Middle-East Journal of Scientific Research 12 (5): 662-673, 2012
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2012.12.5.1685

# A New Symmetric-Key Block Ciphering Algorithm

*Jamal N. Bani Salameh*

Department of Computer Engineering, Faculty of Engineering,
Mu'tah University, Al-Karak, Postal Code 61710, Jordan

**Abstract:** Strong cryptographic algorithms have been used for securing data communication. It is considered as one of the best tools to help people to protect their sensitive information from cryptanalysts when it is stored or transmitted via insecure communication channels. In this paper we present a novel symmetric-key block ciphering algorithm, MJEA (for Modified Jamal Encryption Algorithm). MJEA has a 64-bit block size, 8-rounds and 120-bit key. The design philosophy behind the proposed algorithm is simplicity of design which yields an algorithm that is easier to implement and achieves a good Avalanche Effect as quickly as possible. The motivation for this work is to design a novel encryption algorithm that uses good features of JEA encryption algorithm and try to correct the weakness in its performance. Experimental result shows that the proposed algorithm achieves its goal in modifying JEA. A comparison has been conducted between the proposed algorithm and other block ciphers like TEA and MTEA. Simulation results show the superiority of our proposed algorithm in terms of Avalanche Effect. All codes for the proposed algorithm were captured by using C-language. A detailed description of the design process together with results and analyses are given to define the proposed algorithm precisely.

**Key words:** Encryption · Decryption · Cryptography · Network Security · Block Ciphering Algorithm · Avalanche Effect

## INTRODUCTION

Cryptography plays a significant role in hiding the true nature of data; this is achieved by inducing the factor of confusion through a series of shift and other mathematical functions. In the field of cryptography there exist several algorithms for encryption/decryption; these algorithms can be generally classified into two major groups: symmetric-encryption algorithms and asymmetric encryption algorithms. Symmetric-encryption algorithm is marked by its usage of single key for both encryption and decryption whereas in asymmetric or public key cryptography separate keys are used. Symmetric-key block ciphers have long been used as a fundamental cryptographic element for providing information security and they are primarily designed for providing data confidentiality. The security of symmetric cryptosystem is a function of two parameters: the strength of the algorithm and the length of the key. The algorithm may be public; so the security of the whole system resides in the key. The key must be so secure that there is no better way to break it than with a brute-force attack. Therefore, there is a balance between choosing long key and the time required to complete the enciphering operation [1].The name of block cipher came from the fact that block cipher encrypts plaintext as blocks. These blocks differ in size between block cipher algorithms. If the length of block cipher equal one then, it will become stream cipher.

There are many symmetric-key block ciphers in the literature which offer different levels of security such as ( DES, IDEA, Blowfish, TEA, RC5, SAFER,CAST5, XTEA, AES, 3DES, Serpent, MARS and Camellia) that have received the greatest practical interest. In the following paragraphs we will give brief definitions about each one of those encryption techniques:

DES (The Data Encryption Standard) is 64 bits key size with 64 bits block size. It was first described in 1977 by (IBM) [2]. DES was the first encryption standard to be recommended by NIST (National Institute of Standards and Technology). Since that time many attacks and methods recorded the weaknesses of DES, which made it an insecure block cipher [3] [4].

**Corresponding Author:** Jamal N. Bani Salameh, Department of Computer Engineering,
Faculty of Engineering, Mu'tah University, P.O. Box: (7), Al-Karak, Postal Code 61710, Jordan

IDEA (International Data Encryption Algorithm) is a symmetric-key block encryption algorithm. It was first described in 1991 by (X. Lai and J. Massey) to replace the DES standard. IDEA operates on 64-bit blocks using a 128-bit key and consists of a series of eight identical rounds. IDEA derives much of its security by different operations of addition, multiplication and bitwise XORing [5].

Blowfish is a symmetric-key block ciphering algorithm, was designed by Bruce Schneier in 1993 which can be used as a replacement for the DES algorithm. It has a 64-bit block size and it takes a variable length key, ranging from 32 bits to 448 bits; default 128 bits over 16-rounds. It uses a Feistel network structure [6].

TEA (The Tiny Encryption Algorithm) is a symmetric key block cipher notable for its simplicity of description and implementation. It was developed by (D. Wheeler and R. Needham) in 1994. TEA operates on two 32-bit unsigned integers (could be derived from a 64-bit data block) and uses a 128-bit key. It has a Feistel structure with a suggested 64 rounds. It has an extremely simple key schedule, mixing all of the key material in exactly the same way for each cycle [7].

RC5 was designed by Ron Rivest and first published in 1994. RC5 is a symmetric block encryption algorithm. The key size ranges from 0 to 2040 bits but it is suggested to be 128 bits. RC5 supporting a range of different block sizes (32, 64, 128) bits but typically selected to be 64-bits. RC5 are based on a Feistel network construct. It can perform encryption operations in a range of rounds (1-255), typically used 12 rounds [8].

SAFER K-64 (Secure And Fast Encryption Routine with a Key of length 64 bits) is a symmetric-key block-enciphering algorithm. It was developed by (J. L. Massey) in 1994. It has a block length of 64 bits and only byte operations are used in the processes of encryption and decryption. New cryptographic features in SAFER K-64 include the use of linear transform to achieve the desired diffusion [9].

CAST5 is a symmetric block cipher with a block-size of 64-bit and a variable key-size of up to 128 bits. The algorithm was developed in 1996 by (C. Adams and S. Tavares). CAST-128 is a 12 or 16-round Feistel network. The full 16 rounds are employed when the key size is longer than 80 bits. The main building blocks for this algorithm include large 8 × 32-bits s-boxes based on key-dependent rotations, modular addition/subtraction and XOR operations [10]. There are some attacks on four and six round version of CAST5 [11]. However, the full version of CAST5 is still secure.

AES (The advanced encryption algorithm) was designed by (J. Daemen and V. Rijmen) in 2001. AES is a symmetric block encryption algorithm, also known as Rijndael. AES can encrypt 128-bit data blocks by using 128, 192 or 256-bit keys and operates on 4x4 matrixes, which are called states and each AES round is composed of four stages. AES performs encryption operations in 10, 12, or 14 rounds depending on predetermined key sizes [12]. There is no known practical attack against full version of AES [13].

3DES is an enhancement of DES that was designed in 1998 to overcome the weakness of DES. It has a 64-bit block size with 192 bits key size. In this standard the encryption method is similar to the one in the original DES but applied 3 times to increase the encryption level and the average safe time [2].

Serpent is a symmetric key block cipher. It was developed by (R. Anderson, E. Biham and L. Knudsen). It has a block size of 128 bits and can work with different combinations of key size such as 128, 192 or 256 bits. The cipher is a 32-round substitution-permutation network operating on a block of four 32-bit words. Each round applies one of eight 4-bit-to-4-bit s-boxes 32 times in parallel [14]. In the literature, there are attacks on 10 and 11 round Serpent [15]. However, 32 round Serpent still secure.

MARS is a symmetric-key block cipher, with a 128-bit block size and a variable key size of between 128 and 448 bits (in 32-bit increments). It was developed by IBM in 1998 to meet the requirements for a standard shared-key encryption. MARS was chosen as an AES finalist in August 1999 [16]. There have been various attacks on reduced version of MARS cipher. However, full version of MARS is still secure [17].

