Mathematical Problems of Computer Science 46, 26–36, 2016.

Review of White-box Implementations of AES Block Cipher and Known Attacks

Martun M. Karapetyan

Institute for Informatics and Automation Problems of NAS RA
e-mail: martun.karapetyan@gmail.com

Abstract

Conventional encryption algorithms are designed to be secure in the "black-box" context, i.e. the attacker has access to the input and output of the algorithm, but cannot observe the intermediate values generated during the software execution. Yet in some cases, the encryption algorithm runs in a hostile environment, where the attacker can see not only the input and output values but also has full access to all the internal values and can change the execution at will. White-box cryptography algorithms are designed to be executed in such untrusted environments and are said to operate in the white-box attack context. A white-box implementation of AES cipher was first presented by Chow, Eisen, Johnson and van Oorschot in 2002 [1], which was shown to be insecure against the BGE attack presented by Billet, Gilbert and Ech-Chatbi in 2004 [2]. In 2010, another white-box AES implementation was presented by Karroumi, which was supposed to withstand the BGE attack [3]. In 2013, De Mulder, Roelse, and Preneel showed, that Karroumis and Chows implementations are equivalent, i.e. the BGE attack can be successfully applied to both [4]. They also presented several optimizations, which reduce the work factor of the attack to 2^{22} work steps. In this paper we will review both AES implementations and the BGE attacks.

Keywords: Cryptography, White-box, AES, BGE attack, Review.

1. Introduction

In the "black-box" encryption model a cryptographic operation is executed in a trusted environment. The attacker, whose main goal is to extract the cryptographic key, observes the input and output of encryption/decryption operations, but has no access to the internal values generated by the algorithm. Conventional encryption algorithms were designed to be secure in this context. In some cases, cryptographic software runs on a device controlled by a hostile user. In this case the attacker sees any intermediate value generated by the execution of a cryptographic operation by observing the memory of the device and can change the execution routine at will. Examples of such software are Digital Rights Management (DRM) systems or any client software running on a cloud. White-box implementations of encryption algorithms are designed to run on these devices, and are said to operate in the white-box attack context. White-box algorithms are implemented as series of look-up from tables, which contain the cryptographic key in such a way to prevent its extraction. There were numerous attempts to design a white-box implementation of the Advanced Encryption Standard (AES), all of which were later broken. A white-box implementation of AES cipher was first presented by Chow, Eisen, Johnson and van Oorschot in 2002 [1]. An attack presented by Billet, Gilbert and Ech-Chatbi in 2004 (BGE attack) showed that the secure key can be extracted from Chows implementation in 2^{30} work steps [2]. In 2010, Karroumi presented a modified version of Chows algorithm based on dual ciphers, which was designed to withstand the BGE attack and increase the work factory of it to 2^{93} [3]. In 2013, De Mulder, Roelse , and Preneel proved that Karroumis and Chows implementations are identical and the BGE attack can be applied to Karroumis implementation with minor modifications [4]. Also several speed optimizations were presented, after which just 2^{22} work steps are required to extract the key. In this paper we will review both implementations of AES and the BGE attack applied on them. The paper is organized as follows: in the sections 2 black-box encryption of AES is briefly represented. Section 3 is devoted to the Chow's implementation of AES white-box encryption. Section 4 contains the BGE attack. Section 5 briefly describes Karroumi's implementation. Section 6 comments on the BGE attack for Karoumi's implementation. The paper ends with the conclusion.

2. AES Black-box Encryption

AES is a substitution-permutation network cipher for symmetric encryption also known as the Rijndael cipher [5]. It supports key lengths of 128, 192 or 256 bits and has 10, 12 or 14 rounds, respectively. AES-128 will be considered as the primary setting in the rest of this paper. Each round updates a 16-byte state and consists of four operations: SubBytes, ShiftRows, MixColumns and AddRoundKey, except the final round, where the MixColumns operation is omitted.

The 128-bit state is interpreted as a 4x4 matrix of 8-bit values. SubBytes operation substitutes each value of the matrix $a_{i,j}$ with $S(a_{i,j})$, where $S(a_{i,j})$ values are built using inversion in $GF(2^8)$ modulo an irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$.

