Improved Single-Key Attacks on 8-round AES-192 and AES-256*

Orr Dunkelman[1,2], Nathan Keller[2], and Adi Shamir[2]

[1] Computer Science Department
University of Haifa
Haifa 31905, Israel
orrd@cs.haifa.ac.il
[2] Faculty of Mathematics and Computer Science
Weizmann Institute of Science
P.O. Box 26, Rehovot 76100, Israel
{nathan.keller,adi.shamir}@weizmann.ac.il

Abstract. AES is the most widely used block cipher today, and its security is one of the most important issues in cryptanalysis. After 13 years of analysis, related-key attacks were recently found against two of its flavors (AES-192 and AES-256). However, such a strong type of attack is not universally accepted as a valid attack model, and in the more standard single-key attack model at most 8 rounds of these two versions can be currently attacked. In the case of 8-round AES-192, the only known attack (found 10 years ago) is extremely marginal, requiring the evaluation of essentially all the 2^{128} possible plaintext/ciphertext pairs in order to speed up exhaustive key search by a factor of 16. In this paper we introduce three new cryptanalytic techniques, and use them to get the first non-marginal attack on 8-round AES-192 (making its time complexity about a million times faster than exhaustive search, and reducing its data complexity to about $1/32,000$ of the full codebook). In addition, our new techniques can reduce the best known time complexities for all the other combinations of 7-round and 8-round AES-192 and AES-256.

Keywords: AES, cryptanalysis, single-key attacks, multiset tabulation, differential enumeration, key bridging.

1 Introduction

The Rijndael block cipher [11] was developed in the late 1990's by Joan Daemen and Vincent Rijmen, and was selected as the Advanced Encryption Standard (AES) in 2001. Over the last ten years it replaced the Data Encryption Standard (DES) in most applications, and had become the block cipher of choice for any new security application. It has three possible key sizes (128, 192, and 256 bits), and in 2003 the US government had publicly announced that AES-128 can be

* An extended version of an ASIACRYPT 2010 paper.

used to protect classified data up to the level of "secret", and that AES-192 and AES-256 can be used to protect classified data up to the level of "top secret".

Due to its importance and popularity, the security of AES had attracted a lot of attention, and is considered one of the hottest areas of research in cryptanalysis. A major breakthrough was the recent discovery of related-key attacks on the full versions of AES-192 and AES-256 [6, 7] which are faster than exhaustive search, but have impractical complexities. In another line of research [5], related-key attacks requiring practical time complexity of 2^{45} were found on AES-256 with up to 10 rounds, and related key attacks requiring semipractical time complexity of 2^{70} were found on AES-256 with 11 rounds (the full AES-256 algorithm has 14 rounds, so none of these attacks endanger the security of AES in real applications).

The main weakness of AES-192 and AES-256 exploited in these attacks was their extremely simple key schedule. In a related-key attack model, this made it possible to cancel data differences with corresponding key differences over many rounds of AES. This created a very high probability differential characteristic, which led to a greatly improved time complexity. However, such attacks make a very strong assumption that the attacker can ask the encryption box to modify the unknown key in a known way. Some of these attacks even assume that the attacker can obtain a large number of related keys, or that he can obtain related intermediate subkeys — see [6] for a discussion of these possibilities. Consequently, related-key attacks are important considerations during the design and certification stage of new ciphers, but are not considered a realistic threat in practical security protocols which use the block cipher in a standard way.

In this paper we consider the classical attack model of a single key and multiple known or chosen plaintext/ciphertext pairs. In this model the attacker has to deal with the very well designed data path of AES, and cannot directly benefit from its weak key schedule. Consequently, there are no known attacks which are faster than exhaustive search on any one of the three flavors of AES, and the best we can do is to attack reduced round versions of AES. In the case of AES-256, the largest number of rounds we can attack faster than the 2^{256} complexity of exhaustive search is 8. In the case of AES-192 the reference complexity of exhaustive search is reduced to 2^{192}, and while there is one attack on 8-round AES-192 which was published in FSE'2000 [15], it is extremely marginal: It requires the evaluation of essentially all the possible plaintext/ciphertext pairs under the unknown key, and even then the time required to derive the key is only 16 times faster than the 2^{192} complexity of exhaustive search (one can argue that given the complete codebook of size 2^{128}, there is no need to find the actual key in order to easily decrypt any given ciphertext ...). In the case of AES-128, there is no known attack on its 8-round version, and the best we can do is to attack its 7-round version.

In order to improve all these known attacks, and especially the marginal attack on 8-round AES-192 which no one was able to improve upon in the last ten years, we develop three new cryptanalytic techniques. Our starting point is the attack on 7-round AES developed by Gilbert and Minier [16], which constructs

Rounds		AES-128	AES-192	AES-256	AES-IND
8	Best Published	N/A	2^{188} †	2^{204} †	2^{212} ⋆
	Our Results	N/A	2^{172}	2^{196}	2^{204}

† — Square. ⋆ — Meet in the middle.
AES-IND — AES with independent subkeys.

Table 1. Comparing the time complexities of the best previous attacks and our new attacks

a large table of 2^{72} entries, where each entry contains a sequence of 256 byte values. This idea was extended to 8-round AES by Demirci and Selçuk [12], who constructed an even larger table of 2^{192} entries (again containing sequences of 256 byte values, which are constructed in a slightly modified way). Due to the 2^{200} time required just to construct this table, this attack is worse than exhaustive search for 8-round AES-192, and can only be applied to 8-round AES-256.

Our first new idea (called *multiset tabulation*) is to replace the sequence of 256-byte values in each table entry by the multiset of its values. Even though we lose some information, we show that it is still possible to use such a table in order to discard with very high probability incorrect key guesses. This modification makes it possible to reduce the number of table entries (and thus also the time required to prepare the table) by a factor of 2^8. A much bigger saving (by a factor of 2^{57}) in the size of the table is obtained by another new technique which we call *differential enumeration*. It uses some truncated differential (which need not have particularly high or low probability, as required in standard or impossible differential attacks) in order to enumerate the entries of such a table in a much more efficient way: Instead of directly enumerating state values, the attacker derives them indirectly by enumerating the input and output differential values of certain internal S-boxes. By reducing the space complexity in such a major way, we can now trade it off with the high time complexity of the Demirci and Selçuk attack in order to get greatly improved attacks. Finally, we develop a new *key bridging* technique which exploits the weak key schedule of AES by using the following surprising observation: In the particular case of 8-round AES-192, it is possible to compute one byte of the whitening subkey (used before the first round) directly from four bytes of the last subkey (used at the end of the eighth round), even though they are separated by eight consecutive key mixing stages. Since our attack requires guessing of these five subkey bytes in the first and last rounds, we get an extra saving of 2^8 in our time complexity. By combining these three techniques, we can now break this previously marginal case in about one millionth of the complexity of exhaustive search.

Our new results are summarized and compared with the best previously known single-key attacks in Table 1. As can be seen in this table, our time complexities for 8-round AES are considerably better than the best previous results for both AES-192 and AES-256. In addition, our attack can overcome any possible enlargement of the key size and improvement of the key schedule of

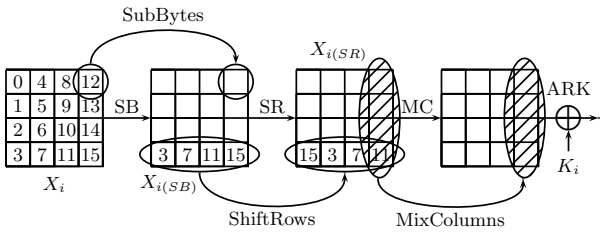

Fig. 1. An AES round

8-round AES, since we can directly find all the subkeys of AES-IND (in which they are independently chosen) with just a little higher complexity.

The rest of this paper is organized as follows. In Section 2 we describe the AES block cipher and introduce our notation. In Section 3 we describe the techniques used in previous attacks on reduced round AES, and analyze their complexity. In Section 4 we introduce the multiset tabulation technique and prove its validity by rigorous probabilistic analysis. The differential enumeration technique is introduced in Section 5. In Section 5.1 we introduce the key bridging technique, prove its validity, and discuss when it can be applied to improve other attacks on AES. We use our new techniques in Section 6 to improve the best known attacks on 7-round AES, and in Section 7 to improve the best known attacks on 8-round AES. In Appendix A we analyze another improvement of the Demirci-Selçuk attack on 7-round AES proposed in [13] and show that its time complexity is significantly higher than claimed by the authors. Finally, we summarize our results in Section 8.

2 A Short Description of AES

The advanced encryption standard (AES) [11] 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 subkey) is applied, and in the last round the MixColumns operation is omitted. Rounds which include the MixColumns operation are called *full rounds*.

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, e.g., AES with 128-bit keys (and thus with 10 rounds) is denoted by AES-128. We use AES to mean all three variants of AES.

The key schedule of AES takes the user key and transforms it into $Nr+1$ subkeys of 128 bits each. The subkey array is denoted by $W[0, \ldots, 4 \cdot Nr+3]$, where each word of $W[\cdot]$ consists of 32 bits. Let the length of the key be Nk 32-bit words, then the first Nk words of $W[\cdot]$ are loaded with the user supplied key. The remaining words of $W[\cdot]$ are updated according to the following rule:

- For $i = Nk, \ldots, 4 \cdot Nr+3$, do
  - If $i \equiv 0 \bmod Nk$ then $W[i] = W[i-Nk] \oplus SB(W[i-1] \lll 8) \oplus RCON[i/Nk]$,
  - else if $Nk = 8$ and $i \equiv 4 \bmod 8$ then $W[i] = W[i-8] \oplus SB(W[i-1])$,
  - 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.

