# Strongest AES with S-Boxes Bank and Dynamic Key MDS Matrix (SDK-AES)

Fatma Ahmed and Dalia Elkamchouchi

***Abstract*—With computers, security is only a matter of software. The Internet has made computer security much more difficult than it used to be. In this paper, we introduce modified AES with S-boxes bank to be acted like rotor mechanism and dynamic key MDS matrix (SDK-AES). In this paper we try to make AES key dependent and resist the frequency attack. The SDK-AES algorithm is compared with AES and gives excellent results from the viewpoint of the security characteristics and the statistics of the ciphertext. Also, we apply the randomness tests to the SDK-AES algorithm and the results shown that the new design passes all tests which proven its security.**

***Index Terms*—Advanced encryption algorithm (AES), S-boxes bank, rotor, frequency analysis, inverse power function, MDS.**

## I. Introduction

### A. AES

AES is short for Advanced Encryption Standard and is a United States encryption standard defined in Federal Information Processing Standard (FIPS) 192, published in November 2001 [1]. It was approved as a federal standard in May 2002. AES is the most recent of the four current algorithms ratified for federal us in the United States. Rijndael submitted by Joan Daemen and Vincent Rijmen, is a symmetric key, iterated block cipher based on the arithmetic in the Galois Field $GF(2^8)$. AES Input and Output consists of 128 bit sequences. The cipher key is 128, 192, or 256 bits. Byte is the unit of processing. Input blocks are 16 bytes each. AES operations are Conducted on a two dimensional array of bytes called the state. The state consists of four rows of bytes each containing $N_b$ bytes where $N_b$ is the block length divided by 32. Rijndael round function acts on a state $N_r$ times, where $N_r$ is equal to the number of rounds that can be 10, 12 or 14 rounds, depending on $N_k$, where $N_k$ is equal to the number of 32-bit words comprising the Cipher Key [2]. Rijndael round is consists of 4 transformations:

**Sub bytes:** Transformation in the Cipher that processes the State using a nonlinear byte substitution table (S-box) that operates on each of the State bytes independently which provides nonlinearity and confusion.

**Shift rows:** Transformation in the Cipher that processes the State by cyclically shifting the last three rows of the State by different offsets to provide inter-column diffusion.

**Mix columns:** Transformation in the Cipher that takes all of the columns of the State and mixes their data (independently of one another) to produce new columns which provides inter-Byte diffusion.

**Add round key:** Transformation in the Cipher and Inverse Cipher in which a RoundKey is added to the State using an XOR operation which provides confusion.

This paper introduces a new modification on AES algorithm to exhibit a substantial avalanche effect, to ensure that no trapdoor is present in the cipher, to make the key schedule so strong that the knowledge of one round key does not help in finding the cipher key or other round keys, and to resist the frequency analysis on ciphertext.

### B. The MDS Matrix

Maximum distance separable matrixes (MDS) are widely used in design of block ciphers and hash functions etc. Based on the character of its differential branch number, MDS matrix is widely used and the arithmetic using MDS matrixes can effective against differential cryptanalysis and linear cryptanalysis. A linear code over Galois field $GF(2^p)$ is denoted as an $(n,k,d)$ code, where $n$ is the symbol length of the encoded message, $k$ is the symbol length of the original message, and $d$ is the minimal symbol distance between any two encoded messages[3].

**Definition 1:** Let $K$ be a finite field and $p$ and $q$ be two integers. Let $x \mapsto M \times x$ be a mapping from $K^p$ to $K^q$ defined by the $q \times p$ matrix $M$. We say that it is a linear multipermutation (or an MDS matrix) if the set of all pairs $(x, M \times x)$ is an MDS code, i.e. a linear code of dimension $p$, length $p+q$ and minimal distance $q+1$ [4].

The following theorem [5] will depict the character of MDS matrix from the angle of a subdeterminant.

**Theorem 1:** A matrix is an MDS matrix if and only if every sub-matrix is non-singular.

MDS matrices are constructed by two types of matrices: circulant and Hadamard matrices.

**Circulant matrices:** Given $k$ elements $\alpha_0, \alpha_1, ....., \alpha_{k-1}$, a circulant matrix $M$ is constructed with each entry $M_{i,j} = \alpha_{(i+j) \bmod k}$.

