New Impossible Differential Attacks on AES

Jiqiang Lu[1,⋆], Orr Dunkelman[2,⋆⋆], Nathan Keller[3,⋆⋆⋆], and Jongsung Kim[4,†]

[1] Department of Mathematics and Computer Science,
Eindhoven University of Technology,
5600 MB Eindhoven, The Netherlands
lvjiqiang@hotmail.com
[2] École Normale Supérieure
Département d'Informatique,
45 rue d'Ulm, 75230 Paris, France.
orr.dunkelman@ens.fr
[3]Einstein Institute of Mathematics, Hebrew University.
Jerusalem 91904, Israel
nkeller@math.huji.ac.il
[4] Center for Information Security Technologies(CIST), Korea University
Anam Dong, Sungbuk Gu, Seoul, Korea
joshep@cist.korea.ac.kr

Abstract. In this paper we apply impossible differential attacks to reduced round AES. Using various techniques, including the early abort approach and key schedule considerations, we significantly improve previously known attacks due to Bahrak-Aref and Phan. The improvement of these attacks leads to the best known impossible differential attacks on 7-round AES-128 and AES-192, as well as to the best known impossible differential attacks on 8-round AES-256.

Keywords: AES, Impossible differential cryptanalysis

1 Introduction

The *Advanced Encryption Standard (AES)* [13] is a 128-bit block cipher with a variable key length (128, 192, and 256-bit keys are supported). Since its selection, AES gradually became one of the most widely used block ciphers. AES has received a great deal of cryptanalytic attention, both during the AES process, and even more after its selection.

In the single-key model, previous results can attack up to 7 rounds of AES-128 (i.e., AES with 128-bit key). The first attack is a SQUARE attack suggested in [15]

⋆ The work was done when this author wasa Ph.D. student at Royal Holloway, University of London.

⋆⋆ The first author was supported by the France Telecome Chaire. Some of the work presented in this paper was done while the first author was staying at K.U. Leuven.

⋆⋆⋆ This author is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

† This author was supported by the Second Brain Korea 21 Project.

which uses $2^{128} - 2^{119}$ chosen plaintexts and 2^{120} encryptions. The second attack is a meet-in-the-middle attack proposed in [16] that requires 2^{32} chosen plaintexts and has a time complexity equivalent to almost 2^{128} encryptions. Recently, another attack on 7-round AES-128 was presented in [1]. The new attack is an impossible differential attack that requires $2^{117.5}$ chosen plaintexts and has a running time of 2^{121} encryptions.

Similar results, but with better attack algorithms and lower complexities were reported in [20]. The resulting impossible differential attack on 7-round AES-192 has a data complexity of 2^{92} chosen plaintexts and time complexity of 2^{162} encryptions, while the attack on AES-256 uses $2^{116.5}$ chosen plaintexts and running time of $2^{247.5}$ encryptions.

There are several attacks on AES-192 [1, 14, 15, 18–20]. The two most notable ones are the SQUARE attack on 8-round AES-192 presented in [15] that requires almost the entire code book and has a running time of 2^{188} encryptions and the meet in the middle attack on 7-round AES-192 in [14] that requires 2^{34+n} chosen plaintexts and has a running time of $2^{208-n} + 2^{82+n}$ encryptions. Legitimate values for n in the meet in the middle attack on AES-192 are $94 \geq n \geq 17$, thus, the minimal data complexity is 2^{51} chosen plaintexts (with time complexity equivalent to exhaustive search), and the minimal time complexity is 2^{146} (with data complexity of 2^{97} chosen plaintexts).

AES-256 is analyzed in [1, 14, 15, 18, 20]. The best attack is the meet in the middle attack in [14] which uses 2^{32} chosen plaintexts and has a total running time of 2^{209} encryptions.

Finally, we would like to note the existence of many related-key attacks on AES-192 and AES-256. As the main issue of this paper is not related-key attacks, and as we deal with the single key model, we do not elaborate on the matter here, but the reader is referred to [21] for the latest results on related-key impossible differential attacks on AES and to [17] for the latest results on related-key rectangle attacks on AES.

The strength of AES with respect to impossible differentials was challenged several times. The first attack of this kind is a 5-round attack presented in [5]. This attack is improved in [11] to a 6-round attack. In [19], an impossible differential attack on 7-round AES-192 and AES-256 is presented. The latter attack uses 2^{92} chosen plaintexts (or $2^{92.5}$ chosen plaintexts for AES-256) and has a running time of 2^{186} encryptions (or $2^{250.5}$ encryptions for AES-256). The time complexity of the latter attack was improved in [20] to 2^{162} encryptions for AES-192.

In [1] a new 7-round impossible differential attack was presented. The new attack uses a different impossible differential, which is of the same general type as the one used in previous attacks (but has a slightly different structure). Using the new impossible differential leads to an attack that requires $2^{117.5}$ chosen plaintexts and has a running time of 2^{121} encryptions. This attack was later improved in [2, 20] to use $2^{115.5}$ chosen plaintexts with time complexity of 2^{119} encryptions.

The last application of impossible differential cryptanalysis to AES was the extension of the 7-round attack from [1] to 8-round AES-256 in [20]. The extended attack has a data complexity of $2^{116.5}$ chosen plaintexts and time complexity of $2^{247.5}$ encryptions.

We note that there were three more claimed impossible differential attacks on AES in [8–10]. However, as all these attacks are flawed [7].

In this paper we present a new attack on 7-round AES-128, a new attack on 7-round AES-192, and two attacks on 8-round AES-256. The attacks are based on the attacks proposed in [1, 19] but use additional techniques, including the *early abort* technique and key schedule considerations.

Our improvement to the attacks on 7-round AES-128 from [1, 20] requires $2^{112.2}$ chosen plaintexts, and has a running time of $2^{117.2}$ memory accesses. Our improvement to the attack on 7-round AES-192 from [19] has a data complexity of $2^{91.2}$ chosen plaintexts and a time complexity of $2^{139.2}$ encryptions. Since the first attack is also applicable to AES-192, the two attacks provide a data-time tradeoff for attacks on 7-round AES-192.

The best attack we present on 8-round AES-256 requires $2^{89.1}$ chosen plaintexts and has a time complexity of $2^{229.7}$ memory accesses. These results are significantly better than any previously published impossible differential attack on AES. We summarize our results along with previously known results in Table 1.

Although the attacks presented in the paper are not the best known attacks on AES, the results are important, both due to the significance of the AES, and since the techniques used in the paper can be useful in other works as well.

This paper is organized as follows: In Section 2 we briefly describe the structure of AES. In Section 3 we discuss the previous impossible differential attacks. In Section 4 we describe the possible improvements and extensions (to 8-round AES-256) of the Bahrak and Aref attack. The improvement of Phan's attack on 7-round AES-192 along with its extension to 8-round AES-256 is presented in Section 5. In Appendix A we describe a technique which is repeatedly used in impossible differential attacks on AES. Appendix B outlines the impossible differentials used in this paper for the sake of completeness. We conclude the paper in Section 6.

2 Description of AES

The advanced encryption standard [13] is an SP-network that supports key sizes of 128, 192, and 256 bits. A 128-bit plaintext is treated as a byte matrix of size 4x4, where each byte represents a value in $GF(2^8)$. An AES round applies four operations to the state matrix:

- SubBytes (SB) — applying the same 8-bit to 8-bit invertible S-box 16 times in parallel on each byte of the state,
- ShiftRows (SR) — cyclic shift of each row (the i'th row is shifted by i bytes to the left),
- MixColumns (MC) — multiplication of each column by a constant 4x4 matrix over the field $GF(2^8)$, and
- AddRoundKey (ARK) — XORing the state with a 128-bit subkey.