2.1 The Notations Used in the Paper

In the sequel we use the following definitions and notations:

The state matrix at the beginning of round i is denoted by X_i, and its bytes are denoted by $0, 1, 2, \ldots, 15$, as described in Figure 1. Similarly, the state matrix after the SubBytes and the ShiftRows operations of round i are denoted by $X_{i(SB)}$ and $X_{i(SR)}$, respectively.

We denote the subkey of round i by k_i, and the first (whitening) key by k_{-1}, i.e., $k_i = W[4 \cdot (i+1)]||W[4 \cdot (i+1)+1]||W[4 \cdot (i+1)+2]||W[4 \cdot (i+1)+3]$. 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 subkey and only then applying the MixColumns operation. We denote the equivalent subkey for the altered version by u_i, i.e., $u_i = MC^{-1}(k_i)$. The bytes of the subkeys are numbered by $0, 1, \ldots, 15$, in accordance with the corresponding state bytes.

We use the following notations for intermediate encryption values: The intermediate state at the beginning of round i in the encryption of P^j is denoted by X_i^j, and its bytes are denoted by $X_{i,l}^j$, for $0 \leq l \leq 15$. Similarly, the intermediate values after the SubBytes and the ShiftRows operations of round i are denoted by $X_{i(SB),l}^j$ and $X_{i(SR),l}^j$, respectively.

In our attacks we mostly consider the encryption of *δ-sets*, which are structured sets of 256 plaintexts $\{P^0, P^1, \ldots, P^{255}\}$ in which one *active* byte assumes each one of the 256 possible values exactly once, and each one of the other 15 bytes is a (possibly different) constant. A state byte is called *balanced* if the XOR of its 256 values during the encryption of a δ-set is zero.

In all the observations considering reduced-round versions of AES, the numbering of the rounds starts with round 0. When we analyze the behavior of some

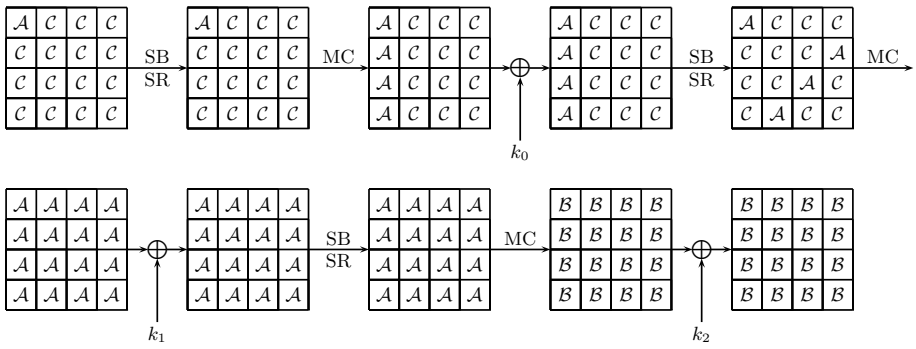

Fig. 2. The development of a δ-set through 3 rounds of AES, where A stands for an active byte, B stands for a balanced byte, and C stands for a constant byte

consecutive inner rounds of AES, we shift the round numbering accordingly, depending on the number of rounds we add at the beginning.

Finally, we measure the time complexity of all the attacks in units which are equivalent to a single encryption operation of the relevant reduced round variant of AES. We measure the space complexity in units which are equivalent to the storage of a single plaintext (namely, 128 bits). To be completely fair, we charge for all the operations carried out during our attacks, and in particular we do not ignore the time and space required to prepare the various tables we use. Note that in this sense, all the standard time/memory tradeoff attacks are worse than exhaustive search due to their lengthy preprocessing phase.

3 Previous Work

The first attack developed against AES was the Square attack, which was found by its designers [10]. The Square attack is based on the following observation:

Observation 1 *Consider the encryption of a δ-set through three full AES rounds. The set of 256 corresponding ciphertexts is balanced, i.e., the XOR of the 256 values in each one of its* 16 *bytes is zero.*

The observation follows easily from the structure of AES, as demonstrated in Figure 2.

This property is the basis of many attacks on reduced round variants of AES. For example, it can be used to attack 6-round AES by adding one round at the top and two rounds at the bottom. In a naive version of such an attack, the adversary guesses four bytes of the key k_{-1} in order to construct a collection of 256 plaintexts which form a δ-set at state X_1 (e.g., if the active byte of the δ-set is byte 0, bytes $0, 5, 10, 15$ of k_{-1}). Then she guesses four bytes of the equivalent subkey u_5 and one byte of the equivalent subkey u_4, and checks whether the 256 intermediate values in one byte of the state X_4 sum up to zero. (For example, if the byte to be checked is byte 0, then the subkey bytes the adversary should guess are byte 0 of u_4 and bytes $0, 7, 10, 13$ of u_5). This naive version requires 2^{32}

chosen plaintexts and about 2^{72} encryptions. In [15], the attack was improved using *partial sums* and other techniques, which reduced the time complexity to the practical value of 2^{42} encryptions. The resulting attack is the best known attack on 6-round AES.

In [16], Gilbert and Minier proposed to refine the information on the intermediate encryption values of the δ-sets exploited in the attack. Their attack is based on the following observation:

Observation 2 *Consider the encryption of a δ-set through three full AES rounds. For each one of the 16 bytes of the ciphertext, we can define a sequence of* 256 *values for this byte by ordering the plaintexts according to the value of their active byte. Then any such sequence is fully determined by just 9 byte parameters, which are complex functions of the constants in the δ-set and the key bytes. Consequently, for any fixed byte position, there are at most 2^{72} 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, and out of the $2^{256+15\times 8} = 2^{376}$ sequences which could be potentially defined by the choice of* 15 *constant bytes and* 256 *key bits) .*

This observation was used in [16] to mount an attack on 7-round AES-128 with time complexity slightly smaller than that of exhaustive key search. Since the attack algorithm is a bit complex and not used in our paper, we omit it here.

In [12], Demirci and Selçuk extended the observation of [16] by another round. They showed the following:

Observation 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^{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, and out of the $2^{256+15\times 8} = 2^{376}$ sequences which could be potentially defined by the choice of* 15 *constant bytes and* 256 *key bits).*[3]

This observation was used in [12] to mount attacks on 7-round and 8-round variants of AES-256. The attack on 7-round AES-256 is roughly as follows:

1. **Preprocessing phase:** Compute all the 2^{192} possible values of the 255-byte sequence given in Observation 3, and store them in a hash table.
2. **Online phase:**

[3] In [12] the authors note that the function $f_{c_1,\ldots,c_{25}}(x)$ can be written as $f_{c_1,\ldots,c_{25}}(x) = g_{c_1,\ldots,c_{24}}(x) \oplus c_{25}$, and thus one can reduce the number of possible sequences by picking some x_0, and considering the augmented function $f'_{c_1,\ldots,c_{24}}(x) = f_{c_1,\ldots,c_{25}}(x) - f_{c_1,\ldots,c_{25}}(x_0) = g_{c_1,\ldots,c_{24}}(x) - g_{c_1,\ldots,c_{24}}(x_0)$. In this case, the number of parameters is reduced to 24, the number of "interesting" entries in each sequence is reduced to 255 (as $f'(x_0) = 0$, independently of the choice of x_0 and $c_1, \ldots, c_{24}$), and the number of possible sequences is reduced to 2^{192}.

(a) Guess the value of four bytes in the whitening key k_{-1} and of one byte in k_0, and for each guess, construct a δ-set from the data. (For example, if the active byte of the δ-set is byte 0, then the guessed bytes are bytes $0, 5, 10, 15$ of k_{-1} and byte 0 of k_0. Note that byte 0 of k_0 is used only to compute the *order* of the values in the δ-set).

(b) Guess four bytes of the equivalent subkey u_6 and one byte of the equivalent subkey u_5 and partially decrypt the ciphertexts of the δ-set to obtain the sequence of 256 intermediate values of one byte of the state X_5. (For example, if the byte to be checked is byte 0, then the subkey bytes the adversary should guess are byte 0 of u_5 and bytes $0, 7, 10, 13$ of u_6).

(c) Check whether the sequence exists in the hash table. If not, discard the key guess.

The data complexity of the attack is 2^{32} chosen plaintexts. The time complexity of the online phase is relatively modest at 2^{80}, but the space complexity and the time complexity in encryption operations required to prepare this large table are about 2^{200}. These complexities are worse than exhaustive search for both AES-192 and AES-128. However, Demirci and Selçuk presented a tradeoff, which makes it possible to decrease the memory complexity at the expense of increasing both the data and the online time complexities. This results in an attack on 7-round AES-192 with data complexity of 2^{96} chosen plaintexts, and time and space complexities of 2^{144}.

The attack in [12] can be extended to 8-round AES-256 by guessing the full subkey of the last round. This increases the time complexity of the online phase from 2^{80} to 2^{208} encryptions, and makes it impossible to rebalance the parameters in order to attack 8-round AES-192.

Finally, in a more recent paper, Demirci et al. [13] claim that by optimizing their technique they can also attack 7-round AES-128 faster than exhaustive search. However, as we show in Appendix A, the analysis of [13] is flawed, and the correct running time of the attack is about 2^{32} times more than claimed, and in particular greater than the complexity of exhaustive key search for the 128-bit key version.

4 The Multiset Tabulation Technique

Our first technique improves Observation 3 by replacing the sequence of 256 values with the multiset of the values. We show by a rigorous probabilistic analysis that although information is lost in the transformation to a multiset, the new table still allows the adversary to discard all the incorrect key guesses with an overwhelming probability.