Camellia is a symmetric-key block cipher that was designed by Mitsubishi and NTT in 2000. It has a block size of 128-bits and can use 128-bit, 192-bit or 256-bit keys like other AES submissions. Camellia is a Feistel cipher with either 18 rounds (when the key is 128 bits) or 24 rounds (when the key is 192 or 256 bits). It utilizes four 8 x 8-bit S-boxes with input and output affine transformations and logical operations [18]. In the literature, there are various attacks on 6, 7 and 11 round Camellia. However, 18 round Camellia is still secure [19].

MTEA (Modified Tiny Encryption algorithm) is a symmetric block cipher designed to correct weaknesses in TEA. MTEA is a 64-bit block Feistel network with a 128-bit key and 64 rounds. It used RC6 as key generation algorithm to overcome the TEA algorithm weaknesses [20].

As we see from the previous survey; most symmetric-key block ciphers (such as DES, RC5, CAST and Blowfish) are based on a Feistel network construct and a round function. This technique involves dividing the plaintext into two halves and repeatedly applying a round function to the data for some number of rounds, where in each round using the round function and a key, the left half is transformed based on the right half and then the right half is transformed based on the modified left half. The round function provides a basic encryption mechanism by composing several simple linear and nonlinear operations such as exclusive-or, substitution, permutation and modular arithmetic [21]. Different round functions provide different levels of security, efficiency and flexibility. The strength of a Feistel cipher depends heavily on the degree of diffusion and non-linearity properties provided by the round function. Many ciphers (such as DES and CAST) base their round functions on a construct called a "substitution box" (S-box) as a source of diffusion and non-linearity. Some ciphers (such as RC5) use bit-wise data-dependent rotations and a few other ciphers (such as IDEA) use multiplication in their round functions for diffusion.

In the design process of our proposed algorithm we used the idea of lattice that was used in Johnson algorithm [22]. This protocol was specifically developed for use in scrambling the transmission of the high frequency radio automatic link. The 24-bit input word represents an automatic link establishment words used by high frequency (HF) radios.

The nearest work to our proposed algorithm is (JEA K-128) that was done by Jamal [23]. It is a symmetric block ciphering algorithm that has a 64-bit word size and 128-bit is the length of the secret key. It uses many successive XORing for the plaintext with sub-keys and it uses a novel technique for sub-keys generation by using multiple multiplexers. JEA needs 23 different 16-bit sub-keys that will be used in all rounds of the algorithm. This algorithm uses 4-different-rounds. After the last iteration, a final transform step produces the 64-bit cipher block.

MJEA (Modified Jamal Encryption Algorithm) is a novel block encryption algorithm proposed in this approach. It encrypts a 64-bit plaintext (Pt) to a 64-bit ciphertext (Ct) in 8 rounds for encryption or decryption process under the control of key (K) that has a size of 120-bit. In this Algorithm, a series of transformations have been used depending on S-BOX and XOR Gates. The cipher itself consists of the following: An initial permutation followed by 8 rounds of encryption, each round consisting of a key mixing operation, a pass through S-boxes and a linear transformation and the last step in this process is the final permutation. The proposed algorithm satisfies the basic requirement of a good algorithm that each ciphertext bit depend upon all bits of the plaintext and all bits of the key, although the degree of dependence is uneven. The proposed mechanism is intended to counter two types of attacks: a passive attack and active cryptanalysis directed to recovery of the current key (brute force attack). The security of this algorithm relies on the key and the S-box for thorough scrambling of the plaintext to produce the ciphertext.

The motivation for this work is to design a new technique that corrects the weakness in JEA algorithm. Several differences from JEA were made which includes the following:

- In MJEA we tried to have the key-schedule to be less complex than JEA.
- For simplicity and fast implementation we used the same design for all rounds in MJEA while JEA has different design for each round.
- To get more diffusion and to have a strong Avalanche Effect we used the idea of substitution table (S-box) in MJEA.

The rest of the paper is organized as follows: Section 2 describes the design process for MJEA cryptographic algorithm. Section 3 discusses the results and checks the performance analyses of the algorithm and section 4 provides some concluding remarks and future work.

**Description of the Proposed Algorithm:** In this section, the proposed algorithm will be introduced in more details. As with all other block ciphers, MJEA uses both confusion and diffusion. MJEA was designed in accordance with Shannon's principles of confusion and diffusion for obtaining security in secret-key ciphers [24]. When a round subkeys are mixed with the plaintext within the round; this acts like a nonlinear combination with respect to the subsequent transformations. This gives the cipher the confusion required to make the statistics of the ciphertext depend in a complicated way on the statistics of the plaintext; provided that small changes diffuse quickly through the cipher. To guarantee this diffusion in MJEA, we spread the redundancy of the plaintext out over the ciphertext. A cryptanalyst looking for those redundancies will have a harder time finding them.

The proposed algorithm encrypts a 64-bit plaintext to a 64-bit ciphertext in 8 rounds under the control of the key. The user key length and the number of rounds are variable, but for the purposes of this submission we use 120-bits and 8-rounds.

**The Cipher Itself Consists of Three Steps:** An initial permutation, 8 rounds (each consisting of a key mixing operation, a pass through S-boxes and a linear transformation) and a final permutation. Each round of the cipher consists of three operations:

- Key Mixing: At each round, a 120-bit key is XORed with the current intermediate data after it divided into several 8-bit subkeys (k0, k1,.....k14).
- S-boxes: The combination of input and key is considered as eight 8-bit words. The S-box applied to these eight words and the result is eight output words.
- Linear Transformation: The 8 bits in each of the output words are linearly mixed.

**Encryption Process:** Figure 1 shows a schematic representation of the algorithm to perform the encryption process. The 64-bit input of plain text is divided into eight 8-bit blocks. The algorithm operates on each of the eight bytes of the 64-bit word individually. These eight blocks become the input to the first round of the algorithm. MJEA encrypts a 64-bit plaintext to a 64-bit ciphertext in 8 rounds under the control of the main key. Each round needs sixteen 8-bit subkeys. The subkeys generation process starts with the 120-bit main key (K); which is divided into two parts $K_L$ and $K_R$. Furthermore, $K_L$ is divided into seven 8-bit subkeys and $K_R$ is divided into eight 8-bit subkeys. More details about the key scheduling will be discussed later. MJEA is based upon a basic function, which is iterated eight times. The first iteration (round) operates on eight input 8-bit plaintext blocks and the successive iterations also operate on the 8-bit blocks that come from the previous iteration. At each round, each byte of the plain text is XORed with one or more of the other data bytes and two 8-bit subkeys (one produced by $K_L$and the other produced by $K_R$).

