International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 3, June 2020, pp. 3125~3135
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i3.pp3125-3135

New modification on feistel DES algorithm based on multi-level keys

Suhad Muhajer Kareem[1], Abdul Monem S. Rahma[2]
[1]Department of Computer Science, University of Basrah, Iraq
[2]Department of Computer Science, University of Technology, Iraq

Article Info

Article history:

Received Jun 7, 2019
Revised Sep 28, 2019
Accepted Dec 6, 2019

Keywords:

Des
Information security encryption
Multi-level keys
Symmetric block cipher

ABSTRACT

The data encryption standard (DES) is one of the most common symmetric encryption algorithms, but it experiences many problems. For example, it uses only one function (XOR) in the encryption process, and the combination of data is finite because it occurs only twice and operates on bits. This paper presents a new modification of the DES to overcome these problems. This could be done through adding a new level of security by increasing the key space (using three keys) during the 16 rounds of the standard encryption algorithm and by replacing the predefined XOR operation with a new # operation. Our proposed algorithm uses three keys instead of one. The first key is the input key used for encrypting and decrypting operations. The second key is used for determining the number of bits, while the third key is used for determining the table numbers, which are from 0 to 255. Having evaluated the complexity of our proposed algorithm, the results show that it is the most complex compared with the well-known DES and other modified algorithms. Consequently, in our proposed algorithm, the attacker try a number of attempts 2^{1173} at minimum to decrypt the message. This means that the proposed DES algorithm will increase the security level of the well-known DES.

Corresponding Author:

Suhad Muhajer Kareem,
Department of Computer science,
University of Basrah,
Basrah, Iraq.
Email: suhad_althaher@yahoo.com

1. INTRODUCTION

Nowadays, the rapid growth of electronic data transferred via unsecure networks makes information security one of the most important challenges. Thus, it is necessary to apply encryption algorithms on electronic data to protect the data from attackers [1, 2]. Moreover, in any encryption system shared by two parties, the sender uses an encryption algorithm to convert the original text into ciphertext, while the receiver uses the reverse process for retrieving the original text [3, 4].

Generally, algorithmic encryption is classified into two categories: *symmetric-key* and *asymmetric-key* encryption. Symmetric algorithms are of two types: block ciphers and stream ciphers. Stream ciphers encrypt each bit individually by adding a bit from a key stream to a plaintext bit. Block ciphers encrypt an entire block of plaintext bits at a time with the same key. Block ciphers could be based on the Feistel network, such as the data encryption standard (DES) and a non-Feistel network, such as an advance encryption system (AES) [5-7].

The Feistel network was invented by Horst Feistel and has been used in many block cipher designs. It can be defined as a common method of converting a function, which is usually called an F function, into a permutation. The principle work of Feistel networks is to combine the numbers of rounds to reiterate the same operations, such as bit-shuffling (using boxes for permutations called P-boxes), simple non-linear

functions (substitution boxes or S-boxes), and a logical operation, which is predefined as XOR (using two states, 0 or 1) [8-10].

In this paper, we focus on the Feistel network applied in the DES algorithm. It is the earliest symmetric encryption algorithm developed by IBM in 1972 and was adopted in 1977 as the Federal Information Processing Standard by the National Bureau of Standards. The DES algorithm is best suitable for implementation in hardware. Conversely, it tends to demonstrate slow implementation in software. The algorithm accepts 64-bit keys, where only 56 bits have previously been used because the remaining 8 bits were used for error detection purposes [11-13].

The principle work of the DES algorithm can be summarised as follows: input the plaintext (64 bits) of the algorithm, which is processed with an initial permutation, then 16 rounds of the key and plaintext are applied, and the inverse permutation is the final step in the algorithm. The structure of the DES algorithm is based on the Feistel network, which divides the input plaintext (64 bits) block into two halves: left (32 bits) and right (32 bits) [14, 15].

The core work of the DES Feistel is the F function, which is key-dependent and consists of four phases as follows [16-18]:

- Expansion phase: The 32-bit input word is expanded into 48 bits by duplicating and reordering the bits of the word.
- Key mixing phase: The result word from the previous phase uses XOR with a round key constructed by selecting 48 bits from the 56-bit key, and in each round, a different selection of bits is used.
- Substitution phase: The step uses eight S-boxes to map the 48 bit for producing 32 new bit.
- Permutation phase: The 32-bit result from the S-boxes is reordered according to a fixed permutation choice table.

In the undo step, the modified right block then uses XOR operation with the left block, and the result from this step is provided in the next right block. The unmodified right block is fed to the next left block register. The same process is iterated sixteen times for making sixteen rounds of DES [19]. The work of well-known DES algorithm in steps cited in [16, 20]:

Any encryption algorithm depends on the key as the significant element can be “defined as a “numeric, “alpha numeric text or special symbol [11, 21]. Most modern cryptographic algorithms “depend on functions “with two “states (0, 1) for “encryption and decryption. DES, as one of the block algorithms uses the classical logical operation (XOR) which depends on two states: simply (0, 1) which has several weak points lead to break the DES, such as being simple where it can be deciphered easily by attackers. Consequently researchers have attempted to replace the two states with four “ones (0, 1, 2, 3) as shown in Figure 1 in the following sections for increasing key space [22]. In this paper, we focus on the weak points of XOR by replacing it with a new # operation with variable block bit sizes (n): (1 or 2 or 4 or 8) instead of one block size. Each block will generate different states tables based on addition in GF (2^n). The overall new # operation is managed by using additional two keys. This work is repeated in each round of DES to increase the security level of the algorithm. Our results show that this new modification on DES algorithm will increase the security level of the encryption by increasing the complexity in each round and thus the protection of encrypted messages will be guaranteed.

