Workshop on Cryptology and Information Security

A Variant of AES against Square Attack

Wei-Chuan Wang
Department of Computer Science and Information Engineering
National Chao Tung University
1001 Ta Hsueh Road
Hsin Chu, Taiwan 30050
email: eric.csie89g@nctu.edu.tw

Jen-Chun Chang
Department of Information Management
Ming-Hsin Institute of Technology
1 Hsin Hsing Road, Hsin Fong
Hsin Chu, Taiwan 30440
email: jcchang@csie.nctu.edu.tw
Tel: 0935-193912, (03)5593142-3440
Fax: (03)5595243

Rong-Jaye Chen
Department of Computer Science and Information Engineering
National Chao Tung University
1001 Ta Hsueh Road
Hsin Chu, Taiwan 30050
email: rjchen@csie.nctu.edu.tw

Contact Author: Jen-Chun Chang

Keywords: block cipher, DES, AES, square attack, differential attack, nonlinearity

A Variant of AES Against Square Attack

Jen-chun Chang[1], Rong-jaye Chen[2] and Wei-chuan Wang[2]

Department of Information Management[1]
Ming-Hsin Institute of Technology
Hsin-Fong, Hsin-Chu, Taiwan 30440
Email: jcchang@csie.nctu.edu.tw

Scientific Computing Laboratory[2]
Department of Computer Science and Information Engineering
National Chiao-Tung University
Hsin-Chu, Taiwan 30050
E-mail: rjchen@csie.nctu.edu.tw
eric.csie89g@nctu.edu.tw

July 11, 2002

Abstract

Since the "Square" attack was introduced by Joan Daemen, Lars R. Knudsen and Vincent Rijmen in 1997, its variants has been reported to go against 9 rounds of AES. We analyse the features of some cryptanalysis and try to find out what properties are used to attack block ciphers.

We propose MAES, a modified version of AES, in order to have higher security level in prevention from the "Square" attack. Furthermore, we also compare the security levels between AES and MAES. Finally we prove that MAES provides higher security than AES.

1 Introduction

The "Square" attack, a dedicated attack on Square, was introduced in the paper presenting the Square cipher itself[DKR97]. It is also valid for AES[DR98] since AES inherits many properties from the Square cipher. This attack takes advantage of the byte-oriented structure of AES and the properties of the outputs of the 3rd round of AES. It is independent of the specific choices of *ByteSub*, the multiplication polynomial of *MixColumn*, and the key schedule in AES.

For 128-bit keys, the “Square” attack associated with the “partial sums” technique[FKL$^+$00] is faster than exhaustive key search for AES variants by up to 7 rounds. For 192-bit keys, it is faster than exhaustive key search by up to 8 rounds. For 256-bit keys the “Square” attack associated with the related key attack is faster than exhaustive key search by up to 9 rounds.

Since the “Square” attack is based on some properties of the Λ-set and the structure of AES, we want to reduce these properties by slightly modifying the specification of AES.

2 Preliminaries

2.1 Λ-set

To describe the “Square” attack we need the notion of a Λ^1-set, a set of 256 states that are all different in the same bytes.

Definition 1: *(state)*
The ***state*** *means the intermediate cipher result.*

Definition 2: *(Λ-set)*
A Λ-set is a set of 2^8 states that are all different in some of the state bytes (the active bytes) and all equal in the other state bytes (the passive bytes). For two distinct states x and y in a Λ-set we always have

$$\forall x,y \in \Lambda : \begin{cases} x_{i,j} \neq y_{i,j} & \textit{if the byte at position (i,j) is active, and} \\ x_{i,j} = y_{i,j} & \textit{otherwise} \end{cases}$$

Moreover, a “Λ^k-set” is a Λ-set with exactly k active bytes.

