

# Cryptanalysis of Reduced Variants of Rijndael

Eli Biham\*      Nathan Keller†

## Abstract

Rijndael was submitted to the AES selection process, and was later selected as one of the five finalists from which one will be chosen in the next summer. The best known attack against Rijndael is still the one presented by the designers. In this paper we describe several attacks against reduced variants of Rijndael which improve the best known attack by a factor of 2, and present considerably larger factors for shorter variants. We also show that if only the key scheduling of Rijndael was reversed, the complexity of the best known attack would be reduced by an additional factor of  $2^8$ .

## 1 Introduction

The cipher Rijndael is one of the five finalists of the Advanced Encryption Standard selection process. The only described attack on this cipher is the Square attack which was already described by the designers and which is applicable to Rijndael reduced to 4, 5, and 6 rounds.

Rijndael is a block cipher. The length of the block and the length of the key can be specified to be 128, 192 or 256 bits, independently of each other. In this paper we discuss the variant with 128-bit blocks and 128-bit keys. In this variant, the cipher consists of 10 rounds. A block of data is represented

---

\*Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel. [biham@cs.technion.ac.il](mailto:biham@cs.technion.ac.il), <http://www.cs.technion.ac.il/~biham/>.

†Department of Mathematics, Technion – Israel Institute of Technology, Haifa 32000, Israel. [nkeller@tx.technion.ac.il](mailto:nkeller@tx.technion.ac.il).

in a  $4 \times 4$  matrix of bytes. Every round except for the last consists of 4 transformations:

1. ByteSubstitution - a single non-linear transformation is applied to each byte of the data.
2. Shift - reorders the bytes of each row cyclicly.
3. MixColumn - a linear transformation applied to columns of the matrix.
4. AddRoundKey - mixes a round key to the intermediate data.

Before the first round AddRoundKey is performed using the key as the round key. In the last round the MixColumn is omitted.

In this paper we present an improvement of all the variants of the Square attack against Rijndael (and against the Square cipher [5]) which reduces the complexity by a factor of 2. In addition, we analyze the 5-round variant of Rijndael using impossible differentials [2, 3]. This attack is faster than the fastest previously known attack against this variant (but requires some precomputation). We also show the importance of the design of the key schedule: If the key schedule would be reversed (either the order of the round keys, or the order of the bytes in each round key), then we can reduce the complexity of the attacks against the 5- and 6-round (modified) variants by another factor of  $2^8$ .

The attacks are based on the following property of MixColumn transformation: if two inputs of this transformation differ by one byte then the corresponding outputs differ by all the four bytes. We can use it to find a property of Rijndael reduced to 4 rounds: if two plaintexts differ by one byte then before the first MixColumn the data differ by only one byte. After MixColumn the data differs in a full column and then before the second MixColumn the data differs in one byte in each column. Therefore, after the second MixColumn the data differs in all the 16 bytes. It leads to the following interesting properties:

1. Consider a set of 256 plaintexts which are equal in all bytes except for one and in this one assume all the possible values (this byte is then called “delta-set” and the others are called “passive”). Then because of the property the inputs of the third MixColumn assume all 256 possible values in each byte. So the XOR of them in each byte is 0. MixColumn

Attack	Number of Rounds	Source	Chosen Plaintexts	Time Complexity
Square	4	[4]	$2^9$	$2^9$
Square	5	[4]	$2^{11}$	$2^{40}$
Square	6	[4]	$2^{32}$	$2^{72}$
Improved Square	4	This paper	$2^9$	$2^8$
Improved Square	5	This paper	$2^{11}$	$2^{39}$
Improved Square	6	This paper	$2^{32}$	$2^{71}$
Impossible differential	5	This paper	$2^{29.5}$	$2^{31}$
Reversed key schedule	5	This paper	$2^{11}$	$2^{31}$
Reversed key schedule	6	This paper	$2^{32}$	$2^{63}$

Table 1: Complexities of attacks on reduced variants of the cipher

is a linear transformation so this property holds after the MixColumn too. This is the property on which Square attack is based.