#0	0	1	2	3
0	3	2	1	0
1	2	3	0	1
2	1	0	3	2
3	0	1	2	3

#1	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

#2	0	1	2	3
0	2	3	0	1
1	3	2	1	0
2	0	1	2	3
3	1	0	3	2

#3	0	1	2	3
0	1	0	3	2
1	0	1	2	3
2	3	2	1	0
3	2	3	0	1

Figure 1. Truth tables for the # operation

2. RELATED WORK

This section presents the overview of the related literature on various modifications of the DES algorithm and uses the truth tables in the key distribution. In 2009 [22], researchers presented the work by combining the curve security methods with quantum cryptography concepts to increase the security and key space to make the encryption operation more secure and robust. In this work, the proposed modification focuses on the use of four different states (0, 1, 2, and 3) instead of two (0 and 1). This is to make variations in the polarised angles that have been used in the quantum description encoded in these four tables in addition to the output descriptions that have used polarised state angles according to the tables. Then, manipulation ciphers convert the plaintext into ciphertext by changing the actual state pattern of each character using a logical operator (#). The operator # has the following figure truth tables.

The work of the # operation involves three inputs. The first input refers to the table number, which should be used to compute the result among the four tables. The other two inputs determine the row and column numbers in the given table to give the result as a cross point. In 2010, [23] researchers introduced a proposal for a new method to improve the performance of the DES algorithm. This improvement is demonstrated by replacing the predefined XOR operation applied during the 16 rounds in the standard algorithm Feistel with a new # operation that depends on using two keys. Each key consists of a combination of four states (0, 1, 2, and 3) instead of the ordinary two-state keys (0 and 1) using different truth tables proposed in [22] Figure 1. The first key is used to determine the table number among the four tables, and the second key is used in the encryption algorithm. This replacement adds a new level of security to the algorithm against attackers through increasing the level of complexity. In our proposed method, we operate on multi-states as combination of 0- or 1- or 4- or 8- bits for representing (0, 1, 2, 3, … 255) while the proposed in [23] operate only 2-bits to represent (0, 1, 2, 3). Consequently, our proposed more randomness and more security. In 2017, [5] the authors have proposed a new modification on DES by extended the standard bit size from 64–bit to 128-bit for both plaintext and key size in order to increase the security of algorithm. This is done by doubling the size of tables, function and keys. By increasing the overall size in cipher will made the algorithm stronger against brute force attack.

3. PROPOSED IMPROVEMENT OF THE DATA ENCRYPTION STANDARD

The DES is considered unsecure for many applications for several reasons. It primarily depends on only a single bit (0 or 1). Similarly, it uses only one function (XOR), as it does not contain enough randomness and is vulnerable to attacks. Therefore, to overcome these problems, in this section, we introduce a new method to modify the DES to improve the encryption performance and make the algorithm more complex against attacks. This is can be achieved by making a modification on a binary function and key generation using multiple keys in each round of the standard DES algorithm instead of one. Each key is generated independently. The first key is the input key used for encrypting and decrypting operations. The second key is generated randomly in binary format called the key $_{\text{number of bits}}$, which is used to determine the number of bits (block bit size) taken from key and message. The third key is called key $_{\text{no. of table}}$, which is used to select one state table among different state tables that is used to apply the # operation. This work is done by replacing the XOR function with a new logical operation called the # operation. This # operation needs three inputs: the first one specifies the state table number, which should be used to calculate the result among different state tables. The other two inputs identify the row and column numbers in the specified state table where their cross point gives the results. However, the number of tables with more states are used in this work to increase the randomness in the algorithm.

In this paper, a new manipulation of the bit process has been introduced because the well-known DES algorithm is based on XOR, which operates only on (0,1), whereas the proposed algorithm uses a new operation (#), which works on different truth state tables. These state tables are generated in the same manner as for the tables of the previous section shown in Figure 1, yet with more spaces. These tables are mainly constructed based on the addition operation in the Galois field GF(2^n), where n is the value depend on the block bit size that specify by the key $_{\text{number of bits}}$. In our work, four variable block bit size:1,2,4 and 8 are used, there are 2-state tables (0 and 1) for GF (2^1), 4-state tables (0, 1, 2, and 3) for GF (2^2), 16-state tables (0, 1, …, 15) for GF (2^4), and 256 –state tables (0, 1, …, 255) for GF (2^8), the samples of theses tables shown in the next section Tables 1 to 6. The following examples illustrate the process how generate state tables based on block bit size and select one state from them.

Let, K_b = key $_{\text{number of bits}}$, and K_c=key $_{\text{no. of table}}$, then:

- $K_{b\,=\,000110110101010}$ … (generated randomly for encryption and decryption);
- *At each round, take two bits from the K_b and check it*:
- If K_b = 00, then the block bit size=1 and recall 2- state tables: K_c select randomly one table either 0 or 1 for encrypting and decrypting.