Observation 4 *Consider the encryption of a δ-set $\{P^0, P^1, \ldots, P^{255}\}$ through four full AES rounds.*

For each $0 \leq l \leq 15$*, the (un-ordered) multiset*[4] $\left[X^0_{4,l} \oplus X^0_{4,l}, X^1_{4,l} \oplus X^0_{4,l}, \ldots, X^{255}_{4,l} \oplus X^0_{4,l}\right]$ *is fully determined by the following 24 byte parameters:*

- *The full 16-byte state* X^0_2*.*
- *Four bytes of the state* X^0_1*. (For example, if the active byte of the* δ*-set is byte* 0 *then these are bytes* $0, 1, 2, 3$*).*
- *Four bytes of the subkey* k_2*. (For example, if* $l = 0$ *then these are bytes* $0, 5, 10, 15$*).*

Moreover, this multiset can assume only 2^{184} *values out of the* $\binom{510}{256} \approx 2^{505.2}$ *"theoretically possible" values.*[5]

Our variant has several advantages over Observation 3:

- In our variant, the parameters upon which the sequence depends are specified explicitly. This improvement will be crucial for the major reduction in the number of parameters which we shall present in the next section.
- The smaller number of possible configurations in our variant (2^{184} instead of 2^{192}) allows to reduce the memory requirements of the attack and the time complexity of the preprocessing phase by a factor of 2^8.
- Since we consider a multiset instead of an ordered sequence, the adversary does not need to know the *order* of the values in the δ-set at the beginning of the four rounds. This allows to reduce the time complexity of the online phase of the attack by a factor of 2^8 (by avoiding the guess of one byte in the subkey k_0).

Proof: The proof emphasizes the meet-in-the-middle nature of the observation.

We start with the "bottom side" of the four rounds. First, we observe that if the values $\{X^0_2, X^1_2, \ldots, X^{255}_2\}$ are known, then the knowledge of bytes $0, 5, 10, 15$ of k_2 yields the knowledge of the entire first column before the AddRoundKey of round 3 in all the 256 encryptions. Since the AddRoundKey preserves differences, this yields the desired values of the vector of differences $\left(X^0_{4,l} \oplus X^0_{4,l}, X^1_{4,l} \oplus X^0_{4,l}, \ldots, X^{255}_{4,l} \oplus X^0_{4,l}\right)$.

Second, we note that in order to know the values $\{X^0_2, X^1_2, \ldots, X^{255}_2\}$, it is sufficient to know the value X^0_2 which is given as part of the parameters, and the differences $\left(X^0_2 \oplus X^0_2, X^1_2 \oplus X^0_2, \ldots, X^{255}_2 \oplus X^0_2\right)$. Since the ShiftRows, the MixColumns and the AddRoundKey operations are linear, it is thus sufficient to know the differences $\left(X^0_{1(SB)} \oplus X^0_{1(SB)}, X^1_{1(SB)} \oplus X^0_{1(SB)}, \ldots, X^{255}_{1(SB)} \oplus X^0_{1(SB)}\right)$.

Now we turn to the "top side" of the four rounds. In round 0, the differences $\left(X^0_{0(SB)} \oplus X^0_{0(SB)}, \quad X^1_{0(SB)} \oplus X^0_{0(SB)}, \ldots, X^{255}_{0(SB)} \oplus X^0_{0(SB)}\right)$ are known — these are exactly the 256 possible differences in byte 0 (the rest of the bytes are equal). Note that the *order* of the differences is not known, but this

[4] Unlike sets, elements can occur multiple times, and the multiset retains this multiplicity along with the values.

[5] The calculation of the number of possible values is explained at the end of this section.

does not disturb the adversary since in our attack she is interested only in the multiset and not in the sequence. Since the ShiftRows, the MixColumns, and the AddRoundKey operations are linear, the differences $\left(X_1^0 \oplus X_1^0, X_1^1 \oplus X_1^0, \ldots, X_1^{255} \oplus X_1^0\right)$ are also known. By the structure of the δ-set, these differences are active in bytes $0, 1, 2, 3$ and passive in the rest of the bytes. Since bytes $0, 1, 2, 3$ of X_1^0 are given as part of the parameters, bytes $0, 1, 2, 3$ of the values $\{X_1^1, \ldots, X_1^{255}\}$ are thus also known, and so are bytes $0, 1, 2, 3$ of $\{X_{1(SB)}^0, X_{1(SB)}^1, \ldots, X_{1(SB)}^{255}\}$. Since the differences $X_{1(SB)}^j \oplus X_{1(SB)}^0$ in all the bytes except for $0, 1, 2, 3$ are zero for all $j = 1, 2, \ldots, 255$, this implies that the full vector of differences $\left(X_{1(SB)}^0 \oplus X_{1(SB)}^0, X_{1(SB)}^1 \oplus X_{1(SB)}^0, \ldots, X_{1(SB)}^{255} \oplus X_{1(SB)}^0\right)$ is known, as required above.

Finally, since the multiset depends on 24 byte parameters, it can assume at most 2^{192} possible values. However, we note that in this count, each δ-set is represented by 2^8 multisets, according to the 256 possible choices of P^0. We can then reduce the number of parameters by one by choosing P^0 such that $X_{1,0}^0 = 0$ (this is possible since byte 0 in state X_1 is active). This reduces the number of possible multisets to 2^{184}, concluding the proof. □

4.1 Analysis of The Distribution of Sequences

While it is easy to see that in the original Demirci-Selçuk attack, all the wrong subkeys are discarded with an overwhelming probability, it is far less clear that the same holds for our multiset tabulation technique. In order to address this issue, we provide in this subsection a rigorous analysis of the distribution of the sequences generated in the attacks described in the paper. The analysis shows that despite the loss of information in our generation of tables, the adversary is still able to discard all the wrong subkey guesses with overwhelming probability.

In the analysis, we assume that the sequences obtained during the attack for wrong key guesses look as they were generated randomly (with the appropriate distribution). This assumption is very common in cryptanalysis, and in our case it is founded on the diffusion properties of AES. A wrong subkey guess (even in a single byte) will either completely change the values in the sequence, or even change the identity of which elements are taken into consideration (or their order).

The first attack we discuss is the original attack of Demirci-Selçuk, discussed in Observation 3. In this attack, a vector of 256 entries is evaluated as $(f(0), f(1), \ldots, f(255))$, where $f(i) = f_{c_1,\ldots,c_{25}}(i)$. As the evaluations of the vector are randomly distributed (for a wrong subkey, the outcome is expected to be random), we can easily conclude that there are 2^{2048} possible vectors, all with the same probability, where the number of "good" vectors (i.e., ones that can be produced by any of the admissible functions), is only 2^{200}. Hence, the probability that a wrong subkey guess generates a vector which is admissible is extremely low, $2^{-2048} \cdot 2^{200} = 2^{-1848}$.

The improved variant of the attack, mentioned in the footnote of the observation, takes into consideration vectors of 255 elements, which are generated

by taking the previous 256-element vectors, and subtracting the first element from all other elements (discarding the first entry which is always 0 after this procedure). It is easy to see that all $256^{255} = 2^{2040}$ vectors can appear with the same probability, and as there are 2^{192} admissible vectors, defined by the function $f'_{c_1,\ldots,c_{24}}(x) = f_{c_1,\ldots,c_{25}}(x) - f_{c_1,\ldots,c_{25}}(0)$, again, the probability that a wrong subkey generates an admissible vector is $2^{-2040} \cdot 2^{192} = 2^{-1848}$.

For the multiset sequences used in our attack, the analysis is more delicate. First we note that each entry $X^0_{4,\ell} \oplus X^i_{4,\ell}$ (besides the entry $i = 0$, which is always zero) is distributed randomly. Hence, we look at 255 values, each chosen uniformly and independently from the set $\{0, 1, \ldots, 255\}$. While this may seem similar to the previous attack, we deal with multisets, where the order has no meaning. This results with a significantly smaller sample space. In other words, a multiset can be considered as a vector of 256 counters, each counting how many times a specific entry value for $X^0_{4,\ell} \oplus X^i_{4,\ell}$ is encountered, such that the sum of all counters is 255 (considering that 0 is always counted at least once, and hence we disregard it).

Using selection with repetitions, it is easy to see that the number of possible multisets can be described by a sequence of 255 place holders and 255 dividers placed in some order in a linear array of 510 entries. Hence, the number of repetitions of value i is defined by the number of place holders between the i-th and $i + 1$-st dividers. This allows counting the number of possible multisets as $\binom{510}{256} \approx 2^{505.2}$.

Additionally, we have to consider the fact that the multiset can is actually a representative of a few other vectors (picking a different X^0, yields a shifted version of the vector). As each multiset is a representative of at most 255 other vectors,[6] we obtain that there are more than $2^{498.2}$ possible counter vectors that may be encountered.

However, unlike the prior cases where the sample space was distributed uniformly, in this case, we obtain a non-uniform distribution. For example, the multiset $\{255, 0, 0, \ldots, 0\}$, occurs with probability of 2^{-2040}, while the multiset $\{254, 1, 0, \ldots, 0\}$ occurs with a larger probability of $255 \cdot 2^{-2040}$ (as it does not matter which entry of the 255 values $X^0_{4,\ell} \oplus X^i_{4,\ell}$ is 1). Thus, we cannot claim that the probability of encountering an admissible multiset when examining a wrong subkey is $2^{184} \cdot 2^{-498.2}$, like in the previous attacks. It may occur that the admissible multisets have a higher probability than the non-admissible ones, and hence the probability of encountering them for a wrong subkey guess is no longer negligible.

In order to overcome this problem, we use Poisson approximation to detect the most probable multisets, and show that there are more than $2^{467.6}$ equiprobable multisets which are the most probable ones, and thus even if all the admissible multisets are contained in this class, the probability of obtain-

[6] Picking a different starting point $X^{0,\star}$, results in changing all the 255 entries of the multiset by XORing them with $X^{0,\star} \oplus X^0$. As there are at most 255 other values for $X^{0,\star}$, each multiset belongs to the same equivalence class with as at most 255 other multisets.

ing an admissible multiset for a wrong key guess is still bounded from above by $2^{184} \cdot 2^{-467.6} = 2^{-283.6}$. Since the adversary checks less than 2^{200} wrong key guesses, it follows that all of them are expected to produce non-admissible multisets with overwhelming probability.

