Journal of Computer Science 2 (2): 144-147, 2006
ISSN 1549-3636

On the Presence of Cascading Effect in the Key Expansion Mechanism of Rijndael-The AES

[1]Y. Talwar, [2]C.E. Veni Madhavan and [3]Navin Rajpal
[1]Guru Gobind Singh Indraprastha University, Delhi, India 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 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 on the Key Expansion Mechanism of Rijndael - The AES. This study highlights on the presence of the cascading effect in its key expansion mechanism. Thus, lowering the brute-force key guess attack by a factor of 2^{31}. Hence, for the key size of 128 bits the key diversity is 2^{97} instead of 2^{128}.

Key words: Cascading effect, key expansion mechanism, AES

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[4-6], 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} [6,7-9]. 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 on its Key Expansion Mechanism; highlighting on the presence of the repeated pattern in the expanded key bytes in a peculiar manner, which we name as the cascading effect. Due to the presence of this pattern in the key expansion mechanism, the brute-force key guess attack on Rijndael key schedule is lowered by a factor of 2^{31}. Hence, the key size of 128 bits has a key diversity of 2^{97} instead of 2^{128}.

Notations: We fix the block size and key size to 128 bits. 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_j^i : j th text byte of ***i***-th round (in particular, X_j^0 is the initial input plain text byte and is fixed)

X_j^{11} : j th cipher text byte.

K_j^i : j th expanded key byte of ***i***-th round (in particular, K_j^0 is the user defined key : $K_j^0 : \langle k_0, k_1, k_2, \cdots, k_{15} \rangle$)

W[i] = i-th key word of 32 bits.

k_n: *n*th key byte,

N_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.

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

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.
We define three functions namely *Rotbyte*(.), *Rc*(.), *Rcon*(.):

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, 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 $(x+1)\left(\equiv\left(03_{\text{base }16}\right)\right)$ 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 $\left(x^6+x^5+x+1\right)\equiv\left(63_{\text{base16}}\right)$. 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, $c(x)=a(x)^{-1}\left(\text{mod}\, g(x)\right)$
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

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

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 $Rcon(a(x))=(Rc(a(x)),0,0,0)$. Here the commas define separation of each byte arranged in big endian format.

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:

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

b) 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 $\boldsymbol{a}(\boldsymbol{x})$ to $\mathbf{bs}(\boldsymbol{a}(\boldsymbol{x}))$ as per the following relation. Let

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

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

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:

$$\left[m(z)\right]=03z^3+01z^2+01z^1+02$$

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:

$$\beta(z) = a_3 z^3 + a_2 z^2 + a_1 z + a_0 \in F_{2^8}[z]$$

Then,

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

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

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.

Key expansion mechanism of rijndael: 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

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

Till now we have discussed briefly the internals of Rijndael algorithm. Now we are going to present the modified form of the ***ak*** in the subsequent sections.

Modified key expansion mechanism of rijndael: 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 $n = \{0,\ldots,15\}$ 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:

when $n = 0$ (mod 4* N_k), the four consecutive key bytes at n to n+3 locations are obtained through:

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

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≤ i <10

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

In order to guess the keys used for encrypting the data we take a close look at the key expansion mechanism of Rijndael and try to derive some relationship between expanded key values and the initial key values that are stored in K^0_j. The following

are our observations for the first round of expanded key values:

Let $\text{in } K_j^0 : \langle k_0, k_1, k_2, k_3, k_{12}, k_{13}, k_{14}, k_{15} \rangle$; the mentioned eight key bytes are assumed to be known. Hence, K^1_j the first round keys can be obtained as per the following relations:

$$\begin{aligned} K_0^1 &= K_0^0 \oplus \left(bs\left(K_{13}^0\right) \oplus Rc(1)\right) = K_0^0 \oplus K'^0_{13} \\ K_1^1 &= K_1^0 \oplus \left(bs\left(K_{14}^0\right)\right) \quad = K_1^0 \oplus K'^0_{14} \\ K_2^1 &= K_2^0 \oplus \left(bs\left(K_{15}^0\right)\right) \quad = K_2^0 \oplus K'^0_{15} \\ K_3^1 &= K_3^i \oplus \left(bs\left(K_{12}^0\right)\right) \quad = K_3^i \oplus K'^0_{12} \end{aligned} \tag{1}$$

$$\begin{aligned} K_4^1 &= K_4^0 \oplus K_0^0 \oplus bs\left(K_{13}^0\right) \oplus Rc(1) \\ &= K_4^0 \oplus K_0^1 \\ K_5^1 &= K_5^0 \oplus K_1^0 \oplus bs\left(K_{14}^0\right) \\ &= K_5^0 \oplus K_1^1 \\ K_6^1 &= K_6^0 \oplus K_2^0 \oplus bs\left(K_{15}^0\right) \\ &= K_6^0 \oplus K_2^1 \\ K_7^1 &= K_7^0 \oplus K_3^0 \oplus bs\left(K_{12}^0\right) \\ &= K_7^0 \oplus K_3^1 \end{aligned} \tag{2}$$