**Hadamard matrices:** Given $k$ elements $\alpha_0, \alpha_1, ....., \alpha_{k-1}$, a Hadamard matrix $M$ is constructed with each entry $M_{i,j} = \alpha_{(i \oplus j)}$.

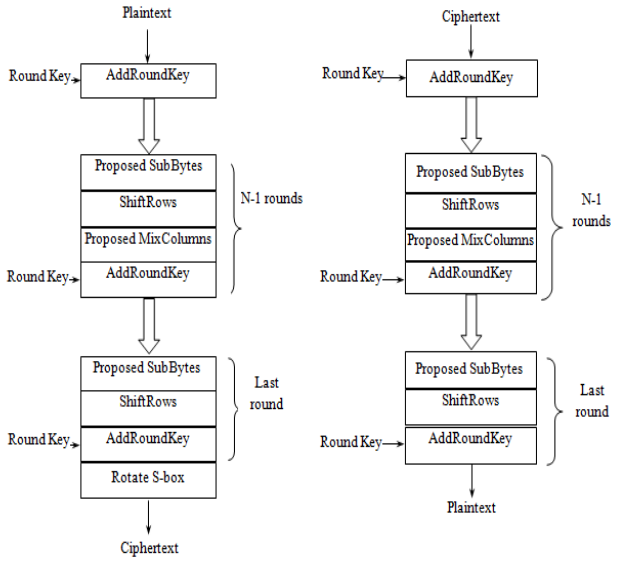
Manuscript received January 11, 2013; revised April 12, 2013.

Fatma Ahmed is with the Dept. of Electrical Engineering, Alexandria Higher Institute of Engineering and Technology (AIET) Alexandria, Egypt (e-mail: moonyally@yahoo.com).

Dalia Elkamchouchi is with the Dept of Electrical Engineering, Faculty of Engineering, Alexandria University Alexandria, Egypt (e-mail: Daliakamsh@yahoo.com).

## II. SDK-AES

SDK-AES is block cipher; it can encrypt blocks ofplaintext of length 128 byte into blocks of the same length. The key length can be 128, 192, or 256 bytes. The total number of rounds depends on the key length that can be 10, 12 or 14 respectively. We assume a key length of 128 byte, which is likely to be the one most commonly implemented. The input to the encryption and decryption algorithms is block of length 128 byte. This block is copied into the $16 \times 8$ matrix of bytes, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix. Similarly the 128 byte key is depicted as $16 \times 8$ matrix of bytes. This key is then expanded into an array of key schedule words; each word is four bytes and the total key schedule is $32 \times 11$ words. The encryption and decryption process of SDK-AES resembles that of AES. The Fig. 1 shows the overall structure of SDK-AES.




(a) Encryption structure (b) Decryption structure

Fig. 1. SDK-AES algorithm

### A. S-Boxes Bank

SDK-AES is a technique seeking to make AES key depending. In this paper, key dependent S-boxes bank act like rotor mechanism [6] is introduced. The S-boxes bank contains two S-boxes, the first S-box is one used for AES algorithm. In order to maintain the structure of cipher to be simple, the second S-box constructs from shifted the first S-box by amount calculated by the user key. At the first step we compute the Checksum "Adler-32" for the user key. The resulting output is thirty-two bits. We use the checksum algorithm because it will yield a different result when the user key is changed. The resulting output from the Checksum will be divided into four sub blocks each one with length 8 bits. These four sub blocks will be XORed together to produce one block with length eight bits representing the number that we use to shift the first S-box to generate the second one. In encryption process, the input byte is mapped into a new byte through the S-boxes bank. At the first, the input byte is mapped by using the first S-box. The output will be the input to the second S-box. If the S-box rotates one byte then after 256 times the S-box will return to its initial position and this operation will be repeated every 256 bytes. In our system, the second S-box will be rotated by irregular step. This rotation consists of two steps: the first step is rotating the second S-box in the tenth round after mapping each byte by odd numbers (1, 3, 5 and 7). First we rotate the second S-box by one byte until we reach 256 bytes then we rotate it by three bytes for the next 256 input bytes. Then we rotate it by five bytes for the next 256 input bytes. Finally we rotate it by seven bytes until we have 256 input bytes. At the second step we rotate the second S-box after the tenth round is completed by one byte. These two steps guarantee that the S-box rotates in irregular manner because after encrypt each block of plaintext the second S-box will be in different arrangement so even we have repeated data, the output will be totally different. The basic idea is make the S-box like rotor cryptosystem with maintain the security and the simple decryption. In decryption algorithm we don't rotate the S-boxes or even scanned for output like rotor, we only subtract the output from the inverse-second S-box with the number of times that second S-box bytes rotated. We rotate the second S-box only and keep the first one stationary because the second one only known to sender and receiver.

