
	AES	LWC-AES	AES	LWC-AES
481 B	314 ms	279 ms	289 ms	260 ms
1.41 KB	318 ms	285 ms	299 ms	271 ms
2.82 KB	276 ms	257 ms	297 ms	266 ms
5.64 KB	270 ms	250 ms	272 ms	267 ms
45.1 KB	294 ms	280 ms	295 ms	291 ms
Average time in sec.	294.4 ms / 0.2944 s	270.2 ms / 0.2702 s	290.4 ms / 0.2904 s	271 ms / 0.271 s
Throughput in Mbps	Average data in Bit = 627586.88	Average data in Bit = 627586.88	Average data in Bit = 627586.88	Average data in Bit = 627586.88
	Throughput = 2.033 Mbps	Throughput = 2.215 Mbps	Throughput = 2.061 Mbps	Throughput = 2.2085 Mbps

7. SECURITY ANALYSIS

There are many security parameters used to evaluate the security strength concept, for instance, avalanche effect which is one of the vital security analysis in the cryptography used to compute the goodness ratio of an encryption technique, the differential cryptanalysis, brute force, linear cryptanalysis, the zero-correlation attack, key schedule attacks, and others. In the following subsections we explained the two most important security analysis metrics, which are avalanche effect that computed by changing one bit in the plaintext while keeping the key constant and changing one bit in the encryption key while keeping the key constant and ciphertext expansion that refers to the lengthening of a message after it has been encrypted. The security analysis is significant in the paper because it allows the investor to determine the projected return and risk for a stock and evaluate its attractiveness in a logical, reasonable manner.

7.1. Avalanche effect

Our proposed lightweight AES algorithm is based on the avalanche effect (AE) parameters [26], which is one of the vital security analysis in the cryptography used to compute the goodness ratio of an encryption technique. The main point in this metric is the change in plaintext or the key may cause a significant change in ciphertext, even if it is one bit. For instance, if the change affects half of the bits, this means 50% of the ciphertext, so this will be a good avalanche effect, and hence higher avalanche effect signifies higher security. The mathematical equation of the avalanche effect is shown in (3):

$$\text{Avalanche Effect} = \frac{\text{NO. of change bits in cipher text}}{\text{NO. of bits in cipher text}} \times 100\% \quad (3)$$

Here, for a particular plaintext block (keeping the key constant), the hamming distance (HD) is changed randomly up to 5 bit during observation and form five different test cases and found satisfactory results as shown in Table 5. After the avalanche effect analysis, the average change in ciphertext is found to be 52.042%. In information theory, the Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In other words, it measures the minimum number of substitutions required to change one string into the other or the minimum number of errors that

could have transformed one string into the other. In a more general context, the Hamming distance is one of several string metrics for measuring the edit distance between two sequences.

Table 5. Avalanche effect for proposed LWC- AES

		HD in bit					
		1	2	3	4	5	Average
Test Case	1	52.88	56.92	58.31	49.49	52.17	53.954
	2	52.81	51.36	50.81	50.00	53.94	51.784
	3	50.04	54.11	50.62	51.08	52.13	51.596
	4	55.61	50.08	49.64	52.33	49.67	51.466
	5	53.01	51.64	50.00	52.07	50.34	51.412

7.2. Cipher-text expansion

Refers to the length increase of a message when it is encrypted. Many modern cryptosystems cause some degree of expansion during the encryption process [27]. It explained the size of cipher text has an effect on storage and transmission cost. Table 6 show the cipher text length and size for both the used AES and the proposed LW-AES. This mean the proposed algorithm has minimum storage and transmission cost.

Table 6. The security Cipher-text Expansion Analysis

Algorithm	Average cipher size in bytes
AES	50.6 bytes
LW-AES	30.6 bytes

8. CONCLUSION

The improvement and development of lightweight cryptography algorithms are necessary to improve the security of constrained devices and embedded systems, such as IoT devices, sensor networks. In this paper, we enhancement processes and improve the AES algorithm to reduce algorithm computation and improve cryptographic performance. Together with shows that the proposed LWC-AES algorithm is directly proportional to high security and low memory and energy consumption and proves to be better as for time execution or latency, with low execution time and less computation power as the memory central processing unit (CPU) usage. Its code is fast and compact on various platforms, and its design is simple, not complicated, based on the ASP.Net Visual studio C# programming language. The programming result showed the proposed LWC-AES text data type of CPU usage idle, 348.7 ms, memory 24 bytes, RAM 326 bytes, code size 587 bytes and bandwidth 2.665 Mbps. Furthermore, the results of LW-AES files data type CPU usage idle 363.6 ms, memory 24 bytes, RAM 433 bytes, code size 1870 bytes and bandwidth 0.387 Mbps. In addition to the better results registered in security analysis with avalanche effect parameter.