2. An impossible differential property: consider two plaintexts which differ by only one byte. If the corresponding ciphertexts of the 4-round variant are equal in bytes number 1, 6, 11, and 16 then before the Shift of round 4 the corresponding data are equal in bytes number 1, 5, 9, and 13 (i.e., in the first column). Therefore, after the MixColumn of the third round the data is equal in the first column. As a result, before the MixColumn of the third round the data is equal in the first column, but according to the discussion above all the bytes in that stage should differ. Therefore, such plaintext pairs cannot be encrypted to such ciphertext pairs. We base our 5-round impossible differential attack on this property.

The complexities of the attacks (the known and those we present here) are summarized in Table 1.

The paper is organized as follows: The description of Rijndael is given in Section 2. The Square attack [5] and our improvement of it are described in Section 3 along with the attack on the variant with the reversed round keys. Finally, in Section 4 we describe the attack against a 5-round variant of Rijndael using impossible differentials.

## 2 Description of Rijndael

The AES requirements state that the length of the key can be specified to be 128, 192 or 256 bits while the length of the block is 128 bits. The designers of Rijndael cipher also allow the length of the block to be 128, 192 or 256 bits, independently of the length of the key. In this paper we deal with the variant in which the lengths of both are 128 bits. In this case the cipher consists of 10 rounds (the number of rounds reaches 14 when the length of the key or the length of the block is 256 bits). The intermediate state is arranged in a  $4 \times 4$  matrix of bytes. Every round except for the last consists of the following transformations:

1. ByteSubstitution: This transformation is applied to each byte separately. Each byte is considered as representing coefficients of a polynomial of degree less than 8 over  $Z_2$  (i.e., elements of  $GF(2^8)$ ). Then we calculate the inverse of this polynomial modulo  $(x^8 + x^4 + x^3 + x + 1)$ , multiply the result by a fixed matrix and add a fixed polynomial. This transformation is the only non-linear transformation in the cipher.
2. Shift: The first row of the matrix remains constant, the second row is shifted one byte to the right, the third row is shifted two bytes to the right, and the last row is shifted three bytes to the right.
3. MixColumn: we consider each column of the matrix as a polynomial of degree less than 4 over  $GF(2^8)$  and multiply this polynomial by the polynomial  $03_x \cdot x^3 + 01_x \cdot x^2 + 01_x \cdot x + 02_x$  (where  $a_x$  denotes hexadecimal value) modulo  $(x^4 + 1)$ . This transformation is linear.
4. AddRoundKey: a 128-bit round key, which is derived of the key by the KeyExpansion algorithm, is bitwise XORed to the state. In the last round the MixColumn transformation is omitted and before the first round an AddRoundKey transformation is performed, using the key itself as a round key.

### 2.1 The Key Expansion

The round key of each round is derived of the key using the Key Expansion algorithm. Each round key is of length 128 bit and by the design knowing a

round key of any round is enough to recover the key. The round key is derived as follows: Let us denote the bytes of the expanded key by  $K_0, K_1, K_2, \dots$  where the key is  $K_0, K_1, \dots, K_{15}$ . Then the expanded key is derived from the formulae

$$K_n = \begin{cases} K_{n-1} \oplus K_{n-16}, & \text{if } 16 \nmid n \\ K_{n-16} \oplus \text{ByteSubstitution}(\text{Shifted } K_{n-1}) \oplus \text{Rcon}, & \text{otherwise} \end{cases}$$

We can see that the Key Expansion is not linear but there is strong connection between bytes of the  $k$ 'th round key and those of the  $(k + 1)$ 'th:  $K_n \oplus K_{n-1} = K_{n-16}$ . It gives us more information when guessing some bytes of a round key.

### 3 The Square Attack

We refer here to 4-round reduced variant of the cipher. Consider a set of 256 plaintexts which differ by one byte in which they assume all of the possible values (the plaintexts are equal in the other bytes). The AddRoundKey and the ByteSubstitution are permutations so after the ByteSubstitution of the first round the data still assumes all the possible values in that byte and equal in the others. Notice now that if two inputs of the MixColumn differ only in one byte, the corresponding outputs differ in all of the bytes: if the inputs are  $(a, b, c, d)$ ,  $(a', b, c, d)$  then (by the linearity of MixColumn) the difference between the outputs is

$$(02_x \cdot (a - a'), 01_x \cdot (a - a'), 01_x \cdot (a - a'), 03_x \cdot (a - a'))$$

