Journal of Computer Science 2 (1): 48-52, 2006
ISSN1549-3636

On Partial Linearization of Byte Substitution Transformation of Rijndael-The AES

[1]Y.Talwar, [2]C.E.Veni Madhavan and [3]Navin Rajpal
[1]Guru Gobind Singh Indraprastha University, Delhi and National Informatics Centre, Delhi, India
[2]Indian Institute of Science, Bangalore, India
[3]Guru Gobind Singh Indraprastha University, Delhi, India

Abstract: Rijndael-The AES[1-3] is 128-bit block cipher based on an elegant algebraic structure over $\mathbf{F}_2{}^8$. This cipher employs a simple approach to its substitution, permutation (SP) operations. We take a close look at its internals; the byte substitution transformation function is the only non-linear function in Rijndael - The AES. This transformation comprises of two steps operating on each byte. Here we are trying to remodel this to one step operation using indicator vector matrix representation. This representation is further extended to mathematically represent one complete encryption or decryption round of Rijndael the using indicator vector matrix representation that can be explored for better crypto-analysis[4,5] of the cipher.

Key words: Rijndael, byte substitution transformation, indicator vector matrix

INTRODUCTION

Rijndael Algorithm[1-3] was designed by two Belgian cryptographers: Vincent Rijmen and John Daemen, as one of the candidates for the Advanced Encryption Standard (AES) selection. The AES committee was formulated by the U.S. Government under the umbrella of National Institute of Standards and Technology (NIST) to find another cryptographic algorithm in order to replace the existing 64-bit block cipher of 1977 - the Data Encryption Standards (DES) to protect sensitive digital information over the next few decades.

After a stringent qualifying process of three rounds involving the whole world's cryptographic community[6], Rijndael algorithm was proposed by the AES committee as Advanced Encryption Standard – The AES on Nov. 26, 2001. Later on May 26, 2002 NIST endorsed it as Federal Information Processing Standard namely FIPS-197 replacing DES (FIPS-46).

Rijndael possesses an elegant algebraic structure over $F_2{}^8$[7-10]. It supports a variable block size and variable key size of 128, 160, 192, 224 or 256 bits each. But for the AES, its block size is fixed to 128-bits and keeping the variable key size of 128, 192 and 256 bits. It has 10, 12 or 14 iterations of round transformations depending on the key size of 128, 192 or 256 bits respectively in conjunction with an initial round of key addition. Each (except the last) round transformation function is composed of the four sub transformation functions: Byte Substitution or *bs,* Row Shift or *rs*, Mix Column or *mc* and Add Round Key or *ak*. The last round transformation does not include the *mc* function.

In this study we present an analysis of the block cipher Rijndael while concentrating on its 128-bit version. This cipher employs a simple approach to its substitution, permutation (SP) operations. We take a close look at its internals, recast some of these and present the cipher in a manner amenable for better analysis.

Notations: We fix the block size and key size to 128 bit. We consider the 10 round version. We use the following notations.
Let for all round index $i = 0,\cdots,10$ and byte index $j = 0,\cdots,15$:

$X^i_{j:j}$ th text byte of ***i***-th round (in particular, X^0_j is the initial input plain text byte and is fixed
$X^{11}_{j:j}$ th cipher text byte.
$\kappa^i_{j:j}$ th expanded key byte of ***i***-th round (in particular κ^0_j is the user defined key : $\kappa^0_{j:}\langle k_0, k_1, k_2, \cdots, k_{15}\rangle$
W[i] = *i*-th key word of 32 bits.
k_n: *n*th key byte, $n = \{0,1,2,\cdots,175\}$
N_k = (key size)/32 = 128/32 = 4.
N_b = (block size)/32 = 128/32 = 4.
N_r = No. of cipher rounds = 10.

We use the standard convention of representing elements of $F_2{}^8$ as polynomials of degree 7, over F_2. We also adopt the standard practice of treating the elements of $F_2{}^8$ as integers in the range 0, … ,255. Thus for example, $\alpha \in F_{2^8}$ with $\alpha = x^7 + x^6 + x^2 + x^1 + 1$ would be referred as $\alpha = 199$, without ambiguity.