ShiftRows transformation shifts 2^{nd}, 3^{rd} and 4^{th} rows of the table 1, 2 and 3 times to the left, respectively.

MixColumns is a transformation applied on the columns of the state matrix. Each column is multiplied with a fixed matrix $MC = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$.

AddRoundKey operation simply XORs the state with the next key, generated by the key schedule.

For purpose of creating a white-box implementation for AES, we can view the algorithm in a different manner. One can notice, that SubBytes and Shiftrows operations can be safely switched without any change in the output. Also because the ShiftRows is a linear transformation, AddRoundKey(K_i) followed by ShiftRows is identical to ShiftRows followed by AddRoundKey($\hat{K}_i$), where $\hat{K}_i$ is the result of applying ShiftRows operations on K_i. These allow us to change the AES structure to the one in Figure 2.

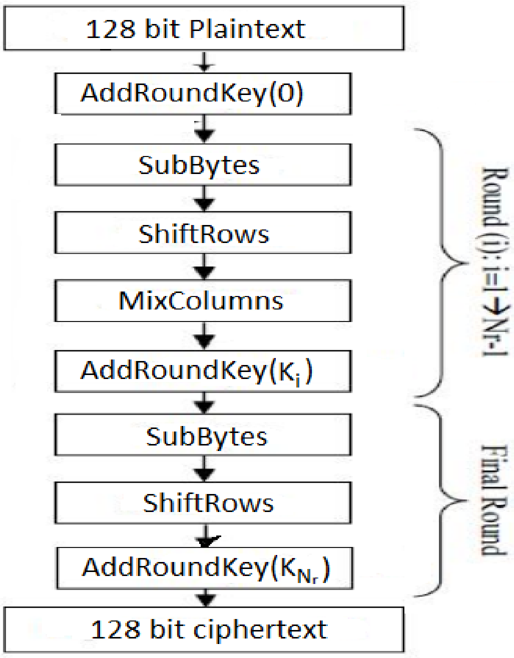

Fig. 1. Structure of AES black-box encryption.

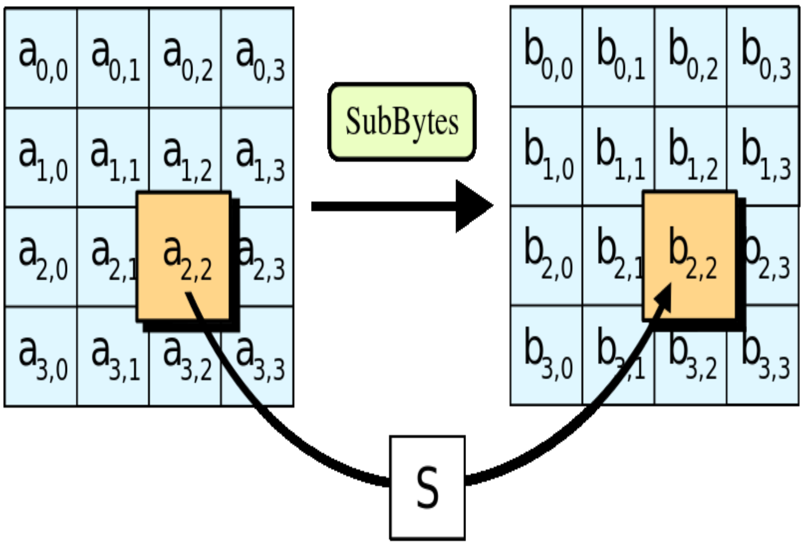

Fig. 2. AES subbytes operation.

3. Chows AES White-box Implementation

In 2002, Chow, Eisen, Johnson and van Oorschot proposed the first white-box implementation of AES-128 [1]. Instead of calculating the function E_K on the plaintext, another function $E_K = G \cdot E_K \cdot F^{-1}$ is computed, where G and F are input and output encodings,