(where all operations are in  $GF(2^8)$ ). Therefore, the outputs of the MixColumn differ in all the 4 bytes. Therefore, in our case in all of the four bytes of the column all of the values are assumed. As we said before, this property remains after the ByteSubstitution of round 2, the Shift transforms these four bytes to one byte in each column so after the MixColumn of round 2 in all of the bytes all of the values are assumed. Therefore, after the Shift of the third round in all of the bytes all of the values are assumed. Then the XOR of all of the values in each byte is 0. The MixColumn is a linear transformation. As a consequence, after the MixColumn still the XOR of the values in each byte is 0. This property remains after the AddRoundKey. Therefore, the

XOR of the inputs of the ByteSubstitution of the fourth round in each byte is 0.

Let us consider now 256 chosen plaintexts which assume all of the values in the first byte and all equal 0 in the other bytes. In the attack we guess one byte in the round key of the fourth round and decrypt the fourth round in the corresponding byte in all of the 256 ciphertexts. We get the 256 inputs of the ByteSubstitution of the fourth round. Now we XOR them and check, whether the result is equal to 0. If not, then the guess is wrong. The probability of this event (that the XOR equals 0) is  $1/256$  so on average only one wrong guess is left. We can check it by looking at another set of plaintexts. In the same way we find the rest of the fourth RoundKey and from that RoundKey we derive the key.

This attack can be extended to Rijndael reduced to 5 and 6 rounds by adding one round in the beginning or in the end or both of them (see [4]). The complexities of those attacks are summarized in Table 1.

### 3.1 Our Improvement of the Square Attack

In the Square attack the XOR of all of the 256 inputs of the ByteSubstitution of the last round is computed. For that, the corresponding ciphertext bytes are XORed with the guessed byte of the round key, then  $ByteSubstitution^{-1}$  is performed, and all the 256 results are XORed. The final result is expected to be zero.

We observe that pairs with equal values do not contribute to the last XOR, and thus pairs with equal values in the corresponding byte of the ciphertext do not contribute to it, independently of the byte of the round key. Therefore, we can avoid applying this process for pairs with the same values in the corresponding byte of the ciphertext. In the random case the probability that any byte value appears in  $n$  ciphertexts out of the 256 is  $prob(n) = \binom{256}{n} 256^{-n} \left(1 - \frac{1}{256}\right)^{256-n} \approx \frac{1}{en!}$  (this is the probability that when we throw 256 balls into 256 boxes, a box is being hit by  $n$  balls).

We throw out all pairs of ciphertexts which are equal in the byte we check, i.e., values which appear an odd number of times remain once, and all others are discarded. As a result a smaller set of byte values is found, and the rest of the analysis should be performed only on this smaller set. In the average case the number of remaining ciphertexts is about  $prob(1) +$

$prob(3) + prob(5) + \dots \approx 0.43$ . Thus, the key inner loop of the attack should be performed only to  $0.43 \cdot 256 \approx 110$  values on average, rather than 256. This reduces the complexity of the attack to 0.43 of the original complexity.

We also observe that in the small number of cases in which more than  $m = 128$  values remain (in the range 129 to 255 values), we can use the complementing set of  $256 - m < 128$  values instead, as if the XOR of the first set is zero, so is the XOR of the complementing set.

Note that in the cases of 4- and 5-round Rijndael we can also improve the data requirements by other techniques, without affecting this improved time complexity (see Appendix).

### 3.2 Analysis of a Variant with Reversed Order of Round Keys

If the order of round keys of Rijndael would be reversed, by either reversing the order of the round keys, or reversing the order of bytes in each round key, we would be able to reduce the complexity of the attack by another factor of  $2^8$  (and in total by  $2^9$  simultaneously with the previous improvement). This attack is based on the fact that given two consecutive bytes of a round key, it is easy to derive one byte of the previous round key. In the original Rijndael this byte cannot be the one guessed in the attack. However, after any of these modifications, we can select particular choices of guessed bytes in such a way that the fifth byte is directly computable from the first four, thus saving a factor of  $2^8$  in the complexity of analysis.

When the order of round keys is reversed, these choices are bytes 1, 6, 11, and 16 of the last round key, which give directly the first byte of the previous round key, which is the one required for the analysis in that round. When the order of bytes in each round key is reversed, bytes 4, 5, 10, and 15 can also be used, giving byte 4 of the previous round which can also be used for the analysis of that round. In both cases we save the complexity of guessing the  $2^8$  possible values of one byte.

