Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds

Alex Biryukov[1], Orr Dunkelman[2], Nathan Keller[3], Dmitry Khovratovich[1], and Adi Shamir[4]

[1] University of Luxembourg, Luxembourg
[2] École Normale Supérieure,
45 rue d'Ulm, 75230 Paris, France.
[3] Einstein Institute of Mathematics, Hebrew University.
Jerusalem 91904, Israel
[4] Computer Science department
The Weizmann Institute
Rehovot 76100, Israel

Abstract. AES is the best known and most widely used block cipher. Its three versions (AES-128, AES-192, and AES-256) differ in their key sizes (128 bits, 192 bits and 256 bits) and in their number of rounds (10, 12, and 14, respectively). In the case of AES-128, there is no known attack which is faster than the 2^{128} complexity of exhaustive search. However, AES-192 and AES-256 were recently shown to be breakable by attacks which require 2^{176} and 2^{119} time, respectively. While these complexities are much faster than exhaustive search, they are completely non-practical, and do not seem to pose any real threat to the security of AES-based systems.
In this paper we describe several attacks which can break *with practical complexity* variants of AES-256 whose number of rounds are comparable to that of AES-128. One of our attacks uses only two related keys and 2^{39} time to recover the complete 256-bit key of a 9-round version of AES-256 (the best previous attack on this variant required 4 related keys and 2^{120} time). Another attack can break a 10 round version of AES-256 in 2^{45} time, but it uses a stronger type of *related subkey attack* (the best previous attack on this variant required 64 related keys and 2^{172} time). While neither AES-128 nor AES-256 can be directly broken by these attacks, the fact that their hybrid (which combines the smaller number of rounds from AES-128 along with the larger key size from AES-256) can be broken with such a low complexity raises serious concern about the remaining safety margin offered by the AES family of cryptosystems.

1 Introduction

AES (Advanced Encryption Standard) is an iterated block cipher which was selected by NIST in October 2000 after a three year competition. It was made a national and international standard, and replaced DES as the most widely deployed block cipher in both software and hardware applications.

The three standardized versions of AES are called AES-128, AES-192, and AES-256. They differ from each other in the key length (128, 192, and 256 bits) and the number of rounds (10, 12, and 14). Their data encryption rounds are all the same, but the details of the key schedule (which turns the given key into a sequence of subkeys for the various encryption rounds) are slightly different since different amounts of key material are available and required in the three variants. Their security was thoroughly analyzed by the NSA, which declared in 2003 that none of them has any known vulnerability and that the longer-key variants AES-192 and AES-256 can be used to protect top secret US governmental data [10].

The situation started to change in the spring of 2009, when Biryukov, Khovratovich and Nikolić [4] found a key recovery attack on AES-256 with time complexity of 2^{131}. The attack was completely non-practical, but it was the first time that anyone had published an attack on the full AES cipher which was faster than exhaustive search. Shortly afterwards, Biryukov and Khovratovich [3] reduced the time complexity of the attack on AES-256 to 2^{119}, and described the first attack on AES-192 which was faster than exhaustive search (requiring 2^{176} instead of 2^{192} time). As a result, AES is no longer considered to be theoretically secure, but the crucial question facing all of us is how far it is from becoming practically insecure.

The practicality of various types of cryptanalytic attacks depends on many factors: Attacks based on few ciphertexts are better than attacks that require many ciphertexts, known plaintext attacks are better than chosen plaintext attacks, nonadaptive attacks are better than adaptive attacks, single key attacks are better than related key attacks, etc. Since it is difficult to quantify the relative importance of all these factors in different scenarios, we usually concentrate on the total running time of the attack, which is a single well defined number. While one can argue about the exact transition point between cryptanalytic attacks of practical and theoretical time complexity, it is reasonable to place it at around 2^{64} basic instructions. Since optimized AES implementations require about 16 clock cycles per byte and each plaintext has 16 bytes, this is approximately equal to 2^{56} AES encryptions. This choice of threshold is supported by the fact that 2^{55} evaluations of DES were carried out on special purpose hardware several years ago, while a collision finding attack on SHA-1 which was expected to take 2^{61} time is still at an early stage of a large distributed effort.

To try to estimate the security margin left in a given cryptosystem, we can take two different approaches. One of them is to compare the time complexity of the best known attack on the full cryptographic scheme with this threshold. This was the approach that motivated [4] and [3], and in this sense AES still seems to be very secure. However, attacks can only get better over time, and in particular they tend to exhibit "round creep" that slowly increases the number of rounds which can be attacked with practical complexity. A second approach (and the one taken in this paper) is thus to compare this number to the total number of rounds in the iterated cryptosystem. An example which demonstrates the difference between these two approaches is the comparison between Serpent and Rijndael made during the AES competition. The best attacks on their full versions had exactly the same time complexity (namely, that of exhaustive search). However, Rijndael was designed to be as fast as possible (with a relatively small security margin), whereas Serpent was designed to have a large security margin (at the expense of speed on some platforms), which made it more resistant against future cryptanalytic developments. As an extreme example, we would feel very uncomfortable using a theoretically secure n round block cipher if we knew that its $n-1$ round version can be attacked with practical complexity, since such a scheme is "one idea away from disaster".