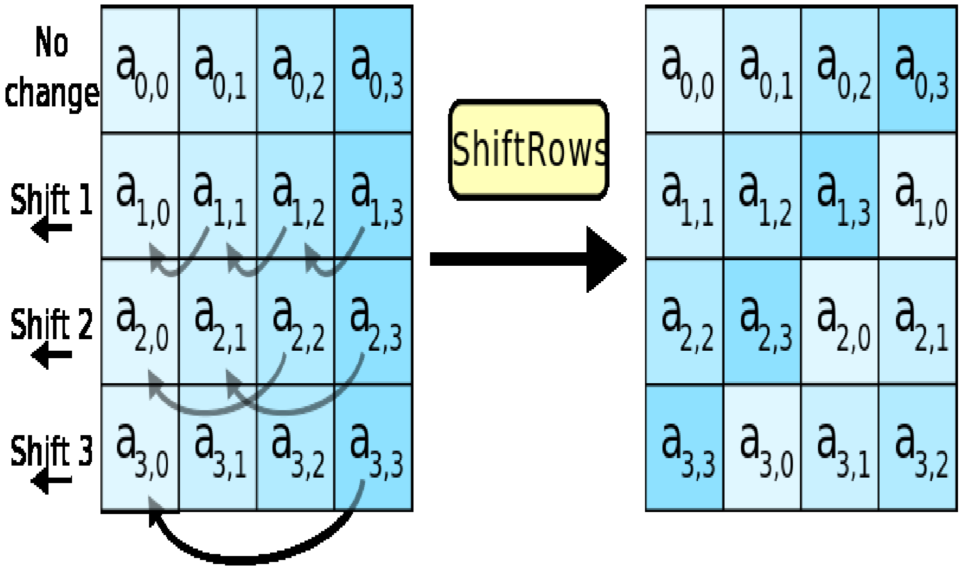

Fig. 3. AES ShiftRows operation.

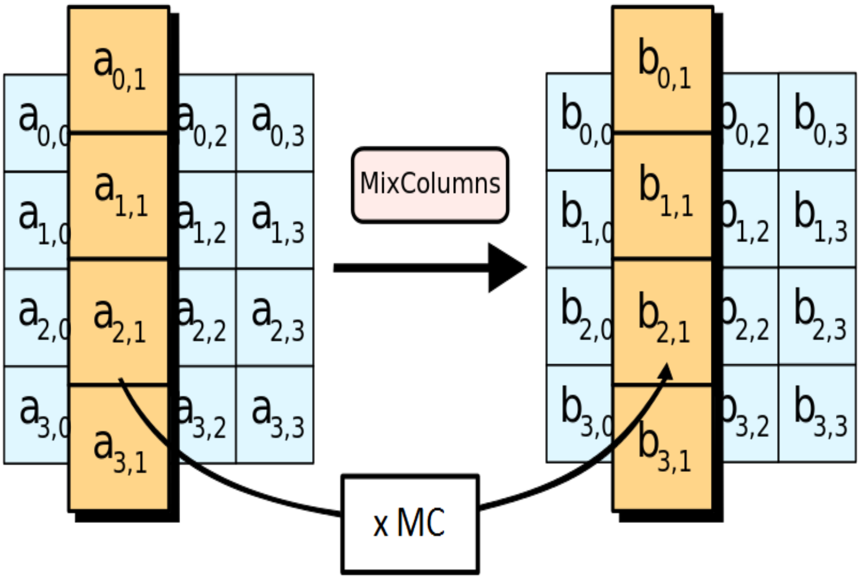

Fig. 4. AES MixColumns operation.

which are randomly generated independent of K. Since in some cases it's infeasible to have input and output encodings of the desired bit length, some encodings can be represented as a concatenation of smaller bijections. A bijection F of size $n = n_1 + n_2 + \quad + n_k$ can be built from a list of smaller bijections F_i, where F_i has size n_i, and for any n-bit vector $b = (b_1, b_2, ..., b_n)$ $F(b) = F_1(b_1, ..., b_{n_1})||F_2(b_{n_1+1}, ..., bn_1 + n_2)...F_k(b_{n_1+...+n_{k_1}+1}, ..., b_n)$. In this case we say $F = F_1||F_2||...||F_k$, and call F a concatenated encoding. The encryption algorithm is implemented as a sequence of look-ups from different look-up tables. The output