- If K_b = 01, then the block bit size=2 and recall 4-state tables: K_c select randomly one table among (0, 1, …, 4) for encrypting and decrypting;
- If K_b = 10, then the block bit size=4 and recall 16-state tables: K_c select randomly one table among (0, 1, …, 15) for encrypting and decrypting;
- If K_b = 11, then the block bit size=8 and recall 256-state tables: K_c select randomly one table among (0, 1, …, 255) for encrypting and decrypting;

The overall process of the # operation for each round in the proposed DES algorithm as shown in Figure 2.

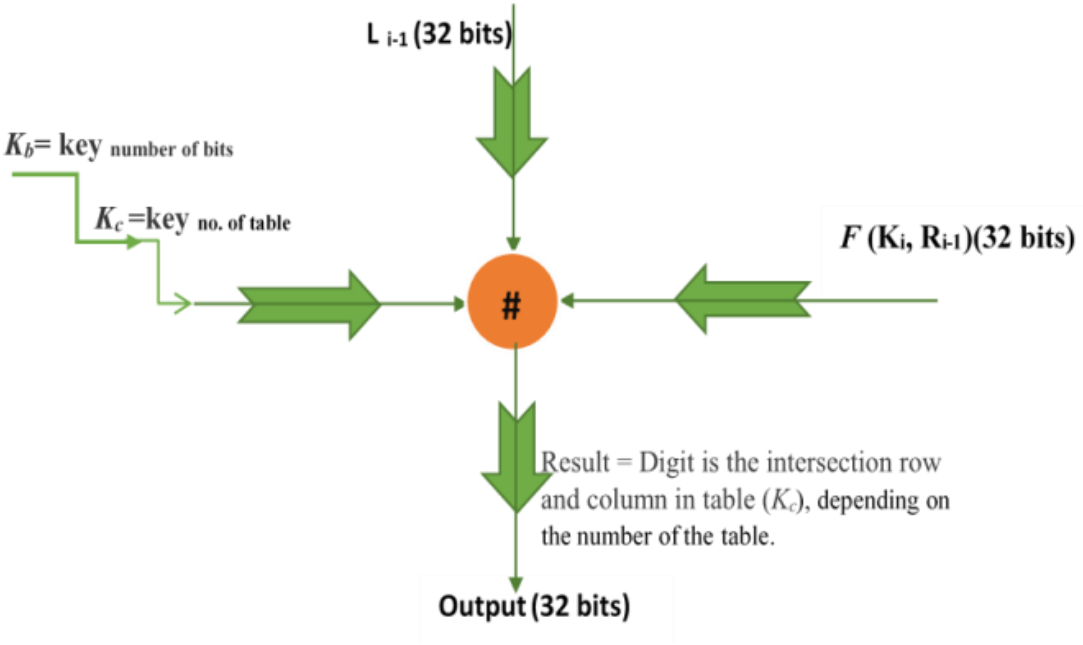

Figure 2. Process of the # operation

3.1. Construction of the state tables

This section show samples of tables that constructed based on the addition mathematical operation in Galois Field (GF (2^n)). Tables 1, 2, 3 and Tables 4, 5, 6 represent the addition in GF (2^4) and GF (2^8) consecutively.

Table 1. State (#0) addition in GF (2^4)

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
#0	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0000	**0**	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0001	**1**	1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
0010	**2**	2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
0011	**3**	3	2	1	0	7	6	5	4	11	10	9	8	15	14	13	12
0100	**4**	4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
0101	**5**	5	4	7	6	1	0	3	2	13	12	15	14	8	9	10	11
0110	**6**	6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
0111	**7**	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
1000	**8**	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
1001	**9**	9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
1010	**10**	10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
1011	**11**	11	10	9	8	15	14	13	12	3	2	1	0	7	6	5	4
1100	**12**	12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
1101	**13**	13	12	15	14	9	8	11	10	5	4	7	6	1	0	3	2
1110	**14**	14	15	12	13	10	11	8	9	6	7	4	5	2	3	0	1
1111	**15**	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 2. State (#3) addition in GF (2^4)

	#3	0	1	2	……	13	14	15
0000	**0**	3	2	1	……	12	13	14
0001	**1**	2	3	0	……	15	12	13
0010	**2**	1	0	3	……	12	15	14
0011	**3**	0	1	2		13	14	15
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.								
1100	**12**	13	12		……	0	3	2
1101	**13**	12	13	14	……	1	2	3
1110	**14**	15	14	13	……	2	1	0
1111	**15**	14	15	12	……	3	0	1

Table 3. State (#15) addition in GF (2^4)

	#15	0	1	2	……	13	14	15
0000	**0**	15	14	13	……	2	1	0
0001	**1**	2	3	0	……	15	12	13
0010	**2**	1	0	3	……	12	15	14
0011	**3**	4	5	6		9	10	11
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1100	**12**	11	10	9	……	6	5	4
1101	**13**	14	15	12	……	3	0	1
1110	**14**	13	12	15	……	0	3	2
1111	**15**	0	1	2	……	13	14	15

Table 4. State (#0) addition in GF(2^8)