$$\begin{aligned} K_8^1 &= K_8^0 \oplus K_4^0 \oplus K_0^0 \oplus bs\left(K_{13}^0\right) \oplus Rc(1) \\ &= K_8^0 \oplus K_4^0 \oplus K_0^1 \\ K_9^1 &= K_9^0 \oplus K_5^0 \oplus K_1^0 \oplus bs\left(K_{14}^0\right) \\ &= K_9^0 \oplus K_5^0 \oplus K_1^1 \\ K_{10}^1 &= K_{10}^0 \oplus K_6^0 \oplus K_2^0 \oplus bs\left(K_{15}^0\right) \\ &= K_{10}^0 \oplus K_6^0 \oplus K_2^1 \\ K_{11}^1 &= K_{11}^0 \oplus K_7^0 \oplus K_3^0 \oplus bs\left(K_{12}^0\right) \\ &= K_{11}^0 \oplus K_7^0 \oplus K_3^1 \end{aligned} \tag{3}$$

$$\begin{aligned} K_{12}^1 &= K_{12}^0 \oplus K_8^0 \oplus K_4^0 \oplus K_0^0 \oplus bs\left(K_{13}^0\right) \oplus Rc(1) \\ &= K_{12}^0 \oplus K_8^0 \oplus K_4^0 \oplus K_0^1 \\ K_{13}^1 &= K_{13}^0 \oplus K_9^0 \oplus K_5^0 \oplus K_1^0 \oplus bs\left(K_{14}^0\right) \\ &= K_{13}^0 \oplus K_9^0 \oplus K_5^0 \oplus K_1^1 \\ K_{14}^1 &= K_{14}^0 \oplus K_{10}^0 \oplus K_6^0 \oplus K_2^0 \oplus bs\left(K_{15}^0\right) \\ &= K_{14}^0 \oplus K_{10}^0 \oplus K_6^0 \oplus K_2^1 \\ K_{15}^1 &= K_{15}^0 \oplus K_{11}^0 \oplus K_7^0 \oplus K_3^0 \oplus bs\left(K_{12}^0\right) \\ &= K_{15}^0 \oplus K_{11}^0 \oplus K_7^0 \oplus K_3^1 \end{aligned} \tag{4}$$

Now, using the relations (1) in conjunction with our assumptions, we can evaluate K^1_0 to K^1_3, as all the variables on right hand side of (1) are known. Then using relations (2); K^0_4 to K^0_7 can be evaluated in 2^{32} operational trials. Using the relations (4) K^0_8 to K^0_{11} (or K^1_{12} to K^1_{15}) can be evaluated in 2^{32} operations. Therefore, in all $2^{32} + 2^{32} = 2^{33}$ operations are required to evaluate all K^1_0 to K^1_{15}. Alternatively, K^0_0 to K^0_{15} can be evaluated in 2^{33} operations due to the presence of the cascading effect in Rijndael key expansion mechanism. Thus, together with the prior knowledge of the eight key bytes specifically $k_0, k_1, k_2, k_3, k_{12}, k_{13}, k_{14}, k_{15}$ we can determine the remaining 8 key bytes in 2^{33} operations instead of 2^{64}. Similarly, if four key bytes i.e. k_0, k_1, k_2, k_3 are known then we require only 2^{65} ($=2^{32}$ x 2^{33}) trials and not 2^{96}, to determine remaining 12 key bytes. In general with 2^{97} ($=2^{32}$ x 2^{65}) trials we can determine the key.

CONCLUSION

Due to the presence of cascading effect in Rijndael's key expansion mechanism the brute-force key guess attack for the key size of 128 bits can be launched in 2^{97} operations instead of 2^{128}. Hence, the key diversity for the key size of 128 bits is lowered by a factor of 2^{31}. Further, the key diversity for the key size of 192 and 256 bits has to be explored on the similar lines.

REFERENCES

1. Daemen, J. and V. Rijmen. The block cipher Rijndael. <http://www.nist.gov/aes>
2. Daemen, J. and V. Rijmen, 1998. AES Proposal: Rijndael. In AES Round 1 Technical Evaluation, NIST 1998. 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. Gladman, B., 1999. Implementation experience with the AES candidate algorithms. Proc. 2nd AES candidate Conf., Mar. 22-23, Rome, pp: 7-14. <http://fp.gladman.plus.com/cryptography_technology/rijndael>
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. 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.
7. Lidl, R. and H. Niederreiter, 1986. Introduction to Finite Fields and their Applications. Cambridge University Press, (Reprinted 1988).
8. McEliece, R.J., 1987. Finite Fields for Computer Scientists and Engineers. Kluwer Academic Publishers.
9. Menezes, A.J., P.C. Van Oorschot and S.A. Vanstone, 1996. Handbook of Applied Cryptography. CRC Press.