We outline an AES round in Figure 1. In the first round, an additional AddRoundKey operation (using a whitening key) is applied, and in the last round the MixColumns

Key Size	Number of Rounds	Complexity Data (CP)	Complexity Time	Attack Type & Source
128	7	$2^{128} - 2^{119}$	2^{120}	SQUARE [15]
	7	$2^{117.5}$	2^{121}	Impossible Differential [1]
	7	$2^{115.5}$	2^{119}	Impossible Differential [2, 20]
	7	2^{32}	2^{128}	Meet in the Middle [16]
	7	$2^{112.2}$	$2^{117.2}$ MA	Impossible Differential (App. 4.1)
192	7	2^{32}	2^{184}	SQUARE [18]
	7	$19 \cdot 2^{32}$	2^{155}	SQUARE [15]
	7	2^{92}	$2^{186.2}$	Impossible Differential [19]
	7	$2^{115.5}$	2^{119}	Impossible Differential [20]
	7	2^{92}	2^{162}	Impossible Differential [20]
	7	2^{34+n}	$2^{208-n} + 2^{82+n}$	Meet in the Middle [14]
	8	$2^{128} - 2^{119}$	2^{188}	SQUARE [15]
	7	$2^{113.8}$	$2^{118.8}$ MA	Impossible Differential (Sect. 4.1)
	7	$2^{91.2}$	$2^{139.2}$	Impossible Differential (Sect. 5.1)
256	7	2^{32}	2^{200}	SQUARE [18]
	7	$21 \cdot 2^{32}$	2^{172}	SQUARE [15]
	7	$2^{92.5}$	$2^{250.5}$	Impossible Differential [19]
	7	2^{32}	2^{208}	Meet in the Middle [14]
	7	2^{34+n}	$2^{208-n} + 2^{82+n}$	Meet in the Middle [14]
	7	$2^{115.5}$	2^{119}	Impossible Differential [20]
	8	$2^{116.5}$	$2^{247.5}$	Impossible Differential [20]
	8	$2^{128} - 2^{119}$	2^{204}	SQUARE [15]
	8	2^{32}	2^{209}	Meet in the Middle [14]
	7	$2^{113.8}$	$2^{118.8}$ MA	Impossible Differential (Sect. 4.1)
	7	2^{92}	2^{163} MA	Impossible Differential (Sect. 5.1)
	8	$2^{111.1}$	$2^{227.8}$ MA	Impossible Differential (Sect. 4.2)
	8	$2^{89.1}$	$2^{229.7}$ MA	Impossible Differential (Sect. 5.2)

CP – Chosen plaintext, MA – Memory Accesses
Time complexity is measured in encryption units unless mentioned otherwise

Table 1. A Summary of the Previous Attacks and Our New Attacks

operation is omitted. As all other works on AES, we shall assume that reduced-round variants also have the MixColumns operation omitted from the last round.

The number of rounds depends on the key length: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. The rounds are numbered $0, \ldots, Nr-1$, where Nr is the number of rounds ($Nr \in \{10, 12, 14\}$). For the sake of simplicity we shall denote AES with n-bit keys by AES-n, i.e., AES with 192-bit keys (and thus with 12 rounds) is denoted by AES-192.

The key schedule of AES takes the user key and transforms it into 11, 13, or 15 subkeys of 128 bits each. The subkey array is denoted by $W[0, \ldots, 59]$, where each word of $W[\cdot]$ consists of 32 bits. The first Nk words of $W[\cdot]$ are loaded with the user supplied key, i.e., $Nk = 4$ words for 128-bit keys, $Nk = 6$ words for 192-bit keys, and

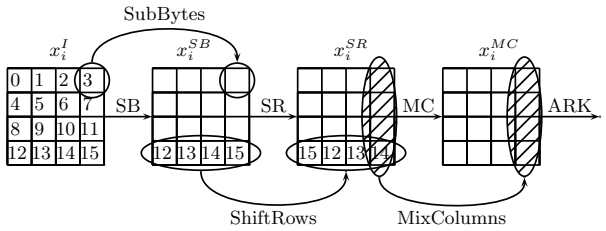


Fig. 1. An AES round

$Nk = 8$ for 256-bit keys. The remaining words of $W[\cdot]$ are updated according to the following rule:

- For $i = Nk, \ldots, 43/51/59$, do
  - If $i \equiv 0 \bmod Nk$ then $W[i] = W[i-Nk] \oplus SB(W[i-1] \lll 8) \oplus RCON[i/Nk]$,
  - Otherwise $W[i] = W[i-1] \oplus W[i-Nk]$,

where $RCON[\cdot]$ is an array of predetermined constants, and $\lll$ denotes rotation of the word by 8 bits to the left. We also note that for 256-bit keys, when $i \equiv 4 \bmod 8$ the update rule is $W[i] = W[i-8] \oplus SB(W[i-1] \lll 8)$.

2.1 The Notations Used in the Paper

In our attacks we use the following notations: x_i^I denotes the input of round i, while x_i^{SB}, x_i^{SR}, x_i^{MC}, and x_i^O denote the intermediate values after the application of SubBytes, ShiftRows, MixColumns, and AddRoundKey operations of round i, respectively. Of course, the relation $x_{i-1}^O = x_i^I$ holds.

We denote the subkey of round i by k_i, and the first (whitening) key is k_{-1}, i.e., the subkey of the first round is k_0. In some cases, we are interested in interchanging the order of the MixColumns operation and the subkey addition. As these operations are linear they can be interchanged, by first XORing the data with an equivalent key and only then applying the MixColumns operation. We denote the equivalent subkey for the altered version by w_i, i.e., $w_i = MC^{-1}(k_i)$.

We denote bytes of some intermediate state x_i or a key k_i (or w_i) by an enumeration $\{0, 1, 2, \ldots, 15\}$ where the byte $4m+n$ corresponds to the n'th byte in the m'th row of x_i, and is denoted by $x_{i,4 \cdot m+n}$. We denote the z'th column of x_i by $x_{i,Col(z)}$, i.e., $w_{0,Col(0)} = MC^{-1}(k_{0,Col(0)})$. Similarly, by $x_{i,Col(y,z)}$ we denote columns y and z of x_i. We define two more column related sets. The first is $x_{i,SR(Col(z))}$ which is the bytes in x_i corresponding to the places after the ShiftRows operation on column z, e.g., $x_{i,SR(Col(0))}$ is composed of bytes 0,7,10,13. The second is $x_{i,SR^{-1}(Col(z))}$ which is the bytes in the positions of column z after having applied the inverse ShiftRows operation.

3 Previous Impossible Differential Attacks on AES

The security of AES against impossible differential attacks was challenged in two lines of research. The first presented in [5, 11, 19, 20], and the second in [1, 20]. Both lines

use very similar impossible differentials as well as similar algorithms. In this section we present the previously known results:

All known impossible differential attacks on the AES, are based on the following 4-round impossible differential of AES, first observed in [5]:

Proposition 1. *Let $\Delta(x_i^I)$ denote the input difference to round i, and let $\Delta(x_{i+3}^{SR})$ denote the difference after the ShiftRows operation of round $i+3$. If the following two conditions hold:*

1. *$\Delta(x_i^I)$ has only one non-zero byte,*
2. *In $\Delta(x_{i+3}^{SR})$, at least one of the four sets of bytes $SR(Col(i))$, for the four different possible columns, is equal to zero,*

then $\Delta(x_i^I) \longrightarrow \Delta(x_{i+3}^{SR})$ is an impossible differential for any four consecutive rounds of AES. We outline one of these impossible differentials in Figure 4 (in Appendix B). We also note that if in round $i+3$ the order of MixColumns and AddRoundKey is swapped, then, one can consider the impossible differential $\Delta(x_i^I) \longrightarrow \Delta(x_{i+3}^O)$.

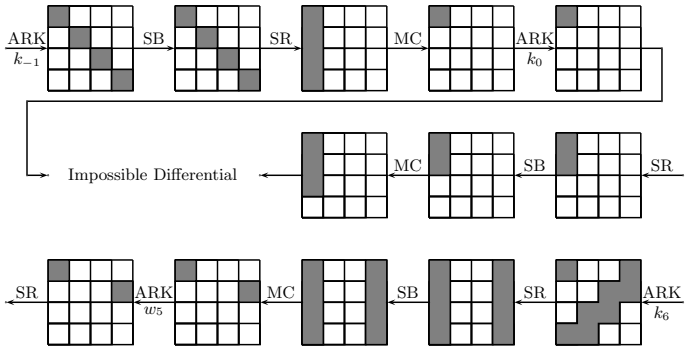
Proof. On the one hand, if $\Delta(x_i^I)$ has only one non-zero byte then $\Delta(x_{i+1}^I)$ has non-zero values in a single column, and therefore, $\Delta(x_{i+2}^I)$ has non-zero values in all the 16 bytes of the table (following the basic diffusion properties of AES, a fact used in many attacks on AES). On the other hand, if Condition (2) holds then $\Delta(x_{i+3}^I)$ has at least one zero column, and hence at least one of the four sets of bytes $SR^{-1}(Col(i))$ in $\Delta(x_{i+2}^I)$ is equal to zero, a contradiction.