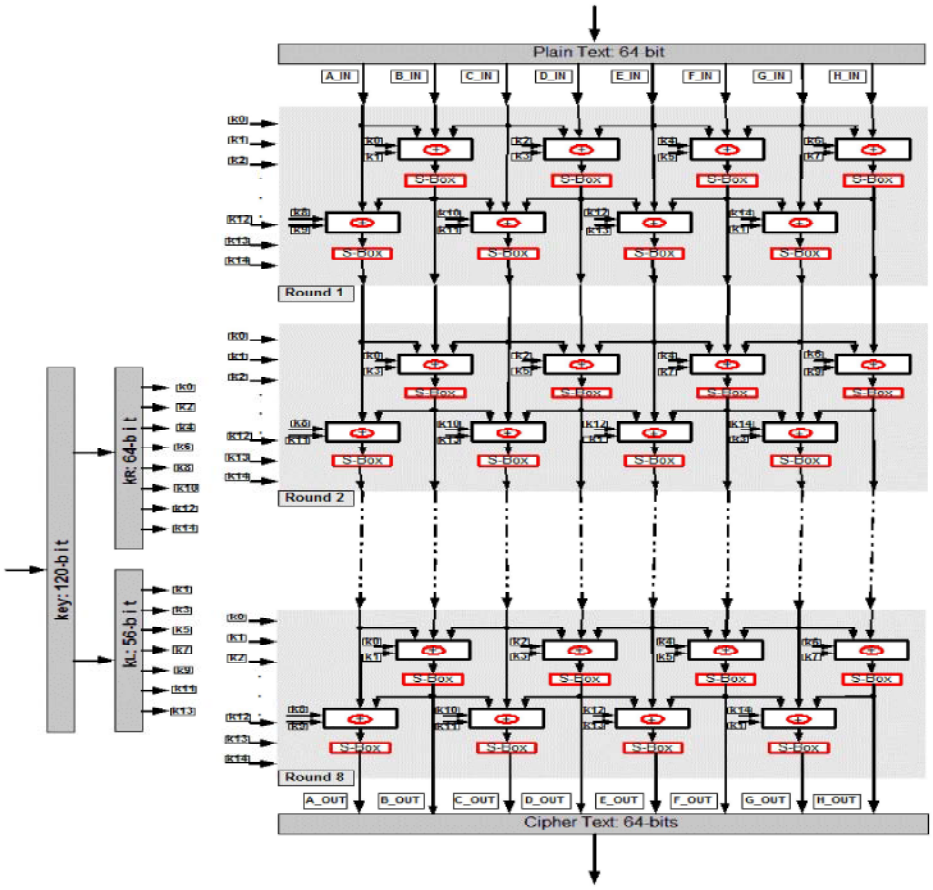


Fig. 1: Explanation of MJEA algorithm (encryption)

The result is then translated using a 256x8 bit substitution table (S-box). More details about the S-box will be discussed later. After the last iteration, a final transform step produces the 64-bit ciphertext.

The following assumptions were considered in describing the operation of the proposed encryption algorithm:

- Let S(●) be an invertible function mapping {0..255}-> {0..255}.
- Let $K_L$ be a vector of key variable bytes that represents the first 56-bits of the key and $K_R$ be a vector of key variable bytes that represents the last 64-bits of the key.
- Each byte in $K_L$ and $K_R$ represents an 8-bit subkey; so we have seven 8-bit subkeys out of $K_L$ and eight 8-bit subkeys out of $K_R$.
- Let A be the most significant of the eight-byte input to each round of encryption, B, C, D, E, F, G be the middle bytes and H be the least significant byte and A', B', C', D', E', F', G' and H' be the corresponding ciphertext bytes of each round.
- The priority of performing ciphertext blocks starts with B', then (D', F', H', A', C', E') and ends with F'.

Mathematically, MJEA encryption algorithm works as follows: For each round; the eight encrypted bytes are formed as follows:

$$B' = S(B \oplus A \oplus C \oplus K_L[] \oplus K_R[]) \quad (1)$$

$$D' = S(D \oplus C \oplus E \oplus K_L[] \oplus K_R[]) \quad (2)$$

$$F' = S(F \oplus E \oplus G \oplus K_L[] \oplus K_R[]) \quad (3)$$

$$H' = S(H \oplus G \oplus K_L[] \oplus K_R[]) \quad (4)$$

$$A' = S(A \oplus B^{/} \oplus K_L[] \oplus K_R[]) \quad (5)$$

$$C' = S(C \oplus B^{/} \oplus D^{/} \oplus K_L[] \oplus K_R[]) \quad (6)$$

$$E' = S(E \oplus D^{/} \oplus F^{/} \oplus K_L[] \oplus K_R[]) \quad (7)$$

$$G' = S(G \oplus F^{/} \oplus H^{/} \oplus K_L[] \oplus K_R[]) \quad (8)$$

The way in which the encryption algorithm chooses the required group of subkeys for the first round starts with the first byte in each of $K_L$ and $K_R$, perform the first block, using the next byte from $K_L$ and $K_R$ to perform the second block in sequence. This sequence is repeated until the end of $K_L$ and $K_R$ arrays. After that using the next byte from $K_L$ and $K_R$ (modulo their lengths) each time a reference to $K_L$ [ ] and $K_R$ [ ] is made. The same scenario is repeated for all other rounds.

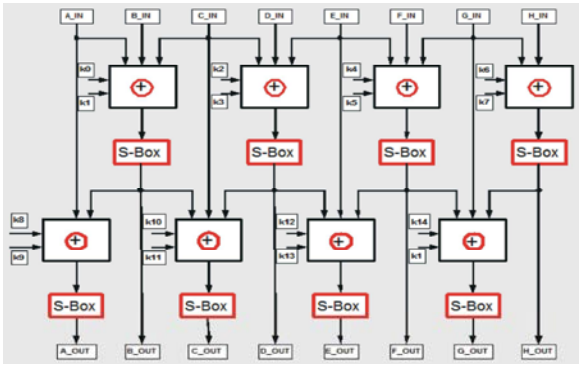

Fig. 2: Schematic diagram for a single round (encryption)

Figure 2 shows a schematic diagram/encryption for a single round. The inputs for this round are eight 8-bit blocks that were received from the initial permutation stage and sixteen 8-bit different sub-keys. As we see in the figure; each byte of the plain text is XORed with one or more of other data bytes and a two 8-bit subkeys, the result is then translated using a 256x8 bit substitution table (S-box). The 8-bit block outputs of this round will form the inputs for round 2. The same scenario is repeated in the following rounds with different mixing scheme between text-blocks and sub-keys.

As a final note on the design process of the encoder; which could be seen in the schematics above; each round in our algorithm consists of a key-dependent permutation, a key and data-dependent substitution and all operations are XORed on 8-bit words.

**Decryption Process:** Decryption for MJEA is relatively straightforward beginning with the ciphertext as input. It is different from encryption in that the inverse of the S-boxes must be used in the reverse order, as well as the inverse linear transformation and reverse order of the subkeys. A block diagram for the decryption process is shown in Figure 3. As we see in the figure, the same 120-bit secret key is used as input to provide the algorithm with the same sub-keys as in the encoder side; the sub-keys are used in reverse order. The 64-bit block of ciphertext goes in one end of the algorithm and then the algorithm runs in the reverse direction, which reconfigures the 64-bit of plaintext at the end. Note that the starting point in the key vectors must be pre-computed, based upon the number of rounds performed and that the bytes are used in reverse order.

