Exhausting Demirci-Selçuk Meet-in-the-Middle Attacks against Reduced-Round AES

Patrick Derbez[1] and Pierre-Alain Fouque[1,2]

[1] École Normale Supérieure, 45 Rue d'Ulm, 75005 Paris, France
[2] Université de Rennes 1, INRIA Rennes
{Patrick.Derbez,Pierre-Alain.Fouque}@ens.fr

Abstract. In this paper, we revisit Demirci and Selçuk meet-in-the-middle attacks on AES. We find a way to automatically model SPN block cipher and meet-in-the-middle attacks that allows to perform exhaustive search of this kind of attacks. This search uses the tool developed by Bouillaguet, Derbez and Fouque at CRYPTO 2011 as a subroutine to solve specific systems. We also take into account ideas introduced by Dunkelman, Keller and Shamir at ASIACRYPT 2010 which can be seen as a new tradeoff of the classical time/memory tradeoff used by Demirci and Selçuk. As a result, we automatically recover all the recent improved attacks of Derbez, Fouque and Jean on AES and we show new improved attacks against 8-rounds of AES-192 and AES-256.

1 Introduction

The AES encryption scheme [19] has been developed in the late nineties and has been specifically designed to resist against differential and linear cryptanalysis. Since 2008, the best attack for the 128-bit version was an impossible differential attacks by Lu *et al.* in [17] going back to a remark of Biham and Keller [1] improved by Bahrak and Aref in 2007. For the 192-bit and 256-bit versions, Demirci and Selçuk have described generalization of the Gilbert-Minier attack [16] which has also been discovered during the AES competition. During almost 10 years, there was no new cryptanalytic result and the first successful direction to analyze the AES encryption function comes from differential attacks in *the related-key setting* in 2009. This is a very powerful adversarial model in theory and it has recently been studied due to its applications in the analysis of hash functions. In this model, many other interesting results have been obtained by carefully studying the key schedule algorithms of AES-192 and AES-256 [4,3,2,5].

Despite important work on side-channel analysis on the AES, no real theoretical improvement on the first analysis performed during the AES competition [9,16,1,14] has been made. In this paper we turn our attention to the standard *single-key model* using meet-in-the-middle attack since these attacks are very efficient and are now the most efficient on all version of AES [11]. The first new theoretical result has been shown by Demirci and Selçuk at FSE 2008 using the old Meet-in-the-Middle cryptanalysis technique [10]. They improve the

Gilbert and Minier attack using meet-in-the-middle technique instead of collision ideas. These results at that time use a very small data complexity 2^{34} but require high precomputation and memory in 2^{216}. They need a hash table parameterized by 24 byte values. These attacks only work for the 256-bit and 192-bit versions thanks to a time/memory tradeoff which significantly increases the data and time complexity. They have been improved by Dunkelman *et al.* in [13] and more recently by Derbez *et al.* in [11]. Finally, recent biclique attacks [6] have been able to attack the full number of rounds of the AES at the price of using an exhaustive loop on all the key bits.

Meet-in-the-middle Attacks on AES. At Asiacrypt 2010, Dunkelman, Keller and Shamir improve Demirci and Selçuk attacks on `AES-192` and `AES-256` using many interesting new ideas in [13]. They introduce the idea of multisets, a clever differential enumeration technique and a remark on the `AES-192` key schedule to present attacks whose complexity is better than [10]. The main technique is the differential enumeration which allows to reduce the high memory complexity. This is mainly the bottleneck of the previous attacks with the precomputation phase. The attack can be seen as a new time/memory tradeoff, while Demirci and Selçuk one was very simple. Indeed, in this latter basic attack the memory is greater than the time. Consequently, they reduce the data in memory by repeting the attack as many times as the inverse of the probability of being in the table. Dunkelman *et al.* tradeoff uses a specific differential path to reduce the memory. This saving allows to consider a new attack on 7 rounds of `AES-128` with basically the same complexity as the impossible differential attack, which is the best attack on this version. They also improve the attacks on the two other versions. However, since these attacks rely on a differential technique, they require a huge amount of data. Basically, they show that the number of parameters can be reduced from 24 to 16 while the time complexity is constant. These attacks have been recently improved by Derbez *et al.* in [11] by showing that the table can be reduced since many sequences in the table are never reached. They exactly compute the size of the memory needed and show that the table can be described by 10 parameters. This leads to the best attack for 7 rounds of `AES-128` and also to the other versions.

Finally, Bouillaguet *et al.* in [7] study low data complexity attacks in reduce-round `AES` and in [8], some the authors build a computer-aided tool to look for the best meet-in-the-middle attacks in this model. A software has been developed allowing to solve linear systems of equations in $\mathbb{F}_{256}$ in the variables $x, S(x)$ where S is the `AES` S-box. This algorithm has been able to find attacks up to 5 rounds, but its complexity is exponential in the number of S-boxes. It is very versatile and has been used to solve systems for other cryptosystems such as the LEX stream cipher, the Pelican-MAC or fault attacks on `AES` [8,12].

Our Results. In this paper, we consider another direction to improve on Demirci and Selçuk (DS) attack using only meet-in-the-middle techniques. Here, we generalize DS attack using DS or DKS time/memory tradeoffs and we automatize the search of these attacks to find the best ones. We discover many efficient attacks and we also rediscovered the recent improved attacks on all the versions of AES presented in [11]. To perform this search, we use the tool of Bouillaguet, Derbez and Fouque, but only on the keyschedule equations instead of the system of equations describing the AES. These equations are sparse in the number of Sbox and consequently, the complexity of the search is very low. In particular, we have been able to improve the complexity on AES-192 and AES-256 by a factor 2^{32} and 2^{40} respectively as it is summarized in table 1. Finally, some of the attacks we discovered have a small data complexity such as the basic DS attack. This leads us to increase the number of rounds attacked using small data complexity as in [7,8]. For instance, we present on AES-128 an attack on up to 6 rounds using 256 data complexity and 2^{106} in time and memory whereas Bouillaguet *et al.* were able to find attack on 5 rounds with complexity 2^{120}. It is possible to extend this last attack to 7 rounds with a marginal improvement over exhaustive search. We refer the reader to table 1 for all the attacks.

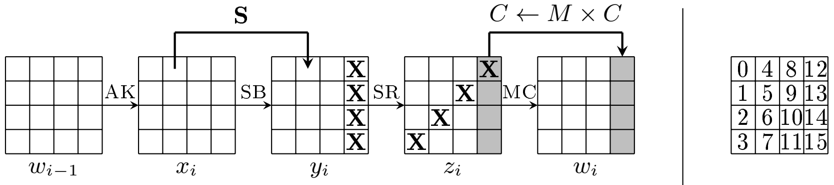
Organization of the Paper. In section 2, we describe the AES cipher and some properties useful to analyze its security for meet-in-the-middle techniques. Then, we present the previous attacks and ideas in section 3 before showing our ideas in section 4. In section 5, we discuss on the results and describe some of our new attacks requiring at most 2^{32} chosen plaintexts. The section 6 is dedicated to the differential enumeration technique introduced by Dunkelman *et al.* and contains the description of new attacks on AES-192 requiring 2^{104} data, 2^{138} in memory and 2^{140} in time and on AES-256 requiring 2^{103} in data, 2^{140} in memory and 2^{156} in time.