### B. ShiftRows Transformation

The input data is arranged in sixteen rows and eight columns. The forward shift row transformation is performed in following way: The first and ninth row of State is not altered. For the second row, a 1-byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed. And so on until the eighth row, a 7-byte circular left shift is performed [7]. Then we circular left shift the tenth row by one byte, the eleventh row by two bytes and so on until the sixteen row, a seven-byte circular left shift is performed. The inverse shift row transformation performs the circular shifts in the opposite direction for each of the fourteenth rows.

### C. The New Efficient MDS Matrix

In SDK-AES, we design new dynamic MDS matrixes which depend on the user key. The number of matrixes is eight each one depends on the user key. In encryption process we only use one matrix from eight for every data block. The choice of this one depends on the subkeys and plaintext. The new matrixes are self inverse so that same matrix can be used for decryption algorithm, which decreases the complexity of system. The new MDS matrix is $4 \times 4$ Circulant matrix. MDS is $(12,8,5)$. MDS property of the matrix is calculated i.e. a $(12,8,5)$ code is MDS if d = n – k + 1.This can be done by checking the branch number of the transformation. The input with one or two active byte column is multiplied with the matrix and the output column is checked, if the total number of active bytes including input and output bytes is equal to 5 then it satisfies the property of MDS. The new MDS matrix is checked for the involution property. We design it by provide the involution conditions which can calculate from the next matrix:

$$\begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ b_3 & b_0 & b_1 & b_2 \\ b_2 & b_3 & b_0 & b_1 \\ b_1 & b_2 & b_3 & b_0 \end{bmatrix} \quad (1)$$

These conditions relative to the above matrix are:

$$b_0 + b_2 = \{01\} \qquad b_1 = b_3 \tag{2}$$

So we choose elements for the new MDS matrix that satisfy these conditions. We choose $b_1 = b_3 = 1$. Because multiplications by 1 are "free" operations so they can improve the computational efficiency of MDS matrices. The elements $b_0$ and $b_2$ are depend on key. At the first step we take the thirty-two bits (the output from the checksum of the key) and divided it into eight sub blocks each one has length four bits. The value of each sub block represents the element $b_0$ in each D-MDS matrix. The element $b_2$ is calculated from the addition inverse for $b_0$ in $GF(2^4)$.

$$\begin{bmatrix} b_0 & 01 & b_2 & 01 \\ 01 & b_0 & 01 & b_2 \\ b_2 & 01 & b_0 & 01 \\ 01 & b_2 & 01 & b_0 \end{bmatrix} \tag{3}$$

In SDK-AES, The data is copied into the 16×8 matrix of bytes. First we divide the state of data into sub states each one is 4×4 matrix of bytes. So we have eight sub states. Each element in the product matrix is the sum of products of elements of one row and one column. In this case, the individual additions and multiplications are performed in $GF(2^8)$. In our cipher, we ensure that each byte will effect in all other bytes by performed "*row exchange*" after the MDS matrix. We use the following equation 4 which represented mathematica 9sub-program to perform "row exchange":

$$\begin{aligned}
&\text{If}[r == 2, aux = row[6]; row[6] = row[1]; row[1] = aux] \\
&\text{If}[r == 3, aux = row[14]; row[14] = row[9]; row[9] = aux] \\
&\text{If}[r == 4, aux = row[12]; row[12] = row[7]; row[7] = aux] \\
&\text{If}[r == 5, aux = row[8]; row[8] = row[3]; row[3] = aux] \\
&\text{If}[r == 6, aux = row[13]; row[13] = row[2]; row[2] = aux] \\
&\text{If}[r == 7, aux = row[10]; row[10] = row[5]; row[5] = aux] \\
&\text{If}[r == 8, aux = row[16]; row[16] = row[11]; row[11] = aux] \\
&\text{If}[r == 9, aux = row[15]; row[15] = row[4]; row[4] = aux]
\end{aligned} \tag{4}$$