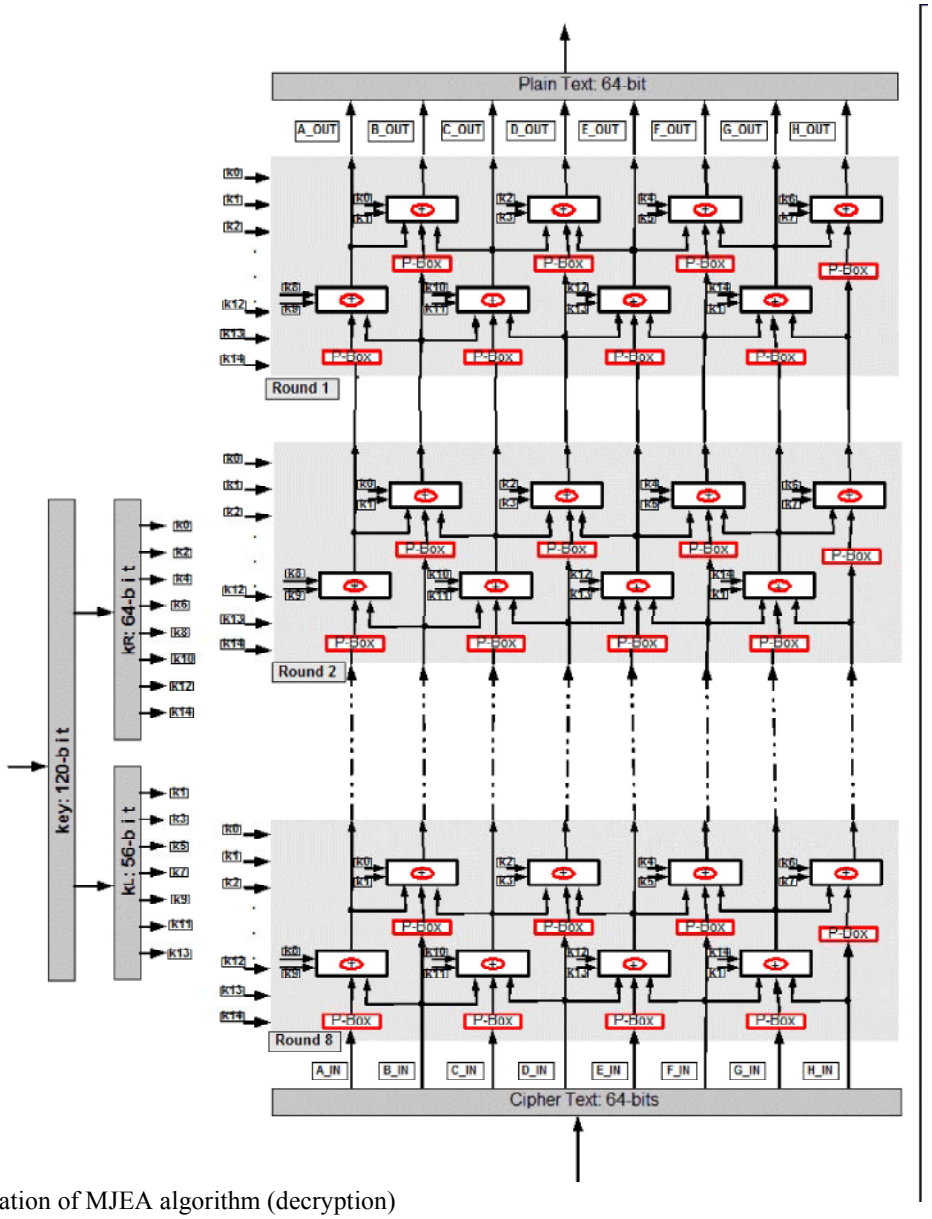


Fig. 3: Explanation of MJEA algorithm (decryption)

Mathematically, the decryption algorithm works as follows:

- Let $S^{-1}(\bullet)$ be the inverse of the $S(\bullet)$ used for encryption.
- Starting with the last elements of the key vectors used in encryption, perform the same number of rounds of the following decryption steps as were used in encryption, working backward through the key vectors. the eight decrypted bytes are formed as follows:

$$G = S^{-1}(G') \oplus F' \oplus H' \oplus K_L[] \oplus K_R[] \quad (9)$$

$$E = S^{-1}(E') \oplus D' \oplus F' \oplus K_L[] \oplus K_R[] \quad (10)$$

$$C = S^{-1}(C') \oplus B' \oplus D' \oplus K_L[] \oplus K_R[] \quad (11)$$

$$A = S^{-1}(A') \oplus B' \oplus K_L[] \oplus K_R[] \quad (12)$$

$$H = S^{-1}(H') \oplus G \oplus K_L[] \oplus K_R[] \quad (13)$$

$$F = S^{-1}(F') \oplus E \oplus G \oplus K_L[] \oplus K_R[] \quad (14)$$

$$D = S^{-1}(D') \oplus C \oplus E \oplus K_L[] \oplus K_R[] \quad (15)$$

$$B = S^{-1}(B') \oplus A \oplus C \oplus K_L[] \oplus K_R[] \quad (16)$$

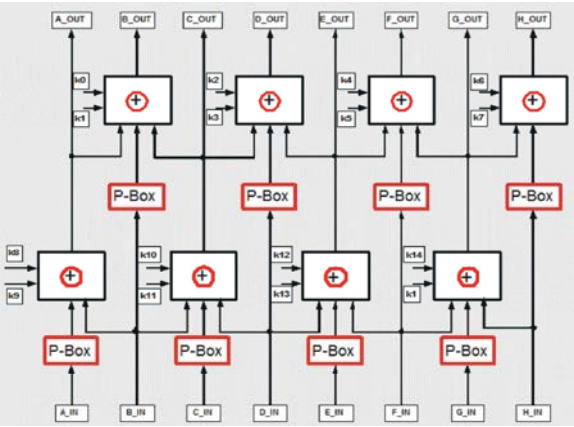


Fig. 4: Schematic diagram for a single round (decryption)

The schematic diagrams for all rounds in the decryption process are exactly the same as in the encryption process but in the reverse direction. Figure 4 shows a sample of this process: A schematic diagram for decryption in a single round.

**The Key Schedule:** The key schedule is an important component of a block cipher; it computes the round keys from the external key. MJEA uses a large number of sub-keys, each round needs sixteen 8-bit subkeys which means we need 128 subkeys for 8-rounds. These sub-keys must be pre-computed before any data encryption or decryption.

The subkeys generation process works as follows: The 120-bit main key (K) is divided into two parts ($K_L$: 56-bits) and ($K_R$: 64-bits); then $K_L$ is divided into seven 8-bit subkeys and $K_R$ is divided into eight 8-bit subkeys. The key array consists of fifteen 8-bit sub-keys (k0, k1,……., k14). Let $K_L$ be a vector of key variable bytes that represents the first 56-bits of the key and $K_R$ be a vector of key variable bytes that represents the last 64-bits of the key. Starting with the first byte in each of $K_L$ and $K_R$, perform several rounds of the sequence, using the next byte from $K_L$ and $K_R$ (modulo their lengths) each time a reference to $K_L$[ ] and $K_R$[ ] is made.