REFERENCES





- [1] A. M. Abed and A. Boyaci, "A lightweight cryptography algorithm for secure smart cities and IOT," *Electrica*, vol. 20, no. 2, pp. 168-176, 2020, doi: 10.5152/electrica.2020.20002.
- [2] A. S. Coronado, "Computer Security: Principles and Practice, Second Edition," *Journal of Information Privacy and Security*, vol. 9, no. 2, pp. 62-65, 2013, doi: 10.1080/15536548.2013.10845680.
- [3] P. Singh, B. Acharya, and R. K. Chaurasiya, "Lightweight cryptographic algorithms for resource-constrained IoT devices and sensor networks," *Security and Privacy Issues in IoT Devices and Sensor Networks*, pp. 153-185, 2021, doi: 10.1016/B978-0-12-821255-4.00008-0.
- [4] H. H. Alyas and A. A. Abdullah, "Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps," in *Conference paper Next Generation of Internet of Things*, 2021, vol. 201, pp. 91-107, doi: 10.1007/978-981-16-0666-3_10.
- [5] Nayancy, S. Dutta, and S. Chakraborty, "A survey on implementation of lightweight block ciphers for resource constraints devices," *Journal of Discrete Mathematical Sciences and Cryptography*, 2020, doi: 10.1080/09720502.2020.1766764.
- [6] A. Ayoub, R. Najat, and A. Jaafar, "A lightweight secure CoAP for IoT-cloud paradigm using elliptic-curve cryptography," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 3, pp. 1460-1470, Dec. 2020, doi: 10.11591/ijeecs.v20.i3.pp1460-1470.
- [7] M. Saffkhani, S. Rostampour, Y. Bendavid, and N. Bagheri, "IoT in medical & pharmaceutical: Designing lightweight RFID security protocols for ensuring supply chain integrity," *Computer Networks*, vol. 181, Nov. 2020, doi: 10.1016/j.comnet.2020.107558.
- [8] G. H. Zhang, C. C. Y. Poon, and Y. T. Zhang, "A review on body area networks security for healthcare," *International Scholarly Research Notices*, vol. 2011, pp. 1-8, 2011, doi: 10.5402/2011/692592.
- [9] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities," in *IEEE Access*, vol. 9, pp. 28177-28193, 2021, doi: 10.1109/ACCESS.2021.3052867.
- [10] H. Martín, P. Peris-Lopez, J. E. Tapiador, and E. San Millán, "An Estimator for the ASIC Footprint Area of Lightweight Cryptographic Algorithms," in *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1216-1225, May 2014, doi: 10.1109/TII.2013.2288576.

Enhancement process of AES: a lightweight cryptography algorithm-AES for ... (Hussein M. Mohammad)