3.1 The Original Bahrak-Aref Attack on 7-round AES-128

The algorithm of the BA attack, as described in [1], is the following (depicted in Figure 2): [1]

1. Encrypt $2^{85.5}$ structures of 2^{32} plaintexts each, such that in every structure, bytes $SR^{-1}(Col(0))$ assume all the 2^{32} possible values and the rest of the bytes are fixed.
2. In each structure, look for ciphertext pairs with zero difference in bytes $SR(Col(1,2))$ and discard the other pairs.
3. Guess the values of $k_{6,SR(Col(3))}$ and partially decrypt all the remaining ciphertext pairs through round 6 to obtain the difference $\Delta(x_{5,Col(3)}^{SR})$. Select only the pairs for which $\Delta(x_{5,Col(3)}^{SR})$ has a non-zero value only in byte 7.
4. Guess the values of bytes $k_{6,SR(Col(0))}$ and partially decrypt all the remaining ciphertext pairs through round 6 to get the difference $\Delta(x_{5,Col(0)}^{SR})$. Select only the pairs for which $\Delta(x_{5,Col(0)}^{SR})$ has a non-zero value only in byte 0.
5. Guess the values of bytes $(0,7)$ of the equivalent key w_5 and partially decrypt all the remaining ciphertext pairs through round 5 to get the difference $\Delta(x_{4,Col(0)}^{SR})$. Select only the pairs for which $\Delta(x_{4,Col(0)}^{SR})$ has one zero byte value.

[1] Note that different notations are used in [1]. We adapted their attack to the standard notations used in AES.



A gray box stands for a non-zero difference in the byte, while a white box stands for a zero difference.

Fig. 2. The 7-Round Impossible Differential Attack on AES by Bahrak and Aref

6. For each of the remaining pairs, consider the corresponding plaintext pair and discard all the values of $k_{-1,SR^{-1}(Col(0))}$ that lead to the input difference of the impossible differential in the input of round 1. If a guess for these bytes remains, guess all the remaining key bytes and check the guess using trial encryption. Otherwise, repeat Steps 4–6 for a different guess of k_6.

Step 1 of the attack consists of encryption of $2^{117.5}$ chosen plaintexts. Step 2 requires $2^{117.5}$ memory accesses, and suggests $2^{84.5}$ pairs to Step 3. Step 3 takes $2^{117.5}$ partial decryptions and passes $2^{60.5}$ pairs, for a given subkey guess, to Step 4, which in turn takes $2^{125.5}$ partial decryptions and outputs $2^{36.5}$ pairs to Step 5. Step 5 requires $2^{117.5}$ partial decryptions, and leaves $2^{30.5}$ pairs to Step 6, which uses $2^{118.5}$ memory accesses.

Hence, the total time complexity of the attack is 2^{121} 7-round AES encryptions.[2] The data complexity of the attack is $2^{117.5}$ chosen plaintexts, and the memory complexity is 2^{109} bytes of memory required for storing the list of discarded key values.

3.2 The Phan Attack Algorithm on 7-Round AES-192

The algorithm of the Phan attack, as described in [19], is the following (depicted in Figure 3):

1. Encrypt 2^{60} structures of 2^{32} plaintexts each such that in every structure, the bytes of $SR^{-1}(Col(0))$ assume all the 2^{32} possible values and the rest of the bytes are fixed.
2. Select only the ciphertext pairs, corresponding to plaintexts in the same structure, for which the difference in bytes $SR(Col(2,3))$ is zero.

[2] In [1] Step 4 was considered as a full one-round decryption, while it is only 1/4 round in reality.

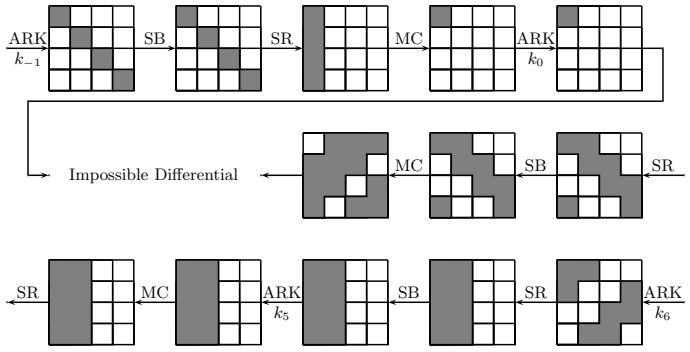


Fig. 3. The 7-Round Impossible Differential Attack on AES-192 by Phan

3. Guess k_6 and partially decrypt the remaining ciphertext pairs through round 6 to get x_6^I.
4. Using the guessed value of k_6, retrieve $k_{5,Col(0,1)}$ by the key schedule algorithm. For each remaining pair, decrypt x_6^I through $ARK^{-1} \circ MC^{-1} \circ SR^{-1} \circ SB^{-1} \circ MC^{-1}$ to get the difference $\Delta(x_4^{SR})$.[3] If the difference does not satisfy Condition (2) of Proposition 1, discard the pair.
5. Consider the plaintext pairs corresponding to the remaining ciphertext pairs. Guess the value of $k_{-1,SR^{-1}(Col(0))}$ and partially encrypt each plaintext pair through $ARK \circ SB \circ SR \circ MC$ to get the difference $\Delta(x_1^I)$. If the difference satisfies Condition (1) of Proposition 1, discard the guess of $k_{-1,SR^{-1}(Col(0))}$.
6. If all the guesses of $k_{-1,SR^{-1}(Col(0))}$ are discarded for a guess of k_6, repeat Steps 3–5 with another guess of k_6. If a candidate of $k_{-1,SR^{-1}(Col(0))}$ remains, the rest of the key bits (or their equivalent) are exhaustively searched.

Step 1 of the attack consists of the encryption of 2^{92} chosen plaintexts. Step 2 of the attack takes 2^{92} memory accesses and proposes 2^{59} pairs for further analysis. Steps 3 and 4 take together 2^{188} two-round decryptions, and suggest 2^{29} pairs for Step 5 (for a given key guess). Step 5 takes 2^{185} 1-round encryptions.

Therefore, the data complexity of the attack is 2^{92} chosen plaintexts, and the time complexity is $2^{186.2}$ 7-round AES-192 encryptions. The attack requires 2^{157} bytes of memory, used for storing the discarded guesses of k_6 and $k_{-1,SR^{-1}(Col(0))}$.

4 Improvement of the Bahrak-Aref Attack

4.1 Improvement of the Bahrak-Aref Attack on 7-Round AES-128

Our improvements to the 7-round attack are based on several techniques and observations. First, instead of partially decrypting each candidate pair under many possible

[3] Note that the ARK^{-1} operation in the end of round 4 can be skipped since it does not affect the difference $\Delta(x_4^{SR})$.

key guesses, we use table look ups to deduce which key it suggests. The second improvement is based on re-using the data, i.e., repeating the attack several times by using a slightly different impossible differential. Using these two ideas along with several additional techniques, we succeed in reducing the data and time complexities of the BA attack.

We first discuss the improvement of the time complexity of the attack. However, as we demonstrate later, the data complexity of the attack can be reduced to $2^{112.2}$ chosen plaintexts. Hence, we use this figure in the computation of the time complexity as well.

Improving the Time Complexity of the BA Attack Steps 1 and 2 of the attack remain unchanged, except for the reduction in the number of plaintexts. As a result, the time complexity of Step 2 is $2^{112.2}$ memory accesses, and the number of pairs remaining after Step 2 is $2^{79.2}$.