The way in which the encryption algorithm chooses the required group of subkeys for the first round starts with the first byte in each of $K_L$ and $K_R$, perform the first block, using the next byte from $K_L$ and $K_R$ to perform the second block in sequence. This sequence is repeated until the end of $K_L$ and $K_R$ arrays; then using the next byte from $K_L$ and $K_R$ (modulo their lengths). To show how the algorithm chooses the required group of subkeys for the first round; let us follow the following explanation:

Table 1: Subkey distribution over different rounds

| | 5 | 1 | 6 | 2 | 7 | 3 | 8 | 4 |
|---|---|---|---|---|---|---|---|---|
| Round | A | B | C | D | E | F | G | H |
| 1 | {K8,K9} | {K0,K1} | {K10,K11} | {K2,K3} | {K12,K13} | {K4,K5} | {K1,K14} | {K6,K7} |
| 2 | {K11,K8} | {K3,K0} | {K13,K10} | {K5,K2} | {K1,K12} | {K7,K4} | {K14,K3} | {K9,K6} |
| 3 | {K8,K13} | {K0,K5} | {K10,K0} | {K2,K7} | {K12,K3} | {K4,K9} | {K5,K14} | {K6,K11} |
| 4 | {K1,K8} | {K7,K0} | {K3,K10} | {K9,K2} | {K5,K12} | {K11,K4} | {K14,K7} | {K13,K6} |
| 5 | {K8,K3} | {K0,K9} | {K10,K5} | {K2,K11} | {K12,K7} | {K4,K13} | {K9,K14} | {K6,K1} |
| 6 | {K5,K8} | {K11,K0} | {K7,K10} | {K13,K2} | {K9,K12} | {K1,K4} | {K14,K11} | {K3,K6} |
| 7 | {K8,K7} | {K0,K13} | {K10,K9} | {K2,K1} | {K12,K11} | {K4,K3} | {K13,K14} | {K6,K5} |
| 8 | {K9,K8} | {K1,K0} | {K11,K10} | {K1,K3} | {K13,K12} | {K5,K4} | {K14,K1} | {K7,K6} |

*KL = {K1, K3, K5, K7, K9, K11, K13}*
*KR = {K0, K2, K4, K6, K8, K10, K12, K14}*

Known that, each one of (k0, k1,……., k14) represents an 8-bit sub-key. According to the previous assumptions in section (2.1); the algorithm starts by assigning the first group of subkeys {(k0, k1)?B}, {(k2, k3)?D}, {(k4, k5)?F}, {(k6, k7)?H}, {(k8, k9)?A}, {(k10, k11)?C}, then{(k12, k13)?E}. Since the elements in $K_L$ array is consumed we take the next subkey in the array modulo its length which is k1 which will be the subkey number sixteen that will be assigned for the last block, so we have {(k14, k1)?G}.The same scenario is repeated for all other rounds. Table (1) shows subkey distribution over 8-rounds of the algorithm.

**Substitution Box (S-Box):** Many cipher algorithms base their round functions on a construct called a "substitution box" (S-box) as a source of diffusion and non-linearity. As other ciphers, we used a linear transformation tables (S-box and P-box) in the proposed algorithm in order to maximize the Avalanche Effect.

As we said before, at each step of the encryption process, each byte of plaintext is XORed with one or both of the other data bytes and two bytes of the key and the result is then translated using 256x8 bit substitution table (S-box) or encryption table.

On the other hand at each step of the decryption process, we used a reverse type of translation using 256x8 bit substitution table (P-box) or decryption table. S-box generation is the backbone of this algorithm. It has eight columns and 8 rows; each element consists of 8-bits. It replaces the input by another code to the output.

As a final notice on the design process for the proposed algorithm is the decision of choosing the number of rounds. As we see in Figure 5; a change in one byte of the plain text need at most 4-rounds to affect other bytes of the plaintext, as does each round key bit. But to have thorough mixing between blocks we decide to have

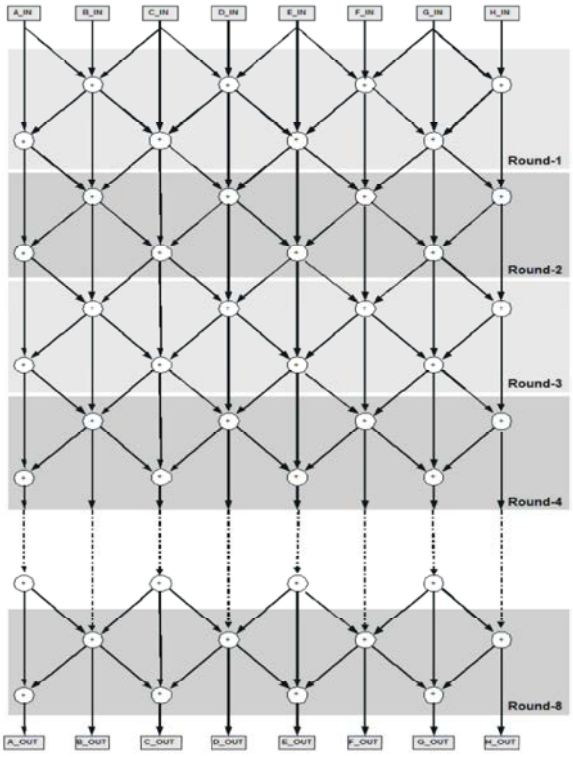


Fig. 5: Schematic for the algorithm showing thorough mixing between the eight blocks of data

8-rounds for our algorithm, this will be sufficient to make differential and linear cryptanalysis of proposed encryption algorithm impractical.

Moreover, for more than 4-rounds of encryption, the resultant Avalanche Effect as we will see in the simulation study is very similar to the one obtained for 4-rounds of encryption. Hence, we conclude that MJEA achieves the desired diffusion after 4-rounds; this result is also verified by experimental tests shown in section 3.

**Performance Analyses:** In this section the performance of the proposed algorithm will be evaluated and compared with other algorithms considering Avalanche Effect as the major metric in this evaluation. William [2] defines the Avalanche Effect as the phenomenon that describes the effect in the output cipher text if a single or few bits are changed in the plain text. If a block cipher does not exhibit the Avalanche Effect to a significant degree, then it has poor randomization and thus a cryptanalyst can make predictions about the input, being given only the output. This may be sufficient to partially or completely break the algorithm. Thus, the Avalanche Effect is a desirable condition from the point of view of the designer of the cryptographic algorithm.