### D. SDK-AES Key Expansion

The SDK-AES key expansion algorithm takes as input a 32-word (128-byte) key and produces a linear array of 32×11 words. This is sufficient to provide a 32-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher. The key expansion procedure of SDK-AES is like the expansion procedure of AES.

## III. Security Analysis

### A. Avalanche Effect

In cryptography, the avalanche effect refers to a desirable property of cryptographic algorithms. The avalanche effect is evident when an input is changed slightly (for example, flipping a single bit) the output changes significantly (e.g. half the output bits flip). In the case of quality block ciphers, such a small change in either the key or the plaintext should cause a drastic change in the ciphertext. Constructing a cipher to exhibit a substantial avalanche effect is one of the primary design objectives. The avalanche effect is calculated as:

$$\text{Avalanche Effect} = \frac{\text{No. of flipped in the ciphered text}}{\text{No. of bits in the ciphered text}} \times 100\% \tag{5}$$

In our case, we take two plaintexts and two blocks of data of length 128 bytes flipping one bit from everyone in different positions and calculate the avalanche effect. Then we flip the user key in different positions and calculate the avalanche effect [8]. The following results are obtained after calculating the respective Avalanche Effects.

TABLE I: AV Effect for 1 Bit Change in the Plaintext

| Plaintext | Length of plaintext in bits | Change first bit in plaintext | | Change last bit in plaintext | | Change middle bit in plaintext | |
|---|---|---|---|---|---|---|---|
| | | AES | SDK-AES | AES | SDK-AES | AES | SDK-AES |
| Case 1 | 158720 | 0.03% | 0.33% | 0.04% | 0.32% | 0.04% | 0.36% |
| Case 2 | 200000 | 0.04% | 0.3% | 0.04% | 0.3% | 0.03% | 0.3% |
| Case 3 | 1024 | 5.8% | 51.4% | 5.9% | 52.3% | 5.3% | 51.9% |
| Case 4 | 1024 | 7.1% | 51.4% | 7.1% | 51.3% | 6.3% | 52.7% |

TABLE II: AVA Effect for 1 Bit Change in the User Key

| Plaintext | Length of plaintext in bits | Change first bit in key | | Change middle bit in key | | Change last bit in key | |
|---|---|---|---|---|---|---|---|
| | | SDK-AES | AES | SDK-AES | AES | SDK-AES | AES |
| Case 1 | 158720 | 53.1% | 49.2% | 52.1% | 49.2% | 53.1% | 49.1% |
| Case 2 | 200000 | 52.1% | 45.3% | 51.5% | 45.3% | 51.1% | 48.5% |
| Case 3 | 1024 | 52.8% | 47.7% | 52.7% | 47.7% | 52.8% | 47.7% |
| Case 4 | 1024 | 52.5% | 48.8% | 52.9% | 48.8% | 52.5% | 49.9% |

The avalanche effect of the proposed algorithm is producing very high change in ciphertext as comparison with AES because in AES if only one bit changes, it effects on its data block not all the blocks, while in SDK-AES because we rotate S-box using the output ciphertext so if only one bit changes it produces different rotation in S-box.

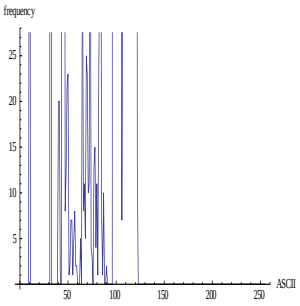
### B. Secret Data Groups

Considering the secret data used in AES, the brute force attack for the key in the case of 128 bit block is $(2^{128} = 3.4 \times 10^{38})$. Considering the secret data used in SDK-AES, the brute force attack for the key in the case of 128 bytes block is $2^{128\times 8} = 1.8 \times 10^{308}$. The S-box in SDK-AES will produce different substitution in the last round because it shifted after every output byte by irregular step. To calculate the brute force attack in this case we will find a huge number of trails to break the system.