Corresponding Author: Y.Talwar, Guru Gobind Singh Indraprastha University, Delhi & National Informatics Centre, Delhi, India

We define three functions namely *Rotbyte*(.), *Rc*(.), *Rcon*(.) & *Iv*[.] an indicator vector:

i. Rotbyte(.) rotates the bytes of key within the word, when word oriented structure is considered for key expansion mechanism. If k_0,k_1,k_2,k_3 are four bytes of *i*-th key-word W[i] arranged in big endian format, $\text{Rotbyte}(W[k_0,k_1,k_2,k_3]) = W[k_1,k_2,k_3,k_0]$

The byte substitution transformation of Rijndael uses an S-box, generated over F_2^8 with $(x+1)\left(\equiv(03_{\text{base }16})\right)$ as primitive element and $g(x)=\left(x^8+x^4+x^3+x+1\right)$ as the defining irreducible polynomial along with an affine transformation of $\left(x^6+x^5+x+1\right)\equiv(63_{\text{base}16})$. Thus, *bs*, using S-box, transforms the individual byte *a(x) to bs(a(x)).* Mathematically:

$$bs(a(x))=\left(x^6+x^5+x+1\right)+c(x)\left(x^4+x^3+x^2+x+1\right)\left(\text{mod}\left(x^8+1\right)\right)$$

where $c(x)=a\left(x\right)^{-1}(\text{mod}\,g(x))$

Similarly

$$bs\left(W\left[k_0,k_1,k_2,k_3\right]\right) = W\left[bs(k_0),bs(k_1),bs(k_2),bs(k_3)\right]$$

and

$$Rotbyte\left(bs\left(W\left[k_0,k_1,k_2,k_3\right]\right)\right) = W\left[bs(k_1),bs(k_2),bs(k_3),bs(k_0)\right]$$

ii. Rc(a(x)) is another round dependent byte oriented constant function defined over F_2^8. *POW(a(x))* contains powers of a(x) in the field. Then

$$Rc(a(x)) = POW(a(x))(\text{mod}\,g(x))$$

In particular, for $a(x)\in\{1,2,\cdots,10\}$

$$Rc(a(x))=\{1,2,4,8,16,32,64,128,27,54\}$$

iii. Rcon(a(x)) is a round dependent word oriented function such that $\text{Rcon}(a(x))=(\text{Rc}(a(x)),0,0,0)$. Here the commas define separation of each byte arranged in big endian format.

iv. Indicator vector representing a byte, say $a(x)=x^4+x^3+1\equiv 25,$ is a 256x1 matrix with 1 only at 25th position and zeros elsewhere, i.e. the vector representing *a*(*x*) has 1 at the place corresponding to the numerical value of the byte and zero at all other positions in the matrix 0 to 255. Hence it will be of the form:

Iv[*a*(*x*)] : [0 0 0 0 0 … 0 1 0 0 … 0]^T
↑0 ↑25 ↑255

Brief description of Rijndael internals: Rijndael has an elegant algebraic structure over F_2^8. The input plain text or the output cipher text of block size of 128-bits is viewed as a 4x4 matrix of 16 bytes arranged in a column major format. Rijndael consists of an initial round of key addition (*ak*) followed by 10 iterations of round transformations for the key size of 128-bits. Each (except the last) round transformation function is composed of the four sub transformation functions: Byte Substitution or *bs,* Row Shift or *rs*, Mix Column or *mc* and Add Round Key or *ak*. The last round transformation does not include the *mc* function.

Byte substitution transformation: *bs:* This is the only non-linear transformation in the entire Rijndael structure. It operates independently on each byte using a substitution table (S-box). The S-box, which is invertible in nature, is composed of two transformations:

1. Taking multiplicative inverse of the desired byte in the finite field GF(2^8) with $(x+1)\equiv(03_{\text{base }16})$ as primitive element and $g(x)=\left(x^8+x^4+x^3+x^1+1\right)$ as the defining irreducible polynomial. The element $00_{\textbf{\textit{base }}16}$ is mapped to itself.
2. Applying an affine transformation of $\left(x^6+x^5+x^1+1\right)$ equivalently $63_{\text{base }16}$.