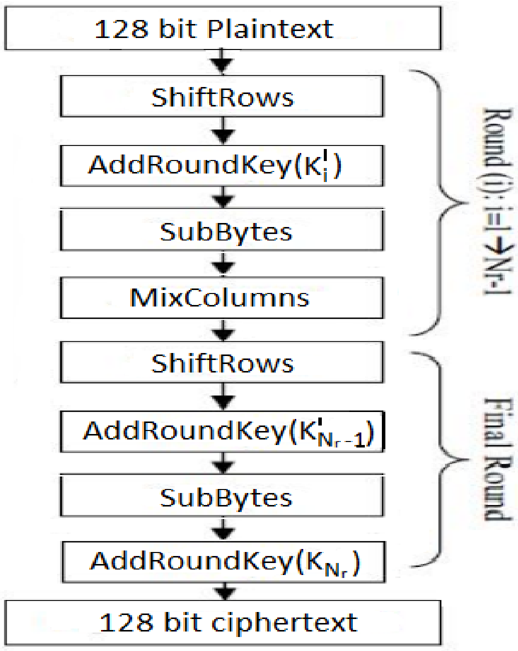

Fig. 5. AES algorithm with modified structure.

encoding of any table matches the input encoding of the table following it, so the encodings can cancel each other. After the encryption routine is over, the value of $E_K = G \cdot E_K \cdot F^{-1}$ is properly computed, as any intermediate encodings are cancelled out. We will present an unprotected implementation first, to describe the tables we'll need, and then describe the modified protected version of the tables.

3.1 Unprotected Implementation

For each round, AddRoundKey and SubBytes transformations can be made using 16 look-up tables that map 1 byte to 1 byte for each round. These look-up tables are called T-Boxes, and are defined as follows:

$$T_i^r(x) = S(x \oplus \widetilde{k}_{r-1}[i]), \quad for \ \ i = \overline{0..15}, \ \ and \ \ r = \overline{1..9}, \tag{1}$$

$$T_i^{10}(x) = S(x \oplus \widetilde{k}_9[i]) \oplus k_{10}[i], \quad for \ \ i = \overline{0..15}. \tag{2}$$

It's obvious, that T-boxes have no security and the attacker can easily extract the keys from the T-boxes, if those were provided. So additional encoding is applied on the T-boxes, before they can be used.

After the state vector passes through the T-boxes, MixColumns operations must be applied on it. MixColumns operation multiplies each 4 bytes of the state vector $[x_1, x_2, x_3, x_4]$ with the $MC = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$ matrix. This can be accomplished by the so called T_{y_i}

tables, where

$$T_{y_0}(x) = x \cdot [02\ 01\ 01\ 03]^T, \tag{3}$$

$$T_{y_1}(x) = x \cdot [03\ 02\ 01\ 01]^T, \tag{4}$$

$$T_{y_2}(x) = x \cdot [01\ 03\ 02\ 01]^T, \tag{5}$$

$$T_{y_3}(x) = x \cdot [01\ 01\ 03\ 02]^T. \tag{6}$$

So T_{y_i}-boxes are 1 byte to 4 byte tables. The outputs of each 4 consecutive T_{y_i} tables must be XOR-ed to get the output of the round, i.e. $T_{y_0}(x)+T_{y_1}(x)+T_{y_2}(x)+T_{y_3}(x)$. XOR tables are used for this purpose:

$$XOR(x, y) = x \oplus y. \tag{7}$$

XOR tables operate on 2 nibbles (4-bit values), so they are 1 byte to 4 bits mapping tables. The XOR of two 32 bit values can be computed using 8 copies of these XOR tables.

One can notice, that T-boxes and T_{y_i} boxes can be combined into a single table. These tables will look as follows:

$$T_{y_0}(T_i^r(x)) = S(x + k^{r-1}[i]) * [03\ 02\ 01\ 01]^T. \tag{8}$$

$$T_{y_1}(T_i^r(x)) = S(x + k^{r-1}[i]) * [03\ 02\ 01\ 01]^T. \tag{9}$$

$$T_{y_2}(T_i^r(x)) = S(x + k^{r-1}[i]) * [03\ 02\ 01\ 01]^T. \tag{10}$$

$$T_{y_3}(T_i^r(x)) = S(x + k^{r-1}[i]) * [03\ 02\ 01\ 01]^T. \tag{11}$$

Then the outputs of these tables must be passed through XOR boxes to get the round's output. So there will be 144 composed $T - box/T_{y_i}$ tables, 864 XOR tables and 16 T-boxes total for all the rounds of AES together. These boxes are not secure against key extraction. The next section will change these tables to apply some defense mechanisms against different attacks.