2 AES and Observations

2.1 Description of the AES

The Advanced Encryption Standard [19] is a Substitution-Permutation Network that can be instantiated using three different key sizes: 128, 192, and 256. The 128-bit plaintext initializes the internal state viewed as a 4×4 matrix of bytes as values in the finite field $\mathbb{F}_{256}$, which is defined using the irreducible polynomial $x^8+x^4+x^3+x+1$ over $\mathbb{F}_2$. Depending on the version of the AES, N_r rounds are applied to that state: $N_r = 10$ for AES-128, $N_r = 12$ for AES-192 and $N_r = 14$ for AES-256. Each of the N_r AES round (Figure 1) applies four operations to the state matrix (except in the last round where the **MixColumns** operation is missing):

- **AddRoundKey** (AK) adds a 128-bit subkey to the state.
- **SubBytes** (SB) applies the same 8-bit to 8-bit invertible S-Box S 16 times in parallel on each byte of the state,
- **ShiftRows** (SR) shifts the i-th row left by i positions,
- **MixColumns** (MC) replaces each of the four column C of the state by $M \times C$ where M is a constant 4×4 maximum distance separable matrix over $\mathbb{F}_{256}$,



0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

An AES round applies MC ∘ SR ∘ SB ∘ AK to the state.

Ordering.

Figure 1: Description of one AES round and the ordering of bytes in an internal state.

After the N_r-th round has been applied, a final subkey is added to the internal state to produce the ciphertext. We refer to the original publication [19] for the key expansion algorithms.

Notations. In this paper, we count the AES rounds from 0 and we refer to a particular byte of an internal state x by $x[i]$, as depicted in Figure 1. Moreover, in the ith round, we denote the internal state after **AddRoundKey** by x_i, after **SubBytes** by y_i, after **ShiftRows** by z_i and after **MixColumns** by w_i. To refer to the difference in a state x, we use the notation Δx. The first added subkey is the master key k_{-1}, and the one added after round i is denoted k_i.

In some cases, we are interested in swapping the order of the **MixColumns** and **AddRoundKey** operations. 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:

$$u_i = MC^{-1}(k_i) = \begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix} \times k_i$$

2.2 Observations on the Structure of AES

In this section we recall two well-known observations on the structure of AES, that will be used later in our attacks. We first consider the propagation of differences through **SubBytes** layer.

Property 1 (the **SubBytes** *property).* Consider pairs $(\alpha \neq 0, \beta)$ of input/output differences for a single S-box in the **SubBytes** operation. For 129/256 of such pairs, the differential transition is impossible, i.e., there is no pair (x, y) such that $x \oplus y = \alpha$ and $S(x) \oplus S(y) = \beta$. For 126/256 of the pairs (α, β), there exist two ordered pairs (x, y) such that $x \oplus y = \alpha$ and $S(x) \oplus S(y) = \beta$, and for the remaining 1/256 of the pairs (α, β) there exist four ordered pairs (x, y) that satisfy the input/output differences. Moreover, the pairs (x, y) of input values corresponding to a given difference pattern (α, β) can be found instantly from the difference distribution table of the Sbox.

Property 1 means that given the input and output difference of an S-box, we can find in constant time the possible absolute values of the input, and there is only a single one on average.

The second observation is a necessary and sufficient condition for a matrix to be MDS applied to the matrix MC used in the **MixColumns** operation.

*Property 2 (***MixColumns** *property).* Consider a pair (a, b) of 4-byte vectors, such that $a = MC(b)$, *i.e.* the input and the output of a **MixColumns** operation applied to one column. Denote $a = (a_0, a_1, a_2, a_3)$ and $b = (b_0, b_1, b_2, b_3)$ where a_i and b_j are elements of $\mathbb{F}_{256}$. Then there is no equation involving less than five bytes and for each choice of five bytes among the eight bytes $(a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3)$ there is a linear equation between them.

Finally, in our attacks we consider the encryption of structured sets of 256 plaintexts in which one active byte takes each one of the 256 possible values exactly once, and each one of the other 15 bytes is a (possibly different) constant. Such a structure is called a δ-set.

3 Related Results from Previous Work

In this section, we remind Demirci and Selçuk attack together with its improvements which are the main results used in our attack. We refer the reader to [10] and [13] for details.

3.1 The Demirci and Selçuk attack

At FSE 2008, Demirci and Selçuk described the following 4-round property for AES.

Property 3. Consider the encryption of a δ-set through four full AES rounds. For each of the 16 bytes of the state, the ordered sequence of 256 values of that byte in the corresponding ciphertexts is fully determined by just 25 byte parameters. Consequently, for any fixed byte position, there are at most $(2^8)^{25} = 2^{200}$ possible sequences when we consider all the possible choices of keys and δ-sets (out of the $(2^8)^{256} = 2^{2048}$ theoretically possible 256-byte sequences).

The 25 parameters are intermediate state bytes for any message of the δ-set and their positions depend on the active byte of the δ-set and on which byte we want to build values. As depicted on Figure 2, if there are both at position 0 then the 25 parameters are the first column of x_{i+1}, the full state x_{i+2}, the first column of z_{i+3} and $x_{i+4}[0]$. Indeed, if those bytes are known for one of the messages, we can compute the value of $x_{i+4}[0]$ for each message of the δ-set as follows:

1. Knowing the 256 differences in the full state z_i we can compute the 256 differences in the full state x_{i+1} because $\Delta x_{j+1} = \mathrm{MC}.\Delta z_j$ for any round number j, where MC is the matrix used in the **MixColumns** operation.