- [11] T. X. Meng and W. Buchanan, "Lightweight cryptographic algorithms on resource-constrained devices," *Preprints*, Sep. 2020, doi: 10.20944/preprints202009.0302.v1.
- [12] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight cryptography for IoT: A state-of-the-art," *arXiv*, pp. 1-19, 2020. [Online]. Available: <https://arxiv.org/ftp/arxiv/papers/2006/2006.13813.pdf>
- [13] National Institute of Standards and Technology, *Advanced encryption standard (AES)*, Gaithersburg, MD, USA: Federal Information Processing Standards, 2001. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf>
- [14] W. Stallings, *Cryptography and network security: Principles and practice*. LDN, UK: Pearson, 2014. [Online]. Available: <https://www.pearson.com/us/higher-education/product/Stallings-Cryptography-and-Network-Security-Principles-and-Practice-6th-Edition/9780133354690.html>
- [15] S. S. Dhanda, B. Singh, and P. Jindal, "Lightweight cryptography: A solution to secure IoT," *Wireless Personal Communication*, vol. 112, pp. 1947-1980, Jun. 2020, doi: 10.1007/s11277-020-07134-3.
- [16] Ritambhara, A. Gupta, and M. Jaiswal, "An enhanced AES algorithm using cascading method on 400 bits key size used in enhancing the safety of next generation internet of things (IOT)," *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 2017, pp. 422-427, doi: 10.1109/CCAA.2017.8229877.
- [17] P. Kawle, A. Hiwase, G. Bagde, E. Tekam, and R. Kalbande, "Modified Advanced Encryption Standard," *International Journal Soft Computing Engineering*, vol. 4, no. 1, pp. 21-23, 2014. [Online] Available: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.495.3243&rep=rep1&type=pdf>
- [18] C. A. Sari, G. Ardiansyah, D. R. I. M. Setiadi, and E. H. Rachmawanto, "An improved security and message capacity using AES and Huffman coding on image steganography," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 5, pp. 2400-2409, doi: 10.12928/TELKOMNIKA.v17i5.9570.
- [19] K. Kumar, K. R. Ramkumar, and A. Kaur, "A lightweight AES algorithm implementation for encrypting voice messages using field programmable gate arrays," *Journal of King Saud University-Computer and Information Science*, pp. 1-15, 2020, doi: 10.1016/j.jksuci.2020.08.005.
- [20] F. Hazzaa, A. M. Shabut, N. H. M. Ali, and M. Cirstea, "Security scheme enhancement for voice over wireless networks," *Journal of Information Security and Applications*, vol. 58, May 2021, doi: 10.1016/j.jjsa.2021.102798.
- [21] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback, "Report on the development of the Advanced Encryption Standard (AES)," *Journal of Research of the National Institute of Standards and Technology*, vol. 106, no. 3, pp. 511-577, 2001, doi: 10.6028/jres.106.023.
- [22] H. Zodpe and A. Sapkal, "An efficient AES implementation using FPGA with enhanced security features," *Journal of King Saud University-Engineering Sciences*, vol. 32, no. 2, pp. 115-122, 2020, doi: 10.1016/j.jksues.2018.07.002.
- [23] A. M. Kane, "On the use of continued fractions for stream ciphers," *IACR Cryptology ePrint Archive*, 2009. [Online]. Available: <https://eprint.iacr.org/2013/319.pdf>
- [24] H. Chon, "A short proof of the simple continued fraction expansion of e," *The American Mathematical Monthly*, vol. 113, no. 1, pp. 57-62, Jan. 2006. [Online]. Available: https://www-fourier.ujf-grenoble.fr/~marin/une_autre_crypto/articles_et_extraits_livres/Cohn_H_A_Short_proof_of_the_simple_convergent_of_e.pdf
- [25] K. N. Prasetyo, Y. Purwanto, and D. Darlis, "An implementation of data encryption for Internet of Things using blowfish algorithm on FPGA," *2014 2nd International Conference on Information and Communication Technology (ICoICT)*, 2014, pp. 75-79, doi: 10.1109/ICoICT.2014.6914043.
- [26] A. Biswas, A. Majumdar, S. Nath, A. Dutta, and K. L. Baishnab, "LRBC: a lightweight block cipher design for resource-constrained IoT devices," *Journal of Ambient Intelligence Humanized Computing*, 2020, doi: 10.1007/s12652-020-01694-9.
- [27] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, "Report on Lightweight Cryptography" *National Institute of Standards and Technology Internal Report 8114*, Mar. 2017, doi: 10.6028/NIST.IR.8114.

BIOGRAPHIES OF AUTHORS



Hussein M. Mohammad     received his B.S. degree in computer technics engineering from the Al-Mustaqbal University College – Babylon/ Iraq in 2014, and a higher diploma in information technology and security from the Higher Institute for Security and Administrative Development – Baghdad/ Iraq. Currently an MSc. research student in the Department of Information Networks/ College of Information Technology in Babylon University / Iraq. He can be contacted at email: hussein.moksad@uobabylon.edu.iq.



Alharith A. Abdullah     received his B.S. degree in Electrical Engineering from Military Engineering College, Iraq, in 2000. MSc. degree in Computer Engineering from University of Technology, Iraq, in 2005, and his PhD. in Computer Engineering from Eastern Mediterranean University, Turkey, in 2015. His research interests include Security, Network Security, Cryptography, Quantum Computation and Quantum Cryptography. He can be contacted at email: alharith@inet.uobabylon.edu.iq.