What we show in this paper is that this type of security margin in AES is dramatically smaller than generally believed. In particular, we describe several key derivation attacks of practical complexity on AES-256 when its number of rounds is reduced to approximately that of AES-128. The best previously published attacks on such variants were far from practical, requiring 4 related keys and 2^{120} time to break a 9-round version of AES-256 [9], and 64 related keys and 2^{172} time to break a 10-round version of AES-256 ([9], see also [2]).[1] In this paper we describe an attack on 9-round AES-256 which can find its complete 256-bit key in 2^{39} time by using only the simplest type of related keys (in which the chosen plaintexts are encrypted under two keys whose XOR difference can be chosen in many different ways). Our best attack on 10-round AES-256 requires only two keys and 2^{45} time, but it uses a stronger type of *related subkey attack*. These attacks can be extended into a quasi-practical 2^{70} attack on 11-round AES, and into a trivial 2^{26} attack on 8-round AES. The attacks are particularly well suited to counter modes of operation (AES-CTR), since the attacker can get all the chosen plaintexts he needs by starting from just two chosen initial values and running the counter mode in a natural way.

This paper is organized as follows. In Section 2 we describe the AES cipher and its different versions, and in Section 3 we discuss various types of related key attacks. Our attacks on 9-round variants of AES-256 are described in Section 4, and our attacks on 10-round variants of AES-256 are described in Section 5. In Section 6 we briefly outline several other attacks on variants of AES-256 with a smaller or larger number of rounds, and in Section 7 we describe how to choose more naturally looking plaintexts in our attack. We summarize the complexity of all our attacks in Section 8, and conclude with a discussion of our results and a list of open problems in Section 9.

[1] For comparison, the best practical single-key attack on AES-256 is a SQUARE attack on 6 rounds which requires $6 \cdot 2^{32}$ chosen plaintexts and has time complexity of 2^{44} [7].

2 Description of AES

AES-256 is an iterated block cipher which encrypts 128-bit plaintexts with 256-bit keys. It has 14 rounds, where each round applies four basic operations in the following order:

– SubBytes (SB) is a nonlinear byte-wise substitution that applies the same 8×8 S-box to every byte.
– ShiftRows (SR) is a cyclic shift of the i'th row by i bytes to the left.
– MixColumns (MC) is a matrix multiplication over a finite field applied to each column.
– AddRoundKey (ARK) is an exclusive-or with the round subkey.

Before the first round an additional whitening ARK operation is performed, and in the last round the MC operation is omitted. AES-128 and AES-192 use exactly the same round function, but the number of rounds is reduced to 10 and 12, respectively.

Next we describe the key schedule of AES-256. The supplied 256-bit key is divided into 8 words of 32 bits each $(W[0], W[1], \cdots, W[7])$. To generate the 15 subkeys of 128 bits (which consist of 60 words of 32 bits), the following algorithm is used:

– For $i = 8$ till $i = 59$ do the following:
  - If $i \equiv 0 \bmod 8$, then $W[i] = W[i-8] \oplus \mathrm{SB}(\mathrm{RotByte}(W[i-1])) \oplus \mathrm{Rcon}[i/8]$,
  - If $i \equiv 4 \bmod 8$, then $W[i] = W[i-8] \oplus \mathrm{SB}(W[i-1]))$,
  - Else $W[i] = W[i-8] \oplus W[i-1]$,

where RotByte represents one byte rotation (i.e., $(a_0, a_1, a_2, a_3) \rightarrow (a_1, a_2, a_3, a_0)$), and *Rcon* denotes an array of fixed constants.

The key schedules of AES-128 and AES-192 are slightly different, since they have to apply more mixing operations to the shorter key in order to produce the slightly smaller number of subkeys for the various rounds. This small difference in the key schedules plays a major role in making AES-256 more vulnerable to our attacks, in spite of its longer key and supposedly higher security.

For more details about all aspects of the design of AES, we refer the reader to [6].

2.1 Our Notation

The rounds of AES-256 are numbered $0, 1, \ldots, 13$. The subkey used at the end of round i is denoted by K^i, and the whitening subkey is denoted by K^{-1}. The XOR difference between the subkeys K^i produced by two related keys is denoted by $\Delta(K^i)$. The 128-bit state at the input to round i is denoted by I^i. The XOR difference between the states I^i produced during two related encryptions is denoted by $\Delta(I^i)$.