2. Knowing the value of the first column of x_{i+1} for one message we can now compute the value of this column for all messages.
3. Then we apply the Sbox on those bytes and get the value of $z_{i+1}[0]$, $z_{i+1}[7]$, $z_{i+1}[10]$ and $z_{i+1}[13]$ for each message of the δ-set.
4. The differences are null in all the other bytes of z_{i+1} so we know the 256 differences in the full state z_{i+1}.
5. In the same way we obtain the 256 differences in the full state z_{i+2} and then in the first column of z_{i+3} to finally compute the 256 values of $x_{i+4}[0]$

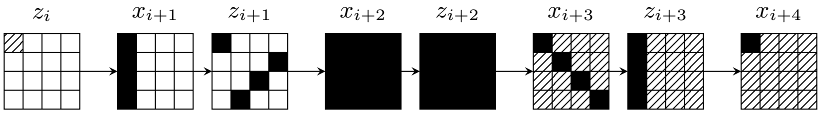


Figure 2: 4 AES-rounds. The 25 black bytes are the parameters of Property 3. Hatched bytes play no role. The differences are null in white squares

They first use this property to mount a basic meet-in-the-middle attack on 7 rounds AES-256 depicted on Figure 3 and its procedure is roughly as follows:

- **Preprocessing phase:** Compute all the 2^{200} possible sequences according to Property 3, and store them in a hash table.
- **Online phase:**
  1. Ask for a structure of 2^{32} chosen plaintexts such that the main *diagonal* can take the 2^{32} possible values and the remaining bytes are constant.
  2. Choose one plaintext and guess the first column of its intermediate state z_0 and byte $z_1[0]$.
  3. For each of the 255 non-zero values of $\varDelta z_1$ compute the corresponding difference in the plaintext using the guessed bytes.
  4. Order the obtained δ-set according to the value of the state byte $z_1[0]$.
  5. Guess the first column of x_6 and the byte $x_5[0]$ for one of the message and deduce those state bytes for the 256 ciphertexts.
  6. Build the sequence and check whether it exists in the hash table. If not, discard the guess.

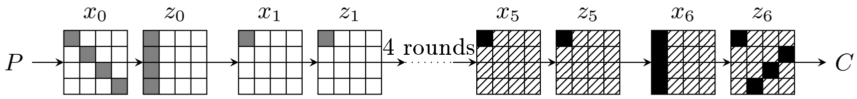


Figure 3: Online phase of Demirci and Selçuk attack. $\mathcal{B}_{on}$ is composed by gray and black bytes. Gray bytes are used to identify a δ-set and to order it. Black bytes are used to build the sequence from ciphertexts. Hatched bytes play no role. The differences are null in white squares.

Note that the parameters of both the online and offline phases are state bytes which we shall refer in the sequel as respectively $\mathcal{B}_{on}$ and $\mathcal{B}_{off}$. The complexity of the attack depends directly on how many values can assume those state bytes and how fast can we enumerate them. Indeed, bytes of $\mathcal{B}_{off}$ (resp. $\mathcal{B}_{on} \cup P \cup C$) are related by the AES equations and thus lead to the knowledge of some linear combinations of the (sub)keys bytes. Then it may exist some relations derived from the key-schedule between them, allowing to reduce the number of assumed values. In the sequel, we will denote by $\mathcal{K}_{off}$ (resp. $\mathcal{K}_{on}$) the vector space generated from these linear combinations. For instance, in the case of the described attack and if the last **MixColumns** is omitted,

- $\{k_{-1}[0,5,10,15], k_0[0], u_5[0], k_6[0,7,10,13]\}$ is a basis of $\mathcal{K}_{on}$,
- $\{u_1[0], u_2[0,7,10,13], k_3[0,5,10,15], k_4[0]\}$ is a basis of $\mathcal{K}_{off}$.

All in all, this attack has a data complexity of 2^{32} chosen plaintexts, a time complexity of $2^{80} \times 2^{8}$ partial encryptions/decryptions, and a memory requirement of 2^{200} 256-byte sequences. The memory complexity of this attack is too high to apply it on the 128 and 192-bit versions. But its time complexity is low enough to mount an attack from it on 8 rounds AES-256. This is done by fully guessing the last subkey, decrypting the last round and applying the 7-round attack, which increases the time complexity by a factor 2^{128}.

3.2 Previous improvements of the original attack

We summarize the main improvements to the original attack of Demirci and Selçuk.

Difference Instead of Value. Demirci and Selçuk showed that the number of parameters can be reduced to 24 in Property 3 by considering the sequence of the differences instead of values because in that case $x_{i+4}[0]$ is not needed.

Data/Time/Memory Trade-Off. They also showed that one can do a classical trade-off by storing in the hash table only a fraction of the possible sequences. Then the attacker has to repeat the online phase many times to compensate the probability of failure if the sequence is not present in the table which will increase the data and time complexities. In other word, if the attack has a complexity (D, T, M) (D for the data, T for the time complexity of the online phase and M for the memory) then it is possible to modify it to reach a complexity equal to $(D \times N, T \times N, M/N)$ for any positive N such that $D \times N$ is smaller than the size of the codebook. This trade-off allows to adapt the attack on 7 rounds of AES-256 to attack the 192-bit version.

Data Recycling. The structure of 2^{32} plaintexts used in the attack contains 2^{24} δ-sets. Thus the data may be reused 2^{24} times in the Data/Time/Memory Trade-Off.

Time/Memory Trade-Off. Kara observed that considering the sequence of the differences instead of values allows to remove $x_5[0]$ from $\mathcal{B}_{off}$ (as Demirci and Selçuk did) or from $\mathcal{B}_{on}$.

Multiset. A multiset is an unordered set in which elements can occur many times. Dunkelman *et al.* introduce them to replace the functional concept used in the DS attack and propose to store in the hash table unordered sequences of 256 bytes instead of ordered sequences. Moreover, they claim that a multiset still contains enough information to make the attack possible. Indeed they showed that given two random functions $f, g : \mathbb{F}_{256} \longrightarrow \mathbb{F}_{256}$, the multisets $[f(0), \dots, f(255)]$ and $[g(0), \dots, g(255)]$ are equal with a probability smaller than $2^{-467,6}$. Combined to the fact that the Sbox is a bijection, the main gain is to remove $z_1[0]$ from $\mathcal{B}_{on}$ since it was used only to ordered the δ-set, and thus the time complexity is decreased by a factor 2^8. Finally, we note that a multiset contains about 512 bits of information and its representation can be easily compressed into 512 bits of space while an ordered sequence needs $256 \times 8 = 2048$ bits.

Differential Enumeration. In [13], Dunkelman *et al.* introduce a more sophisticated trade-off which reduces the memory without increasing the time complexity. The main idea is to add restrictions on the parameters used to build the table such that those restrictions can be checked (at least partially) during the online phase. More precisely, they impose that sequences stored come from a δ-set containing a message m which belongs to a pair (m, m') that follows a well-chosen differential path. Then the attacker first focus on finding such pair before to identify a δ-set and build the sequence. Section 6 is dedicated to this technique.

4 Generalization of the Demirci and Selçuk Attack

The basic attack of Demirci and Selçuk requires a huge memory and a relatively small time complexity. The classical data/time/memory trade-off allows to *balance* these complexities by increasing the data complexity and randomizing the attack. In this section we present new improvements to reduce the data complexity increase which leads to almost 2^{16} variants of the Demirci and Selçuk attack and we explain how to find the best ones between them.

4.1 New improvements of the original attack

In this section we summarized our new improvements that allow us to reduce the increase of the data complexity and, sometimes, to keep the deterministic nature of the original attack.