		0000 0000	0000 0001	0000 0010	0000 0011	…	0001 0010	0001 0011	0011 0100	…	0111 1101	0111 1110	0111 1111	…	1111 1101	1111 1110	1111 1111
	#0	**0**	**1**	**2**	**3**	**…**	**50**	**51**	**52**	**…**	**125**	**126**	**127**	**…**	**253**	**245**	**255**
0000 0000	**0**	0	1	2	3	…	50	51	52	…	125	126	127	…	253	254	255
0000 0001	**1**	1	0	3	2	…	51	50	53	…	124	127	126	…	252	255	254
0000 0010	**2**	2	3	0	1	…	48	49	54	…	127	124	125	…	255	252	253
0000 0011	**3**	3	2	1	0	…	49	48	55	…	126	125	124	…	254	253	252
0000 0100	**4**	4	5	6	7	…	54	55	48	…	121	122	123	…	249	250	251
0000 0101	**5**	5	4	7	6	…	55	54	49	…	120	123	122	…	148	251	250
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1111 1010	**250**	250	251	248	249	…	200	201	206	…	135	132	133	…	7	4	5
1111 1011	**251**	251	250	249	248	…	201	200	207	…	134	133	132	…	6	5	4
1111 1100	**252**	252	253	254	255	…	206	207	200	…	129	130	131	…	1	2	3
1111 1101	**253**	253	252	255	253	…	207	206	201	…	128	131	130	…	0	3	2
1111 1110	**254**	254	255	252	253	…	204	205	202	…	131	128	129	…	3	0	1
1111 1111	**255**	255	254	251	252	…	205	204	203	…	130	129	128	…	2	1	0

Table 5. State (#170) addition in GF (2^8)

		0000 0000	0000 0001	0000 0010	0000 0011	0000 0100	0000 0101	……..	1111 1010	1111 1011	1111 1100	1111 1101	1111 1110	1111 1111
	#170	**0**	**1**	**2**	**3**	**4**	**5**	**……..**	**250**	**251**	**252**	**253**	**254**	**255**
00000000	**0**	170	171	168	169	174	175	……..	80	81	86	87	84	85
00000001	**1**	2	3	0	1	6	7	……..	248	249	254	255	252	253
00000010	**2**	1	0	3	2	5	4	……..	251	250	253	252	255	254
00000011	**3**	4	5	6	7	0	1	……..	254	255	248	249	250	251
00000100	**4**	3	2	1	0	7	6	……..	249	248	255	254	253	252
00000101	**5**	6	7	4	5	2	3	……..	252	253	250	251	248	249
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	,	.	.	.	.	.	.	.	.
11111010	**250**	251	250	249	248	255	254	……..	1	0	7	6	5	4
11111011	**251**	250	251	248	249	254	255	……..	0	1	6	7	4	5
11111100	**252**	253	252	255	254	249	248	……..	7	6	1	0	3	2
11111101	**253**	252	253	254	255	248	249	……..	6	7	0	1	2	3
11111110	**254**	255	254	253	252	251	250	……..	5	4	3	2	1	0
11111111	**255**	254	255	252	253	250	251	……..	4	5	2	3	0	1

Table 6. State (#200) addition in GF (2^8)

		0000 0000	0000 0001	0000 0010	0000 0011	0000 0100	0000 0101	……..	1111 1010	1111 1011	1111 1100	1111 1101	1111 1110	1111 1111
	#200	**0**	**1**	**2**	**3**	**4**	**5**	**……..**	**250**	**251**	**252**	**253**	**254**	**255**
00000000	**0**	200	201	202	203	204	205	……..	50	51	52	53	54	55
00000001	**1**	2	3	0	1	6	7	……..	248	249	254	255	252	253
00000010	**2**	1	0	3	2	5	4	……..	251	250	253	252	255	254
00000011	**3**	4	5	6	7	0	1	……..	254	255	248	249	250	251
00000100	**4**	3	2	1	0	7	6	……..	249	248	255	254	253	252
00000101	**5**	6	7	4	5	2	3	……..	252	253	250	251	248	249
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
11111010	**250**	251	250	249	248	255	254	……..	1	0	7	6	5	4
11111011	**251**	250	251	248	249	254	255		0	1	6	7	4	5
11111100	**252**	253	252	255	254	249	248	……..	7	6	1	0	3	2
11111101	**253**	252	253	254	255	248	249	……..	6	7	0	1	2	3
11111110	**254**	255	254	253	252	251	250	……..	5	4	3	2	1	0
11111111	**255**	254	255	252	253	250	251	……..	4	5	2	3	0	1

3.2. Proposed data encryption standard in steps

This section proposes the proposed DES algorithm in steps as shown hereunder. Modified steps have been highlighted with the red colour:

Algorithm 1: Proposed DES using multi-level keys

Input: Plaintext message 64 bits and key 64 bits called K.
Output: Produce cipher block of 64 bits
Begin
Step 1: Key is processed to produce sixteen (48-bit) sub_keys Ki from K as follows:
Step1.1: The 8 parity bits are removed from the key using the initial Permutation table.
Step 1.2: Split K into two halves: Ci and Di.
Step 1.3: Each half of the key is shifted by one or two bits depending on the round.
Step 1.4: The halves are recombined and subject to compression.
Step 1.5: Permutation is used to reduce the key from 56 bits to 48 bits.
Step 2: Generate randomly control key 32 bits in binary form called K_{bi} (number of bits) for16 round.
Step 3: Use the IP table to permute the bits of plaintext block (64 bits).
Step 4: The block result from Step 3 is split into two 32-bit halves (L_0, R_0) left and right.
For 16 rounds, compute L_i and R_i as follows:
Step 4.1: $L_i = R_{i-1}$
Step 4.2: $R_{i-1} = L_{i-1} \# \grave{P}$, where $\grave{P} = f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$, computed as follows:
Step 4.2.1: Expand R_i-1 = r1, r2,..., r32 from 32 to 48 bits $T \leftarrow E(R_{i-1})$).
Step 4.2.2: Apply the XOR operation T, $T` \leftarrow (T \oplus K_i)$
Step 4.2.3: Output of Step 4.2.2 is fed into an S-box, which substitutes key bits and reduces the 48-bit block back down to 32 bits. $\acute{T} \leftarrow S(T`)$.
Step 4.2.4: Output of Step 4.2.3 is subject to a P-box to permute, $\grave{P} \leftarrow (P(\acute{T}))$.

a. Compute the operation # in R_i-1 = L_i-1# $\grave{P}$ as follows:
b. Split K_{bi} to two bits (N_b), then test as follows:
where n is the block bit size selected from L_i and $\grave{P}$
1. If N_b = 00, then n = 1; go to b;
2. If N_b = 01, then n = 2; go to b;
3. If N_b = 10, then n = 4; go to b;
4. If N_b = 11, then n = 8; go to b.
c. Depending on the previous step, recall states tables using addition on GF (2^n).
d. Generate random key (K_{ci}) for selecting one state table that create in step (c).
e. Compute R_i by applying the operation on $L_{i-1} \# \grave{P}$ according to three inputs (index = number of state table, row = L_{i-1}, and column = $\grave{P}$).
f. The output of step (d) as the cross point between the row and column in the specified state table to give the result.

End for
Step 5: Exchange final blocks L_{16}, R_{16}.
Step 6: Transpose the results using inverse permutation (IP^{-1}).
End.

4. SIMULATION RESULTS

Cryptography is the science that provides secure communication over unsecure channels. The message is encrypted using the key by applying mathematical operations to produce the ciphertext. Consequently, without knowing the key, an attacker as a third party cannot calculate the message from the ciphertext. Moreover, the number of attempts to estimate the key by the attackers can be defined as brute force attacks. As an illustration, one of the algorithms that is vulnerable to this type of attack is the DES algorithm [24, 25]. Thus, this paper introduces a new modification of the DES algorithm. This is to enhance the security level by increasing the key space using multiple control keys to determine the number of bits used from the block to encrypt into the specified table. Hence, it will be very difficult to estimate the key. This section presents three metrics evaluation (complexity, encryption time, throughout, NIST tests and histogram analysis) of the proposed DES algorithm as shown below, where the simulation of the algorithm is done to perform the evaluation tests on Intel Core i7-8550U@2.00 GHz processor using Microsoft visual studio c# 2017.

4.1. Security complexity analysis

We calculate the complexity of the proposed algorithm by computing the number of possibilities of keys, which the attacker needs to decrypt the cipher-text with 64 bits using three keys with four blocks of (1, 2, 4, or 8) bit size and different state tables. First, we compute the complexity of the well-known DES algorithm using a predefined XOR binary operation (0, 1); thus, computing the number of possible keys used in the encryption and decryption is calculated as follows:

$$2 \times (2)^8 \times 32 \times 2 = 2 \times (2)^8 \times 2^5 \times 2 = 2^{15} \quad (1)$$

Second, when using the # operation n the modified DES algorithm [23] with four states (0, 1, 2, and 3) and two bits instead of one bit, the number of key possibilities used in the encryption and decryption is computed as follows:

$$(2^2)^{16} \times (2^2)^8 \times 2^2 \times 32 \times 2 = 2^{32} \times 2^{32} \times 2^2 \times 2^5 \times 2 = 2^{72} \quad (2)$$

Finally, we compute the complexity of the proposed algorithm using three keys. The overall complexity of our proposed algorithm is as follows:

$$((21)32 \times (22)16 \times (24)8 \times (28)4) \times ((21)32 \times (22)16 \times (24)8 \times (28)4)8 \times$$
$$(2 \times 22 \times 24 \times 28) \times 32 \times 2 = 2128 \times 21024 \times 215 \times 25 \times 21 = 21173 \quad (3)$$

Table 7 summarises the results based on computing the complexity as the comparison between our proposed algorithm with the well-known DES algorithm and the modified algorithm cited in [23]. The findings show that our proposed algorithm is more complex than the others. Figure 3 shows the security complexity of proposed DES for 16 round with two algorithms (well-known DES and DES modified with 4-states cited in [23]. These results have shown in Table.7 and Figure 3 proved that our proposed algorithm has been more complex than others. Consequently, our proposed become stronger against brute force attacks.

4.2. Encryption time and throughput

As another metric for measuring the performance of the algorithm, the encryption time is computed by the time required for converting the plaintext into an unrecognised form. The throughput metric as applied in this context is calculated as [26]:

$$\text{Throughput} = \text{plaintext size (in kilobyte)}/\text{total encryption time (ms)} \quad (4)$$

From these Table 8 and Figure 4, the original, modified, and the proposed DES algorithms are equivalent in terms of computation time. However, our proposed method offers more effective results related to the complexity evaluation against attacks, which enables our DES algorithm to be more difficult for an attacker to retrieve the original message.