3.2 Protected Implementation

As there are just 256 possible values for $k^{r-1}[i]$, one can build a $T - box/T_{y_i}$ table for each of these values, and check if the table we provide matches any of those. This will allow an attacker to extract the key from $T - box/T_{y_i}$ tables. In order to prevent this, input and output encodings are applied to all the tables. The encodings are applied in a networked fashion, so all the encodings except the input encoding of the first round F and the output encoding of the last round G cancel out each other, and the function $E_K = G \cdot E_K \cdot F^{-1}$ is calculated by using these tables. Here G and F are concatenated encodings of 128 bits, made of 16 bijections of 8 bits each. These encodings dramatically increase the total number of possible tables. There are $(16!)^2$ possible input encodings and (16!)8 output encodings per table. It's obvious that for a fixed input key, the tables constructed for different output encodings are all different. This means that there are at least $(16!)^8$ possible tables, which makes it impossible to the attacker to enumerate. As the input/output encodings provide the confusion step for the tables, linear transformations are applied to produce the diffusion step. The table input/outputs are multiplied with randomly generated matrices over $GF(2)$ which are called mixing bijections. 16 8-bit to 8-bit mixing bijections are randomly generated and applied at the inputs of each round except the first. Lets denote mixing bijection for byte i in round r as Lir. Another 4 32-bit to 32-bit mixing bijections $MB_i^r, i = \overline{1..4}, r = \overline{1..9}$ are applied to all the rounds outputs except the last one. So after these changes, the results that we get after applying XOR tables will need to be multiplied with the inverse of the mixing bijection $(MB_i^r) - 1$ and $(L_i^{r+1}) - 1$, so the effect of MB is cancelled out, and the inverse of L_i^{r+1} is applied, so it can get cancelled in the next round. This is done with the same technique as the MixColumns step, and additional XOR tables are created for this.

The total size of the lookup tables of this implementation is 770,048 bytes, and there are 3104 lookups during each execution [1]. For a more detailed tutorial on Chows whitebox AES refer to [6].

4. BGE Attack

BGE attack, introduced by Billet, Gilbert and Ech-Chatbi in 2004 [2], targets not a single table from Chows implementation, but a group of tables, which form the AES round. As each table has good confusion and diffusion properties, it's hard to extract the key from a single table, but attacking a group of tables is more reasonable. The AES round is viewed as four 32-bit to 32-bit mappings Rjr, where the structure of R_j^r is shown in figure 6. P_i^r are a combination of input encodings and mixing bijections, Q_i^r are a combination of mixing bijections and output encodings. All the intermediate encodings between different tables have been cancelled out. The BGE attack consists of 3 phases:

1) The values of P_i^rs and Q_i^rs are recovered up to an unknown affine transformation. This allows us to change transformations P_i^rs and Q_i^rs with affine transformations $\widetilde{P}_i^r$ and $\widetilde{Q}_i^r$.
2) The values of P_i^r and Q_i^r are recovered completely.
3) Having P_i^r and Q_i^r, the AES-128 key is extracted.

Let's denote the inputs of R_0^r as x_0, x_1, x_2, x_3, and the outputs as y_0, y_1, y_2, y_3.

$$y_0 = Q_0(02 \cdot T_0^r \cdot P_0^r(x_0) \oplus 03 \cdot T_1^r \cdot P_1^r(x_1) \oplus 01 \cdot T_2^r \cdot P_2^r(x_2) \oplus 01 \cdot T_3^r \cdot P_3^r(x_3)), \tag{12}$$

$$y_1 = Q_1(01 \cdot T_0^r \cdot P_0^r(x_0) \oplus 02 \cdot T_1^r \cdot P_1^r(x_1) \oplus 03 \cdot T_2^r \cdot P_2^r(x_2) \oplus 01 \cdot T_3^r \cdot P_3^r(x_3)), \tag{13}$$

$$y_2 = Q_2(01 \cdot T_0^r \cdot P_0^r(x_0) \oplus 01 \cdot T_1^r \cdot P_1^r(x_1) \oplus 02 \cdot T_2^r \cdot P_2^r(x_2) \oplus 03 \cdot T_3^r \cdot P_3^r(x_3)), \tag{14}$$

$$y_3 = Q_3(03 \cdot T_0^r \cdot P_0^r(x_0) \oplus 01 \cdot T_1^r \cdot P_1^r(x_1) \oplus 01 \cdot T_2^r \cdot P_2^r(x_2) \oplus 02 \cdot T_3^r \cdot P_3^r(x_3)); \tag{15}$$

Now, having these tables, we must find a transformation $\widetilde{Q}_i^r$, such that $\widetilde{Q}_i^r = Q_i^r * A_i^r$, i.e. differs from Q_i^r by an unknown affine transformation A_i^r.