In Step 3 of the BA attack the attacker decrypts a full column through round 6 by guessing 32 subkey bits. This step is significantly improved by using a well-known observation related to differential cryptanalysis: Given an input and an output differences of the SubBytes operation, there is on average one pair of actual values that satisfies these differences.[4] Since for any ciphertext pair, the difference $\Delta(x^{SB}_{6,Col(3)})$ is known, the knowledge of the difference $\Delta(x^{I}_{6,Col(3)})$ can be used to find the actual values of $x^{SB}_{6,Col(3)}$, and hence the value of $k_{6,SR(Col(3))}$ as well. ere are only $2^8 - 1$ possible values of $\Delta(x^{SR}_{5,Col(3)})$ in which only byte 7 is non-zero. Thus, there are only 255 possible differences $\Delta(x^{I}_{6,Col(3)})$ that may be considered for further analysis. Hence, the attacker can perform Step 3 of the attack in the following way:

- Initialize 2^{32} empty lists, each corresponds to a different guess of $k_{6,SR(Col(3))}$.
- For each remaining ciphertext pair, and for each of the 255 possible differences in $\Delta(x^{I}_{6,Col(3)})$, compute the key which leads this specific pair to this specific difference. Add this pair to the list corresponding to that specific key guess.

For each pair of ciphertexts and a guess of the difference in $\Delta(x^{I}_{6,Col(3)})$ we expect one key suggested on average. These $2^{79.2} \cdot 255 = 2^{87.2}$ suggestions are distributed over 2^{32} possible subkeys, and thus, for a given subkey guess, we expect $2^{55.2}$ pairs to remain.

As finding the key suggestion can be done by a simple look-up table, the time complexity of the modified Step 3 is $255 \cdot 2^{79.2} = 2^{87.2}$ memory accesses. We note that by storing the results in the table in an efficient way, we can obtain the result in one memory access even if more than one key is suggested. Actually, most of the queries are expected to fail, while about 1/16 of the queries are expected to return 16 options (which can be efficiently stored in one 32-bit memory block), and even smaller fraction are expected to return more options. Hence, in total we expect $2^{87.2}$ memory accesses to perform this step.

[4] More accurately, for randomly chosen input and output differences we expect that about half of the combinations are not possible, about half propose two actual values, and a really small fraction suggest four values.

Step 4 of the attack can be modified similarly to Step 3. As a result, the time complexity of the modified Step 4 is $2^{95.2}$ memory accesses, and the number of remaining pairs (for a given subkey guess of the 64 bits of $k_{6,SR(Col(0,3))}$) after Step 4 is $2^{31.2}$.

Step 5 of the attack can be also improved using the observation presented above. Recall that the goal of the attacker in Step 5 is to check whether $\Delta(x^{SR}_{4,Col(0)})$ has one zero byte value. There are $4 \cdot 255^3$ values $u = \Delta(x^{SR}_{4,Col(0)})$ satisfying this requirement, and for each such value u, there is a unique $v = \Delta(x^{I}_{5,Col(0)})$ such that $v = MC(u)$. Of the $4 \cdot 255^3$ possible v's there are $4 \cdot 255 \approx 2^{10}$ with zero difference in bytes $(8, 12)$. Hence, the goal of the attacker is to check, for every guess of bytes $(0, 7)$ of w_5, whether the value in bytes $(0, 4)$ of $\Delta(x^{I}_{5,Col(0)})$ falls into these 2^{10} values.

As explained before, given the input/output differences of the SubBytes operation, the actual inputs and outputs can be determined. In turn, these values can be used to determine w_5 in a similar way to that described in Step 3.

Thus, at the end of this process, the $2^{31.2}$ pairs and the 2^{10} possible differences propose $2^{41.2}$ candidate keys which are scattered among 2^{16} candidate values. Hence, for a given guess for bytes (0,7) of w_5 we expect $2^{41.2}/2^{16} = 2^{25.2}$ pairs which lead to the output difference of the impossible differential. As finding the key suggestion can be done by a simple look-up table, this step takes $2^{31.2} \cdot 2^{10}$ memory accesses for a given subkey guess of k_6, or a total of $2^{64} \cdot 2^{31.2} \cdot 2^{10} = 2^{105.2}$ memory accesses.

We note that the overall complexity of these steps can be reduced to about 2^{90} memory accesses. This is done by observing that in the first step of the filtering, i.e., Step 2, we obtain $2^{79.2}$ pairs with 2^{64} possible differences. Our analysis can be easily augmented to take this into consideration and analyze only one pair with a given difference. However, we note that this is not the bottleneck in our algorithm (and thus, this step is not as optimized as possible).

Improving Step 6 of the BA Attack and Reducing the Data Complexity Since our improved attack uses less data than the original attack, it is expected that for every guess of the 80 bits of k_6 and w_5, only $2^{25.2}$ pairs are analyzed in Step 6 of the attack. This number of pairs is far from being sufficient to discard all the possible values of $k_{-1,SR^{-1}(Col(0))}$. Hence, Step 6 of the original attack has to be modified.

We start with a simple observation: In step 6 of the BA attack, the attacker uses only pairs that have a non-zero difference only in byte 0 (while the remaining bytes have a zero difference). As there are actually 4 impossible differentials which originate from this column (one for each possible non-zero byte of $x_{1,Col(0)}$), we can use other pairs as well. This observation increases the number of keys each pair discards from 2^8 to 2^{10}.[5]

Furthermore, we observe that four variants of the original attack can be applied using the same data set. For each guess of $k_{6,SR(Col(0,3))}$ we apply four similar attacks: The first is the BA attack, and in the other three variants, bytes $(0, 7)$ of w_5 are replaced by the pairs of bytes $(4, 11), (8, 15)$, and $(12, 3)$, respectively. Each attack discards possible values of 112 subkey bits, including 96 common bits (the bits of

[5] We note that the same improvement was independently suggested in [20].

$k_{6,SR(Col(0,3))}$ and $k_{-1,SR^{-1}(Col(0))}$) and 16 bits that differ between the attacks. After applying the four attacks separately, we aggregate the remaining subkey values to get a small number of suggestions for part of the key, and check the suggestions by exhaustive search over the remaining key bits.

This is done efficiently by running the modified steps 3–5 each time for each guess of $k_{6,SR(Col(0,3))}$ with different two active bytes in x_4^{MC}. For each guess of $k_{6,SR(Col(0,3))}$ we obtain the remaining candidates for $k_{-1,SR^{-1}(Col(0))}$ and the two respective bytes of w_5. We store these in a table indexed by the $k_{-1,SR^{-1}(Col(0))}$ values. The $2^{25.2}$ analyzed pairs (for a guess of $k_{6,SR(Col(0,3)}$ and two bytes of w_5) suggest $(1-2^{-32})^{2^{35.2}} = 2^{-13.25}$ remaining candidates for $k_{-1,SR^{-1}(Col(0))}$ on average. Equivalently, for each value of $k_{-1,SR^{-1}(Col(0))}$, there are $2^{2.75}$ remaining values of the two w_5 bytes for each of the four pairs of two bytes from w_5 (for a given $k_{6,SR(Col(0,3))}$ guess). After applying the four attacks separately, we go over the $k_{-1,SR^{-1}(Col(0))}$ values, and for each value we compute all the possible combinations of $w_{5,Col(0,3)}$ that correspond to the value of $k_{-1,SR^{-1}(Col(0))}$ in the respective attack. As a result, for each value of $k_{-1,SR^{-1}(Col(0))}$ we expect $(2^{2.75})^4 = 2^{11}$ values of $w_{5,Col(0,3)}$. Combining all the $k_{-1,SR^{-1}(Col(0))}$ values, we get 2^{43} possible values for $w_{5,Col(0,3)}$, and consequently, a list of 2^{43} values for $k_{5,Col(0,3)}$.

Hence, for each guess of $k_{6,SR(Col(0,3))}$ we are left with 2^{43} possible values of $k_{5,Col(0,3)}$. Now we observe that by the key schedule of AES-128, there is a relation between $k_{6,SR(Col(0,3))}$ and $k_{5,Col(0,3)}$. The following relations hold due to the key schedule:

1. $k_{6,0} = k_{5,0} \oplus SubBytes(k_{5,7}) \oplus RCON[7]$,
2. $k_{6,12} = k_{5,12} \oplus SubBytes(k_{5,3}) \oplus RCON[7]$,
3. $k_{6,7} = k_{5,7} \oplus k_{6,6}$.