The properties of applying the 4 transformations of AES on the elements of a Λ-set are described in [DR98]. By the structure of AES, consider a Λ^1-set as inputs. The evolution of the positions of the active bytes forms the Λ^{16}-set through 2 rounds. Then these would lead to that *XORing* with all relevant bytes of the output of *MixColumn* in the 3rd round equals zero. And the subsequent *AddRoundKey* does not affect this property. Therefore, all bytes at the output of the 3th round are still balanced. Nevertheless this does not mean the bytes range over all possible values and the balance is destroyed by the subsequent application of *ByteSub*.

2.2 Differential Attack

Differential attack introduced by Biham and Shamir[BS91] is one of the most significant advances in cryptanalysis. It is known as a chosen-plaintext attack or a statistical attack against block ciphers. The main idea is to compare two separate encryptions which use the same key and to look at the *XOR* of the S-box inputs and outputs. This step is independent of the key being used. Then it would be known as various input *XOR* - output *XOR* pairs, called characteristics, occurring with particular probabilities.

Definition 3: *(characteristic)*
A particular input XOR value and output XOR value pair which occurs with some probability is called a characteristic.

Definition 4: *(differential distribution table)*
A table that shows the distribution of the input XOR and output XOR of all the possible pairs of an S-box is called the ***differential distribution table*** *of the S-box. In this table each row corresponds to a particular input XOR, each column corresponds to a particular output XOR and the entries themselves count the number of possible pairs with such an input XOR and output XOR.*

Definition 5: *(δ_f)*
*Let **f** be a substitution function from $GF(2^n)$ into $GF(2^n)$. Let **a** be the input XOR and **b** be the output XOR. Then each entry $\delta_f(a,b)$ is defined by:*

$$\delta_f(a,b) = \#\{x \in GF(2^n) : f(x+a) \oplus f(x) = b\}$$

Let δ_f denote the maximal value of the entries of the differential distribution table. We have

$$\delta_f = \max_{a \neq 0} \max_{b} \delta_f(a,b) \tag{1}$$

We usually use the equation (1) to measure the resistance to the differential attack. The more δ_f increases, the less resistance to the differential attack it is.

2.3 Nonlinearity

Although the S-boxes were designed to be non-linear, it turns out that some of the inputs/outputs can be approximated by linear functions. Using this feature to attack the cipher can reveal some key bits and the remaining key bits can be gained by exhaustive search.

Definition 6: *(Nonlinearity $\mathcal{N}_f$)*
*Let **f** be a substitution function from $GF(2^n)$ into $GF(2^n)$. Let **a** and **b** be two binary vectors in $\{0,1\}^n$. We have*

$$\begin{aligned}
\lambda_f(a,b) &= |\#\{x \in GF(2^n) : a \cdot x + b \cdot f(x) = 0\} - 2^{n-1}| \\
\lambda_f &= \max_{a} \max_{b \neq 0} \lambda_f(a,b) \qquad (2) \\
Nonlinearity\mathcal{N}_f &= 2^{n-1} - \lambda_f \qquad (3)
\end{aligned}$$

We usually use the equation (3) to measure the nonlinearity of a S-box. The more λ_f is, the less the nonlinearity is.

3 Modification

The “Square” attack exploits the byte-oriented structure of AES. Moreover, 4 bytes output depend on 4 bytes input of an intermediate round of AES and 1 byte output depend on

1 byte input of the final round of AES. We wish to make this relationship more complex but keep the feature of uniform diffusion. Then we need to think about the influences on changes of any components of AES.

The *ByteSub* transformation operates on each state byte independently. It does not change the position of each byte, and has no effect on diffusion. If we replace it with another format of S-boxes, other transformations are not affected. The round function still works normally.

The *ShiftRow* transformation cyclically shifts the rows of the state over different offsets. Each byte of the state is unchanged and its position is simply shifted over a offset. If we replace it with other offsets, either it has no influence on the round function or the round function is not able to spread uniformly.