## 4 An Attack on 5 Rounds Using Impossible Differentials

The impossible differential property on which the attack is based is the following: Consider Rijndael reduced to 4 rounds. If a pair of plaintexts differ by only one byte then the ciphertexts cannot be equal in any of the following combinations of bytes: (1,6,11,16), (2,7,12,13), (3,8,9,14), nor (4,5,10,15). The reason is that the difference before the first MixColumn is in one byte, so after it there is difference in one column, and then after the second MixColumn the data differs in all the bytes. On the other hand, if the ciphertexts are equal in one of the four prohibited combinations of bytes then after the third MixColumn the data is equal in one column, and thus before the MixColumn the data in this column is also equal. Therefore, after the second MixColumn there are 4 bytes in which the data is equal. This is a contradiction since we showed that all the bytes of the data differ after that MixColumn. This property is indeed impossible. Note that the probability of this test to pass when testing a random pair is about  $2^{-30}$ .

The attack analyzes Rijndael reduced to 5 rounds. The attack eliminates wrong round keys of the first round by showing that the impossible property holds in the last 4 rounds if these keys were used. At a precomputation stage we consider all of the pairs of the form  $((a, b, c, d), (a', b, c, d))$  in a column where  $a \neq a'$  (this is the data after the first round). For these pairs we undo the encryption of the first round, i.e., perform  $\text{MixColumn}^{-1}$ ,  $\text{Shift}^{-1}$  and  $\text{ByteSubstitution}^{-1}$  and create a hash table containing one of the inputs of the  $\text{ByteSubstitution}$   $x$  and the XOR of the two inputs  $x \oplus y$ , indexed by  $x \oplus y$ , where  $x, y$  are the inputs of the  $\text{ByteSubstitutions}$ . There are  $2^{32}$  possible values for  $x \oplus y$  and  $2^{40}$  values in the table so on average there are about  $2^8$  values of  $x$  which correspond to each value of  $x \oplus y$ .

In the attack we consider  $2^{32}$  chosen plaintexts which assume all the possible values in these four bytes and 0 in the others. There are about  $(2^{32})^2/2 = 2^{63}$  pairs of plaintexts so there are about  $2^{63}/2^{30} = 2^{33}$  pairs in which the ciphertexts are equal in one of the prohibited combinations. For these pairs we compute  $x \oplus y$ , and use the hash table to fetch the about  $2^8$  possibilities of  $x$  which correspond to the computed  $x \oplus y$ . This process identifies about  $2^8$  wrong keys by XORing the plaintext and the  $x$ 's. Due to



collisions the number of remaining wrong keys is about

$$2^{32}(1 - 2^{-32})^{2^{33} \cdot 2^8} \approx 2^{32}(e^{-1})^{2^9} = 2^{32}e^{-512} \approx 0$$

so only the right key remains. In the same way we get the rest of the key.

Notice that this number of remaining wrong pairs is very small, and thus that less chosen plaintexts can be used. A careful analysis shows that about  $2^{28}$  pairs which are formed from about  $2^{29.5}$  chosen plaintexts (randomly chosen out of the  $2^{32}$  former ones) suffice for this attack.

The complexity of this attack is as follows: In the precomputation we decrypt one column of one round of  $2^{40}$  pairs. This is equivalent to about  $\frac{1}{4.5} \cdot 2^{41} \approx 2^{37}$  encryptions. The resultant hash table requires  $2^{42}$  bytes of memory.

The attack requires  $2^{29.5}$  chosen plaintexts, which contain  $2^{28}$  pairs. For each pair we look at the table and get about  $2^8$  values of  $x$ , from which we compute the corresponding impossible values for  $k$ . These values are then deleted from the table of the possible keys. The time complexity of this stage (and of the attack) is about  $2^{31}$  encryptions. The attack requires  $2^{29.5}$  chosen plaintexts,  $2^{31}$  time,  $2^{42}$  bytes of memory (including  $2^{32}$  bits for the table of deleted keys), and  $2^{36}$  time of precomputation.