4.3. Nist tests analysis

The output of the encryption algorithm should be more random and unpredictable. Several methods exist for computing the randomness, such as NIST (National Institute of Standards and Technology), Diehard tests, and TestU01. In this paper, we use 15 statistical tests from NIST statistical for testing the "randomness" of the binary sequences, as shown in Table 9. This and the modified tests are calculated over multiple cyphertext produced from the well-known DES. The probability value (p-value) is set to a value of 0.01 to confirm if the output is random. The average tests are computed and listed in Table 9.

If the test results provide a p-value "asymptotically approaching 1, then the output should appear to have complete randomness. A p-value equal to zero signifies that the output is non-random. The pass status represents that the p-value of these tests is greater than 0.001 and denotes the output is acceptable (e.g., offers good randomness). The p-values of most of the tests from the proposed DES algorithm are greater than the p-values of the well-known DES, as shown in Table 9. Consequently, the proposed DES is better than the original DES in most tests.

Table 7. The results of security complexity analysis

Algorithm	The complexity
Well-known DES	2^{15} =32,768
Modified DES [23]	2^{72} =4,722,366,482
Our proposed DES	2^{1173}=1.2828766894627921743741 1e+353

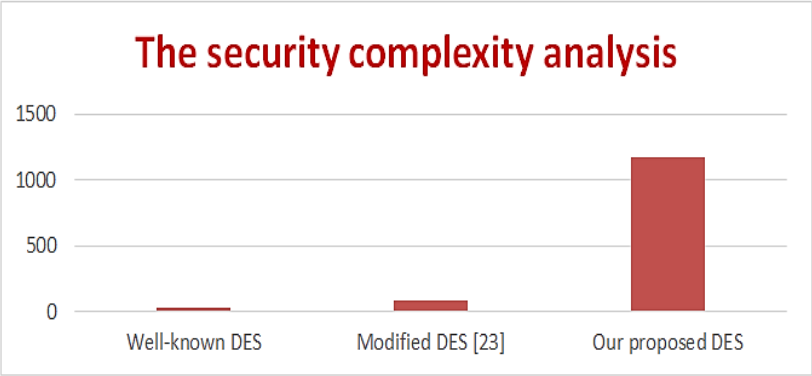

Figure 3. The security complexity analysis

Table 8. The encryption time and throughput of original, previously modified, and proposed DES algorithms

File sizes(kb)	Execution times (in milliseconds)for:		
	Original DES algorithm	Modified DES algorithm in[11]	Our proposed DES algorithm
10	5	5	5
50	13	14	14
100	29	29	30
450	113	113	117
Average time	40	40.25	41.5
Throughput	3.815	3.788	3.674

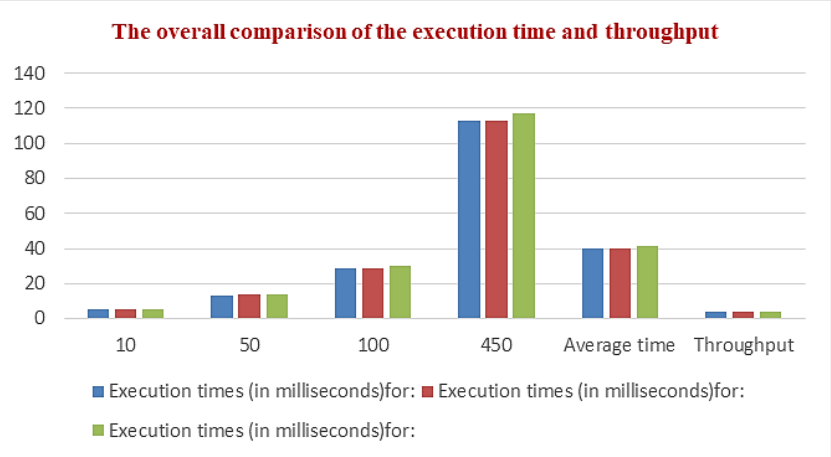

Figure 4. The overall comparison of the execution time and throughout

Table 9. Result of Running NIST on the generated key by DES and the proposed DES

Test no.	Statistical Test Name	Well-known DES		Proposed DES	
		P-Value	Status	P-Value	Status
1	Approximate Entropy	0.624	pass	0.910	pass
2	Block Frequency	0.639	pass	0.849	pass
3	Cumulative Sums	0.068	pass	0.430	pass
4	FFT	0.082	pass	0.562	pass
5	Frequency	0.116	pass	0.310	pass
6	Linear complexity	0.884	pass	0.623	pass
7	Longest Run	0.25	pass	0.847	pass
8	Non Overlapping Template	0.527	pass	0.517	pass
9	Overlapping Template	0.480	pass	0.787	pass
10	Random Excursions	0.591	pass	0.757	pass
11	Random Excursions Variant	0.761	pass	0.630	pass
12	Rank	0.432	pass	0.434	pass
13	Runs	0.001	pass	0.388	pass
14	Serial	0.649	pass	0.670	pass
15	Universal	0.326	pass	0.973	pass

4.4. Histogram analysis

A histogram is used to measure the security of the original, encrypted, and decrypted images using the well-known and proposed DES. The experimental results of three images are shown in Figures 5, 6 and 7.