Thus, the byte substitution operation transforms a byte *a*(*x*) to bs(*a*(*x*)) as per the following relation. Let

1. $c(x)=a(x)^{-1}(\text{mod}\,g(x))$
2. $bs(a(x))=\left(x^6+x^5+x+1\right)+c(x)\left(x^4+x^3+x^2+x^1+1\right)(\text{mod}\left(x^8+1\right))$

The inverse S-box is constructed by taking an inverse affine transform followed by a multiplicative inverse in the finite field F_{2^8}.

1. $\boldsymbol{c(x)=(x^2+1)+bs(a(x))(x^6+x^3+x)(\text{mod}(x^8+1))}$
2. $a(x)=c(x)^{-1}(\text{mod}\,g(x))$

Row shift transformation: *rs*: The 16 input bytes are arranged in a column major format of a 4x4 matrix. To achieve the desired confusion, a linear transformation *rs* is applied. Here, the bytes in each row of the matrix are given a cyclic left shift. For *i* = 1, 2, 3, 4 the bytes in the i-th row are circularly left shifted by (*i*-1) bytes.

The inverse of a row shift transformation is obtained by cyclically shifting the bytes in the reverse direction i.e. circularly right shifting 0, 1, 2, and 3 bytes in the first, second, third and fourth row of the 4x4 input matrix, respectively.

Mix column: *mc*: The linear transformation mix column provides the diffusion by mixing the bits of each column. The function $\beta(z)$, given below, operates on the input column by treating it as a degree three polynomial in $F_{2^8}[z]$. This polynomial is multiplied by a rotated version of a standard polynomial $m(z) \in F_{2^8}[z]$ given by $[m(z)] = 03z^3 + 01z^2 + 01z^1 + 02$ and reduced modulo the polynomial $(z^4+1) \in F_{2^8}[z]$. Here the coefficients denote elements of F_{2^8}. It is known that the coefficients of $m(z)$ are so chosen that the result $\beta(z).m(z)$is invertible modulo $(z^4 + 1)$ although this polynomial is reducible over F_2.

For example, a column of mc, $[a_0, a_1, a_2, a_3]^T$ is considered as:

$$\beta(z) = a_3z^3 + a_2z^2 + a_1z + a_0 \in F_{2^8}[z]$$

Then,

$$\begin{aligned} m(z)\cdot\beta(z) = {} & (03\cdot a_3)z^6 + (03\cdot a_2 + 01\cdot a_3)z^5 \\ & + (03\cdot a_1 + 01\cdot a_3 + 01\cdot a_2)z^4 \\ & + (03\cdot a_0 + 02\cdot a_3 + 01\cdot a_2 + 01\cdot a_1)z^3 \\ & + (01\cdot a_0 + 02\cdot a_2 + 01\cdot a_1)z^2 \\ & + (01\cdot a_0 + 02\cdot a_1)z + (02\cdot a_0)z^0 \\ & \left(\mathrm{mod}\left(z^4+1\right)\right) \end{aligned}$$

$$m(z).\beta(z) = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

The inverse of Mix column transformation is similar to the forward operation with the only difference that the inverse of the fixed polynomial i.e. $[m(z)]^{-1}$ is used and it is given by

$$[m(z)]^{-1} = 11z^3 + 13z^2 + 09z^1 + 14$$

Hence,

$$[m(z)]^{-1}.\beta(z) = \begin{bmatrix} 14 & 11 & 13 & 09 \\ 09 & 14 & 11 & 13 \\ 13 & 09 & 14 & 11 \\ 11 & 13 & 09 & 14 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Add Round Key: *ak*: In this function, the round key is added to the current byte as bit-wise exclusive OR. The XOR operation is the inverse of itself.

Modified Rijndael's key expansion mechanism: The Key expansion mechanism for 128-bit key size, in Rijndael is defined in the following manner.