The *MixColumn* transformation replaces each column with another column by multiplying a polynomial $c(x)$ over $GF(2^8)$. It has high intra-column diffusion. By composing the *ShiftRow* transformation with the *MixColumn* transformation, it has high diffusion over multiple rounds. If we replace the *MixColumn* transformation with another transformation, we also need to replace the *ShiftRow* transformation. Otherwise, the round function is not able to spread uniformly.

The *RoundKeyAddition* transformation applies the round key to the state by a simple bitwise XOR. If we replace it with another format of the round key addition, we may also need to redesign the key schedule.

We think that modifying the *ByteSub* transformation would cost minimally. Therefore, we design another transformation instead of the *ByteSub* transformation. The *TwoByteSub* transformation is a new component designed by us and the detail is described below. Unlike *ByteSub*, it treats every two bytes in a state as the basic unit for the operation and divides a state into 8 units.

Definition 7: *(unit)*
Each unit is a two-byte variable of a state. The unit $u_{i,j}$ denotes the bytes in position (i,2j) and (i,2j+1) of a state.

Each unit operates with the *TwoByteSub* transformation independently. The detailed design criteria for the S-box used in the *TwoByteSub* transformation are listed below:

1. Invertibility
2. Mapping $T : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$
3. High nonlinearity
4. Minimization of the greatest non-trivial value in the differential distribution table

Although the size of S-box grows 512 times bigger than the original one, it just needs 128 kilo bytes. We think this is acceptable in modern computer technologies.

Similar to the *ByteSub* transformation, we construct a S-box of the *TwoByteSub* transformation from the method mapping $p \Rightarrow p^{-1}$ in $GF(2^{16})$ mentioned in [Nyb94]. Since $GF(2^{16})$ is equivalent to $F_2[x]/M(x)$, where $M(x)$ is a irreducible polynomial with degree 16. We can treat every two bytes as a binary polynomial and compute its inverse in the finite field. The related properties of the inverse mapping in $GF(2^n)$ are described below:

1. Nonlinearity $\mathcal{N}_f \geq 2^{n-1} - 2^{\frac{n}{2}}$

2. The S-box is differentially 2-uniform if n is odd and it it differentially 4-uniform if n is even.
3. x^{-1} is computed by the Euclidean algorithm in polynomial time with respect to n.

By the properties listed above the *TwoByteSub* transformation is guaranteed to be low λ_f (see Equation 2) and low δ_f (see Equation 1). Furthermore, the nonlinearity $\mathscr{N}_f$ is at least 32512 and the δ_f is 4. In order to do multiplication of polynomials modulo an primitive binary polynomial of degree 16, we choose a primitive polynomial $M(x)$ from [BMS87]. The primitive polynomial $M(x)$ is given by:

$$M(x) = x^{16} + x^5 + x^3 + x^2 + 1$$

or '$1002D'$ in hexadecimal representation. Moreover, we apply an affine transformation to the result of the multiplicative inverse for complicating the algebraic expression in $GF(2^{16})$. In fact, this affine transformation does not affect the properties claimed above. We have written a program to test it and got the same properties. The goal of it is just to avoid some weak mapping.

Definition 8: *(TwoByteSub)*
The TwoByteSub transformation $T(p)$ is defined by:

$$\forall p \in GF(2^{16}), \qquad T(p) = A \cdot p^{-1} + B \quad mod \;\; M(x)$$
$$where \begin{cases} A & = x^{14} + x^{11} + x^{10} + x^8 + x^6 + x^2 + x \\ B & = x^{14} + x^{11} + x^7 + x^6 + x^5 + x^2 + 1 \\ M(x) & = x^{16} + x^5 + x^3 + x^2 + 1 \end{cases}$$

These two polynomials A and B are chosen in such a way that the S-box has neither fixed points (S-box(a) = a) nor opposite fixed points (S-box(a)=$\overline{a}$).