As each value of the multiset (up to the first entry) is chosen randomly, we can approximate the number of times a specific value appears in the multiset using a Poisson distribution with a mean value of 255/256. This way, we can conclude that on average out of the 256 possible values $X^0_{4,\ell} \oplus X^i_{4,\ell}$ (after removing the $X^0_{4,\ell} \oplus X^0_{4,\ell} = 0$ entry), about 94 do not appear, 94 appear once, 47 appear twice, 16 three times, four values appear four times, and one is expected to appear five times.

As this is the most probable outcome, we look at these cases, and show that there are sufficiently many of these. Notably, there are

$$\binom{256}{94} \cdot \binom{162}{94} \cdot \binom{68}{47} \cdot \binom{21}{16} \cdot \binom{5}{4} \cdot \binom{1}{1} = 2^{238.5} \cdot 2^{155.0} \cdot 2^{57.4} \cdot 2^{14.3} \cdot 2^{2.3} = 2^{467.6}$$

possible multisets of this structure. Hence, we can conclude that even though the outcome space is not uniformly distributed, there is a sufficient number of multisets with the highest probability, to ensure that the attack succeeds.

5 The Differential Enumeration Technique

Observation 4 shows that the possible multisets depend on 24 explicitly stated parameters. In order to reduce the size of the precomputed table, we would like to choose the δ-set such that several of these parameters will equal to pre-determined constants. Of course, the key bytes are not known to the adversary and thus cannot be "replaced" by such constants. At first glance, it seems that the bytes in the intermediate states X^0_1 and X^0_2 also cannot be made equal to pre-determined constants by choosing the plaintexts appropriately, since they are separated from the plaintexts by operations involving an unknown key. However, we show that by using an expected-probability differential (i.e., a differential whose probability is not assumed to be especially high or especially low) for 4-round AES, the plaintext P^0 can be chosen such that the full 128-bit state X^0_2 will assume one of at most 2^{64} particular values (which can be computed in advance and are independent of the choice of key) instead of 2^{128} possible values.

Consider a truncated differential for four full AES rounds, in which both the input and the output differences are non-zero in a single byte (e.g., byte 0 both in the input and in the output). The probability of this differential is expected to be about 2^{-120}, [7] and thus it is expected that 2^{120} randomly chosen pairs with

[7] The probability of 2^{-120} is based on the assumption that 4-round AES behaves like a random permutation with respect to this differential, and thus forcing 120 bits to be equal has this probability. Theoretically, it may be the case that due to the algebraic structure of AES, this differential is impossible, which would lead to very strong impossible differential attacks on reduced-round variants of AES. However,

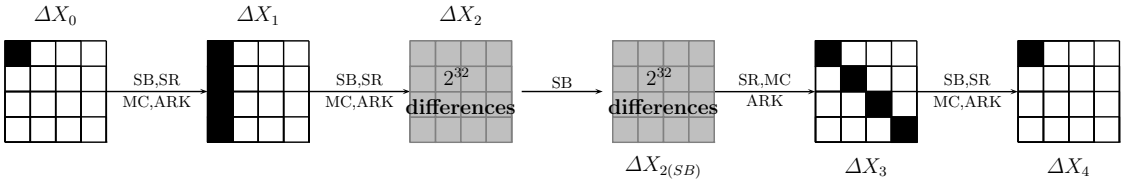

Fig. 3. The 4-Round Differential Characteristic Used in Our Attack

difference only in byte 0 would contain one pair that satisfies the differential. Moreover, since each δ-set contains 2^{15} pairs with difference in a single byte, a collection of 2^{105} randomly chosen δ-sets in which byte 0 is active is expected to contain a right pair with respect to the differential. For right pairs, we show the following:

Observation 5 *Let (P^1, P^2) be a right pair with respect to the differential (i.e., the difference $P^1 \oplus P^2$ is non-zero only in byte 0, and the difference between the corresponding ciphertexts, $C^1 \oplus C^2$, is also non-zero only in byte 0). Then the intermediate state X_2^1 assumes one of at most 2^{64} prescribed values.*

Proof:

The proof is a meet-in-the-middle argument. We start with the "top side" of the four rounds. Due to the structure of AES, the difference between the states $X^1_{1(SB)}$ and $X^2_{1(SB)}$ (i.e., the intermediate values after SubBytes of round 1) is non-zero only in bytes $0, 1, 2, 3$. Thus, this difference can assume at most 2^{32} distinct values. Since the ShiftRows, the MixColumns, and the AddRoundKey operations are linear, this implies that the difference $X_2^1 \oplus X_2^2$ can assume at most 2^{32} different values.

On the other hand, from the "bottom side" we see that the difference $X_3^1 \oplus X_3^2$ is non-zero only in bytes $0, 5, 10, 15$. Since the ShiftRows, the MixColumns, and the AddRoundKey operations are linear, this implies that the difference $X^1_{2(SB)} \oplus X^2_{2(SB)}$ can assume at most 2^{32} different values.

It is well-known that given the input and output differences of the SubBytes operation, there is one possibility on average for the actual pair of input/output values.[8] Moreover, this pair of actual values does not depend on the key, and can be easily found by precomputing the full difference distribution table of the SubBytes operation. Since for the right pair we consider, there are at most $2^{32} \cdot 2^{32} = 2^{64}$ possible pairs of input/output difference of the SubBytes operation in round 2, there are at most 2^{64} possible values of the full state X_2^1, as asserted. □

we could not find any specific reason why this should be the case, and unfortunately, we cannot check this differential experimentally due to its extremely low probability.

[8] Actually, given the input/output differences, with probability of about 1/2 there are no such pairs, with probability of about 1/2 there are two pairs, and with probability of about 1/256 there are four pairs.

It follows from the observation that if we choose the δ-set such that P^0 is a member of a right pair with respect to this expected-probability differential, we are assured that the state X_2^0 can assume at most 2^{64} possible values. Moreover, since these values do not depend on the key and can be computed in advance, this allows to construct the "table of possible multisets" only for these 2^{64} values, which reduces the size of the table and the time complexity of the preprocessing phase by a huge factor of 2^{57} as shown below.

Three additional remarks are due.

- First, we note that in order to exploit the expected-probability differential we have to consider as many as 2^{113} chosen plaintexts, which increases the data complexity of the attack. However, the resultant tradeoff is advantageous since the data complexity was smaller than the other complexities.
- Second, in order to detect the right pair with respect to the differential, the adversary has to guess several key bytes in the rounds before and after the differential. However, it turns out that if the differential is chosen such that the non-zero differences are in the bytes which are active in the δ-set, these key bytes coincide with the key bytes that should be guessed in the original Demirci-Selçuk attack. Hence, this does not increase the time complexity of the online phase of the attack.
- Finally, we note that the total number of possible multisets after the combination with the differential is not $2^{184} \cdot 2^{-64} = 2^{120}$, but rather 2^{127}. The reason for this increase is that in the original attack, the number of multisets is reduced by a factor of 2^8 since each δ-set corresponds to 2^8 different multisets, according to the possible choices of P^0 (see proof of Observation 4). In the new version of the attack, we are forced to choose P^0 to be one of the members of the right pair w.r.t. the differential, and thus each δ-set corresponds to only two "special" multisets. [9] Therefore, the memory complexity and the time complexity of the preprocessing phase are reduced by a factor of 2^{57} rather than 2^{64}, compared to Observation 4.

5.1 The Key Bridging Technique

In this section we show that the time complexity of the online phase in the attacks on 8-round AES-192 can be reduced significantly by using key schedule considerations. While most of these considerations are simple, one of them is a

[9] We note that while the table of possible multisets is constructed according to one member of the right pair, it may occur that in the actual attack, the other member is chosen as P^0, and thus the multiset does not match the table (even for the right key guess). A simple solution is to repeat the attack for both members of the right pair. A more advanced solution, which allows to save the extra factor two in the time complexity of the attack, is to store the multisets only up to XOR with a constant value. This can be achieved by a small modification to the preprocessing phase, consisting of XORing each multiset with the 256 possible byte values and storing in the table the resulting multiset which is the least in the lexicographic order amongst the 256 possibilities.

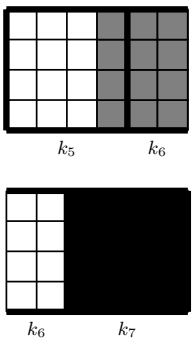

Fig. 4. The Subkeys k_5, k_6, and k_7 in the Key Schedule of AES-192. The known bytes are colored in black, and the retrieved bytes are colored in gray.

novel observation which we call *key bridging technique*, that allows the adversary to deduce some subkey bytes from some other subkey bytes, even though they are separated by many key mixing steps. At the end of the section, we show that except for its application in our attack, the key bridging technique can be used to improve two other previously known attacks on 8-round AES.

We start with the attack on 8-round AES-192. Recall that in the online phase of this attack, the adversary has to guess four bytes of the subkey k_{-1}, one byte of the equivalent subkey u_5, four bytes of the equivalent subkey u_6, and the full k_7. The exact number of bytes that should be guessed depends on the choice of the active byte of the δ-set and of the byte in which the multiset is constructed. It turns out that if the byte to be examined at the end of round 4 is one of the bytes $1, 6, 11, 12$, then the number of guessed key bytes is reduced by three. Indeed, by the key schedule of AES-192, the knowledge of k_7 yields the knowledge of the first two columns of k_6 (and thus also of u_6) and of the last column of k_5 (and thus also of u_5), see Figure 4.