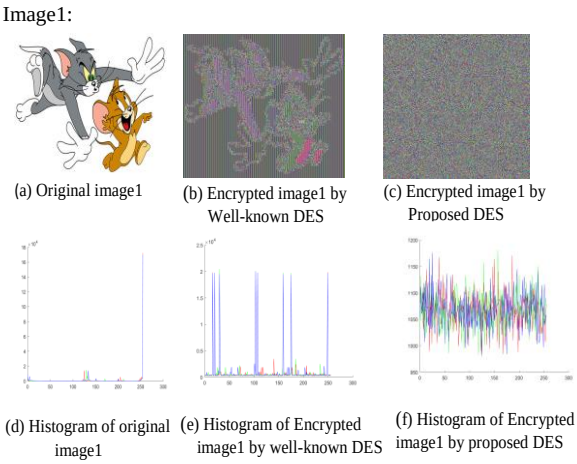

Figure 5. Results and histogram of well-known DES and proposed DES for original image1 and encrypted image1

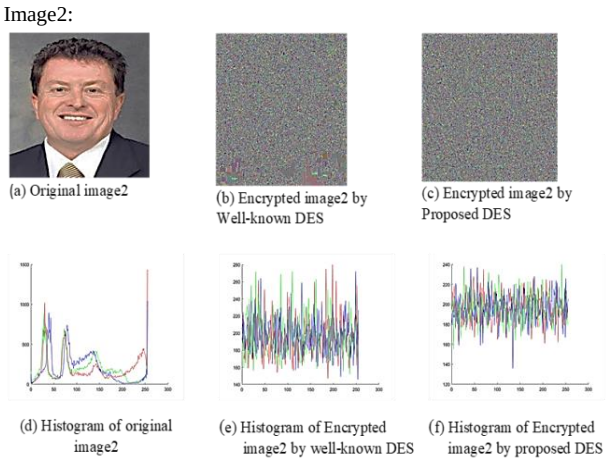

Figure 6. Results and histogram of well-known DES and proposed DES for: original image2 and encrypted image2

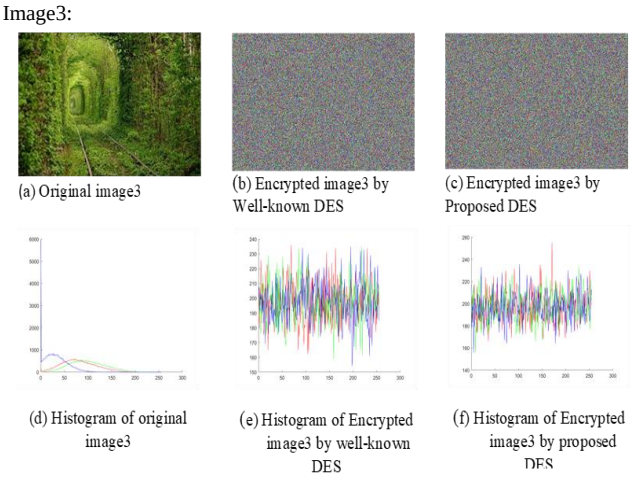

Figure 7. Results and histogram of well-known DES and proposed DES for original image3 and encrypted image3

4.5. Discussion the results

Our proposed DES uses three keys through 16 rounds with each round using three keys, including an input key for encryption and decryption, a second key for determining the number of blocks of 1-, 2-, 4- or 8-bits then generating the number of state tables, and a third key to select one state table for encryption and decryption. This approach indicates that using these coefficient parameters (i.e., three keys with dynamic block sizes) in each round increases the complexity of our proposed DES against attack. When comparing with a triple DES, which works by using DES three times, then we use only three keys through a total of 48 rounds. Moreover, our proposed DES manipulates the bits with different states since the triple DES manipulates the bits only with two states (0 and 1). So, the complexity of the triple DES is less than our proposed DES.

From the histogram analysis, the correlation between the pixels in an image allows us to select three different images. In image1, the correlation between the pixels is high while being less in image2. In image3, little correlation exists. The effect of the dynamic block size is selected each time for encryption and decryption. As shown in the previous figures, the distribution of the pixels in our proposed DES is equivalent to a uniform distribution suggesting that it is stronger compared to the original DES for these three types of images.

5. CONCLUSION

The DES is one of the most popular and earliest encryption algorithms that has been used until very recently. The DES is considered insecure for many applications due to its many weaknesses. Examples of such weaknesses include the key length, using only one function, containing less randomness, etc. Thus, it is necessary to increase the security of this algorithm by adding new levels of security to make it more secure. An additional key is added, and the old XOR is replaced by a new operation called the # operation, with more truth tables. In this paper, this change is suggested to give more strength to the DES algorithm. This will make it more powerful against any kind of snooping. Using multi-keys instead of one key increases the reliability of the key. Using a variable block bit size in each round increase the security of the algorithm. However, this will increase the efficiency of the encryption and decrease the probabilities of a break against differential analysis from brute force attacks. The modification of the DES algorithm adds complexity in computing the key but saves the time taken in mathematical computation, as shown in the previous section.