So y_0 is a function of 4 parameters x_0, x_1, x_2, x_3. Let's fix the values of x_1 and x_2 to c_1 and c_2, respectively, and give 2 different values to x_3, namely c_3 and c_3. We will get 2 functions:

$$y_0(x_0, c_1, c_2, c_3) = Q_0(02 \cdot T_r \cdot P_r(x_0)) \oplus B_{c_1,c_2,c_3}, \tag{16}$$

$$y_0(x_0, c_1, c_2, c_3) = Q0(02 * T_r * P_r(x_0)) \oplus B_{c_1,c_2,c_3}. \tag{17}$$

From these 2 equations we get:

$$y_0(x_0, a_1, c_2, c_3) \cdot y_0(x_0, a_2, c_2, c_3)^{-1} = Q_0(Q_0^{-1}(x) + B) where B = B_{c_1,c_2,c_3} \oplus B_{c_1,c_2,c_3}. \tag{18}$$

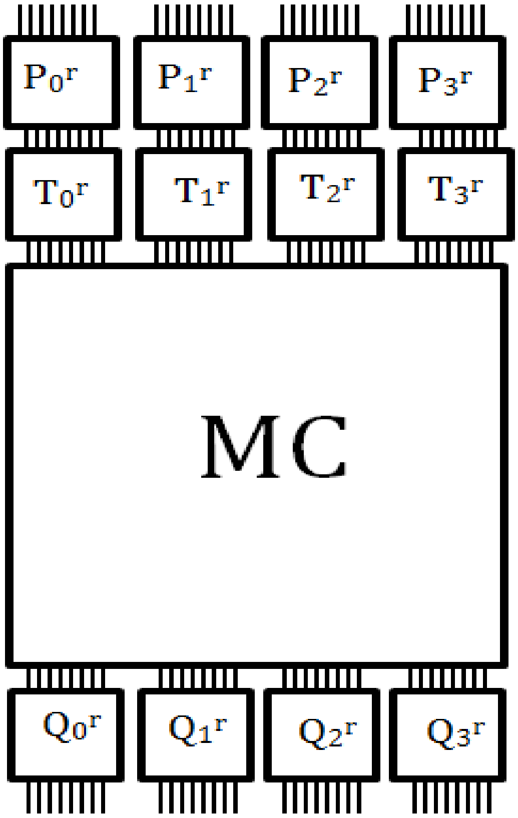

Fig. 6. R_0^r mapping, 1 of the 4 mappings that make the AES round.

If we vary the value of c_3 so it can take all the possible 256 values in $GF(2^8)$, the value of B will also take all the 256 different values. This gives us all the 256 functions $Q_0(Q_0^{-1}(x) + B)$ for all values of B. This set of bijections forms a commutative group. Bilet et al provide a technique to recover the value of Q_0 up to an unknown affine transformation, given these group of bijections. Once this is done, we can change all the tables to use $\widetilde{Q_i}$ instead of Q_i, which significantly weakens the confusion properties of the tables. This allows key extraction, which is described in detail in [2].