Since all the key bytes in the above three equations are included in $k_{6,SR(Col(0,3))}$ and $k_{5,Col(0,3)}$, the equations yield a 24-bit filtering on the remaining key possibilities. Therefore, for each guess of $k_{6,SR(Col(0,3))}$, we are left with $2^{43} \cdot 2^{-24} = 2^{19}$ possible values of $k_{5,Col(0,3)}$.

Finally, we observe that due to the key schedule, the values of $k_{6,SR(Col(0,3))}$ and $k_{5,Col(0,3)}$, along with three additional key bytes, are sufficient to recover the secret key. Hence, the remaining possible key values can be checked by exhaustive key search with a time complexity of $2^{64} \cdot 2^{19} \cdot 2^{24} = 2^{107}$ encryptions.

The total time complexity of the modified attack is dominated by the four applications of the basic attack, which are in turn dominated by discarding the possible key values in Step 6 of the attack. Hence, the total time complexity of the improved attack is $2^{80} \cdot 2^{25.2} \cdot 2^{10} \cdot 4 = 2^{117.2}$ memory accesses.

For AES-192 and AES-256 the attacker has to use a stronger filtering (as the last part of exhaustive key search is dominating). For AES-192, by using $2^{113.8}$ chosen plaintexts, the attacker can find the key using $2^{118.8}$ memory accesses (the increase follows the increase in the data complexity). For AES-256, since the attack retrieves only 128 key bits an d thus the complexity of retrieving the rest of the key by exhaustive search is too high, the attacker has to repeat the attack using different columns in order to retrieve additional subkey material. The second application of the attack is

expected to be much faster than the first one due to the already known key material. The resulting data complexity is $2^{113.8}$ chosen plaintexts and the time complexity is $2^{118.8}$ memory accesses.

Improving the Memory Complexity We conclude this section by improving the memory complexity of the attack. We observe that there is no need to store the discarded values of the 112 guessed subkey bits. Instead, for each guess of the 64 key bits of k_6 the attack is performed separately, and if the guess is not discarded it is checked immediately by exhaustive search. Thus, most of the memory is used to store the $2^{87.2}$ remaining pairs, i.e., a total of $2^{93.2}$ bytes of memory.

Summarizing the improved attack, the data complexity of the improved attack is $2^{112.2}$ chosen plaintexts, the time complexity is $2^{117.2}$ memory accesses, and the memory complexity is $2^{93.2}$ memory bytes.

4.2 Extension of the Bahrak-Aref Attack to 8-Round AES-256

The obvious extension which is based on guessing the 8th round subkey and applying the 7-round attack, has a time complexity of $2^{237.2}$ encryptions and $2^{245.7}$ memory accesses.

We improve this basic attack by exploiting the key schedule and by modifying the way the partial decryption of round 7 is handled. We first note that the knowledge of k_7 leads to the knowledge of $k_{5,Col(1,2,3)}$. By slightly changing the attack to use an impossible differential which depends on bytes of $w_{5,Col(1,2,3)}$, we eliminate the need to "guess" bytes of w_5.

Furthermore, we observe that the partial decryption can be handled in a more efficient manner. Rather than guessing the entire subkey of the last round, we guess only 96 bits of the key, and then try all the possible 2^{16} differences in the remaining column to find the key guesses that fit the analyzed pair in a more efficient way.

The resulting attack algorithm is as follows:

1. Guess bytes $k_{7,SR(Col(0,1,2))}$ and decrypt all the ciphertexts through $ARK^{-1} \circ SR^{-1} \circ SB^{-1} \circ MC^{-1}$ in three columns to get $\Delta(x^{SR}_{6,Col(0,1,2)})$. Discard all pairs in which $\Delta(x^{SR}_{6,Col(0,1,2)})$ is not equal to zero in at least one of the bytes $(0, 1, 4, 10, 12, 13)$.
2. Initialize for each guess of $k_{7,Col(3)}$ an empty list.
3. For each of the remaining pairs, and each of the possible differences in $\Delta(x^{SR}_{6,Col(3)})$ in which bytes (7,11) are zero, find the key that this pair suggests (using the knowledge of the input/output differences to the SubBytes operation in Round 7), and add the pair to the list corresponding to this key.
4. Repeat the 7-round attack with the following changes:
   (a) In Step 3 of the 7-round attack the attacker guesses $w_{6,SR(Col(3))}$ and selects only pairs in which $\Delta(x^{SR}_{5,Col(3)})$ has non-zero value only in byte 3.
   (b) In Step 4 of the 7-round attack the attacker guesses $w_{6,SR(Col(2))}$ and selects only pairs in which $\Delta(x^{SR}_{5,Col(2)})$ has non-zero value only in byte 6.
   (c) The attacker then partially decrypts the pair (using the knowledge of the relevant bytes of w_5), and checks whether the obtained difference is active only in three bytes of x^{MC}_4.

Analysis of the Improved Partial Decryption The result of the improved partial decryption is the same as the result obtained by a naive approach: There are $2^{79.2}$ remaining pairs for each 128-bit guess of k_7, and Steps 3–6 of the 7-round attack can be applied. The time complexity of the Steps 1–3 of the 8-round attack is $3 \cdot 2^{112.2} \cdot 2^{96} = 2^{209.8}$ one column decryptions for Step 1 and $2^{96} \cdot 2^{95.2} \cdot 2^{16} = 2^{207.2}$ memory accesses (there are about $2^{95.2}$ pairs remaining after Step 1 for each subkey guess). These steps can be further optimized, but due to the complexity of other steps in the 8-round attack, we refrain from doing so.

Analysis of Steps 3–4 of the 7-Round Attack in the 8-Round Attack Steps 3 and 4 of the 7-round attack are repeated for each guess of the 128-bit key w_7. The time complexity of Step 3 is therefore $2^{79.2} \cdot 2^8 \cdot 2^{128} = 2^{215.2}$ memory accesses. The time complexity of Step 4 is $2^{223.2}$ memory accesses. As before, the number of pairs remaining after Step 4 is $2^{25.2}$ for a given guess of k_7 and $w_{6,SR(Col(2,3))}$.

Analysis of Steps 5–6 of the 7-Round Attack in the 8-Round Attack While the time complexity for Steps 3–4 of the 7-round attack is the same as in the 7-round attack, the time complexity of Step 5 of the 7-round attack is reduced considerably, since bytes $(3, 6)$ of w_5 are known to the attacker and there is no need to guess them. The time complexity of Step 5 is $2 \cdot 2^{25.2} \cdot 2^{192} = 2^{217.2}$ partial decryptions that are equivalent to $2^{215.2}$ encryptions. We note that there is no change in the number of pairs which are passed for analysis in Step 6.

The time complexity of Step 6 is also reduced due to the smaller amount of guessed subkey bits (i.e., Step 6 is repeated less times). Its time complexity is $2^{25.2} \cdot 2^{10} \cdot 2^{192} + 2^{31.8} \cdot 2^{189} = 2^{227.8}$ memory accesses.[6]

Another Improvement to the 8-Round Attack We observe that the 7-round attack can be repeated for three different sets of columns, i.e., for $w_{6,SR(Col(1,2))}$, $w_{6,SR(Col(1,3))}$, and $w_{6,SR(Col(2,3))}$ (with the appropriate changes to the pairs of active bytes in round 5). We note that as each 7-round attack is in itself a repetition of 4 times the BA attack, we obtain 12 parallel applications of the improved BA attack (for each guess of k_7). The results of these attacks are combined into suggestions for $w_{6,SR(Col(1,2,3))}$.

Let 2^{-x} be the probability that in a given run of the 7-round attack (for any of the 12 runs), a given guess of $k_{-1,SR^{-1}(Col(0))}$ remains. The probability that for a given guess of $w_{6,SR(Col(i,j))}$, this guess of $k_{-1,SR(Col(0))}$ remains after the four corresponding runs of the 7-round attack is 2^{-4x}. This statement can be read as "the expected number of remaining $w_{6,SR(Col(i,j))}$ guesses is 2^{64-4x} for each of the three pairs of values for (i, j)". After finishing the attacks, the attacker goes over all possible values of $k_{-1,SR^{-1}(Col(0))}$ and combines the lists to generate a list of candidate 96-bit

[6] This time complexity is the outcome of first using only $2^{110.2}$ chosen plaintexts. This approach eliminates 7 out of 8 keys. Then, the remaining ciphertexts are analyzed to achieve a stronger filtration of wrong keys.

guesses (for $w_{6,SR(Col(1,2,3))}$). The total number of expected values is 2^{96-4x} for a given guess of k_7 and $w_{1,SR(Col(3))}$, or a total of 2^{128-4x} for a given guess of k_7.