Table 2: Encryption process over 8-rounds

key variable (K) = a1 h2 c3 d4 e5 f0 697869a1 h2 c3 d4 e5 f0
Plain Text (Pt) = a789b456c123defa

| Round | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0 | A7 | 89 | B4 | 56 | C1 | 23 | DE | FA |
| 1 | 76 | 28 | C0 | 05 | 85 | 82 | 2A | 5D |
| 2 | 9A | B9 | 8D | BF | FD | D4 | 72 | 03 |
| 3 | E1 | 4A | 5B | C7 | C8 | 04 | C2 | E4 |
| 4 | 5C | BA | 9F | FF | D0 | 33 | 12 | 2C |
| 5 | 86 | D8 | B2 | 56 | 98 | 5C | B1 | DE |
| 6 | 9F | FE | A7 | D2 | 4C | 5B | A1 | 87 |
| 7 | 51 | 7F | B4 | AE | 25 | 4D | 7F | A2 |
| 8 | 33 | 28 | 0D | D8 | 42 | 5B | B7 | D5 |

Table 3: Decryption process over 8-rounds

key variable (K) = a1 h2 c3 d4 e5 f0 697869a1 h2 c3 d4 e5 f0
Cipher Text (Ct): 33280DD8425BB7D5

| Round | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 0 | 33 | 28 | 0D | D8 | 42 | 5B | B7 | D5 |
| 1 | 51 | 7F | B4 | AE | 25 | 4D | 7F | A2 |
| 2 | 9F | FE | A7 | D2 | 4C | 5B | A1 | 87 |
| 3 | 86 | D8 | B2 | 56 | 98 | 5C | B1 | DE |
| 4 | 5C | BA | 9F | FF | D0 | 33 | 12 | 2C |
| 5 | E1 | 4A | 5B | C7 | C8 | 04 | C2 | E4 |
| 6 | 9A | B9 | 8D | BF | FD | D4 | 72 | 03 |
| 7 | 76 | 28 | C0 | 05 | 85 | 82 | 2A | 5D |
| 8 | A7 | 89 | B4 | 56 | C1 | 23 | DE | FA |

This section is prepared for making statistical test on the ciphertext that produced from encryption the plaintext in hexadecimal: The evaluation process includes a set of experiments each one focuses on a certain issue.

**Example Showing the Use of the Proposed Algorithm:** At the beginning we present an example showing the use of our algorithm. A random 120-bit key was generated and the plain text chosen represents a random block of data in hexadecimal. Both the key and the data were set then we run the simulator. As we see in Table (2), the plain text is encrypted and the eight byte result of the cipher text of each round of the encryption process is given. The final result of this encryption process is: (OX33280DD8425BB7D5).

After that, we provide the cipher text that we got out of the encoder along with the same key to the decoder then we run the simulator, the final result of the decryption process is (OXA789B456C123DEFA) which means that the ciphertext is decrypted successfully. The decoder is able efficiently to recover the original plain text; this result is shown in Table (3).

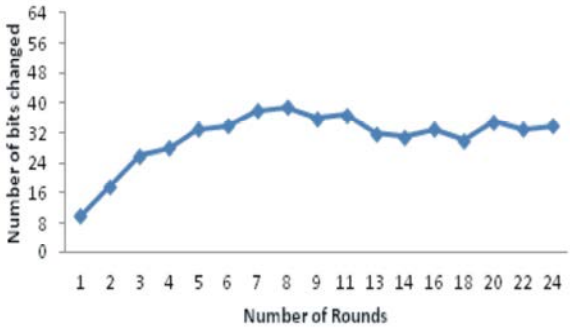


Fig. 6: Avalanche Effect on ciphertext by changing 1-bit in the plaintext over different number of rounds

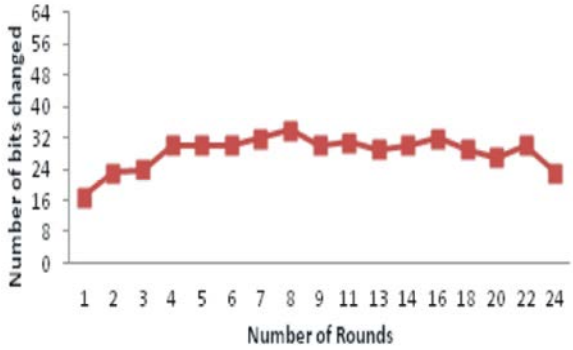


Fig. 7: Avalanche Effect on ciphertext by changing 1-bit in the key over different number of rounds

**Evaluate the Proposed Algorithm Separately:** In the following set of experiments we present the results of simulations of the proposed algorithm separately, showing the following metrics:

- The average number of ciphertext bits changed for single-bit changes in the plaintext.
- The average number of ciphertext bits changed for single-bit changes in the key.
- The average number of decrypted plaintext bits changed for single-bit changes in the key.

**The Results Were Obtained as Follows:** A random 120-bit key was generated and used for all simulations. The plain text chosen represents random blocks of data in hexadecimal. The main goal of the first test is to check the Avalanche Effect on ciphertext by changing 1-bit in the plaintext over different number of rounds; here we set the key fixed (K1 = FFFFFFFFFFFFF0000000000000000) and we change the plain text by 1-bit (Pt1 = 9999999999999999, Pt2 = 8999999999999999). The plot in Figure 6 shows the result for this test versus the number of rounds used in encryption. However, it is observed from this figure that

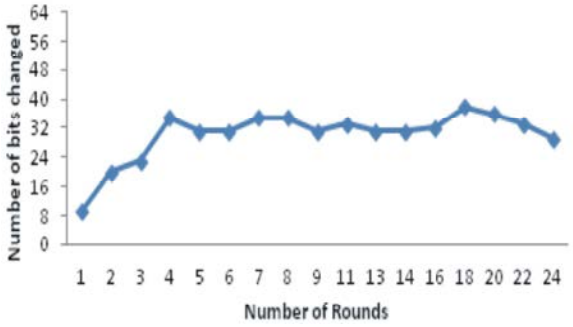


Fig. 8: Avalanche Effect on plaintext of single-bit changes in ciphertext over different number of rounds

the proposed algorithm needs at least 4-rounds to give a reasonable Avalanche Effect on the ciphertext (more than 50% of the ciphertext bits were changed).

The second test aims to check the Avalanche Effect on ciphertext by changing 1-bit in the key over different number of rounds; here we set the plaintext fixed (Pt = 9999999999999999) and we change the key by 1-bit (K1 = FFFFFFFFFFFFF0000000000000000,k2 = EFFFFFFFFFFFF0000000000000000). Figure 7 shows the result for this test versus the number of rounds used in encryption. As we note in this figure the proposed algorithm needs at least 4-rounds to give a strong Avalanche Effect on the ciphertext (more than 50% of the ciphertext bits were changed).

The last test in this section checks the Avalanche Effect on plaintext by changing single bit in ciphertext over different number of rounds. Here we set the key fixed (K = FFFFFFFFFFFFF0000000000000000) and we set the plaintext also to be fixed (9999999999999999) then we change the cipher text by 1-bit for each trial. However, as observed from Figure 8; this test ensures the same result as we got before (a strong avalanche effect), more than 50% of the plain text bits were changed by changing just 1-bit in the cipher text starting from round 4.

From the previous series of simulations, it appears that the algorithm achieves its goal of thoroughly scrambling decrypted plaintext given a single-bit change in ciphertext or key when run for at least 4 rounds. To count for possible interruption due to choice of key and input data word, eight rounds shall be employed when this algorithm is used.

**Performance Evaluation of the Proposed Algorithm Compared with JEA:** The proposed algorithm used good features of JEA algorithm that was introduced before.