5. Karroumis AES White-box Implementation

In 2010 Mohamed Karroumi presented a modification of Chows AES white-box algorithm, which was supposed to withstand the BGE attack [3]. The algorithm is based on usage of AES dual ciphers. AES dual ciphers were first presented in [7] with a list of 240 ciphers. This list was further expanded to 61,200 ciphers that are dual to AES in [8]. For each of these dual ciphers, there exists an affine transformation Δ that maps AES plaintext P, ciphertext C and key K into plaintext P_{dual}, ciphertext C_{dual} and key K_{dual} of a dual AES, i.e. $P_{dual} = \Delta(P)$, $C_{dual} = \Delta(C)$ and $K_{dual} = \Delta(K)$. In Karroumis white-box implementation, for each of 10 rounds of AES a dual-AES is selected randomly. SubBytes constants, MixColumns matrix and the key of the corresponding dual-AES cipher are used for building the $T - box/T_{y_i}$

tables. In order to get the same output values for the same input values as Chow's AES algorithm, affine transformation Δr must be applied at the input of each round, and $\Delta r-1$ at the output of the round. The outputs of each round of this implementation will be different from Chows outputs as different SubBytes and MixColumns values are used, but the final round outputs will match.

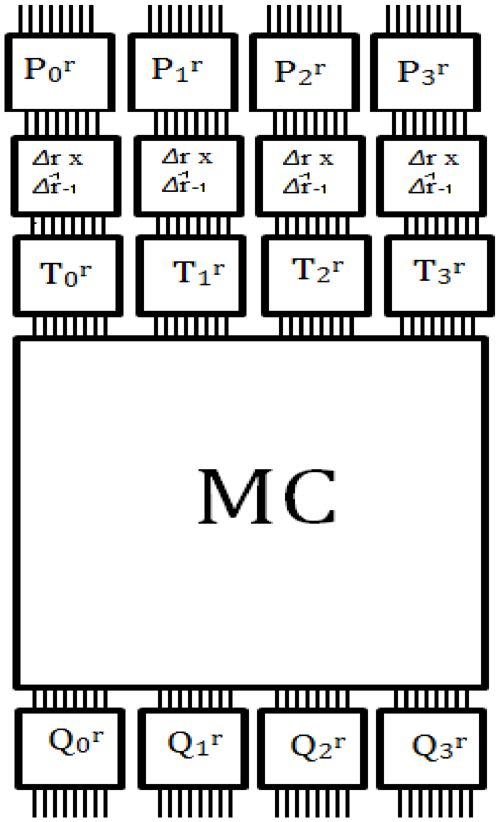

Fig. 7. AES white-box round structure of Karroumi's modification.

One can notice, that as each 4 output bytes of an AES round depend on only 4 input bytes, 4 different dual ciphers may be used in each round, so employing 40 different randomly chosen dual ciphers in total. These changes will not affect the general structure of the tables, so the speed and memory requirement will be identical to Chows implementation. Karroumi argues that the attacker will need to brute-force these randomly chosen dual ciphers, which will increase the security to 2^{91}. The detailed description of this implementation can be found in [3].

6. BGE attack on Karoumis Implementation

It is shown in [4] that an encoded dual AES subround can be represented as an encoded AES subround with the same key. This lets the attacker convert Karroumi's tables into Chow's tables, and apply the BGE attack the exact same way. A detailed explanation on how to do the conversion can be found in section 4.1 of [4].

7. Conclusion

In this paper we reviewed Chow's and Karroumi's white-box implementations of AES algorithm and briefly described the BGE attack which was successfully applied on both implementations. So far no known secure AES white-boxes were created, all the known implementations were broken with a work factor less than 2^{30}.

References

[1] S. Chow, P. Eisen, H. Johnson and P. C. van Oorschot, "White-box cryptography and an AES implementation", *In 9th Annual Workshop on Selected Areas in Cryptography (SAC 2002)*, Aug.15-16, pp. 1–18, 2002.

[2] O. Billet, H. Gilbert and C. Ech-Chatbi, "Cryptanalysis of a white-box AES implementation", *In Selected Areas in Cryptography(SAC)*, pp. 227-240, 2004.

[3] M. Karroumi, "Protecting white-box AES with dual ciphers", *In Kyung-Hyune Rhee and DaeHun Nyang, editors, Information Security and Cryptology - ICISC 2010, of Lecture Notes in Computer Science, Springer Berlin Heidelberg*, vol.6829, pp. 278–291, 2011.

[4] Y. De Mulder, P. Roelse and B. Preneel, "Revisiting the BGE attack on a white-box AES implementation", [Online]. Available: http://eprint.iacr.org/2013/450.pdf.

[5] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (aes)", fips Publication 197, 26 Nov. 2001.