If the byte to be checked at the end of round 4 is byte 1, then the bytes to guess are byte 13 of u_5, bytes $3, 6, 9, 12$ of u_6, and the full subkey k_7. However, as shown earlier, once k_7 is guessed, bytes $3, 6$ of u_6 and byte 13 of u_5 can be computed from the key schedule, thus reducing the time complexity of the online phase of the attack by a factor of 2^{24}.

The complexity can be further reduced by another factor of 2^8 using the following novel observation:

Observation 6 *By the key schedule of AES-192, knowledge of columns* $0, 1, 3$ *of the subkey* k_7 *allows to deduce column* 3 *of the whitening key* k_{-1} *(which is actually Column* 3 *of the master key).*

The main novelty in this observation is that it exploits the weak key schedule of AES-192 in order to provide a surprisingly long "bridge" between two subkeys which are separated by 8 key mixing steps (applied in the reverse direction). In

particular, it makes it possible to compute one byte in the whitening subkey k_{-1} directly from four bytes in the last subkey k_7[10], which saves a factor of 2^8 in the time complexity of any attack which has to guess these five subkey bytes. Since guessing key material in the first and last round is a very common cryptanalytic technique, this observation can have wide applicability (for example, it can reduce the time complexity of the impossible differential attack on 8-round AES-192 presented in [24] from 2^{180} to 2^{172}, which is the same as our time complexity but in the much stronger attack model of related keys).

Proof:

We start with a simpler observation first presented in [14]:

By the key schedule of AES-192, for any $k \geq 2$ and for $0 \leq j \leq 3$, we have

$$\begin{aligned} &W[6k+j] \oplus W[6k+j+2] = \\ &\quad = \big(W[6k+j] \oplus W[6k+j+1]\big) \oplus \big(W[6k+j+1] \oplus W[6k+j+2]\big) \\ &\quad = W[6(k-1)+j+1] \oplus W[6(k-1)+j+2] \\ &\quad = W[6(k-2)+j+2], \end{aligned} \tag{1}$$

where $W[\cdot]$ are the 32-bit words generated by the key schedule algorithm. Thus, the knowledge of words $W[6k+j]$ and $W[6k+j+2]$ is sufficient to retrieve $W[6(k-2)+j+2]$. Similarly, it was observed in [14] that for any $k \geq 2$, the knowledge of $W[6k+1]$ and $W[6(k-1)+5]$ is sufficient to retrieve $W[6(k-2)+1]$. Indeed, we have

$$\begin{aligned} &W[6k+1] \oplus SB(W[6(k-1)+5] \lll 8) = \\ &\quad = \big(W[6k+1] \oplus W[6k]\big) \oplus \big(W[6k] \oplus SB(W[6(k-1)+5] \lll 8)\big) \\ &\quad = W[6(k-1)+1] \oplus W[6(k-1)] \oplus RCON[k] \\ &\quad = W[6(k-2)+1] \oplus RCON[k]. \end{aligned} \tag{2}$$

Both observations allow to "jump" over one row in the key schedule algorithm (see Figure 5).

Combining the two observations, we see that for any $k \geq 4$, the knowledge of $W[6k+3]$ and $W[6(k-1)+5]$ is sufficient to retrieve $W[6(k-4)+3]$. Indeed, we have

$$\begin{aligned} &W[6k+3] \oplus SB(W[6(k-1)+5] \lll 8) = \\ &\quad = \big(W[6k+3] \oplus W[6k+1]\big) \oplus \big(W[6k+1] \oplus SB(W[6(k-1)+5] \lll 8)\big) \\ &\quad = W[6(k-2)+3] \oplus W[6(k-2)+1] \oplus RCON[k] \\ &\quad = W[6(k-4)+3] \oplus RCON[k]. \end{aligned} \tag{3}$$

This already allows us to "jump" over three rows in the key schedule algorithm.

Finally, in order to prove Observation 6, note that by assumption, the words $W[32], W[33]$, and $W[35]$ are known. Since we have $W[33] \oplus W[35] = W[23]$ and $W[32] \oplus W[33] = W[27]$, this implies that $W[27]$ and $W[23]$ are known. By Equation (3) above (with $k = 4$), this is sufficient to retrieve $W[3]$, as asserted. □

[10] The four bytes of k_7 are 0 and 4 (for obtaining byte 0 of $W[27]$) and bytes 7 and 15 (for obtaining byte 3 of $W[23]$).

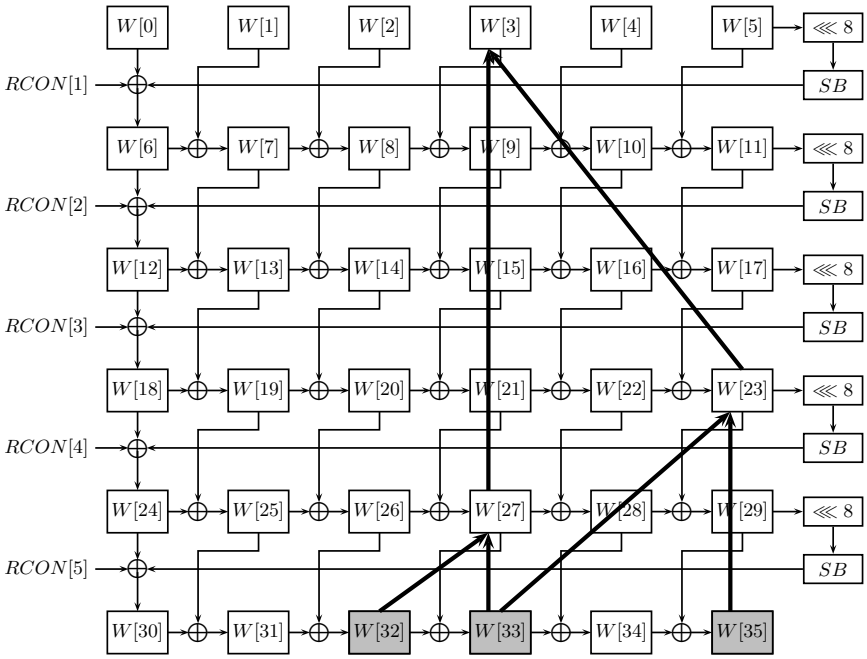

Fig. 5. Deducing Subkeys Using the Key Schedule Algorithm of AES-192

Since in the 8-round attack, one of the subkey bytes guessed by the adversary is included in the column $W[3]$ (regardless of the active byte in the δ-set, since the adversary guesses a shifted column), this reduces the time complexity by another factor of 2^8. In total, the key schedule considerations reduce the time complexity of the online phase of the attack on AES-192 by a factor of 2^{32}.

In the attack on 8-round AES-256, key schedule considerations can help the adversary only a little. By the key schedule, the subkey u_6 is independent of the subkey k_7, and thus the only subkey byte the adversary can retrieve is the single byte of u_5. The novel observation presented in the case of AES-192 does not hold for AES-256, and thus the time complexity can be reduced only by a factor of 2^8.

It is interesting to note that the search for such long key bridges does not require tedious hand calculations or great intuition, since it can be easily automated: By choosing a random key and observing the effect of changing one of its subkey bytes on all the other subkey bytes, one could discover all the cases in which some bytes depend only on a limited number of far away other bytes due to a weak avalanche effect. We recommend to apply this procedure to any newly designed cryptosystem in order to detect such unpleasant surprises in advance.

5.2 Application of the Key Bridging Technique to Other Attacks on 8-round AES-192

The key bridging technique reduces the need to guess subkey material in attacks on 8-round AES-192. As there are other attacks on 8-round AES-192 (most of which are in the related-key model), we tried to locate these attacks that can benefit from the new technique.

Before considering the various attacks, we tried to evaluate what type of attacks can enjoy this technique. We came to the conclusion that such attacks should need a huge amount of subkey material in the last stages, and at least one byte from $W[3]$ in the first whitening key. Additionally, it appears that the attack needs to guess both subkeys simultaneously (rather then guessing one of the subkeys, and computing the second one using some other technique), as we demonstrate later.

The SQUARE Attack on 8-Round AES-192 [15] The first attack which we consider as a candidate for improvement is the 8-round attack SQUARE attack of [15]. In this attack, the adversary guesses the full k_7 as well as one byte of the last column of k_{-1}.

The attack starts with guessing four bytes in k_{-1}, and only then the bytes of k_7, which does not affect the usability of the key bridging, as it is easy to reformulate the relations such that one byte of k_7 is deduced from two bytes of k_7 and the byte of k_{-1}. While this suggests that the key bridging technique may be used, the special nature of the attack, prevents gaining the expected factor of 2^8 in the time complexity of the attack.

The way the attack of [15] works, after guessing the four bytes of k_{-1}, a set of 2^{104} ciphertexts, called a herd, is identified, and is partially decrypted. The partial decryption is done in steps. Firstly, a few bytes of k_7 are guessed, and the partial decryption reduces the set of values for further decryption to a smaller set, which are then partially decrypted under a newly guessed subkey byte(s), which in turn results in a smaller set, and so on, until one byte is determined. Following this fact, obtaining the "free" byte of k_7 is done after a sufficient number of key bytes were already guessed. Hence, the outcome is that the peak number of operations (key guesses times the size of the set of values, and which is met several times throughout the execution of the attack), is not reduced. What can be reduced, is the number of times this peak is reached, implying that instead of having 10 such peaks, we can reduce the number of "peaks" to just three. This suggests an improvement of about 3 times in the running time of the attack, i.e., to $2^{186.3}$.