4 Security Analysis

As we concentrates the modification on the property in resistance to the "Square" attack, we will show that the necessary property needed by the "Square" attack is reduced by our modification.

Since our modification just replaces the *ByteSub* transformation with the other transformation and never attends to change the structure of AES, the advantages of the security level is not affected by it. In fact the original 8×8 S-box used in the *ByteSub* transformation has good properties in resistance to linear attacks and differential attacks. MAES still has at least the same degree of security as long as we choose the 16×16 S-box carefully.

4.1 Square Attack

After replacing the *ByteSub* transformation with the *TwoByteSub* transformation, we now trace the procedure of attacking MAES by the "Square" attack. Since the *TwoByteSub* transformation treats every two close bytes as a unit, we need to choose the Ω-set instead of the Λ-set. If we still use a Λ^1-set as the inputs of the *TwoByteSub* transformation, it does not lead to a Λ^1-set as the outputs. That is to say we need 2^{16} plaintexts as a set for our analysis.

Definition 9: *(Ω-set)*
An Ω-set is a set of 2^{16} states that are all different in some of the state units (the active units) and all equal in the other state units (the passive units). For two distinct states x and y in an Ω-set we always have

$$\forall x, y \in \Omega : \begin{cases} x_{i,j} \neq y_{i,j} & \text{if the unit at position (i,j) is active, and} \\ x_{i,j} = y_{i,j} & \text{otherwise} \end{cases}$$

Moreover, an "Ω^k-set" is an Ω-set with exactly k active units.

The properties of applying the 4 transformations of MAES on the elements of an Ω-set are described below. Notice that one active unit can denote two active bytes repeated 2^8 times.

Lemma 1: *Applying the TwoByteSub transformation on an Ω-set results in an Ω-set with the positions of the active units unchanged.*

Proof: Since *TwoByteSub* is a one-to-one and onto mapping and operates on each unit independently, applying it on an Ω-set properly results in an Ω-set with the positions of the active units unchanged. □

Lemma 2: *Applying the $ShiftRow$ transformation on an Ω^1-set results in two active bytes repeated 2^8 times where the active bytes are transposed by $ShiftRow$.*

Proof: Since $ShiftRow$ just cyclically shifts each row independently and an Ω^1-set can be viewed as two active bytes repeated 2^8 times, applying it on an Ω^1-set results in two active bytes repeated 2^8 times. □

Lemma 3: *Applying the MixColumn transforation on an Ω^1-set results in an Ω^4-set. The 8 active bytes are in two adjacent columns and each active byte is repeated 2^8 times.*

Proof: Since *MixColumn* can only operate with the bytes in the same column and two bytes in a unit must be in the same row, applying it on a Λ^1-set repeated 2^8 times results in a Λ^4-set repeated 2^8 times and the 4 active bytes are in the same column. Therefore, applying *MixColumn* on an Ω^1-set results in an Ω^4-set and the 8 active bytes are in two adjacent columns. □

Lemma 4: *Applying the AddRoundKey transformation on an Ω-set results in an Ω-set with the positions of the active units unchanged.*

Proof: Since *AddRoundKey* just applies the round key to the state by a simple bitwise XOR, applying it on an Ω-set can be viewed as permuting each active byte independently. □

Lemma 5: *Applying the MixColumn transforation on a set with 2 active bytes repeated 2^8 times in the same column results in a set with 4 active bytes repeated 2^8 times in it.*

Proof: Two active bytes can be viewed as a two-layered loop. Since *MixColumn* is a one-to-one and onto function, each output byte is affected by changing these two active bytes and is still an active byte repeated 2^8 times. □

By the structure of MAES, consider a Ω^1-set. We now trace the evolution of the positions of the active bytes through 2 rounds.