As in the 7-round attack, we set the probability that a wrong subkey guess of k_7 and w_6 passes Step 6 (i.e., there is at least one candidate value for $k_{-1,SR^{-1}(Col(0))}$ left) to 2^{-43}. Thus, out of the 2^{224} guesses of bytes in k_7 and w_6, about 2^{181} guesses remain. For each such guess, the attacker guesses the remaining 32 bits of w_6, uses the full value of w_6 to compute k_6, and then retrieves the secret key from the values of k_6 and k_7. Then, the suggested key can be checked by exhaustive key search. The time complexity of this step is $2^{181} \cdot 2^{32} = 2^{213}$ encryptions.

The above computation requires that $x = -6.25$, i.e., that for each of the 12 iterations of the attack, the number of pairs analyzed in Step 6 of the 7-round attack is $2^{24.1}$. This reduces the data complexity by a factor of $2^{1.1}$, in exchange of repeating each attack several times (but now each attack takes less time).

Summarizing the attack, the data complexity of the attack is $2^{111.1}$ chosen plaintexts, and the time complexity is $2^{227.8}$ memory accesses. The memory complexity is less than $2^{116.1}$ bytes of memory.

5 Improving and Extending the Phan Attack

In this section we improve the Phan attack on 7-round AES-192 and extend it to an attack on 8-round AES-256.

5.1 Improvement of the Phan Attack on 7-Round AES-192

The improvement of the Phan attack is based on the early abort technique and key schedule considerations, as well as on a reuse of the data. Our approach reduces the data and time complexities of the attack significantly.

Reducing the number of guessed key material We observe that the amount of guessed key bytes can be reduced for AES-192. This observation was made independently in [20] and was used there only to gain an immediate reduction in the time complexity of the Phan attack by factor 2^{24}. This follows the fact that the 16 subkeys bytes are needed by the attack: $k_{6,SR(Col(0,1))}$ and $k_{5,Col(0,1)}$ (rather than the entire k_6 and $k_{5,Col(0,1)}$ as done in the original version of the Phan attack). In the Phan attack, the attacker needs to guess these 16 subkey bytes. However, using the key schedule of AES-192, the amount of guessed bytes can be reduced:

1. The knowledge of $k_{6,(10,11)}$ yields the knowledge of $k_{5,9}$.
2. The knowledge of $k_{6,(10,13)}$ yields the knowledge of $k_{5,8}$.
3. The knowledge of $k_{6,(1,14)}$ yields the knowledge of $k_{5,12}$.

Hence, it is sufficient to guess 13 key bytes instead of 16.

Reducing the Time Complexity of Steps 3–4 of Phan's Attack The time complexity of Steps 3 and 4 of the attack can be further reduced. We first note that in the Phan attack the attacker can use four "output" differences for the impossible differential, i.e., requiring one of the four sets $SR(Col(0))$, $SR(Col(1))$, $SR(Col(2))$ or $SR(Col(3))$ of bytes to have a zero difference. Thus, the attacker repeats Steps 3–4 four times, each time under the assumption that the (shifted) column with zero difference is different. We shall describe the steps the attacker performs under the assumption that $x^{SR}_{4,SR(Col(0))}$ is zero.

In the improved attack, the attacker guesses the 80 bits of the key which compose $k_{6,SR(Col(0,1))}$ and $k_{5,Col(0)}$ (there are 2 bytes of $k_{5,Col(0)}$ which are known due to the key schedule). Then, all the remaining pairs are decrypted to find the differences in $\Delta x^{MC}_{4,SR^{-1}(Col(0))}$ (we note that the actual values of $x^{MC}_{4,SR^{-1}(Col(0,2,3))}$ are also known to the attacker). Under the assumption that the pair has a difference which satisfies Condition (2) of Proposition 1 for $x^{SR}_{4,SR(Col(0))}$, the attacker can immediately deduce the actual difference in each column of x^{MC}_4. This follows the fact that the MC operation is linear, and as the attacker knows for each column the byte with zero difference before the MC operation, and the difference in three bytes after the MC operation, she can determine the difference in the fourth byte of each column.

Once Δx^{MC}_4 is computed, the attacker knows the input differences to SubBytes of round 5 as well as the output differences (in all bytes), and thus, she can compute the exact inputs and outputs. Given an input and an output difference of the SubBytes operation, there is on average one pair of actual values that satisfies these differences.[7] Once the outputs are known, the attacker encrypts the values through Round 5 and retrieves the key bytes in $k_{5,Col(1)}$ suggested by this pair. Of course, if the suggested key disagrees with the known byte (recall that $k_{5,9}$ is known due to the key schedule) then the pair is discarded (for the specific 80-bit subkey guess). Otherwise, the pair is passed for further analysis in Steps 5–6 of the attack (for a specific guess of 104 bits of the key, 80 that were guessed and 24 that were computed).

The attacker starts with 2^{59} pairs, and for each 80-bit key value and shifted column, partially decrypts these pairs through three columns (two in one round, and then another one in the second round), and analyzes the fourth column. Hence, the time complexity of this step is roughly $2 \cdot 2^{59} \cdot 2^{80} \cdot 4 = 2^{142}$ 1-round encryptions, which are equivalent to $2^{139.2}$ 7-round encryptions.

Each of the $2^{59} \cdot 2^{80} \cdot 4 = 2^{141}$ partially decrypted pairs is expected to suggest one value for $k_{5,Col(1)}$. With probability $1 - 2^{-8}$ this value is discarded, and thus, for a given 104-bit guess, we expect $2^{141} \cdot 2^{-8}/2^{104} = 2^{29}$ pairs which are analyzed in Steps 5–6.

Optimizing Steps 5–6 of the Phan 7-Round Attack Step 5 of the Phan attack can be performed efficiently using the hash table method described in [5]. A short description of this technique can be found in Appendix A. For each guess of the

[7] More accurately, for randomly chosen input and output differences we expect that about half of the combinations are not possible, about half propose two actual values, and a small fraction suggest four values.

104 key bits in k_5 and k_6 there are 2^{29} pairs, each suggesting 2^{10} values of the key to be removed. The time complexity of this step is $2^{104} \cdot 2^{29} \cdot 2^{10} = 2^{143}$ memory accesses. Therefore, it is expected that all the wrong guesses of the 104 guessed bits are discarded, and the attacker is left with the right value of 104 subkey bits. The rest of the key can be easily found using an exhaustive key search.

The memory complexity of the attack also can be significantly improved. We observe that there is no need to store the discarded values of the 136 guessed subkey bits. Instead, for each 80-bit guess of $k_{6,SR(Col(0,1))}$ and $k_{5,Col(0)}$, the attacker repeats Steps 3–4 and stores for each value of $k_{5,Col(1)}$ the pairs which can be used for analysis.

Therefore, the amount of memory required for the attack is smaller, as we mainly need to store the data. The memory complexity of the attack is therefore roughly 2^{65} bytes of memory.

Finally, we slightly reduce the data complexity (and thus the time complexity) of the attack. We observe that in the Phan attack, a wrong subkey for k_6 has probability $2^{-152.7}$ to remain after Step 5.[8] As the time complexity of the attack is already above 2^{130} encryptions, even if more subkeys remain, the attack can be completed by exhaustive key search without affecting the overall time complexity.

We first note that out of $W[24\text{–}29]$ (whose knowledge is equivalent to the knowledge of the key) the attacker already knows 96 bits (for a given 104-bit guess). Thus, as long as Step 5 does not suggest more than 2^{34} values for the 104-bit key, the exhaustive key search phase of the attack would be faster than 2^{130}. Hence, we can reduce the data complexity by a factor of $2^{0.8}$, which in turn reduces the time complexity of the attack by a similar factor.