The expanded key of $N_b*(N_r+1)(=44)$ words is derived from the 4 words of the user defined key. The first N_k (= 4) words, $W[0]$, … , $W[3]$ of the expanded key are filled with the user defined original cipher key bits. The subsequent key words for all $N_k \le i < (N_b * (N_r + 1))$ i.e. $4 \le i < 44$ alternatively $i = \{4, \ldots, 43\}$ are given by:

$$W[i] = \begin{cases} [W[i-N_k] \oplus \mathrm{Rotbyte}(bs(W[i-1])) \\ \quad \oplus \mathrm{Rcon}(i/N_k) \quad \forall i = 0(N_k)] \\ [W[i-N_k] \oplus W[i-1] \\ \quad \forall i \neq 0(N_k)] \end{cases}$$

We have modified the key expansion algorithm in the following manner: As the functions *bs*(.) and *Rcon*(.) transformations inherently operate on individual bytes of every input word, thus, a modified byte oriented version for key expansion algorithm can be derived. Therefore, for the present study with key size and block size of 128 bits and 10 cipher rounds, a total of 176 [= 4* (N_b * (N_r + 1))] bytes from the 16 bytes (=128 bits) of the user defined key k_n with n = {0,…,15} are to be expanded.

First $4*N_k$ (=16) bytes, defined as $K_j^0 : \langle k_0, k_1, k_2, \cdots, k_{15} \rangle$ of the expanded key are filled with the original 128 user defined key bits stored in big endian format. For subsequent rounds, the expanded key bytes at n = {16, …, 175} are given by the following relations:

i. when $n = 0 \pmod{4*N_k}$, the four consecutive key bytes at n to n+3 locations are obtained through:

$$\begin{aligned} k_n &= k_{n-16} \oplus bs(k_{n-3}) \oplus Rc(n/16) \\ k_{n+1} &= k_{(n+1)-16} \oplus bs(k_{n-2}) \\ k_{n+2} &= k_{(n+2)-16} \oplus bs(k_{n-1}) \\ k_{n+3} &= k_{(n+3)-16} \oplus bs(k_{n-4}) \end{aligned}$$

ii. The subsequent expanded key bytes for a particular round i.e. from (n+4)th byte to (n+15)th byte of k_n, are obtained through: $k_n = k_{n-16} \oplus k_{n-4}$

Alternatively, these expanded key bytes can be obtained in the form of round keys K_j^i through the following relations with the original key bytes filled at i =0 & j = 0, …, 15 in K^0_j. For $0 \le i < 10$

$$\begin{aligned} K_0^{i+1} &= K_0^i \oplus bs(K_{13}^i) \oplus Rc(i+1) \\ K_1^{i+1} &= K_1^i \oplus bs(K_{14}^i) \\ K_2^{i+1} &= K_2^i \oplus bs(K_{15}^i) \\ K_3^{i+1} &= K_3^i \oplus bs(K_{12}^i) \\ K_4^{i+1} &= K_4^i \oplus K_0^i \oplus bs(K_{13}^i) \oplus Rc(i+1) \\ K_5^{i+1} &= K_5^i \oplus K_1^i \oplus bs(K_{14}^i) \\ K_6^{i+1} &= K_6^i \oplus K_2^i \oplus bs(K_{15}^i) \\ K_7^{i+1} &= K_7^i \oplus K_3^i \oplus bs(K_{12}^i) \\ K_8^{i+1} &= K_8^i \oplus K_4^i \oplus K_0^i \oplus bs(K_{13}^i) \oplus Rc(i+1) \\ K_9^{i+1} &= K_9^i \oplus K_5^i \oplus K_1^i \oplus bs(K_{14}^i) \\ K_{10}^{i+1} &= K_{10}^i \oplus K_6^i \oplus K_2^i \oplus bs(K_{15}^i) \\ K_{11}^{i+1} &= K_{11}^i \oplus K_7^i \oplus K_3^i \oplus bs(K_{12}^i) \\ K_{12}^{i+1} &= K_{12}^i \oplus K_8^i \oplus K_4^i \oplus K_0^i \oplus bs(K_{13}^i) \oplus Rc(i+1) \\ K_{13}^{i+1} &= K_{13}^i \oplus K_9^i \oplus K_5^i \oplus K_1^i \oplus bs(K_{14}^i) \\ K_{14}^{i+1} &= K_{14}^i \oplus K_{10}^i \oplus K_6^i \oplus K_2^i \oplus bs(K_{15}^i) \\ K_{15}^{i+1} &= K_{15}^i \oplus K_{11}^i \oplus K_7^i \oplus K_3^i \oplus bs(K_{12}^i) \end{aligned}$$

Till now we have discussed briefly the internals of Rijndael algorithm. Now we are going to present the modified form of the *bs* followed with indicator vector matrix representation of one complete round involving

all the four transformation functions namely: *bs, rs mc* and *ak* in the subsequent sections.