1. *MixColumn* of the 1st round converts the active bytes to two complete columns of active bytes and forms the Ω^4-set.
2. *ShiftRow* of the 2nd round spread the eight active bytes of two columns over four columns. Each column consists of 2 active bytes and 2 passive bytes.
3. Then each byte of the output of the *MixColumn* transformation of the 2nd round is affected by 2 active bytes and it is still an active byte repeated 2^8 times.
4. Then this stays as a Λ^{16}-set repeated 2^8 times until the input of *MixColumn* of the 3rd round.

Then these would still lead to that *XORing* with all relevant bytes of the output of *MixColumn* in the 3rd round equals zero. And the subsequent *AddRoundKey* does not affect this property. Therefore, all bytes at the output of the 3th round are still balanced. Nevertheless this does not mean the bytes range over all possible values and the balance is destroyed by the subsequent application of *TwoByteSub*.

When the 4th round is the last round, it does not contain the *MixColumn* transformation. Every input byte of the 4th round affects two output bytes of the 4th round. Let s^4 be the output of the 4th round, s^3 be the output of the 3rd round and $SubKey^4$ be the round key of the 4th round. We have:

$$s^3_{i',j'} = Sbox^{-1}\left(s^4_{i,j} \oplus SubKey^4_{i,j}, \quad s^4_{k,m} \oplus SubKey^4_{k,m}\right)$$

By assuming two key values for $SubKey^4_{i,j}$ and $SubKey^4_{k,m}$, we can compute the value of $s^3_{i',j'}$ for all elements of the Λ^{16}-set from the ciphertexts. If the values of $s^3_{i',j'}$ are not balanced over Λ^{16}-set, the assumed value for the key must be wrong. Otherwise, it is not always true. This is expected to reject all but approximately 1 key value, and can be repeated for the other bytes of the round key $SubKey^4$. This attack needs 2^{16} plaintexts and the complexity is $O(2^{16})$.

Square attack	AES		MAES	
	# Plaintexts	# Operations	# Plaintexts	# Operations
4 Rounds	2^8	2^8	2^{16}	2^{16}
5 Rounds (extended at the end)	2^8	2^{40}	2^{16}	2^{80}
5 Rounds (extended at the beginning)	2^{32}	2^{40}	2^{96}	2^{112}
6 Rounds	2^{32}	2^{72}	2^{96}	2^{176}

Table 1: The Comparison of complexity of the “Square” attack

Table 1 presents the comparison of complexity of the “Square” attack against AES and MAES. Although the “Square” attack can still be used to attack MAES, its complexity grows fast. Moreover, the complexity of the 6 round atack of MAES with 128-bit keys is higher than the exhaustive key search. We claim that MAES provides higher security level than AES in resistance to the “Square” attack.

4.2 Differential Attack

Since the differential attack is based on the differential distribution table of a S-box and the invertible S-box of MAES is constructed by the method mapping $p \Rightarrow p^{-1}$ in $GF(2^{16})$ mentioned in [Nyb94]. We now build the differential distribution table of the S-box of MAES and analyse it.

There are some regular rules in this differential distribution table. Each row (except the row 0) of it consists of 32769 0’s, 32766 2’s, and one 4. As this S-box is invertible, the distribution of the columns is similar to the rows. The row 0 means the input *XOR* equals 0, and the same inputs always lead to the same outputs. Therefore, the δ_f is 4 and one of them appears in (input *XOR*, output *XOR*) = (1, 19782) with probability $\frac{4}{65536}$. In this situation it is difficult to attack MAES by the differential attack.

Moreover, we show the comparison between DES, AES, and MAES about the resistance to the differential attack in table (2). Since the maximal probability of characteristics

Block ciphers	S-box size	The differential distribution table size	δ_f	Max. characteristic probability
DES	6×4	$2^6 \times 2^4$	14	$\frac{14}{64}$
AES	8×8	$2^8 \times 2^8$	4	$\frac{4}{256}$
MAES	16×16	$2^{16} \times 2^{16}$	4	$\frac{4}{65536}$

Table 2: The Comparison about the resistance to the differential attack

of MAES is $\frac{4}{65536}$ far small than that of AES, we claim that MAES provides higher security than AES in resistance to the differential attack.