### C. Language Statistics

Language redundancy [9] is the greatest problem for any cryptosystem. The cryptanalyst uses the language redundancy to attack cryptosystems ciphertext. If the message is long enough, the cryptanalyst computes the frequency of each of the characters and consider different number of combinations up to the length of the cryptosystem block. The cryptanalyst will then try to estimate the plaintext from this statistical result. A cryptosystem is considered unbreakable against statistical analysis if its ciphertext has

flat distribution. To implement the strength of new SDK-AES, Figs 2&3 show the plaintext statistics of the used file. The ciphertext statistics of AES and new SDK-AES are plotted in Figs 4 to 7. From these figures we find that our new system effectively hides the characteristics of ciphertext especially for repeated data.



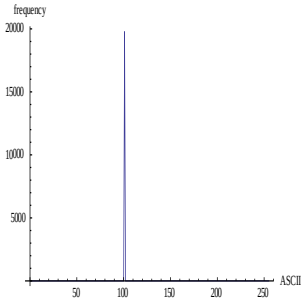




Fig. 2. Plaintext statistics of a text file. Fig. 3. Repeated plaintext statistics

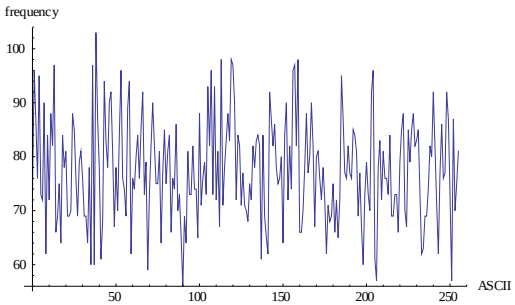



Fig. 4. SDK-AES system ciphertext statistics

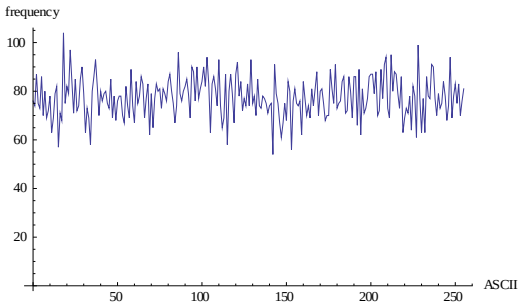



Fig. 5. AES ciphertext statistics

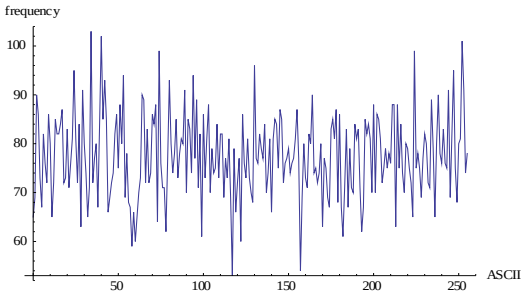



Fig. 6. SDK-AES ciphertext statistics of a message consisting of 20 Kbytes of character "*e*"

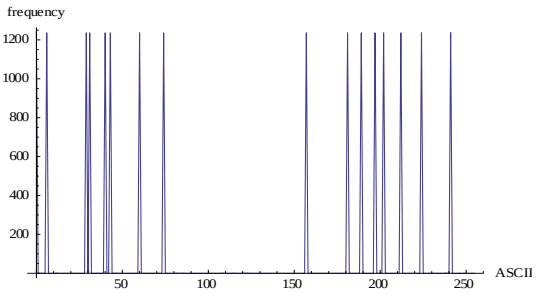



Fig. 7. AES ciphertext statistics of a message consisting of 20 Kbytes of character "*e*"

### *D. NIST Statistical Suite*

The National Institute of Standards and Technology (NIST) [10] develops a Test Suite as a statistical package consisting of 16 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based Cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non randomness that could exist in a sequence. Some tests are decomposable into a variety of subtests. The average values of the statistical tests for both algorithms were given in Table III.