Table 4: Simulation parameters for comparing JEA and MJEA over different number of keys

| Pt1: 0122034405660788, Pt2: 1122034405660788 | | | |
|---|---|---|---|
| key No. | Key | JEA | MJEA |
| 1 | k: C6FAE4500056738BFCEA3434F57A2B84 | 15 | 35 |
| 2 | k: 1122E4500056738BFCEA3434F57A2B84 | 15 | 35 |
| 3 | K: 1122334400567 38BFCEA3434F57A2B84 | 15 | 36 |
| 4 | K: 1122334455567 38BFCEA3434F57A2B84 | 15 | 35 |
| 5 | K: 1122334455667 38BFCEA3434F57A2B84 | 15 | 33 |
| 6 | K: 1122334455667788FCEA3434F57A2B84 | 15 | 33 |
| 7 | K: 1122334455677889900A ABBF57A2B84 | 15 | 34 |
| 8 | K: 1122334455677889900A ABBCCDD2B84 | 15 | 36 |
| 9 | K: 00000000000000000000000000000000 | 15 | 38 |
| 10 | K: 11111111111111111111111111111111 | 15 | 36 |

Table 5: Simulation parameters for comparing JEA and MJEA over different number blocks of data

| K1: 00000000000000000000000000000000, K2: 00000000000000000000000000000001 | | | |
|---|---|---|---|
| Block No. | Block | JEA | MJEA |
| 1 | Pt: 0000000000000000 | 15 | 38 |
| 2 | Pt: 1111111111111111 | 15 | 36 |
| 3 | Pt: 6666666666666666 | 15 | 37 |
| 4 | Pt: 9999999999999999 | 15 | 41 |
| 5 | Pt: bbbbbbbbbbbbbbbb | 15 | 37 |
| 6 | Pt: cccccccccccccccc | 15 | 40 |
| 7 | Pt: 5555555566666666 | 15 | 38 |
| 8 | Pt: 3333333311111111 | 15 | 38 |
| 9 | Pt: 1111111122222222 | 15 | 40 |
| 10 | Pt: 4444444433333333 | 15 | 37 |

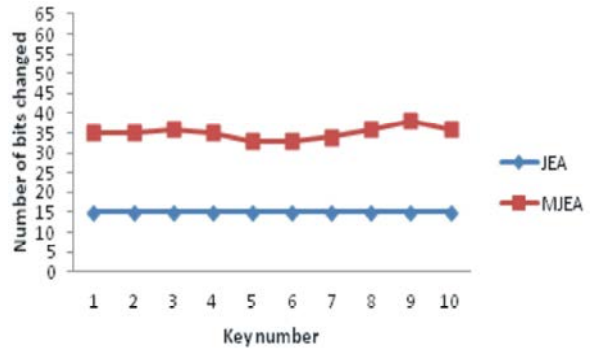


Fig. 9: Avalanche Effect on ciphertext by changing 1-bit in the plaintext over different number of keys

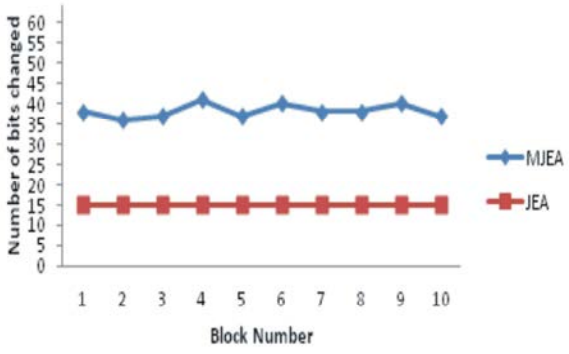


Fig. 10: Avalanche Effect on ciphertext by changing 1-bit in the key over different number blocks of data

The main goal of the following tests in this experiment is to compare the performance of both algorithms together in order to proof the effort that was made to enhance the performance of JEA algorithm.

The first test checks the Avalanche Effect on ciphertext by changing 1-bit in plintext over different number of keys. Table (4) shows the simulation parameters for this test and Figure 9 shows the result; the plot is built versus the number of keys used in encryption. As we see in the figure, JEA algorithm has a poor Avalanche Effect; the total number of bits changed in ciphertext due to 1-bit change in plaintext is fixed (just 15-bits) for all different number of keys. On the other hand, MJEA has a good avalanche effect; the average number of bits changed in ciphertext due to 1-bit change in plaintext is 35 out of 64 which means that 55% of the bits changed.

The second test checks the Avalanche Effect on ciphertext by changing 1-bit in the key over different number blocks of data. Table (5) shows the parameters for this test and Figure 10 shows the result; the plot is built versus the number of blocks of data used in encryption. As we see in Figure 10, JEA algorithm has a poor Avalanche Effect; also in this test; the total number of bits changed in ciphertext due to 1-bit change in plaintext is fixed (just 15-bits) for all different blocks of data. On the other hand, MJEA has a strong Avalanche Effect; the average number of bits changed in ciphertext due to 1-bit change in the key is 38.2 out of 64 which means that 60% of the bits changed.

As shown from the previous simulations in this experiment; the proposed MJEA algorithm achieves its goal in improving the performance of JEA algorithm thoroughly.

**Avalanche Effect Comparison with TEA and MTEA:** In this section we will make a comparison between the proposed algorithm and TEA which is a well known algorithm that has the same structure as MJEA. Also TEA is notable with its simplicity of description and implementation. In the literature I found one more new algorithm MTEA that tries to overcome the weakness in TEA and the author tries to compare its performance with TEA. So, in this test I took the same experimental metrics

Table 6: Data setup for encryption with 1-bit change in the key to compare the performance of different algorithms

| K1: 00000000000000000000000000000000, K2: 00000000000000000000000000000001 | | | | |
|---|---|---|---|---|
| Block No. | Block | MJEA | TEA | MTEA |
| 1 | Pt: 0000000000000000 | 38 | 31 | 39 |
| 2 | Pt: 1111111111111111 | 36 | 30 | 35 |
| 3 | Pt: 6666666666666666 | 37 | 33 | 38 |
| 4 | Pt: 9999999999999999 | 41 | 24 | 39 |
| 5 | Pt: bbbbbbbbbbbbbbbb | 37 | 34 | 31 |
| 6 | Pt: cccccccccccccccc | 40 | 33 | 38 |
| 7 | Pt: 5555555566666666 | 38 | 32 | 38 |
| 8 | Pt: 3333333311111111 | 38 | 29 | 33 |
| 9 | Pt: 1111111122222222 | 40 | 33 | 43 |
| 10 | Pt: 4444444433333333 | 37 | 31 | 36 |

Table 7: Results of comparing Avalanche Effect for different algorithms

| Encryption Algorithm | Average of Avalanche Effect | Percentage (%) |
|---|---|---|
| MJEA | 38.2 - bits | 60% |
| TEA | 31 - bits | 48% |
| MTEA | 37 - bits | 58% |

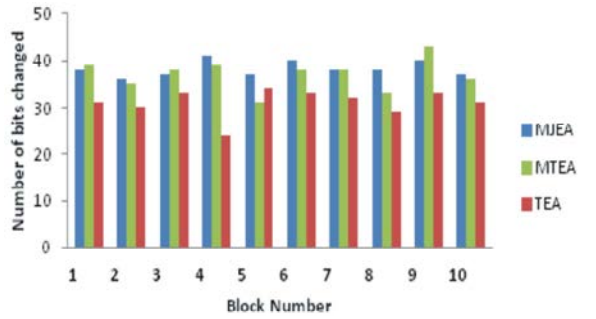


Fig. 11: Avalanche Effect comparison between different algorithms

as in [20] for both algorithms and compare them with MJEA. This test compares the three algorithms on the basis of Avalanche Effect on ciphertext by changing 1-bit in the key over different number blocks of data. Table (6) shows the data that was setup for encryption for this test and Figure 11 shows the result. However, it is observed from the figure that TEA algorithm has the least Avalanche Effect with an average number of bits changed in ciphertext due to 1-bit change in the key equals to (31-bits) while MTEA has an average of (37-bits) and MJEA has an average of ( 38.2-bits) out of 64-bits changed. This result plus the percentage of the avalanche effect is shown in Table (7).