4.3 Nonlinearity

If we want to analyse AES in resistance to linear attacks, we need to do $2^{128} \times 2^{128} \times 2^{128}$ comparisons in order to get the effective linear expressions. It would take too much time to analyse. The only nonlinear component in AES is the *ByteSub* transformation, and furthermore all other components are linear. Since our modification focuses on the S-box of AES, we would do the analysis of the nonlinearity of the S-box of MAES. Then we compare it with the nonlinearity of the S-box of AES in order to show the security level of MAES.

By the equation (3) we can measure the nonlinearity and compute λ_f. Accordingly we need to compute $2^{16} \times 2^{16}$ $\lambda_f(a,b)$ to get λ_f.

Finally we get λ_f of the S-box of MAES equal to 256. One of these approximate affine functions is as follows:

$$\forall p \in GF(2^{16}) \text{ and s = } T\text{(p)}$$
$$p_0 \oplus s_{12} \oplus s_6 \oplus s_5 \oplus s_1 \oplus s_0 = 0$$
$$where \begin{cases} p_i \text{ is a bit of input } p \text{ in position } i \\ s_j \text{ is a bit of output } s \text{ in position } j \end{cases}$$

Although λ_f of MAES is greater than that of AES, the nonlinearity grows with the size of the S-boxes. Consequently the probability of the successes of this equation is $\frac{256}{65536}$. Table (3) shows the comparison of the nonlinearity of S-boxes. Therefore, the nonlinearity of the

Block ciphers	S-box size	λ_f	$\mathcal{N}_f$	Probability of successful attack
AES	8×8	16	112	6.25 %
MAES	16×16	256	32512	0.39 %

Table 3: The Comparison of the nonlinearity $\mathcal{N}_f$ of S-boxes

S-box of MAES is higher than that of AES.

5 Conclusion

We analyse the structure of AES and the procedures of several cryptanalysis in order to find out their relations. Since AES is designed to resist any attacks which have been reported, we have to guarantee the same security level after modifying AES. However, the “Square” attack is successful on the reduced-round variants of AES. We want to conquer it.

We introduce a modified version of AES in order to have the higher security level in opposition to the “Square” attack. Since the “Square” attack is based on some properties of the Λ-set and the structure of AES, we reduce these properties by slightly modifying the specification of AES. By replacing the *ByteSub* transformation with the *TwoByteSub* transformation we achieve the goal. By reserving the other structures of AES it still has most of the advantages of AES. Furthermore, we compare AES and MAES in terms of resistance to the “Square” attack, the differential attack and the linear attack. We claim that MAES provides higher security than AES.

References

[BMS87] Paul H. Bardell, William H. McAnney, and Jacob Savir. ***Built-in Test for VLSI: Pseudorandom Techniques***. John Wiley & Sons, October 1987. ISBN: 0-471-62463-2.

[BS91] Eli Biham and Adi Shamir. **Differential cryptanalysis of DES-like cryptosystems**. *Journal of Cryptology*, 4(1):3–72, 1991.

[DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. **The Block Cipher Square**. *Fast Software Encryption*, pages 149–165, 1997.

[DR98] Joan Daemen and Vincent Rijmen. **AES Proposal: Rijndael**. *The First Advanced Encryption Standard Candidate Conference, N.I.S.T.*, 1998. http://csrc.nist.gov/encryption/aes/rijndael/.

[FKL$^+$00] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting. **Improved Cryptanalysis of Rijndael**. *Seventh Fast Software Encryption Workshop, Springer-Verlag, 2000,* 2000.

[Nyb94] Kaisa Nyberg. **Differentially uniform mappings for cryptography**. *Advances in Cryptology-EUROCRYPT '93, Lecture Notes in Comput. Sci.,* 765:55–64, 1994.