The Related-Key Impossible Differential Attack on 8-Round AES-192 [24] In [24] three related-key impossible attacks on 8-round AES-192 are reported. In all of these attacks parts of the key k_7 are guessed, and some pairs (which satisfy some differential conditions) are then analyzed in the first round, and candidate values for k_{-1} are obtained. Then, the candidate subkey (as a

whole) is found to be illegitimate (as it suggests that an impossible event has occurred). All three variants of the attack analyze two bytes in k_{-1} which are part of $W[3]$. They differ by the number of bytes from k_7 that are guessed, and the amount of data used in each attack.

The first variant of the attack requires $2^{64.5}$ chosen plaintexts, and takes 2^{177} time. This variant allows computing the two bytes of k_0 immediately (due to the key bridging). For each subkey guess (composed of 14 bytes from k_8), there are 2^{71} pairs, each suggesting one value on average for 8 bytes of k_0. Obviously, if the two bytes of k_0 disagree with the bridged key bytes, we can discard the pair. On the other hand, for each 14-byte guess of k_8, there are only 2^{48} (rather than 2^{64}) possible subkeys in k_0. Hence, while the number of useful pairs is reduced to $2^{71} \cdot 2^{-16} = 2^{55}$ pairs, the probability that a wrong subkey guess for k_0 remains is reduced to $2^{48} \cdot (1 - 2^{-48})^{2^{55}} = 2^{48} \cdot e^{-128} \approx 2^{-136}$. This in turn, implies that the number of wrong 14-byte key guesses that remain is $2^{112} \cdot 2^{-136} = 2^{-24}$. This probability is slightly smaller than in the original attack (where the analysis reveals that the probability of a wrong subkey to remain is 2^{-8}), and hence, one may consider reducing the data complexity with no effect on the data complexity. This can be done, but the amount of data needed is reduced to $2^{64.43}$, which in turn suggests a negligible reduction in the time complexity.

The other two variants faces very similar results. In these variants less key material is guessed in the last round, which in turn allows to compute only one byte of k_{-1} using the key bridging technique. This time, each analyzed pair has probability 2^{-8} to offer a consistent solution with the key byte suggested by the key bridging, and there are 2^{56} possible subkeys to discard in the first round.

It is easy to see that the advantage of applying our key bridging technique in this situation is quite small. This follows the fact that in this specific impossible differential attack, the majority of the time complexity is identifying the pairs that we need to analyze. Once the pairs are detected, the suggested subkey in the first round can be easily computed rather than guessed.

On Key Bridging in Attacks on 8-Round AES-256 One may consider applying the same key bridging technique to AES-256. The main problem we faced when trying to apply this technique to AES-256 is the fact that our attack, like many other attacks on AES, require that the last round is without MixColumns. This can be easily justified by the fact that one can switch the order of the last MixColumns with the AddRoundKey (with the appropriate change to the last subkey from k_i to u_i). While knowing a full column from u_i allows computing the respective column of k_i, in our attack on AES-256, the bytes from u_6 that are guessed (along with the full u_7), give one byte of u_{-1} in the third column. Unfortunately, this is insufficient to gain information about a byte of k_{-1}.

6 Our New Attack on 7-round AES

In this section we present our new attack on 7-round AES. First we present the basic variant of the attack, which is used later as part of the 8-round attack. Then

we show how to improve the attack using alteration of the expected-probability differential and time/memory/data tradeoffs, such that the resulting time complexity will be lower than the complexity of all previously known attacks on 7-round AES (in all its three flavors).

6.1 The Basic Attack

In this attack, the byte with non-zero difference in the expected-probability differential is byte 0, both in the input and in the output differences. The active byte of the δ-set and the byte that is checked in the state X_5 are taken to be byte 0 as well. The attack works similarly if these bytes are replaced by any other pair of bytes, as long as the correspondence between the differential and the δ-set is preserved.

The algorithm of the basic attack is as follows.

1. **Preprocessing phase:** Compute the 2^{127} possible values of the "special" multisets defined by Observations 4 and 5, and store them in a hash table.
2. **Online phase:**
   (a) **Phase A – Detecting the right pair:**
      i. Ask for the encryption of 2^{81} structures of 2^{32} plaintexts, such that in each structure, bytes $0, 5, 10, 15$ assume the 2^{32} possible values and the rest of the bytes are constant.
      ii. For each structure, store the ciphertexts in a hash table and look for pairs in which the difference in bytes $1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15$ is zero.[11] Since this is a 96-bit filtering, only 2^{48} pairs are expected to remain.
      iii. For each remaining pair, guess bytes $0, 5, 10, 15$ of k_{-1} and check whether the difference in the state X_1 is non-zero only in byte 0. For each key guess, about 2^{24} pairs are expected to remain for each key guess.
      iv. For each remaining pair, guess bytes $0, 7, 10, 13$ of u_6 and check whether the difference in the state X_5 is non-zero only in byte 0. For each key guess, only one pair is expected to remain.
   (b) **Phase B – Checking the δ-set**
      i. For each guess of the eight subkey bytes made in Phase A and for the corresponding pair, take one of the members of the pair, denote it by P^0, and find its δ-set using the knowledge of bytes $0, 5, 10, 15$ of k_{-1}. This can be done by considering the state X_1^0, XORing it with the 255 possible values which are non-zero only in byte 0, and decrypting the 255 obtained values through round 0 using the known subkey bytes. The resulting plaintexts are the other members of the δ-set.

[11] In the description of our attack we assume that the last round does not contain the MixColumns operation. If it does contain it, one can swap the order of the last round's MixColumns and AddRoundKey and apply the attack with the respective changes.

ii. Guess byte 0 of u_5, and using the knowledge of bytes $0, 7, 10, 13$ of u_6, partially decrypt the ciphertexts of the δ-set to obtain the multiset $[X^0_{5,0} \oplus X^0_{5,0}, X^1_{5,0} \oplus X^0_{5,0}, \ldots, X^{255}_{5,0} \oplus X^0_{5,0}]$.
iii. Check whether the multiset exists in the hash table. If not, discard the key guess (possibly using auxiliary techniques such as repetition of the attack with a different output byte).

(c) **Retrieving the rest of the key:** For each remaining key guess, retrieve the rest of the key by exhaustive key search.

It is clear that the time complexity of the online phase of the attack is dominated by encrypting 2^{113} plaintexts, and hence, the data and time complexity of this part of the attack is 2^{113}. The memory complexity is 2^{129} 128-bit blocks, since each multiset contains about 512 bits of information and its representation can be easily compressed into 512 bits of space. The time complexity of the preprocessing phase of the attack is approximately $2^{127} \cdot 2^8 \cdot 2^{-3} = 2^{132}$ encryptions.

6.2 Altering the Expected-Probability Differential

Our first improvement reduces the data and time complexities of the attack by a factor of 2^8 without affecting the memory requirements.

We observe that the time complexity of most components of the attack is significantly lower than the time required to encrypt the plaintexts. Therefore, a tradeoff that would decrease the data complexity, even at the price of increasing the time complexity of the other parts of the attack, may reduce its overall complexity.

Such tradeoff is achieved by slightly modifying the expected-probability differential used in the attack. Instead of requiring the input difference to be non-zero only in byte 0, we can allow the difference to be non-zero also in one of the bytes $5, 10, 15$. These bytes are chosen such that the number of possible differences in the state X_2 is not increased, and thus the memory complexity is preserved.

This change reduces the data complexity of the attack to 2^{105}, since it allows the adversary to use structures of size 2^{16} that contain 2^{31} pairs with the input difference of the differential. On the other hand, the change requires to guess four additional bytes of k_{-1} in order to detect the right pair (if the additional byte is byte 5, then the additional guessed bytes are $3, 4, 9, 14$). As a result, the number of pairs remaining after the first filtering step of the attack is increased to 2^{72} (instead of 2^{48}). For each such pair, there are 2^{24} possible values of 12 subkey bytes (8 bytes of k_{-1} and 4 bytes of u_6) for which that pair satisfies the expected-probability differential. As in the 8-round attack, these values can be found with time complexity of 2^{24} table look-ups for each pair, using the early abort technique. Thus, the time complexity of Phase A of the modified attack is 2^{96} table look-ups.

At Phase B, we observe that since the value of bytes $3, 4, 9, 14$ of k_{-1} is irrelevant to the examination of the δ-set, the phase has to be performed only

2^{16} times for each of the 2^{72} pairs (instead of 2^{24} times). Thus, its time complexity is $2^{72} \cdot 2^{16} \cdot 2^8 \cdot 2^8 \cdot 2^{-3} = 2^{101}$ encryptions. Therefore, the overall time complexity of the attack is still dominated by the encryption of the plaintexts, and thus both the data and the time complexity of the attack are reduced to 2^{105}.

6.3 Using Several Differentials in Parallel

Our second improvement further reduces the data and time complexities by a factor of 5 without affecting the memory requirements.

We observe that the data complexity can be reduced by using several differentials in parallel. Since there is no specialty in the choice of the active byte at the input and the output of the original differential, there are 256 possible differentials that can be used in parallel. In the basic 7-round attack this improvement leads to a data/memory tradeoff: The attack requires the "active" bytes of the δ-set to correspond to the non-zero difference bytes of the differential, and altering the active bytes of the δ-set requires preparing a different precomputed table for each choice of the bytes. As a result, the data complexity can be reduced by factor of up to 256, but the memory requirement is increased by the same factor. Since the memory complexity is the dominant one in the 7-round attack, this tradeoff is not profitable.

However, in the modified attack the data complexity can be reduced (though, by a small factor) without affecting the memory complexity. We observe that since the additional "active" byte in the expected-probability differential is not used in the analysis of the δ-set, it can be chosen without affecting the memory complexity. There are six possible ways to choose this byte (bytes $5, 10, 15$ in the input and bytes $1, 2, 3$ in the output), and five of them can be used in parallel with the same set of chosen plaintexts.[12] This reduces the data complexity of the attack by a factor of 5 without affecting the memory complexity. Since the time complexity is dominated by encrypting the plaintexts, it is also reduced by a factor of 5. Therefore, the data and time complexities of the modified attack are smaller than 2^{103}. In the sequel, we assume for the sake of simplicity that these complexities are equal to 2^{103}.