Summarizing the improved attack, the data complexity of the attack is $2^{91.2}$ chosen plaintexts, the time complexity is $2^{139.2}$ encryptions. The memory complexity is 2^{65} bytes of memory. For AES-256, as the attacker cannot exploit the key schedule, the data complexity is 2^{92} chosen plaintexts and the time complexity is 2^{163} memory accesses.

5.2 Extension of the Phan Attack to 8-Round AES-256

The trivial extension of the Phan attack to 8-round AES-256 (by guessing the last round subkey, partial decryption, and application of the 7-round attack) leads to an attack whose time complexity is significantly higher than 2^{256}. By using key schedule arguments, changing the used impossible differentials, using a more advanced attack algorithm, and reusing the data, we can present an attack on 8-round AES-256.

Our attack still maintains the above general approach, i.e., the attack is of the form:

- Encrypt 2^{60} structures of 2^{32} plaintexts each such that in every structure, the bytes of $SR^{-1}(Col(0))$ assume all the 2^{32} possible values and the rest of the bytes are fixed.

[8] In this step of the attack in [19] there are 2^{29} remaining pairs after the analysis, each discarding 2^{10} subkeys out of 2^{32} possible ones. Hence, about $2^{32} \cdot (1-2^{-22})^{2^{29}} = 2^{32}e^{-2^7} \approx 2^{-152.7}$ subkeys remain after each application of this step.

– For each value of k_7 determine the pairs that are to be analyzed with this subkey guess.
– Apply the 7-round attack with the selected pairs.

To perform the actual attacks, we need to make several modifications in the internal 7-round attack. The first change is to use an impossible differential in which the two active columns in x_5^O are (2,3) (rather than (0,1)). As a result, in the 7-round attack there is no need to guess bytes from k_5 since these key bytes are known due to the key schedule, given the knowledge of k_7. Thus, in each iteration of the 7-round attack only 8 bytes from k_6 are guessed.

As we show later, $2^{90.7}$ chosen plaintexts are sufficient for the attack. Hence, we describe the results while taking this figure into account. As the partial decryption takes only $2^{90.7} \cdot 2^{128} = 2^{218.7}$ 1-round decryptions, which is less than the time complexity of the remainder of the attack, we do not optimize this step (though a method similar to the one we suggest in Section 4.2 can be used).

Analysis of Steps 3–4 of the 7-Round Attack in the 8-Round Attack The most time consuming steps of the new 8-round attack are Steps 3–4 of the 7-round attack. This step is repeated 2^{128} times, where each time the attacker has to analyze $2^{57.7}$ pairs under 2^{64} possible subkey guesses. However, the time complexity of these steps can be further reduced.

We observe that if $\Delta x^{SR}_{4,SR(Col(0))}$ has a zero difference (recall that the attack is repeated four times, once for each possible shifted column), and if x_4^{MC} has eight bytes with a zero difference, there are $2^8 - 1$ possible differences in each of the columns of x_4^{MC}. As there is a difference only in two bytes of each column, we deduce that there are only $2^{16} \cdot (2^8 - 1) \approx 2^{24}$ different pairs of actual values in the two active bytes in the pair (rather than 2^{32}). Thus, for $x^{MC}_{4,SR^{-1}(Col(2,3))}$ there are 2^{96} possible pairs of intermediate encryption values which satisfy the required differences. As we are dealing with the actual values, we can partially encrypt these values through the SubBytes operation, and the following ShiftRows operation and MC (applied to Columns (2,3) of x_5^{SR}). Given the value of $k_{5,Col(2,3)}$, the attacker is able to further compute the actual values which enter the SubBytes of round 6, and its outputs.

Hence, Steps 3–4 can be performed in a slightly different manner: For each guess of k_7, the attacker computes $k_{5,Col(1,2,3)}$. Then, for each of the 2^{96} possible actual values of $x^{SB}_{5,SR^{-1}(Col(2,3))}$, the attacker computes the respective values of $x^{SR}_{6,SR(Col(2,3))}$. Then, the attacker partially decrypts all the ciphertexts through round 7, and obtains x_6^O. For each of the $2^{57.7}$ expected pairs with zero difference in $\Delta(x^O_{6,SR(Col(0,1))})$, the attacker then computes the equivalent key $w_{6,SR(Col(2,3))}$ suggested by the pair for each of the 2^{96} possible pairs of values of $x^{SR}_{6,SR(Col(2,3))}$. For each of the $2^{57.7}$ pairs, about $2^{96}/2^{64} = 2^{32}$ values of $w_{6,SR(Col(2,3))}$ are suggested.

An efficient implementation would therefore require only 2^{32} memory accesses for any remaining pair to retrieve this list of suggested $w_{6,SR(Col(2,3))}$ values. Then, each pair is added to the lists corresponding to the suggested values of $w_{6,SR(Col(2,3))}$. Steps 5 and 6 of the 7-round attack are repeated with these pairs (for a guess of k_7 and $w_{6,SR(Col(2,3))}$). We note that as there are $2^{57.7}$ pairs, each suggesting 2^{32} out of the 2^{64} keys, we expect each key to be suggested by $2^{25.7}$ pairs.

To further optimize the above attack, we note that the 2^{96} pairs of values of $x^{SR}_{6,SR(Col(2,3))}$ (computed from the 2^{96} possible pairs of values of $x^{MC}_{4,SR^{-1}(Col(2,3))}$) are not changed as long as the value of $k_{5,Col(2,3)}$ is not changed. Thus, an optimized implementation would try all possible values of k_7 in the order of the values of $k_{5,Col(2,3)}$. This reduces the total computational time of generating these 2^{96} pairs of values to about $2^{64} \cdot 2^{96}$ encryptions of two columns for two rounds, which is negligible with respect to the time complexity of the attack.

To conclude, this approach reduces the time complexity of this step to only $2^{128} \cdot 2^{57.7} \cdot 2^{32} \cdot 2 = 2^{218.7}$ memory accesses for a given shifted column with zero difference after round 4, or a total of $2^{220.7}$ memory accesses.

Reducing the Time Complexity of Step 5 of the 7-Round Attack As in the 7-round attack, Step 5 (of the 7-round attack) can be performed efficiently using the hash table method described in [5]. The time complexity of this step is $2^{192} \cdot 2^{37.7} = 2^{229.7}$ memory accesses.

The data complexity of the attack can be reduced as the attack can tolerate wrong keys which remain with probability higher than 2^{-152}. Given the time complexity of the rest of the attack, it is sufficient to set the probability at 2^{-43}, i.e., we expect that out of the 2^{192} guesses of bytes in k_7 and w_6, only 2^{149} guesses remain. For each such guess, the attacker guesses the remaining 64 bits of w_6, computes k_6 from w_6, recovers the secret key, and tests it using trial encryptions. The time complexity of this step is close to $2^{149} \cdot 2^{64} = 2^{213}$ encryptions, which is negligible with respect to the other steps of the attack.

Another observation is that it is possible to reuse the data and repeat the 7-round attack using different pairs of columns. The attack can be repeated with $Col(1,3)$ or $Col(1,2)$ in round 5 instead of $Col(2,3)$. Thus, the attacker repeats the above analysis, assuming that there is no difference in columns 1 and 2 (or 1 and 3) of x^O_5. The attack algorithm is similar (with slight modifications of the columns and the bytes involved). Each of these attacks retrieves a candidate value for $k_{6,SR(Col(1,2))}$ (or $k_{6,SR(Col(1,3))}$) in the inner 7-round attack. As these subkeys share bits, if a candidate value is discarded in one of the attacks, it is sufficient to deduce that this value cannot be true. Hence, it is sufficient to use $2^{89.1}$ chosen plaintexts. The time complexity does not increase despite the 3 repetitions of the attack, as the data analyzed each time is reduced by a similar factor.

Summarizing the 8-round attack, the data complexity of the attack is $2^{89.1}$ chosen plaintexts, the time complexity is $2^{229.7}$ memory accesses, and the memory complexity is about 2^{101} bytes of memory (used mostly to store the table of 2^{96} pairs).

6 Summary and Conclusions

In this paper we improved the previously known impossible differential attacks on 7-round AES and presented new attacks on 8-round AES-256. This research shed more light on the security of AES, especially on the way to exploit the relatively slow diffusion in the key schedule algorithm.