Difference Instead of Value. The sequences stored in the table have the form $[f(0)+f(0), \dots, f(0)+f(255)]$ where f is a function that maps the value of $z_i[0]$ to the value of $x_{i+4}[0] + k_{i+3}[0]$. But, as shown Section 3.1, the procedure used to build the table produces functions that map the value of $\Delta z_i[0]$ to the value of $\Delta x_{i+4}[0]$ and then the only effect of mapping the value of $z_i[0]$ is to set the value of the subkey byte $u_i[0]$ (i.e. $u_i[0] \in \mathcal{K}_{off}$). In another hand, if we store in the table sequences of the form $[f(0), \dots, f(255)]$ where f is a function that maps the value of $\Delta z_i[0]$ to the value of $\Delta x_{i+4}[0]$, then each δ-set can be ordered in 256 ways, saving data in the classical data/time/memory trade-off described Section 3.2. Furthermore, in the case of a δ-set encryption, each byte of the first columns of x_{i+1} assumes the 256 values. As a consequence, to set one of those bytes to 0 when building the hash table can be compensated by trying the 256 orders of a δ-set without randomizing the attack.

Multiset. Note that, given a sequence of 256 bytes $b_0, \ldots, b_{255}$, $b_i = b_j$ implies that the multisets $[b_i + b_0, \ldots, b_i + b_{255}]$ and $[b_j + b_0, \ldots, b_j + b_{255}]$ are equal too. But Dunkelman et al. shown that given a random function $f : \mathbb{F}_{256} \longrightarrow \mathbb{F}_{256}$, the multiset $[f(0) + f(1), \ldots, f(0) + f(255)]$ contains on average 162 different values out of 256. Thus we conclude that a δ-set can be reused $162 \approx 2^{7,34}$ times on average. This remark holds on for the multisets stored in the hash table during the precompution phase and so the memory requirement must be corrected by a factor $2^{-0,66}$.

Time/Memory Trade-Off. To improve the attack of Demirci and Selçuk our idea is to store in the sequences the 256 differences in a linear combinations of bytes of x_5 instead of the 256 differences in a byte of x_5. Thanks to Property 2, minimal equations involving Δz_i and Δx_{i+1} contains exactly 5 variables such that k are on a column c of Δz_i and $5 - k$ are on the column c of Δx_{i+1}, with $1 \leq k \leq 4$ for any round number i. We emphase that Demirci and Selçuk only consider cases $k = 1$ and $k = 4$. The size of the set $\mathcal{B}_{on}$ (resp. $\mathcal{B}_{off}$) is determined by k and it decreases (resp. increases) when k is increased. Thus we can trade time by memory and vice-versa without affecting the data complexity. Furthermore, contrary to the other data/time/memory trade-offs, the attack need not to be randomized. Attacks taking advantage of this trade-off are described Section 5.2 and 5.4.

New Data/Time/Memory Trade-Off. The idea of the previous trade-off can be applied to the δ-set. Instead of considering sets of 256 plaintexts such that one byte assumes the 256 values and the others are constant, we consider set of 256 plaintexts such that exactly 5 bytes of z_i and x_{i+1} are active. We still call such a set a δ-set. The consequences on the attack are the same as the previous trade-off but it now affects the size of the structure needed and bytes of z_i must be guessed in the online phase despite the use of unordered sequences. An attack taking advantage of this trade-off is described Section 5.3.

4.2 Finding the best attack

Once the round-reduced AES is split into three parts, the new improvements allow to mount $(4 \times \binom{8}{5})^2 \approx 2^{15.6}$ different attacks but there are only $(4 \times (\binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4}))^2 \approx 2^{11.8}$ possible sets $\mathcal{B}_{on}$ (resp. $\mathcal{B}_{off}$) to study. To exhaust all of them and find the best attacks we decide to automatize the search. Thus for each set we need to answer to the two following questions:

- How many values can assume those state bytes?
- How fast can we enumerate them?

A priori, this is not an easy task because S-boxes are involved in the keyschedule. To perform it we used the tool developed in [8], originally designed to find the best solver for an AES-like system of equations among a particular class of solvers based on the meet-in-the-middle technique.

Algorithm 1: OriginalTool

```
Data: System of equations $E$ in variables $X$ involving some S-boxes.
Result: An optimal algorithm to enumerate all the solutions of $E$ with
        predictable time and memory complexities.
```

The problem we seek to solve is very close to the problem solved by this tool but is still different and so we have slightly tweaked it.

Algorithm 2: TweakedTool (naive implementation)

```
Data: System of equations $E$ in variables $X$ involving some S-boxes and
      a subset $Y \subseteq X$.
Result: A list of optimal algorithms to enumerate all the possible values
        of $Y$ according to the system of equations $E$ with predictable
        time and memory complexities.

$L \leftarrow \emptyset$;
foreach $Y \subseteq Z \subseteq X$ do
    $F \leftarrow$ the biggest subspace of $E$ in variables $Z$;
    $\mathcal{A} \leftarrow \mathsf{OriginalTool}(F)$;
    $L \leftarrow$ best algorithms from $L \cup \{A\}$;
end
return $L$
```

The output of our tweaked tool is a list because the number of possible values of Y enumerated by considered algorithms is not necessary constant and if an algorithm is slower than an other but finds less possible values for Y than it then both of them must be studied. Note that the tweaked tool can be applied directly to the set $\mathcal{B}_{off}$ (resp. $\mathcal{B}_{on}$) and the system of equations describing the AES but it is faster to apply it on a basis of $\mathcal{K}_{off}$ (resp. $\mathcal{K}_{on}$) and the keyschedule equations since the complexity of the original tool is exponential in the number of S-box.

Finally we were able to perform an exhaustive search over all the parameters for all round-reduced versions of AES for the three key lengths in less than an hour on a personal computer.

5 Results

In this section we present the results obtained by exhausting the variants of the attack of Demirci and Selçuk. We give an overview of the complexities reached and describe three new attacks requiring at most 2^{32} chosen plaintexts and minimizing the maximum between the time complexity (counted in AES encryption) and the memory complexity (counted in 128-bit block).

5.1 Overview of the results

Some of our best results on 7 and 8 rounds are summarized on Figures 4 and 5. They give the ($\log_{256}$ of) data complexity reached as a function of the number of guess to perform in the online phase and in the offline phase. A gray cell means that the corresponding attack is deterministic while the other attacks are obtained by applying the classical data/time/memory trade-off.

number of guess in the offline phase

number of guess in the online phase

	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
10																	4
11																4	
12															4		
13													4				
14												4					3
15											4					3	
16										4							
17									4					3			
18								4					3				2
19							4					3					1
20						4			8		3		1				
21					4			3				1					
22		4					3	1									
23	4				3		1										

last mixcolumns performed

number of guess in the offline phase

number of guess in the online phase

	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
10																	4
11																4	
12															4		
13														4			
14													4				
15												4					3
16											4					3	
17										4							
18									4					3			
19								4					3				1
20							4					3	1				
21						4		3				1					
22		4					3	1									
23	4				3		1										

last mixcolumns omitted

Figure 4: Best variants on 7 rounds AES-192.

offline

online

	19	20	21	22	23
22					4
23	4				

AES-192

number of guess in the offline phase

number of guess in the online phase

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
17																9
18															10	4
19														11	4	
20													12	4		
21												13	4			
22											14	4				
23										15	4				3	
24									4					3		
25								4					3			
26							4					3			2	
27						4				3				2		
28				4					3				2		1	
29			4					3						1		
30		4			8		3			2						
31	4				3				2							

AES-256

Figure 5: Best variants on 8 rounds.