Modified byte substitution *bs* transformation: The *bs* transformation, as stated in previous section comprises of two steps – first step is to calculate multiplicative inverse of the desired byte followed by the second step of an affine transformation. Let $b_i \langle b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7 \rangle$ represents the bits of a byte as a vector in big endian format. In matrix form, the affine transformation component of the S-box can be expressed as:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix}_{8\times1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}_{8\times8} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}_{8\times1} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}_{8\times1}$$

The above matrix can be compactly represented as:

$$B' = A \cdot B \oplus AF \quad (1)$$

Where, $A \equiv \left(x^4 + x^3 + x^2 + x^1 + 1\right)$ over $F_2{}^8$

and $AF \equiv \left(x^6 + x^5 + x + 1\right)$ over $F_2{}^8$

The matrix representation of byte *B* can further be represented as product of two matrices: *FF*[8 x 256] and *Iv*[256 x 1]. Each row of *FF* matrix represents the multiplicative inverse in bit vector form of the corresponding individual byte. *Iv* matrix, as described earlier, gives the indicator vector representation of the byte under consideration. Hence, the equation (1) transforms to:

$$B' = A \cdot FF \cdot Iv \oplus AF \quad (2)$$

The inverse byte substitution transformation can similarly be represented as:

$$B' = A^{-1} \cdot FF \cdot Iv \oplus AF^{-1} \quad (3)$$

Where, $A^{-1} \equiv \left(x^6 + x^3 + x\right)$ over $F_2{}^8$

and $AF^{-1} \equiv \left(x^2 + 1\right)$ over $F_2{}^8$

Here, we partially linearized the *bs* transformation. In the next section we extend this formulation idea of *Iv* to recast the Rijndael round functions to represent mathematically as a simple *Iv* matrix relation.

Recasting of Rijndael internals

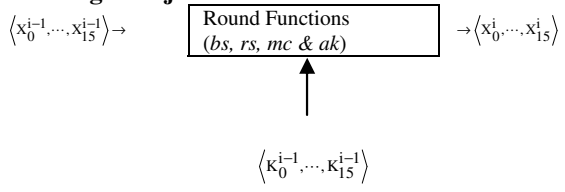

Fig. 1: Pictorial representation of an *i*-th round transformation function

The Fig. 1 gives the pictorial representation of an *i*-th round transformation function. We recast the cipher round with an abuse of notation in the following manner:

Let $\mu_j, j = 1,2,3$ are the "operators" such that:

$$\mu_1 = \mu = bs(a(x)),$$

$$\mu_2 = 2 \cdot \mu = x \cdot bs(a(x)) \pmod{g(x)},$$

$$\mu_3 = 3 \cdot \mu = (x+1) \cdot bs(a(x)) \pmod{g(x)},$$

These μ_j`s correspond to the *mc* transformation of the *bs* transformed byte and their position in the matrix *R* corresponds to the ***rs*** transformation on the byte *X*.

$$R = \begin{bmatrix} \mu_2 & 0 & 0 & 0 & 0 & \mu_3 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_1 \\ \mu_1 & 0 & 0 & 0 & 0 & \mu_2 & 0 & 0 & 0 & 0 & \mu_3 & 0 & 0 & 0 & 0 & \mu_1 \\ \mu_1 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_2 & 0 & 0 & 0 & 0 & \mu_3 \\ \mu_3 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_2 \\ 0 & 0 & 0 & \mu_1 & \mu_2 & 0 & 0 & 0 & 0 & \mu_3 & 0 & 0 & 0 & 0 & \mu_1 & 0 \\ 0 & 0 & 0 & \mu_1 & \mu_1 & 0 & 0 & 0 & 0 & \mu_2 & 0 & 0 & 0 & 0 & \mu_3 & 0 \\ 0 & 0 & 0 & \mu_3 & \mu_1 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_2 & 0 \\ 0 & 0 & 0 & \mu_2 & \mu_3 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_1 & 0 \\ 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_1 & \mu_2 & 0 & 0 & 0 & 0 & \mu_3 & 0 & 0 \\ 0 & 0 & \mu_3 & 0 & 0 & 0 & 0 & \mu_1 & \mu_1 & 0 & 0 & 0 & 0 & \mu_2 & 0 & 0 \\ 0 & 0 & \mu_2 & 0 & 0 & 0 & 0 & \mu_3 & \mu_1 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 \\ 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_2 & \mu_3 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 \\ 0 & \mu_3 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_1 & \mu_2 & 0 & 0 & 0 \\ 0 & \mu_2 & 0 & 0 & 0 & 0 & \mu_3 & 0 & 0 & 0 & 0 & \mu_1 & \mu_1 & 0 & 0 & 0 \\ 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_2 & 0 & 0 & 0 & 0 & \mu_3 & \mu_1 & 0 & 0 & 0 \\ 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_1 & 0 & 0 & 0 & 0 & \mu_2 & \mu_3 & 0 & 0 & 0 \end{bmatrix}_{4096\times4096}$$