6.4 Time/Memory/Data Tradeoffs

Our third improvement is a fine tuning of the complexities using a simple tradeoff between data, time, and memory as proposed in [12]. In the preprocessing phase, we precompute the table only for some of the values, and then for each key guess, we perform the attack for several δ-sets in order to compensate for the missing part of the table. For each $n \geq 0$, this tradeoff decreases the memory

[12] In order to do it, the adversary considers structures of size 2^{96} each, in which bytes $1, 6, 11, 12$ are constant and the other bytes take all the 2^{96} possible values. This allows to use bytes 5 and 10 as the additional active byte in the input of the differential. All three additional bytes cannot be used in parallel, since this would require structures of size 2^{128}.

complexity and the time complexity of the preprocessing phase by a factor of 2^n, and increases the data complexity and the online time complexity by the same factor 2^n. The resulting complexities lie on the following tradeoff curve: Data complexity – 2^{103+n} chosen plaintexts, Time complexity – 2^{103+n} encryptions, Memory requirement – 2^{129-n} AES blocks, for any $n \geq 0$. Choosing $n = 13$, all the three complexities are equalized at 2^{116}, which is lower than the time complexities of all known attacks on 7-round AES, in all its three flavors (see Table 2).

7 Extension to Attacks on 8-round AES-192 and AES-256

In this section we present the first non-marginal attack on 8-round AES-192. The data complexity of the attack is 2^{113} chosen plaintexts, the memory requirement is 2^{129} 128-bit blocks, and the time complexity is 2^{172} encryptions. A variant of the attack can be applied to 8-round AES-256. The data and memory requirements remain unchanged, but the time complexity is increased to 2^{196} encryptions, since most of the key schedule considerations presented in Section 5.1 apply only to AES-192. We present the attack on AES-192; the attack on AES-256 is similar.

In the attack presented below, we choose the non-zero byte in the output difference of the expected-probability differential to be byte 1. Accordingly, the byte to be checked in the δ-set is also chosen as byte 1. This change is required in order to apply the key schedule considerations presented in Section 5.1. The only non-zero byte in the input difference of the differential and the only active byte of the δ-set can be still chosen arbitrarily, as long as they are the same. Without lose of generality, in the sequel we assume that this byte is byte 0.

A trivial generalization of the 7-round attack presented in Section 6 to eight rounds is to guess the full k_7, and for each guess, decrypt all the ciphertexts through the last round and apply the 7-round attack. While this generalization is sufficiently good for the basic Demirci-Selçuk attack where the data and time complexities of the online phase of the 7-round attack are low, in our attack it leads to an extremely high time complexity. Specifically, the first part of the online phase (namely, detecting the right pair) would require time complexity of $2^{113} \cdot 2^{128} = 2^{241}$ encryptions, which is significantly higher than the 2^{192} computations of exhaustive search.

Instead, we use an *early abort* technique that was described in [18]. We present here the technique only briefly, and refer the reader to [18] for the full details.

In the following, the adversary examines each of the $2^{113} \cdot 2^{31} = 2^{144}$ pairs separately, and her goal is to detect the subkey candidates for which that pair satisfies the expected-probability differential. Note that this approach differs from the usual approach where subkey material is guessed and for each guess of the subkey, the adversary obtains the corresponding right pairs.

Note that if (P^1, P^2) is a right pair, then the corresponding intermediate states $(X^1_{6(SR)}, X^2_{6(SR)})$ have non-zero difference only in bytes $3, 6, 9, 12$. Hence, in each column of $X_{6(SR)}$ there are only 2^8 possible differences. Since the Mix-

Columns and AddRoundKey operations are linear, this implies that in each column of X_7 there are only 2^8 possible differences, and thus only $2^{32} \cdot 2^8 = 2^{40}$ possible pairs of actual values. In the technique presented in [18], the adversary considers these 2^{40} pairs in advance, encrypts them through round 7, and stores the actual values before the last AddRoundKey operation in a hash table, sorted by the output difference. In the online phase of the attack, for each examined pair, the adversary considers each shifted column (e.g., bytes $0, 7, 10, 13$) independently, and accesses the hash table in the row corresponding to the ciphertext difference. It is expected that $2^{40} \cdot 2^{-32} = 2^8$ values appear in each row. Since the table gives the actual values before the AddRoundKey operation, and the ciphertexts are the values after that operation, each of the pairs in the table suggests one value for the 32-bit subkey corresponding to that shifted column.

Therefore, for each examined pair, and for each shifted column, the adversary obtains a list of 2^8 candidates for the 32-bit subkey corresponding to that column. In a basic variant of the attack, the adversary aggregates these suggestions to 2^{32} suggestions for the full k_7, and for each suggestion, she decrypts the ciphertext pair through round 7. Then she uses a similar precomputed table for round 6 to get a list of 2^8 possible values of bytes $3, 6, 9, 12$ of u_6. For each such value, the adversary checks whether the relations between bytes $3, 6$ of u_6 and the subkey k_7 described in Section 5.1 hold. If not, the subkey guess is discarded. Since this is a 16-bit filtering, the adversary is left with 2^{24} candidates for the full k_7 and bytes $3, 6, 9, 12$ of u_6. Finally, using a precomputed table also in round 0, the adversary obtains a list of 2^8 possible values of bytes $0, 5, 10, 15$ of k_{-1}. For each such value, the adversary checks whether the relation between byte 15 of k_{-1} and the subkey k_7 described in Section 5.1 holds. If not, the subkey guess is discarded. Since this is an 8-bit filtering, the adversary is left with 2^{24} candidates for the full k_7, bytes $3, 6, 9, 12$ of u_6, and bytes $0, 5, 10, 15$ of k_{-1}. For each of these candidates, (P^1, P^2) is a right pair w.r.t. the expected-probability differential, and the second-phase of the attack can be applied.

The time complexity of this procedure is 2^{40} simple operations for each examined pair, or $2^{144} \cdot 2^{40} \cdot 2^{-8} = 2^{176}$ encryptions in total.

The time complexity can be slightly reduced by using a more sophisticated precomputed table in order to check the consistency between bytes $3, 6$ of u_6 and the subkey k_7. The table takes bytes 3,6 of $MC^{-1}(X_6)$ in both pairs, along with bytes 2,3,5,6 of u_7, and returns the consistent values for bytes 3,6 of u_6, if there are any. The precomputation is done by trying all possible candidates for the pair of bytes for $MC^{-1}(X_6)$ along with the corresponding bytes of u_6, to see if the decrypted values satisfy the linear relation on the differences before the SubBytes operation of round 5. If this is the case, the entry corresponding to the $MC^{-1}(X_6)$ values and all subkeys of u_7 which satisfy the key relation is stored with the respective u_6 bytes. We note that for each key and each pair, there is probability of 2^{-8} that the condition is satisfied, and thus, only 2^{56} of the entries in the table are nonempty.

At the second part of the online phase of the attack, performed for each of the 2^{144} pairs (P^1, P^2) and each of the 2^{24} subkeys corresponding to the pair,

the adversary constructs a δ-set and checks whether the corresponding multiset appears in the table. Note that while in the 7-round attack this phase requires guessing an additional subkey byte (which is byte 13 of u_5), in this attack that subkey byte can be derived from the subkey k_7. The time complexity of the second part is $2^{168} \cdot 2^8 \cdot 2^{-4} = 2^{172}$ encryptions.

Therefore, the overall memory requirement of the attack is 2^{129} 128-bit blocks (as in the basic version of the 7-round attack), the data complexity is 2^{113} chosen plaintexts, and the time complexity is 2^{172} encryptions. These complexities improve significantly over the only previously known attack on AES-192, which is a Square attack [15] requiring almost the entire codebook and time complexity of 2^{188} encryptions.

8 Summary

In this paper we introduced three new cryptanalytic techniques which can be used to improve the best known complexities of all the known attacks on 7 and 8 round versions of AES, as detailed in Table 2. In particular, we describe the first real attack on 8-round AES-192 which does not use the full codebook in order to marginally improve the time complexity of exhaustive search. However, all our attacks have impractical complexities, and thus they do not endanger the security of any fielded system.

References

1. Behnam Bahrak and 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 and Mohammad Reza Aref, *Impossible Differential Attack on 7-round AES-128*, IET (IEE) J. on Information Security, Vol. 2, No. 2, pp. 28–32, 2008.
3. Eli Biham and Nathan Keller, *Cryptanalysis of Reduced Variants of Rijndael*, unpublished manuscript, 1999.
4. Eli Biham and Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer, 1993.
5. Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir, *Key Recovery Attacks of Practical Complexity on AES-256 Variants With Up To 10 Rounds*, Advances in Cryptography, proceedings of EUROCRYPT 2010, Lecture Notes in Computer 6110, pp. 299–319, Springer, 2010.
6. Alex Biryukov and Dmitry Khovratovich, *Related-Key Cryptanalysis of the Full AES-192 and AES-256*, Advances in Cryptography, proceedings of ASIACRYPT 2009, Lecture Notes in Computer Science 5912, pp. 1–18, Springer, 2009.
7. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic, *Distinguisher and Related-Key Attack on the Full AES-256*, Advances in Cryptography, proceedings of CRYPTO 2009, Lecture Notes in Computer Science 5677, pp. 231–249, Springer, 2009.