## A An Improvement of the Square Attack Against the 4- and 5-Round Variants

### A.1 The 4-Round Variant

We first present an attack on 4-round variant which is not so good by itself but can be combined with the Square attack, reducing the required data by a factor of 2.

For this attack we denote by “delta-set” a byte in which all the values are assumed and by a “passive byte” a byte in which all of the data is equal. In this attack we use the same property as Square attack does: if in the plaintexts one byte is a delta-set and the others are passive then in the inputs of the MixColumn transformation of the third round all of the bytes are delta-sets. Similar ideas were presented in [1].

Notice that we can switch between the MixColumn and AddRoundKey of the third round by using  $RoundKey^* = \text{MixColumn}^{-1}(\text{RoundKey})$  instead of the RoundKey.

We consider a set of 256 chosen plaintexts in which one byte is a delta-set and the others are passive. Now, we guess 4 bytes of the fourth RoundKey and decrypt the last round in one column (we choose the four bytes in such way that before the Shift of the last round these bytes are in the same column so we can decrypt this column), perform  $\text{MixColumn}^{-1}$  and get the inputs of the MixColumn in these four bytes. As we said before, all of the values have to appear in each byte. The probability of this event to occur in an arbitrary case is  $256!/256^{256} \approx 0$  so that all of the wrong keys fail this test. As a result we find 4 bytes of the fourth RoundKey. In the same way we can complete the rest of this RoundKey. Then, the original key can easily be derived from the Round Key. As we have seen, the probability of all 256 values assumed in each one of the 4 bytes is very low. So we can check less plaintexts and still with overwhelming probability all of the wrong guesses fail the test. With more accuracy, the probability of  $k$  values chosen in random between 0 and 255 to be different is  $\frac{256(256-1)\dots(256-k+1)}{256^k}$ . The complexity of this attack is big because we guess 32 bits of a Round Key. But we can combine this attack with the Square attack on 4-round Rijndael: in the Square attack after checking the XOR of the 256 (or 110, using our improvement) values, one wrong key remains on average. The designers of the attack then suggest to check the remaining keys by an another delta-set, thus requiring  $2^8$  additional chosen plaintexts.

We observe that instead we can perform the Square attack on 4 bytes of a column (we have to do it anyway in order to find these key bytes too). Then we are left with 16 possibilities of the 32 bits of the last Round Key. Now we perform the attack described above using only these 16 guesses and some of the chosen plaintexts we already have. We can see that even 16 plaintexts are enough because then by the Birthday Paradox with probability 0.5 two values in a byte are equal so the probability that the 16 values are different in all the 4 bytes is  $1/16$  so only the right key remains (if no, we check some more of the plaintexts).

In this way we reduce the data requirement by a factor of 2 without affecting the complexity.

## A.2 The 5-Round Variant

In the Square attack against 5-round Rijndael which is derived of the 4-round attack by extension in the end the attacker guesses 40 bits of key so we have to perform the attack on 6 delta-sets in order to find the key. We observe that it is enough to do it for 4 delta-sets. Then about  $2^8$  guesses of that bits of the RoundKeys remain. We perform it for all of the 4 columns and get  $(2^8)^4 = 2^{32}$  possibilities for the fifth RoundKey. Now we find the key by exhaustive search which requires only  $2^{32}$  encryptions so it doesn't contribute the complexity. Thus, we reduce the data requirements by a factor of 1.5 without affecting the time complexity.

## References

- [1] Eli Biham, *Cryptanalysis of Ladder-DES*, proceedings of FSE'97, lecture notes in computer science 1267, pp. 134–138, 1997.
- [2] Eli Biham, Alex Biryukov, Adi Shamir, *Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials*, proceedings of EUROCRYPT'99, lecture notes in computer science 1592, pp. 12–23, 1999.
- [3] Eli Biham, Alex Biryukov, Adi Shamir, *Miss in the Middle Attacks on Idea and Khufu* proceedings of FSE'99, lecture notes in computer science 1636, pp. 124–138, 1999.
- [4] Joan Daemen, Vincent Rijmen, *AES proposal: Rijndael*, submitted as a candidate to the AES.
- [5] Joan Daemen, Lars Knudsen, Vincent Rijmen, *The Block Cipher Square*, proceedings of FSE'97, lecture notes in computer science 1267, pp. 149–165, 1997.