REFERENCES

[1] P. Patil., P. Narayankar, DG. Narayan., and Meena S., "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish," *International Conference on Information Security & Privacy* (ICISP2015), 11-12 December 2015, Nagpur, India, *Procedia Computer Science*, vol. 78, pp. 617–624, 2016.

[2] Sh. Kandar, Dh. Chaudhuri, A. Bhattachajee, B. Chandra, "Image encryption using sequence generated by cyclic group," *Journal of Information Security and Applications*, vol. 44, pp. 117-129, 2019.

[3] S. Rani and H. Kaur, "Technical Survey on Cryptography Algorithms for Network Security," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6, no. 9, pp. 204-209, Sep. 2016.

[4] A. M. Abdullah, "Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data," *Cryptography and Network Security*, 2017.

[5] B. cruz and K. Domingo, "Expanded 128-bit Data Encryption Standard," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 6, Aug. 2017.

[6] William Stallings, "Cryptography and network Security: Principles and Practice," Pearson Education/Prentice Hall, 5th Edition, 2010.

[7] M. A. Hameed, Ahmed I. Jaber, Jamhoor M. Alobaidy, and Alaa A. Hajer, "Design and Simulation DES Algorithm of Encryption for Information Security," *American Journal of Engineering Research (AJER)*, vol. 7, no. 4, pp. 13-22, 2018.

[8] E. Thambiraja, G. Ramesh, "A Survey on Various Most Common Encryption Techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 7, pp. 226-223, Jul. 2012.

[9] HN. Shandi, A.S. Raut, Sh. R. Vidhale, R.V. Sawant, V.A. Kotkar, "A Review of Various Encryption Techniques," *International Journal Of Engineering And Computer Sci*ence, vol. 3, no. 9, pp. 8092-8096, Sep. 2014.

[10] S. Albermany, F. Radi, "Survey: Block cipher Methods," *International Journal of Advancements in Research & Technology*, vol. 5, no. 11, pp. 11-22, Nov. 2016.

[11] G. Singh, Supriya, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security," *International Journal of Computer Applications*, vol. 67, no. 1, pp. 33-38 Apr. 2013.

[12] S. D. Rihan, A. Khalid, S.E. F, Osman, "A Performance Comparison of Encryption Algorithms AES and DES," *International Journal of Engineering Research & Technology (IJERT)*, vol. 4, no. 12, pp. 151-154, Dec. 2015.

[13] M. Kaur, N. Kaur, B. Singh, "Comparative Study Of Different Cryptoghraphic Algorithms," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 4, pp. 352-354, May 2017.

[14] M. F. Mushtaq, Sapiee J., Abdulkadir H. Disina, Z. A. Pindar, N. Sh. Ahmed, M. M. Deris, "A Survey on the Cryptographic Encryption Algorithms," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 8, no. 11, pp. 333-344, Nov. 2017.
[15] A. Antontony, "Performance Analysis of Data Encryption Algorithms for Secure Data Transmission," *International journal for science and advanced research in technology (IJSART*), vol. 2, no. 12, pp. 388-390, Dec. 2016.
[16] P. Patel, K. Shah, Kh. Shah,"Enhancement Of Des Algorithm With Multi State Logic," *International Journal of Research in Computer Science*, vol. 4, no. 3, pp. 13-17, 2014.
[17] Sh. Kruti, Bh. Gambhava, "New Approach of Data Encryption Standard Algorithm," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 1, Mar. 2012.
[18] B. Rajesh, "Real time implementation of des algorithm by using tms3206713 dsk," a thesis submitted in partial fulfillment of the requirements for the degree of master of technology in telematics & signal processing, Rourkela, 2008.
[19] S. Singh, S. Maakar, S. Kumar, "Enhancing the Security of DES Algorithm Using Transposition Cryptography Techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 6, pp. 464-471, Jun. 2013.
[20] A. Kumar, H. Sharma, "New Approach to DES with Enhanced Key Management and Encryption/Decryption System," *International Journal of Advances in Engineering & Technology*, vol. 8, no. 3, pp. 368-377, Jun. 2015.
[21] P. Verma, J. Shekhar, Preety, A. Asthana, "A Survey for Performance Analysis Various Cryptography Techniques Digital Contents," *International Journal of Computer Science and Mobile Computing*, vol. 4, no. 1, pp. 522-531, Jan. 2015.
[22] H. Bahjat, A. S. Rahma, "Proposed New Quantum Cryptography System Using Quantum Description techniques for Generated Curves," *The 2009 International conference on security and management*, SAM2009, July 13-16 2009, LasVegas, USA, SAM 2009.
[23] R. Faleh, "New Approach for Modifying DES Algorithm using 4-States Multi keys," *Eng. & Tech. Journal*, vol. 28, no. 20, 2010.
[24] B. Zyacob, *A. S. Rahma*, "An improved algorithm for partial cryptography of Digital video," A thesis to university of technology, 2012.
[25] N. Mavrogiannopoulos, "Secure communications protocols and the protection of cryptographic keys," Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering, Royal Holloway, and University of London, Jun. 2013.
[26] D. S. Abdul, Elminaam, H. M. Abdul Kader, M. M. Hadhoud, "Performance Evaluation of Symmetric Encryption Algorithm," *Communications of the IBIMA*, vol. 8, pp. 58-64, 2009.