We observe that almost all the best attacks work with only 2^{32} chosen-plaintexts. For comparison, to reach balanced complexities on seven rounds from the original attack by using the classical data/time/memory trade-off, the amount of data needed will be approximately 2^{71} chosen plaintexts. Furthermore, we have been able to increase by one the number of rounds attacked with 2^{32} chosen-plaintexts for the three key length but with time and memory complexities very close to the natural bound of the exhaustive search. We also obtained competitive results in the very low data complexity league with, for instance, attacks on 8 rounds of AES-256 requiring only 2^8 chosen plaintexts.

5.2 Attack on six rounds AES-128 with 2^8 chosen-plaintexts.

If the data available is limited to 2^8 chosen-plaintexts, the best attack found is based on the attack depicted on Figure 6 and the meet-in-the-middle is performed on the equation

$$03.\Delta z_3[8] + \Delta z_3[9] = 07.\Delta x_4[8] + 07.\Delta x_4[9] + 02.\Delta x_4[11].$$

Let be $e_{in} = 03.z_3[8] + z_3[9]$ and $e_{out} = 07.x_4[8] + 07.x_4[9] + 02.x_4[11]$.

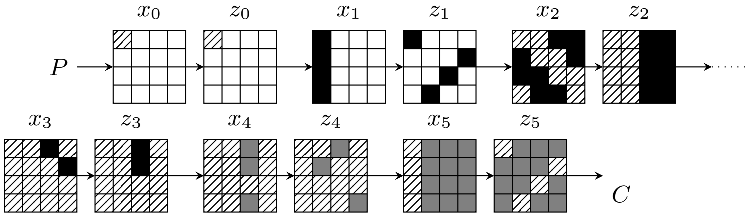


Figure 6: Attack on 6 AES rounds. Bytes of $\mathcal{B}_{off}$ are in black. Bytes of $\mathcal{B}_{on}$ are in gray. Hatched bytes play no role. The differences are null in white squares

The bytes of $\mathcal{B}_{off}$ are the first column of x_1, the two last columns of z_2, and bytes 8 and 9 of z_3. They can assume $2^{8\times14}$ different values and so the memory requirement is $2^{112-0,66} = 2^{111,34}$ multisets on average according to the remark made in Section 4.1.

As the S-box is a bijection and as we consider a δ-set in which only one byte is active, we do not need to guess $x_0[0]$ in order to identify the corresponding set of 256 plaintexts to build the multiset. As a consequence, the bytes of $\mathcal{B}_{on}$ are the entire state x_5 except the first column, and the third column of x_4 except byte 10. Thanks to the keyschedule equations, they can take only $2^{8\times12}$ values instead of $2^{8\times15}$ since we have the three equations $u_4[5] = u_5[1]+u_5[5]$, $u_4[8] = u_5[4]+u_5[8]$ and $u_4[15] = u_5[11] + u_5[15]$.

All in all this leads to the following attack:

- **Preprocessing phase:**
  1. Set $\Delta_i z_0[0]$ to i for $0 \leq i \leq 255$. Then $\Delta_i z_0$ is known since the other differences are null.

2. Guess $x_1[0..3]$ (for one of the 256 messages) and use $\Delta_i z_0$ to compute $\Delta_i z_1[0]$, $\Delta_i z_1[7]$, $\Delta_i z_1[10]$ and $\Delta_i z_1[13]$. Then $\Delta_i z_1$ is known since the other differences are null.
3. Guess bytes 1, 2, 6, 7, 8, 11, 12 and 13 of x_2. Use them with $\Delta_i z_1$ to compute $\Delta_i z_2[8..15]$.
4. Guess $x_3[8]$ then compute $\Delta_i z_3[8]$ using $\Delta_i z_2[8..11]$.
5. Guess $x_3[13]$ then compute $\Delta_i z_3[9]$ using $\Delta_i z_2[12..15]$.
6. Compute the multiset $[\Delta_0 e_{in}, \dots, \Delta_{255} e_{in}]$ and store it in a hash table (if it was not already in it).

- **Online phase:**
  1. Ask for a structure of 256 plaintexts such that byte 0 assume the 256 possible values and others bytes are constant.
  2. Choose one of them to be the one from which difference will be computed.
  3. Guess bytes 1, 2, 4, 5, 8, 11, 14 and 15 of u_5. Compute $u_4[5]$ and $u_4[8]$ and then partially decrypt the ciphertexts to obtain $\Delta_i x_4[8]$ and $\Delta_i x_4[9]$ for $0 \leq i \leq 255$.
  4. Guess bytes 3, 6 and 9 of u_5, and continue to partially decrypt the ciphertexts.
  5. Guess byte 12 of u_5. Compute $u_4[15]$ and then partially decrypt the ciphertexts to obtain $\Delta_i x_4[11]$.
  6. Build the multiset $[\Delta_0 e_{out}, \dots, \Delta_{255} e_{out}]$ and check whether the multiset exists in the hash table. If not, discard the key guess.

Finally, the time complexity is equivalent to $2 \times 2^{-6} \times 2^8 \times 2^{96} = 2^{99}$ encryptions and the memory requirement is $2^{113,34}$ AES-blocks. The probability for a wrong guess to succeed is approximatively $2^{111,34} \times 2^{-467,6} = 2^{-356,26}$ and, as we try 2^{96} key guess, we expect that only the right value remains after the last step.

Trade-Off. Since the memory is higher than the time complexity, the data/time/memory trade-off presented Section 3.2 is possible. This leads to an attack using 2^8 chosen plaintexts (as the data is reused $2^{7,17}$ times), with a time complexity equivalent to $2^{106,17}$ encryptions and requiring $2^{106,17}$ 128-bit blocks.

Key Recovery. This attack retrieves the right value of u_5 except on bytes 0, 7, 10 and 13 and so can easily be turned into a key-recovery attack. The attacker guesses the four missing bytes of u_5 to retrieve the master key and try it. This step has a negligible complexity compared to the previous one.

5.3 Attack on 7 rounds AES-256 with 2^{16} chosen-plaintexts

The best attack on seven rounds AES−256 with 2^{16} chosen-plaintexts is depicted on Figure 7.

The bytes of $\mathcal{B}_{off}$ are bytes 0,2 and 3 of x_1, the three first columns of x_2 and the third column of z_3. The bytes of $\mathcal{B}_{on}$ are bytes 0 and 15 of x_0, the entire state x_6, the second column of x_5 and byte 9 of x_4. The number of values assumed by the bytes of $\mathcal{B}_{on}$ is reduced by a factor 2^8 using the equation $u_4[5] = u_6[1] + u_6[5]$.