This promising result shows the superiority of our proposed algorithm when compared with TEA and MTEA in terms of Avalanche Effect.

As a final note on the performance evaluation of the proposed algorithm: The reason behind the behaviour of MJEA compared to (JEA, TEA and MTEA) is the main building blocks that build the algorithm. In MJEA, we have more diffusion that was maintained through using the S-box to ensure a strong Avalanche Effect. Also, the structure of the lattice that combines the blocks of the algorithm ensures a thorough and fast scrambling of the subblocks of plaintext together with subkeys.

## CONCLUSIONS AND FUTURE WORK

This paper has proposed a novel block encryption algorithm; its design is highly conventional. The design philosophy behind the proposed algorithm is that simplicity of design which yields an algorithm that is easier to implement, achieves a good Avalanche Effect as quickly as possible, achieves better security properties and complete the encryption/decryption process with a high speed.With 64-bit block size and 120-bit key, we believe it is secure and fast as most of well known block encryption algorithms, yet we believe also our algorithm to be more secure against most of known types of attack. From the small series of simulations that was done in this work, several points can be concluded from the experimental results.

- It appears that the algorithm achieves its goals of thoroughly scrambling the plaintext with the key when run for at least four rounds. To account for possible attacks eight rounds shall be employed when this algorithm is used. The proposed algorithm achieved a good Avalanche Effect when it is tested separately; on average more than 50% of bits were changed when we change a bit in the plaintext, key or the ciphertext.
- Experimental results are given to demonstrate the effectiveness of the modified technique compared to JEA algorithm at different settings such as different data blocks and different keys. The difference of efficiency between our proposed algorithm and JEA algorithm is very high, which means that MJEA achieves its goal in enhancing the performance of JEA.
- A comparison has been conducted for different encryption algorithms. Simulation results clearly show the superiority of our proposed technique when compared with TEA and MTEA in terms of Avalanche Effect.

**Further Work Would Include the Following:**

- More thorough testing and analysis to get better S-boxes.
- Use the sub-key generator that was used in JEA algorithm to generate the required subkeys for the proposed algorithm.
- Calculate the execution time for encryption and decryption for the proposed algorithm and compare it with other well known algorithms.
- Try to use the proposed algorithm with other types of data (image data and sound data).
- Do more performance comparison with up-to-date block ciphering algorithms.

## REFERENCES


1. Shnier, B., 1996. Applied Cryptography Protocols, Algorithms and Source Code in C, 2nd Edition. John Wiley and Sons.
2. Stallings, W., 2005. Cryptography and Network Security Principle and Practices. Prentice Hall.
3. Coppersmith, D., 1994. The data encryption standard (DES) and its strength against attacks, IBM Journal of Research and Development, pp: 243-250.
4. Biham, E. and A. Shamir, 1993. Differential Cryptanalysis of the Data Encryption Standard. Springer Verlag Publishing.
5. Schneier, B., 1990. The IDEA encryption algorithm, Dr. Dobb's Journal, 18(13): 50-56.
6. Schneier, B., 1994. The Blowfish encryption algorithm, Dr. Dobb's Journal, 19(4): 38-40.
7. Wheeler, D.J. and R.M. Needham, 1995. TEA, a tiny encryption algorithm. In the Proceedings of Fast Software Encryption, Second International Workshop, pp: 97-110.
8. Rivest, R.L., 1995. The RC5 encryption algorithm, Dr. Dobb's Journal, 20(1): 146-148.
9. Massey, J.L., 1994. SAFER K-64: A byte-oriented block-ciphering algorithm, fast software encryption. In the Proceedings of the Security Workshop, pp: 1-17.
10. Adams, C.M., 1997. Constructing symmetric ciphers using the CAST design procedure, Design, Codes and Cryptography, 12(3): 283-316.
11. Wang, M., X. Wang and C. Hu, 2009. New Linear Cryptanalytic Results of Reduced-Round of CAST-128 and CAST-256. Lecture Notes in Computer Sciences, Volume 5381, Springer Berlin /Heidelberg.
12. Daemen, J. and V. Rijmen, 2002. The design of Rijndael: AES-the advanced encryption standard. Springer-Verlag, ISBN, 3-540-42580-2.
13. Biryukov, A. and D. Khovratovich, 2009. Related-key cryptanalysis of the full AES-192 and AES-256. In the Proceedings of Advances in Cryptology-ASIACRYPT, pp: 1-18.
14. Anderson, R., E. Biham and L. Knudsen, 1998. Serpent: A flexible block cipher with maximum assurance. In the Proceedings of the First AES Candidate Conference, pp: 20-22.
15. Biham, E., O. Dunkelman and N. Keller, 2002. Linear Cryptanalysis of Reduced Round Serpent. Lecture Notes in Computer Sciences, Volume 2355, Springer Berlin /Heidelberg.
16. Burwick, C., D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla and M. Peyravian, 1998. MARS-a candidate cipher for AES, AES submission.
17. Kelsey, J. and B. Schneier, 2000. MARS attacks preliminary cryptanalysis of reduced-round MARS variants. In the Proceedings of the Third AES Candidate Conference, pp: 169-185.
18. Aoki, K., T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima and T. Tokita, 2001. Camellia: a 128-bit block cipher suitable for multiple platforms-design and analysis. Selected Areas in Cryptography, 2012.
19. Yee, W., J. Doumen and P. Hartel, 2006. Survey and benchmark of block ciphers for wireless sensor networks, ACM Trans. Sensor Networks, 2(1): 65-93.
20. Gaidaa, S.M., 2011. A modification of TEA block cipher algorithm for data security (MTEA), Eng. and Tech. Journal, 29(5).
21. Menezes, A.J., P.C. Van Oorschot and S.A. Vanstone, 1996. Handbook of Applied Cryptography. CRC Press.
22. Johnson, E.E., 1992. A 24-bit Encryption Algorithm for Linking Protection, USAISEC Technical Report.
23. Salameh, J.S., 2009. JEA-128: a novel encryption algorithm using VHDL, WSEAS Transactions on Computers, 12(8): 1875-1885.
24. Shannon, C.E., 1994. Communication theory of secrecy systems, Bell Systems Technical Journal, 28(1): 656-715.