TABLE III: SDK-AES VS. AES STATISTICAL TESTS

| **Test name** / **Algorithm** | SDK-AES | | AES | |
|---|---|---|---|---|
| Frequency (Monobit) Test | 100% | Pass | 100% | Pass |
| Frequency Test within a Block | 100% | Pass | 100% | Pass |
| Runs Test | 100% | Pass | 100% | Pass |
| the Longest Run of 1's in a Block Test | 100% | Pass | 100% | Pass |
| Binary Matrix Rank Test | 100% | Pass | 100% | Pass |
| Discrete Fourier Transform Test | 100% | Pass | 100% | Pass |
| Non-overlap Template Matching Test | 100% | Pass | 100% | Pass |
| Overlap Template Matching Test | 100% | Pass | 97% | Failed |
| Maurer's "Universal Statistical" Test | 100% | Pass | 100% | Pass |
| Lempel-Ziv Compression Test | 100% | Pass | 99% | Pass |
| Linear Complexity Test | 100% | Pass | 98% | Failed |
| Serial Test | 99% | Pass | 97% | Failed |
| Approximate Entropy Test | 100% | Pass | 100% | Pass |
| Cumulative Sums (Cusum) Test | 99% | Pass | 100% | Pass |
| Random Excursions Test | 100% | Pass | 97% | Failed |
| Random Excursions Variant Test ($\alpha$ = 0.05) | 98% | Pass | 96% | Pass |

## IV. CONCLUSION

In this paper, a new improved version of AES has been proposed. SDK-AES doesn't contradict the security and simplicity of the AES algorithm. We tried to keep all the mathematical criteria for AES without change. We have improved the security of AES by increase the size of data block to 128 bytes and the size of key to 128, 192 and 256 bytes. Also we make its S-box to be acting like the rotor cryptosystem with maintaining the decryption operation simple and make it to be key dependent. . We have improved the MDS matrix and make it depends on the user key and be itself at decryption (involution property). In SDK-AES system if only one bit change in the plaintext or the user key, it causes more than half of the ciphertext to be change. Finally, our proposal is rigid to withstand the well-known methods of brute-force.

## REFERENCES


[1] J. Daemen and V. Rijmen, *The Design of Rijndael: AES–The Advanced Encryption Standard*, Springer-Verlag, 2002.

[2] W. Stallings, *Cryptography and Network Security Principles and Practices*, Prentice Hall, Fourth Edition, 2005.

[3] B. A. Forouzan, *Cryptography and Network Security*, TATA-Mcgraw hill publication 2007 edition.

[4] P. Junod and S. Vaudenay, "Perfect diffusion primitives for block ciphers: building efficient MDS matrices," in *Proc. Selected Areas in Cryptography 2004*, Waterloo, Canada, pp. 84-99, August 9-10, 2004.

[5] F. MacWilliams and N. Sloane, "The theory of error-correcting codes," North-Holland Mathematical Library, vol. 16, pp. 762, 1977.

[6] W. O. Chan, "Cryptanalysis of SIGABA," Master's Thesis, Department of Computer Science, San Jose State University, May 2007.

[7] J. Daemen and V. Rijmen, *AES Proposal: Rijndael, AES Algorithm: Submission*, September 3, 1999, available at [2].

[8] A. Kumar, "Effective implementation and avalanche effect of AES," *International Journal of Security, Privacy and Trust Management (IJSPTM)*, vol. 1, no. 3/4, pp. 31-35, August 2012.

[9] B. Schneier, *Applied Cryptography, Protocols, Algorithms, and Source Code in C*, Wiley Computer Publishing, Second Edition, John Wiley & Sons, Inc.

[10] NIST, *A Statistical Test Suite for Random and Pseudorandom Generators for Cryptographic Applications*, NIST Special Publication, 2003.


**Dalia Elkamchouchi** held a Masters' of science in Electrical Engineering from Faculty of Engineering, Alexandria University. She works on Alexandria Higher Institute of Engineering and Technology. She Held a Ph.D. in Electrical Engineering from Faculty of Engineering, Alexandria University.

**Fatma Ahmed** held a Masters' of science in Electrical Engineering from Faculty of Engineering, Alexandria University. She works on Alexandria Higher Institute of Engineering and Technology. She studies for Ph.D. in Electrical Engineering from Faculty of Engineering, Alexandria University.