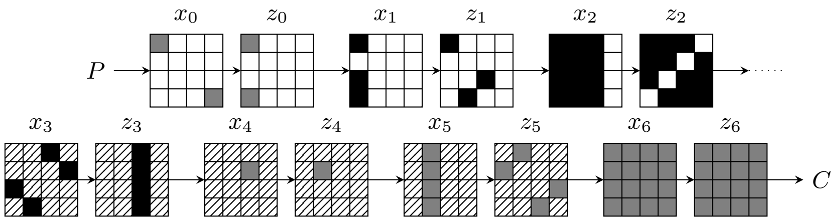


Figure 7: Attack on 7 AES rounds (key length : 256 bits). Bytes of $\mathcal{B}_{off}$ are in black. Bytes of $\mathcal{B}_{on}$ are in gray. Hatched bytes play no role. The differences are null in white squares

The time complexity is equivalent to 2^{178} encryptions and the memory is $2^{153,34}$ AES-blocks.

Key Recovery. This attack can easily be turned into a key-recovery attack without increasing the complexity since only 12 key bytes are sufficient to recover the master key.

5.4 Attack on 7 rounds AES-192 with 2^{32} chosen-plaintexts

The best attack on seven rounds AES−192 with 2^{32} chosen-plaintexts is depicted on Figure 8.

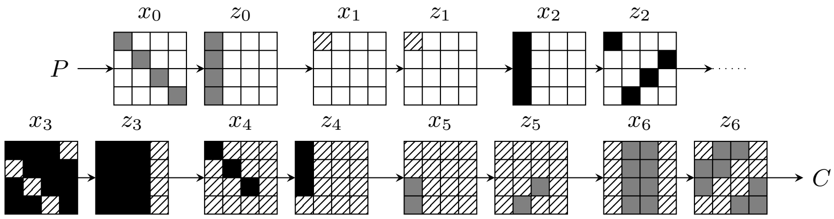


Figure 8: Attack on 7 AES rounds (key length : 192 bits). Bytes of $\mathcal{B}_{off}$ are in black. Bytes of $\mathcal{B}_{on}$ are in gray. Hatched bytes play no role. The differences are null in white squares

The bytes of $\mathcal{B}_{off}$ are the first column of x_2, the three first columns of z_3, and bytes 0, 1 and 2 of z_4. The bytes of $\mathcal{B}_{on}$ are the first column of z_0, the second and third columns of x_6 and bytes 2 and 3 of x_5. Thanks to the keyschedule equations, we can reduce the number of possible values assumed by them by a factor 2^8 since $u_5[7] = u_6[11] + u_6[15]$. The time complexity is equivalent to 2^{106} encryptions and the memory requirement is $2^{153,34}$ AES-blocks.

Trade-Off. Applying the classical data/time/memory trade-off leads to an attack using 2^{32} chosen plaintexts, with a time complexity equivalent to $2^{129,67}$

encryptions and a memory requirement of $2^{129,67}$ AES-blocks. Note that the data complexity remains 2^{32} because the structure may be divided into 2^{24} δ-sets and each of them may be reused $2^{7,34}$ times on average. **Key Recovery.**

This attack can easily be turned into a key-recovery attack without increasing the complexity since only 15 key bytes are sufficient to recover the master key.

6 The Differential Enumeration Technique

We present here our results using the differential enumeration technique first introduced by Dunkelman *et al.* in [13] and improved by Derbez *et al.* in [11]. We explain how this technique works by describing a new attack on 8 rounds and then we give an overview of our results.

6.1 Attack on 8 rounds AES-192

Without restriction on data, the best attack on eight rounds AES−192 begins by considering the attack depicted on Figure 9.

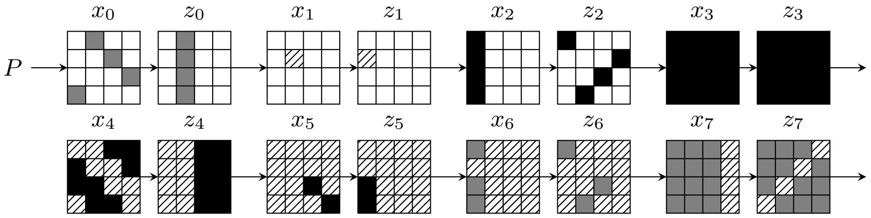


Figure 9: Attack on 8 AES rounds. Bytes of $\mathcal{B}_{off}$ are in black. Bytes of $\mathcal{B}_{on}$ are in gray. Hatched bytes play no role. The differences are null in white squares

The bytes of $\mathcal{B}_{off}$ are the first column of x_2, the entire state x_3, the two last columns of z_4 and bytes 2 and 3 of z_5. The bytes of $\mathcal{B}_{on}$ are the second column of z_0, the three first columns of x_7, and the first column of x_6 excepted byte 1. Thanks to Property **??**, they take only $2^{8\times 17} = 2^{136}$ values because $u_6[0] = u_7[4] + u_7[8]$ and $u_6[7] = u_7[11] + u_7[15]$. Finally, the time complexity is equivalent to 2^{138} encryptions and the memory requirement is $2^{241,34}$ AES-blocks.

Differential Enumeration. The idea of Dunkelman *et al.* is to store in the hash table only the multisets built from a δ-set containing a message m that belongs to a pair (m, m') following a well-chosen differential path. In our case this is the truncated differential $4 \to 1 \to 4 \to 16 \to 8 \to 2 \to 3 \to 12$ depicted on Figure 10. Then the bytes of $\mathcal{B}_{off}$ can take only $2^{16\times 8}$ values for such a pair. Indeed, if we guess the differences in circled bytes then we obtain the difference before and after the S-box for each bytes of $\mathcal{B}_{off}$ and thus we can derive their absolute value thanks to Property 1. As a consequence, the memory requirement

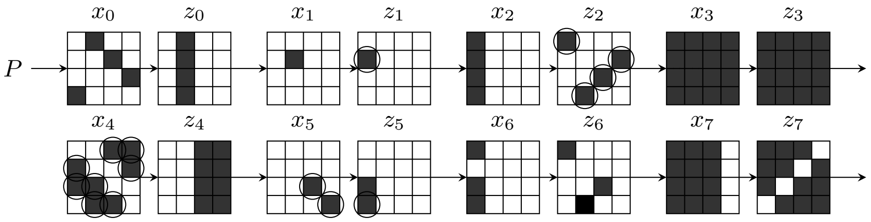


Figure 10: Differential characteristic on 8 AES rounds. The differences are null in white squares. The value of bytes of $\mathcal{B}_{off}$ can be derived from the differences in circled bytes.

is decreased by a factor 2^{112}. However, we now need to find a pair that follows this truncated differential path and so the procedure of the online phase becomes:

1. Ask for a structure of 2^{32} plaintexts such that the second *diagonal* assume the 2^{32} possible values and others bytes are constant.
2. Store the corresponding ciphertexts in a hash table to identify the pairs that have a non-zero probability to follow the differential path.
3. For each of these pairs:
   (a) Guess $\Delta z_6[0]$, $\Delta z_6[7]$ and $\Delta z_6[10]$ and compute the difference in the three first columns of x_7.
   (b) Deduce the value of the three first columns of x_7 using Δz_7.
   (c) Deduce $u_6[0]$ and $u_6[7]$ using $u_7[4]$, $u_7[8]$, $u_7[11]$ and $u_7[15]$.
   (d) Deduce $z_6[0]$ and $z_6[7]$ and compute $\Delta x_6[0]$ and $\Delta x_6[3]$.
   (e) Check if the equation between $\Delta x_6[0]$ and $\Delta x_6[3]$ is satisfied.
   (f) Deduce $\Delta x_6[2]$ and then compute $x_6[2]$ using $\Delta z_6[10]$.
   (g) Guess $\Delta x_1[5]$ and compute the difference in the second column of z_0.
   (h) Deduce the value of the second column of z_0 using Δx_0.
   (i) Get the δ-set associated to one of the message of the pair and build the multiset from the corresponding ciphertexts.
   (j) Check whether the multiset exists in the hash table. If not, discard the key guess.
4. Restart with a new structure if no check found.

As each structure contains 2^{63} pairs and each of these pairs follows the differential with probability 2^{-144}, we need 2^{81} structures on average. Then, for each structure we have to study only $2^{63-32} = 2^{31}$ pairs and for each of them we have to perform $2^{24} \times 2^8$ partial encryptions that is equivalent to 2^{28} encryptions. All in all, this leads to an attack with 2^{113} chosen plaintexts, a time complexity equivalent to 2^{140} encryptions and a memory requirement of 2^{130} AES-blocks.

Reducing the data complexity. Note that for each possible choice of the active *diagonal* in the plaintext we found 96 attacks with the same complexity. As the corresponding differential paths are different it is possible to perform many attacks in parallel to save data in exchange of memory. For instance, if we use structure with three active *diagonals*, it is possible to reach a complexity of $2^{104,83}$ chosen plaintexts and $2^{138,17}$ AES-blocks, the time remaining unchanged.

Key Recovery. This attack can easily be turned into a key-recovery attack without increasing the complexity since only 9 key bytes are sufficient to recover the master key.

AES-256. This attack can be applied to the AES-256 excepted that the keyschedule does not allow us to reduce the time complexity anymore. This leads to an attack with 2^{113} chosen plaintexts, a time complexity equivalent to 2^{156} encryptions and a memory requirement of 2^{130} AES-blocks. For each possible choice of the active *diagonal* in the plaintext we found 384 attacks with the same complexity so it is possible to save more data than previously. For instance, if we use structure with three active *diagonals*, it is possible to reach a complexity of $2^{102,83}$ chosen plaintexts and $2^{140,17}$ AES-blocks, the time remaining unchanged.

6.2 Results

As in the previous section, we have exhausted the almost 2^{16} variants to find the best attacks. For instance our results on the AES-192 reduced to 8 rounds are summarized on Figure 11. As expected we have automatically rediscovered the attacks found by Dunkelman *et al.* and the ones found by Derbez *et al.*, but we have also obtained many new attacks including the best known attacks on 8 rounds for both AES-192 and AES-256 described Section 6.1.

number of guess in the offline phase

number of guess in the online phase

	10	11	12	13	14	15	16	17	18	19	20	21	22
12												6.0	
13												7.0	
14												10.9	9.9
15											9.2	9.9	
16											6.6	7.6	11.0
17							8.6				6.6	7.6	11.0
18							9.6				7.6	10.2	13.6
19							10.7	10.5		6.2	8.0	8.5	8.5
20					4.2	7.7	13.3	10.0		7.6	8.6	10.0	8.5
21	6.6	8.0			5.2	6.5	10.9	8.4		6.9	6.9	10.1	
22	7.3	10.8			3.3	4.3	12.2					8.0	
23		13.4					11.9						

last mixcolumn performed

number of guess in the offline phase

number of guess in the online phase

	10	11	12	13	14	15	16	17	18	19	20	21	22
14												11.1	9.9
15											9.2	9.9	
16													
17											6.2	7.2	10.6
18							8.6				8.3	10.4	13.9
19							10.3	10.5		6.2	8.0	8.5	8.5
20					4.2	7.7	13.4	10.0		7.6	8.6	10.0	8.5
21	6.6	8.0			5.2	6.5	10.9	8.4		6.9	6.9	10.1	
22	7.3	11.0			3.3	4.3	12.2					8.0	
23		13.3					11.9						

last mixcolumn omitted

Figure 11: Differential Enumeration: results on 8 rounds AES-192. All attacks have a data complexity of 2^{113} chosen plaintexts. Numbers in cells are the $\log_2$ of the numbers of attacks found with the same complexity.

Limitations. To save more data, Dunkelman *et al.* propose to consider differential paths with a bigger probability. We have exhausted the simple case where the new differential paths do not have active new bytes in the middle rounds. However, we did not try interesting cases where the active bytes of the pair and bytes of $\mathcal{B}_{on}$ and $\mathcal{B}_{off}$ are *desynchronized* since, besides the number of cases to handle, the complexity of our tweaked tool tends to explode as we cannot apply it to the keyschedule only.

7 Conclusion

We have presented new attacks on AES by generalizing Demirci and Selçuk meet-in-the-middle attacks. We took into account various time/memory tradeoffs including more advanced techniques introduced by Dunkelman *et al.* in [13]. We automatized the search of the best attacks of this kind using the tool developed by Bouillaguet *et al.* in [8] solving linear systems of equations involving S-boxes. As a result, we recovered all best attacks on AES-128, including the recent one of Derbez *et al.* in [11] and found new more efficient attacks for AES-192 and AES-256.