[6] (2012)J. A. Muir , "A Tutorial on White-box AES", *Mathematics in Industry* [Online]. Available: http://www.ccsl.carleton.ca/ jamuir/papers/wb-aes-tutorial.pdf.

[7] (2002) E. Barkan and E. Biham, "The book of Rijndaels", Cryptology ePrint Archive, Report 2002/158, [Online]. Available: http://eprint.iacr.org/2002/158.

[8] A.Biryukov, C. De Canni‘ ere, A. Braeken, B. Preneel, "A toolbox for cryptanalysis: Linear and affine equivalence algorithms", *EUROCRYPT 2003. LNCS,* , vol. 2656, pp. 33–50, 2003.

Submitted 08.07.2016, accepted 04.11.2016.

Բաց կոդով (whitebox) ծածկագրման համակարգերի վերլուծություն և հայտնի հարձակման մեթոդների հետազոտություն

Մ. Կարապետյան

Ամփոփում

Ստանդարտ ծածկագրման ալգորիթմները նախատեսված են “սև տուփի” համատեքստում անվտանգ աշխատելու համար, այսինքն̀հարձակվողը կարող է ուսումնասիրել ալգորիթմի մուտքի և ելքի արժեքները, բայց չի կարող տեսնել ալգորիթմի աշխատանքի ընթացքում ծրագրի գեներացրած միջանկյալ արժեքները: Սակայն, որոշ դեպքերում, ծածկագրման ալգորիթմն աշխատում է օտար միջավայրում, որտեղ հարձակվողը տեսնում է ոչ միայն ալգորիթմի մուտքն ու ելքը, այլ նաև ցանկացած միջանկյալ արժեք` գեներացված ալգորիթմի կողմից և ցանկության դեպքում կարող է փոփոխել

ալգորիթմի աշխատանքը: Բաց կոդով ծածկագրման ալգորիթմները նախատեսված են աշխատելու այսպիսի օտար միջավայրերում: AES ծածկագրման ալգորիթմի բաց կոդով իրականացման սխեմա առաջինը առաջարկվել է Չոի, Այսենի, Ջոնսոնի և Վան Օրշոտի կողմից 2002թ.-ին, որի վրա հաջող հարձակման մեթոդ առաջարկվեց 2004թ.-ին, իսկ հարձակման մեթոդը կոչվեց “BGE հարձակում”: 2010-ին AES-ի մեկ այլ բաց կոդով իրականացում առաջարկվեց Կառումիի կողմից, սակայն 2013-ին Դե Մուլդերը, Ռոլսը և Պրենիլը ցույց տվեցին, որ Կառումիի և Չոի իրականացումները համարժեք են, այսինքն՝ “BGE հարձակումը” հաջողությամբ աշխատում է նաև այդ սխեմայի վրա: Այս հոդվածում մենք կվերլուծենք AES-ի առաջարկած բաց կոդով ծածկագրման սխեմաները և դրանց նկատմամբ հաջողությամբ կիրառված հարձակման մեթոդները:

Криптографические алгоритмы работающие по принципу белого ящика и известные атаки

М. Карапетян

Аннотация

Обычные симметричные криптографические алгоритмы расчитаны на безопасность в так называемой среде “черного ящика", т.е. атакующий имеет доступ к входным и выходным данным алгоритма, но не может видеть промежуточные значения, генерируемые при исполнении. Иногда криптографические программы работают в незащищенной среде, где атакующий имеет доступ не только к входным и выходным данным алгоритма, но также к любому промежуточному значению генерируемому алгоритмом. Атакующий также может изменить промежуточные значения или сам алгоритм по собственному желанию. Алгоритмы, работающие по принципу белого ящика, расчитаны для безопасной работы в такой среде.

Первое исполнение алгоритма AES, работающее по принципу белого ящика, было создано Чо, Енсеном, Джонсоном и Ван Оршотом в 2002 году, который был удачно атакован методом атаки “BGE", предложенном в 2004 году. В 2010 году другой метод реализации алгоритма AES по принципу белого ящика был предложен Каруми, но в 2013 году Де Мюлдер, Ролсе и Пренил показали, что методы Каруми и Чо идентичны, т.е. атака "BGE" может быть успешно применена к обоим методам. В данной статье представлены методы исполнения алгоритма AES по принципу белого ящика и известные атаки на них.