We presented two attacks on 7-round AES. The first attack (applicable to AES-192 and AES-256) has a data complexity of about $2^{91.2}$ chosen plaintexts and a time complexity of $2^{139.2}$ encryptions for AES-192 (or 2^{163} memory accesses for AES-256). The second attack requires $2^{112.2}$ chosen plaintexts, and has a running time of $2^{117.2}$ memory accesses (when attacking AES-128, a slightly higher complexities are needed for AES-192 and AES-256).

We also presented two attacks on 8-round AES-256. The first and better one requires $2^{89.1}$ chosen plaintexts and has a time complexity of $2^{229.7}$ memory accesses. The second one has a slightly smaller running time, in exchange for much more data ($2^{111.1}$ chosen plaintexts and $2^{224.3}$ memory accesses).

References

1. Behnam Bahrak, Mohammad Reza Aref, *A Novel Impossible Differential Cryptanalysis of AES*, proceedings of the Western European Workshop on Research in Cryptology 2007, Bochum, Germany, 2007.
2. Behnam Bahrak, Mohammad Reza Aref, *Impossible Differential Attack on Seven-Round AES-128*, IET Information Security journal, Vol. 2, Number 2, pp. 28–32, IET, 2008.
3. Eli Biham, Alex Biryukov, Adi Shamir, *Miss in the Middle Attacks on IDEA and Khufu*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 124–138, Springer-Verlag, 1999.
4. Eli Biham, Alex Biryukov, Adi Shamir, *Cryptanalysis of Skipjack Reduced to 31 Rounds*, Advances in Cryptology, proceedings of EUROCRYPT '99, Lecture Notes in Computer Science 1592, pp. 12–23, Springer-Verlag, 1999.
5. Eli Biham, Nathan Keller, *Cryptanalysis of Reduced Variants of Rijndael*, unpublished manuscript, 1999.
6. Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
7. Jie Chen, *personal communications*, August 2008.
8. Jie Chen, Yongzhuang Wei, Yupu Hu, *A New Method for Impossible Differential Cryptanalysis of 7-round Advanced Encryption Standard*, Proceedings of International Conference on Communications, Circuits and Systems Proceedings 2006, Vol. 3, pp. 1577-1579, IEEE, 2006.
9. Jie Chen, YuPu Hu, Yongzhuang Wei, *A New Method for Impossib le Differential cryptanalysis of 8-Round Adanced Encryption Standard*, Wuhan Univeristy Jouranl of National Sciences, vol. 11, number 6, pp. 1559-1562, 2006.
10. Jie Chen, YuPu Hu, YueYu Zhang, *Impossible differential cry ptanalysis of Advanced Encryption Standard*, Science in China Series F: Information Sciences, vol. 50, number 3, pp. 342–350, Springer-Verlag, 2007.
11. Jung Hee Cheon, MunJu Kim, Kwangjo Kim, Jung-Yeun Lee, SungWoo Kang, *Improved Impossible Differential Cryptanalysis of Rijndael and Crypton*, proceedings of Information Security and Cryptology — ICISC 2001, Lecture Notes in Computer Science 2288, pp. 39–49, Springer, 2002.
12. Joan Daemen, Vincent Rijmen, *AES Proposal: Rijndael*, NIST AES proposal, 1998.
13. Joan Daemen, Vincent Rijmen *The design of Rijndael: AES — the Advanced Encryption Standard*, Springer-Verlag, 2002.
14. Hüseyin Demirci, Ali Aydin Selçuk, *A Meet-in-the-Middle Attack on 8-Round AES*, proceedings of Fast Software Encryption 15, Lecture Notes in Computer Science 5806, pp. 116–126, Springer, 2008.

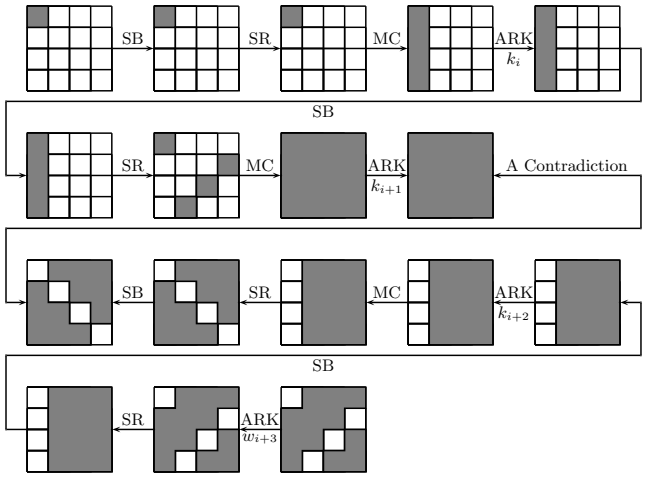
15. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, Doug Whiting, *Improved Cryptanalysis of Rijndael*, proceedings of Fast Software Encryption 7, Lecture Notes in Computer Science 1978, pp. 213–230, Springer-Verlag, 2001.
16. Henri Gilbert, Marine Minier, *A collision attack on 7 rounds of Rijndael*, proceedings of the Third AES Candidate Conference (AES3), pp. 230–241, New York, USA, 2000.
17. Jongsung Kim, Seokhie Hong, Bart Preneel, *Related-Key Rectangle Attacks on Reduced AES-192 and AES-256*, Proceedings of Fast Software Encryption 14, Lecture Notes in Computer Science 4593, pp. 225–241, Springer-Verlag, 2007.
18. Stefan Lucks, *Attacking Seven Rounds of Rijndael under 192-bit and 256-bit Keys*, proceedings of the Third AES Candidate Conference (AES3), pp. 215–229, New York, USA, 2000.
19. Raphael Chung-Wei Phan, *Impossible Differential Cryptanalysis of 7-round Advanced Encryption Standard (AES)*, Information Processing Letters, Vol. 91, Number 1, pp. 33-38, Elsevier, 2004.
20. Wentao Zhang, Wenling Wu, Dengguo Feng, *New Results on Impossible Differential Cryptanalysis of Reduced AES*, proceedings of ICISC 2007, Lecture Notes in Computer Science 4817, pp. 239–250, Springer-Verlag, 2007.
21. Wentao Zhang, Wenling Wu, Lei Zhang, Dengguo Feng, *Improved Related-Key Impossible Differential Attacks on Reduced-Round AES-192*, Proceedings of Selected Areas in Cryptography 2006, Lecture Notes in Computer Science 4356, pp. 15–27, Springer-Verlag, 2007.

A The Biham-Keller Technique for Efficiently Eliminating Wrong Subkeys

In [5] a technique for eliminating wrong subkey candidates in the round before the impossible differential is presented. The attack has two stages: In the precomputation stage, the attacker considers all possible pairs (z_1, z_2) of values of $x^{MC}_{0,Col(0)}$ that have difference in a single byte. For all these $2^{10} \cdot 2^{32} = 2^{42}$ pairs, the attacker computes the corresponding $x^{I}_{0,SR^{-1}(Col(0))}$ values (denoted by (w_1, w_2)), and stores in a table the values $(w_1 \oplus w_2, w_1)$. The table is sorted according to the $w_1 \oplus w_2$ values.

In the online stage, for each input pair, the attacker computes the XOR difference between the two plaintexts in the bytes $SR^{-1}(Col(0))$, and uses the table to detect the 2^{10} pairs of $x^{I}_{0,SR^{-1}(Col(0))}$ values corresponding to this difference. Since the AddRoundKey operation does not change the XOR difference between the two plaintexts, by XORing the 2^{10} corresponding w_1 values with one of the plaintexts, the attacker gets a list of 2^{10} values of $k_{-1,SR^{-1}(Col(0))}$ that lead the plaintext pair to the input difference of the impossible differential at the beginning of Round 1.

These values are then marked in a list of all the possible $k_{-1,SR^{-1}(Col(0))}$ values. Once all the values in the list are marked, the attacker concludes that a contradiction occurred, and discards the value of the corresponding subkeys in the rounds after the impossible differential (i.e., in k_5, k_6, and k_7).



A gray box stands for a non-zero difference in the byte, while a white box stands for a zero difference.

Fig. 4. An Example for a 4-Round Impossible Differential of AES

B The basic differential used in all the impossible differential attacks on the AES

In Figure 4 we give the structure of the impossible differentials used in all the impossible differential attacks on AES.