References

1. Biham, E., Keller, N.: Cryptanalysis of Reduced Variants of Rijndael. Technical report, Computer Science Department, Technion – Israel Institute of Technology (2000)
2. Biryukov, A., Dunkelman, O., Keller, N., Khovratovich, D., Shamir, A.: Key recovery attacks of practical complexity on aes-256 variants with up to 10 rounds. [15] 299–319
3. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Matsui, M., ed.: ASIACRYPT. Volume 5912 of Lecture Notes in Computer Science., Springer (2009) 1–18
4. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and related-key attack on the full aes-256. In Halevi, S., ed.: CRYPTO. Volume 5677 of Lecture Notes in Computer Science., Springer (2009) 231–249
5. Biryukov, A., Nikolic, I.: Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to aes, camellia, khazad and others. [15] 322–344
6. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In Lee, D.H., Wang, X., eds.: ASIACRYPT. Volume 7073 of Lecture Notes in Computer Science., Springer (2011) 344–371
7. Bouillaguet, C., Derbez, P., Dunkelman, O., Fouque, P.A., Keller, N., Rijmen, V.: Low-data complexity attacks on aes. IEEE Transactions on Information Theory **58**(11) (2012) 7002–7017
8. Bouillaguet, C., Derbez, P., Fouque, P.A.: Automatic search of attacks on round-reduced AES and applications. In Rogaway, P., ed.: CRYPTO. Volume 6841 of Lecture Notes in Computer Science., Springer (2011) 169–187
9. Daemen, J., Rijmen, V.: AES proposal: Rijndael (1998)
10. Demirci, H., Selçuk, A.A.: A meet-in-the-middle attack on 8-round AES. In Nyberg, K., ed.: FSE. Volume 5086 of Lecture Notes in Computer Science., Springer (2008) 116–126

Table 1: Current cryptanalysis of `AES` variants in the single-key model.

Version	Rounds	Data (CP)	Memory	Time	Technique	Reference
128	6	2^{8}	$2^{106.17}$	$2^{106.17}$	MITM	Section 5.2
	7	2^{32}	$2^{126.47}$	$2^{126.47}$	MITM	Full ver.
	7	$2^{90.4}$	2^{106}	$2^{117.2}$ MA	ID	[18]
	7	2^{97}	2^{98}	2^{99}	MITM	[11]
	8	2^{88}	2^{8}	$2^{125.3}$	Bicliques	[6]
	10 (full)	2^{88}	2^{8}	$2^{126.2}$	Bicliques	[6]
192	6	2^{8}	$2^{109.67}$	$2^{109.67}$	MITM	Full ver.
	7	2^{8}	$2^{153.34}$	2^{163}	MITM	Full ver.
	7	2^{32}	$2^{129.67}$	$2^{129.67}$	MITM	Section 5.4
	7	$19 \cdot 2^{32}$	$19 \cdot 2^{32}$	2^{155}	Square	[14]
	7	$2^{91.2}$	$2^{139.2}$	2^{101}	ID	[17]
	7	2^{95}	2^{143}	2^{143}	MITM	[10]
	7	2^{97}	2^{98}	2^{99}	MITM	[11]
	8	2^{32}	$2^{182.17}$	$2^{182.17}$	MITM	Full ver.
	8	2^{41}	2^{186}	$2^{187.63}$	MITM	[20]
	8	$\mathbf{2^{104.83}}$	$\mathbf{2^{138.17}}$	$\mathbf{2^{140}}$	**MITM**	**Section 6.1**
	8	2^{107}	2^{96}	2^{172}	MITM	[11]
	8	$\mathbf{2^{113}}$	$\mathbf{2^{130}}$	$\mathbf{2^{140}}$	**MITM**	**Section 6.1**
	8	2^{113}	2^{82}	2^{172}	MITM	[11]
	9	2^{80}	2^{8}	$2^{188.8}$	Bicliques	[6]
	12 (full)	2^{80}	2^{8}	$2^{189.4}$	Bicliques	[6]
256	6	2^{8}	$2^{114.34}$	2^{122}	MITM	Full ver.
	7	2^{8}	2^{186}	$2^{170.34}$	MITM	Full ver.
	7	2^{16}	$2^{153.34}$	2^{178}	MITM	Section 5.3
	7	2^{32}	$2^{133.67}$	$2^{133.67}$	MITM	Full ver.
	7	$21 \cdot 2^{32}$	$21 \cdot 2^{32}$	2^{172}	Square	[14]
	7	2^{95}	2^{143}	2^{143}	MITM	[10]
	7	2^{97}	2^{98}	2^{99}	MITM	[11]
	8	2^{8}	$2^{234.17}$	$2^{234.17}$	MITM	Full ver.
	8	2^{32}	$2^{193.34}$	2^{195}	MITM	Full ver.
	8	$2^{34.2}$	$2^{205.8}$	$2^{205.8}$	MITM	[10]
	8	$\mathbf{2^{102.83}}$	$\mathbf{2^{140.17}}$	$\mathbf{2^{156}}$	**MITM**	**Section 6.1**
	8	2^{107}	2^{96}	2^{196}	MITM	[11]
	8	$\mathbf{2^{113}}$	$\mathbf{2^{130}}$	$\mathbf{2^{156}}$	**MITM**	**Section 6.1**
	8	2^{113}	2^{82}	2^{196}	MITM	[11]
	9	2^{32}	$2^{254.17}$	$2^{254.17}$	MITM	Full ver.
	9	2^{120}	2^{203}	2^{203}	MITM	[11]
	9	2^{120}	2^{8}	$2^{251.9}$	Bicliques	[6]
	14 (full)	2^{40}	2^{8}	$2^{254.4}$	Bicliques	[6]

CP: Chosen-plaintext. ID: Impossible Differential. MITM: Meet-in-the-Middle.

11. Derbez, P., Fouque, P.A., Jean, J.: Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting (2013) To appear. Available at http://eprint.iacr.org/.
12. Derbez, P., Fouque, P.A., Leresteux, D.: Meet-in-the-middle and impossible differential fault analysis on aes. In Preneel, B., Takagi, T., eds.: CHES. Volume 6917 of Lecture Notes in Computer Science., Springer (2011) 274–291
13. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In Abe, M., ed.: ASIACRYPT. Volume 6477 of Lecture Notes in Computer Science., Springer (2010) 158–176
14. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved cryptanalysis of rijndael. In Schneier, B., ed.: FSE. Volume 1978 of Lecture Notes in Computer Science., Springer (2000) 213–230
15. Gilbert, H., ed.: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings. In Gilbert, H., ed.: EUROCRYPT. Volume 6110 of Lecture Notes in Computer Science., Springer (2010)
16. Gilbert, H., Minier, M.: A collision attack on 7 rounds of Rijndael. In: AES Candidate Conference. (2000) 230–241
17. Lu, J., Dunkelman, O., Keller, N., Kim, J.: New impossible differential attacks on AES. In Chowdhury, D.R., Rijmen, V., Das, A., eds.: INDOCRYPT. Volume 5365 of Lecture Notes in Computer Science., Springer (2008) 279–293
18. Mala, H., Dakhilalian, M., Rijmen, V., Modarres-Hashemi, M.: Improved Impossible Differential Cryptanalysis of 7-Round AES-128. In Gong, G., Gupta, K.C., eds.: INDOCRYPT. Volume 6498 of Lecture Notes in Computer Science., Springer (2010) 282–291
19. NIST: Advanced Encryption Standard (AES), FIPS 197. Technical report, NIST (November 2001)
20. Wei, Y., Lu, J., Hu, Y.: Meet-in-the-middle attack on 8 rounds of the aes block cipher under 192 key bits. In Bao, F., Weng, J., eds.: ISPEC. Volume 6672 of Lecture Notes in Computer Science., Springer (2011) 222–232