R represented above seems to be a 16x16 matrix but actually is 4096 x 4096 with each of μ_j as 256x256 matrix and each '0' also represents a null matrix of size 256x256. Further, each row of μ_j is an indicator vector representation of corresponding byte of *mc* operated *S*-box.

Thus, one round of Rijndael can completely be characterized as:

$$X^i = R \cdot X^{i-1} \oplus K^{i-1} \quad (4)$$

where $X^i = \left[X_0^i, \cdots, X_{15}^i\right]^T$: vector of 16 indicator vectors form 4096 x 1

$X^{i-1} = \left[X_0^{i-1}, \cdots, X_{15}^{i-1}\right]^T$: vector of 16 indicator vectors form 4096 x 1

$K^{i-1} = \left[K_0^{i-1}, \cdots, K_{15}^{i-1}\right]^T$: vector of 16 indicator vectors form 4096 x 1

The output X^i does not result in an indicator vector.

CONCLUSION

The algorithm proposed in this paper can be successfully used to remodel the two-step byte substitution transformation to one step. Further, it is possible to represent one complete round of Rijndael using indicator vector matrix representation. The output vector X^i, so obtained deviates from the indicator vector representation. Finding a way of conversion of this byte vector X^i so obtained, to an indicator vector form is still an open problem.

REFERENCES

1. Daemen, J. and V. Rijmen. The block cipher Rijndael. Available from NIST's AES homepage, (http://www.nist.gov/aes)
2. Daemen, J. and V. Rijmen, 1998. AES proposal: Rijndael. In AES Round 1 Technical Evaluation, NIST. (http://www.esat.kuleuven.ac.be/\~rijmen/rijndael/, http://www.nist.gov/aes)
3. Daemen, J. and V. Rijmen. The Design of Rijndael. AES-Advanced Encryption Standard. Springer-Verlag Berlin, Heidelberg, New York.
4. Courtois, N.T. and J. Pieprzyk, 2002. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Asiacrypt 2002, LNCS 2501,Springer-Verlag, pp: 267-287. (http://eprint.iacr.org/2002/044/)
5. Ferguson, N., J. Kelsey, B. Schneier, M. Stay, D. Wagner and D.Whiting, 2001. Improved Cryptanalysis of Rijndael. Fast Software Encryption 2000, LNCS 1978, B. Schneier, Ed., Springer-Verlag, pp: 213-231.
6. Gladman, B., 1999. Implementation experience with the AES candidate algorithms. Proc. 2nd AES Candidate Conf., March 22-23, 1999, Rome, pp: 7-14. (http://fp.gladman.plus.com/cryptography_technology/rijndael)
7. Ferguson, N., R. Schroeppel and D. Whiting, 2001. A simple algebraic representation of Rijndael. Selected Areas in Cryptography, SAC 2001, LNCS 2259, Springer-Verlag, pp: 103-111.
8. Lidl, R., H. Niederreiter, 1988. Introduction to Finite Fields and their Applications. Cambridge University Press.
9. McEliece, R.J.,1987. Finite Fields for Computer Scientists and Engineers. Kluwer Academic Publishers.
10. Menezes, A.J., P.C. van Oorschot and S.A. Vanstone, 1996. Handbook of Applied Cryptography. CRC Press.