8. Alex Biryukov and David Wagner, *Slide Attacks*, proceedings of Fast Software Encryption 1999, Lecture Notes in Computer Science 1636, pp. 245–259, Springer, 1999.
9. Jung Hee Cheon, MunJu Kim, Kwangjo Kim, Jung-Yeun Lee, and 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.
10. Joan Daemen and Vincent Rijmen, *AES Proposal: Rijndael*, NIST AES proposal, 1998.
11. Joan Daemen and Vincent Rijmen, *The design of Rijndael: AES — the Advanced Encryption Standard*, Springer, 2002.
12. Hüseyin Demirci and Ali Aydin Selçuk, *A Meet-in-the-Middle Attack on 8-Round AES*, proceedings of Fast Software Encryption 2008, Lecture Notes in Computer Science 5086, pp. 116–126, Springer, 2008.
13. Hüseyin Demirci, Ihsan Taskin, Mustafa Çoban, and Adnan Baysal, *Improved Meet-in-the-Middle Attacks on AES*, proceedings of INDOCRYPT 2009, Lecture Notes in Computer Science 5922, pp. 144–156, Springer, 2009.
14. Orr Dunkelman and Nathan Keller, *A New Attack on the LEX Stream Cipher*, Advances in Cryptography, proceedings of ASIACRYPT 2008, Lecture Notes in Computer Science 5350, pp. 539–556, Springer, 2008.
15. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting, *Improved Cryptanalysis of Rijndael*, proceedings of Fast Software Encryption 2000, Lecture Notes in Computer Science 1978, pp. 213–230, Springer, 2001.
16. Henri Gilbert and 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, and Bart Preneel, *Related-Key Rectangle Attacks on Reduced AES-192 and AES-256*, proceedings of Fast Software Encryption 2007, Lecture Notes in Computer Science 4593, pp. 225–241, Springer, 2007.
18. Jiqiang Lu, Orr Dunkelman, Nathan Keller, and Jongsung Kim, *New Impossible Differential Attacks on AES*, proceedings of INDOCRYPT 2008, Lecture Notes in Computer Science 5365, pp. 279–293, Springer-Verlag, 2008.
19. 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.
20. Hamid Mala, Mohammad Dakhilalian, Vincent Rijmen, and Mahmoud Modarres-Hashemi, *Improved Impossible Differential Cryptanalysis of 7-Round AES-128*, proceedings of INDOCRYPT 2010, Lecture Notes in Computer Science 6498, pp. 282–291, Springer-Verlag, 2010.
21. US National Institute of Standards and Technology, *Advanced Encryption Standard*, Federal Information Processing Standards Publications No. 197, 2001.
22. 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.
23. Wentao Zhang, Wenling Wu, and 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, 2007.
24. Wentao Zhang, Wenling Wu, Lei Zhang, and 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, 2007.

A Analysis of the Meet-in-the-Middle Attack on 7-round AES proposed in [13]

For the sake of completeness, we present in this appendix a detailed analysis of the improved meet-in-the-middle attack on 7-round AES proposed in [13] and show that the time complexity of this attack is much higher than that of exhaustive key search (for AES-128) or higher than claimed (for AES-192 and AES-256).

The attack of [13] is based on several improvements of the observations used in [12]:

1. The number of parameters that determine the values of the examined byte in the output of 4-round AES can be reduced from 25 to 24 by picking some x_0, and considering the augmented function $f'_{c_1,\ldots,c_{24}}(x) = f_{c_1,\ldots,c_{25}}(x) - f_{c_1,\ldots,c_{25}}(x_0)$. This improvement is used in our attack as well.
2. The number of parameters can be further reduced to 15, under a restriction on the plaintexts that holds with probability 2^{-72}. In order to find a δ-set that satisfies the restriction, the authors suggest to repeat the attack for 2^{72} different δ-sets. We note that this improvement is equivalent to the time/memory tradeoff presented in [12] that suggested to prepare the precomputed table only for some values of the 25 parameters and compensate for it by repeating the attack with more sets of plaintexts. Actually, the proposal of [13] is a partial case of the time/memory tradeoff, where the precomputed table is prepared only for those $2^{120} = 2^{192} \cdot 2^{-72}$ values of the parameters which satisfy the 72-bit restriction. Another equivalent suggestion would be to fix 9 of the 24 constants to zero. Thus, this suggestion does not improve over [12].
3. The time complexity and the memory requirements can be slightly reduced by keeping only 32 of the ciphertext values corresponding to a δ-set, instead of all the 256 values. This improvement is not used in our attack since it cannot be applied simultaneously with our multiset tabulation technique, and the gain of the multiset tabulation technique is greater than that of this improvement.

The attack algorithm in [13] is essentially similar to that of the basic attack in [12] and thus is omitted here.

The authors analyse the attack and conclude that the data complexity is 2^{80} chosen plaintexts, the time complexity of the online phase is 2^{113} encryptions, the memory complexity is 2^{122} 128-bit blocks, and the time complexity of the preprocessing is 2^{123} encryptions. Unfortunately, there is a flaw in the analysis. The exact flaw is in the time complexity of Steps (5)–(6) of the attack. The authors write:

> In the key search phase, for every combination of K_{final}, we do partial decryption over 2^{80} ciphertexts which makes 2^{120} partial decryptions and for every combination of K_{init} and $K_{11}^{(1)}$, we do partial encryption over 2^{80} plaintexts which makes 2^{120} partial encryptions ... Therefore the processing complexity of the attack is comparable to 2^{113} encryptions.

The complexity described by the authors is indeed the complexity of Steps (3)–(4) of the algorithm. However, the time complexity of the matching phase (Steps (5)–(6)) that is not mentioned in the analysis is much higher. Since the matching phase has to be performed for every combination of guesses of K_{final} (bytes 0, 7, 10, and 13 of k_7 and byte 0 of u_6 in this paper's notations), K_{init} (bytes 0,5,10, and 15 of k_{-1} in our notations), and $K_{11}^{(1)}$ (byte 0 of k_0), the equivalent of , its time complexity is at least $2^{40} \cdot 2^{40} \cdot 2^{80} = 2^{160}$ operations, which is much higher than claimed (and suppresses exhaustive key search time for AES-128). Hence, the improved attack presented in [13] cannot be considered a valid attack on 7-round AES-128.

We note that in [13], the authors also present an extension of the collision attack presented by Gilbert and Minier [16]. Since this extension is not used by the authors to mount an attack on AES, we do not discuss it here.

Rounds	Key Size	Complexity				Attack Type & Source
		Data (CP)	Memory	Time	MinMax*	
7	128	$2^{112.2}$	$2^{112.2}$	$2^{117.2}$ MA	$2^{117.2}$	Impossible Differential [18]
		$2^{90.4}$	2^{106}	$2^{117.2}$ MA	$2^{117.2}$	Impossible Differential [20]
		$\mathbf{2}^{103+n}$	$\mathbf{2}^{129-n}$	$\mathbf{2}^{103+n}$	$\mathbf{2}^{116}$	**Our Results (Sect. 6)**
	192	$19 \cdot 2^{32}$	$19 \cdot 2^{32}$	2^{155}	2^{155}	SQUARE [15]
		2^{46+n}	2^{192-n}	2^{94+n}	2^{143}	Meet in the Middle [12]
		$2^{91.2}$	$2^{139.2}$	2^{101}	$2^{139.2}$	Impossible Differential [18]
		$2^{113.8}$	$2^{113.8}$	$2^{118.8}$ MA	$2^{118.8}$	Impossible Differential [18]
		$\mathbf{2}^{103+n}$	$\mathbf{2}^{129-n}$	$\mathbf{2}^{103+n}$	$\mathbf{2}^{116}$	**Our Results (Sect 6)**
	256	$21 \cdot 2^{32}$	$21 \cdot 2^{32}$	2^{172}	2^{172}	SQUARE [15]
		2^{34+n}	2^{204-n}	2^{82+n}	2^{143}	Meet in the Middle [12]
		2^{92}	2^{125}	2^{163} MA	2^{163}	Impossible Differential [18]
		$2^{113.8}$	$2^{113.8}$	$2^{118.8}$ MA	$2^{118.8}$	Impossible Differential [18]
		$\mathbf{2}^{103+n}$	$\mathbf{2}^{129-n}$	$\mathbf{2}^{103+n}$	$\mathbf{2}^{116}$	**Our Results (Sect 6)**
8	192	$2^{127.997}$	2^{128}	2^{188}	2^{188}	SQUARE [15]
		$\mathbf{2}^{113+n}$	$\mathbf{2}^{129-n}$	$\mathbf{2}^{172+n}$	$\mathbf{2}^{172}$	**Our Results (Sect. 7)**
	256	2^{34+n}	2^{206-n}	$2^{205.6+n}$	$2^{205.8}$	Meet in the Middle [12]†
		$2^{34+\max(n-24,0)}$	2^{208-n}	2^{206+n} MA	2^{208}	Meet in the Middle [13]‡
		$2^{89.1}$	2^{97}	$2^{229.7}$ MA	$2^{229.7}$	Impossible Differential [18]
		$2^{127.997}$	2^{128}	2^{204}	2^{204}	SQUARE [15]
		$\mathbf{2}^{113+n}$	$\mathbf{2}^{129-n}$	$\mathbf{2}^{196+n}$	$\mathbf{2}^{196}$	**Our Results (Sect. 7)**

* — the lowest time complexity which exceeds the other complexities via the tradeoff option (if such a tradeoff exists).

† — [12] estimates the cost of partial encryption as 2^{-8} of an encryption. As there are at least six columns which take part in the partial encryption/decryption, we believe that $2^{-2.4}$ is a more accurate estimate.

‡ — The complexity is higher than claimed in [13] due to a flaw in the analysis.

CP – Chosen plaintext. MA – Memory Accesses.

Time complexity measures the online time in encryption units unless mentioned otherwise.

Memory complexity is measured in AES blocks.

Table 2. A Comparison of Previous Results with Our New Attacks