Any 128-bit intermediate state or subkey is described by a 4×4 byte matrix, whose bytes are numbered as described in Figure 1. The rows and the columns of this matrix are numbered 0,1,2,3. Byte j of subkey K^i or of state I^i is denoted by K^i_j or I^i_j, respectively. Similarly, a difference in this byte is denoted by $\Delta(K^i_j)$ or $\Delta(I^i_j)$, respectively. When we want to refer to more than one byte, we list the relevant bytes in the subscript, as in $\Delta(I^i_{j,k,l,m})$.

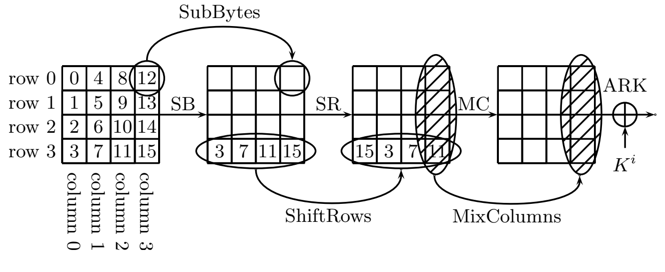


Fig. 1. The AES round function and byte marking conventions.

3 Related-Key Attacks

The related-key attack model [1] is a class of cryptanalytic attacks in which the attacker knows or chooses a relation between several keys and is given access to encryption/decryption functions with all these keys. The goal of the attacker is to find the actual keys. The relation between the keys can be an arbitrary bijective function R (or even a family of such functions) chosen in advance by the attacker. In the simplest form of this attack, this relation is just an XOR with a constant: $K_2 = K_1 \oplus C$, where the constant C is chosen by the attacker. This type of relation allows the attacker to trace the propagation of XOR differences induced by the key difference C through the key schedule of the cipher. However, more complex forms of this attack allow other (possibly non-linear) relations between the keys. For example, in some of the attacks described in this paper the attacker chooses a desired XOR relation in the second subkey, and then defines the implied relation between the actual keys as: $K_2 = F^{-1}(F(K_1) \oplus C) = R_C(K_1)$ where F represents a single round of the AES-256 key schedule, and the constant C is chosen by the attacker. We call such attacks *related subkey attacks*, and we emphasize that once the second key is computed via the relation, all its subkeys are computed in a normal way, consistent with the full evolution of the key schedule.

The choice of the relation between secret keys gives additional power to the attacker compared to other cryptanalytic attacks in which the attacker can manipulate only the plaintexts and/or the ciphertexts. The simpler the relation is, the easier it is for an adversary to manipulate the key in the desired fashion. For example, the key exchange protocol 2PKDP [12], allows an attacker to XOR the unknown key with a constant. Other related key attacks, such as those presented in [8, 11], discuss practical attacks on well known schemes under different key relations.

Even though related-key attacks may not be a realistic threat in many cryptographic applications, resistance to such attacks is an important design goal for new block ciphers, and in fact it was one of the stated design goals of the Rijndael algorithm, which was selected as the Advanced Encryption Standard. Designers usually try to build primitives which can be automatically used without further analysis in the widest possible set of applications, protocols, or modes of operation. Moreover, history shows that users of cryptography tend to use (or misuse) such cryptographic primitives in very creative manners, forcing the cryptographers to design schemes which resemble ideal primitives under the broadest possible set of scenarios.

4 Attacks On 9 Round Variants of AES-256

4.1 A Related-Key (XOR Difference) Attack

In this section we present an attack on 9-round AES-256, which is based on the simplest form of a related key attack, in which plaintexts can be encrypted under two unknown keys, whose XOR difference can be chosen by the attacker. The number of chosen plaintexts used in the attack is 2^{38}, and the most time consuming part of the attack is to ask the legitimate user to prepare all their 2^{39} corresponding ciphertexts under the two keys. Once this is done, the derivation of the complete 256-bit key requires less than 2^{39} time, and in this sense we can say that attacking the scheme is faster than using it...

4.1.1 The Related-Key Differentials Used in Our Attacks The basic differential characteristic we use is an 8-round differential, whose first seven rounds are part of the longer differential path introduced in [4], but its eighth round is different (see Figure 2). Since we use a related key attack, each one of the characteristics is a combination of a key characteristic and a state characteristic, which are intertwined in a way that maximizes their total probability. We first describe the key difference part of the characteristics (which is independent of the data), and how it generates the various subkey differences we would like to have.

Let a and b be *any two 32-bit words*. When we choose the key difference of the differential as $\Delta(K) = (b, b, b, b, a, 0, a, 0)$, we can easily show that the ten 128-bit subkeys used in the 9-round version of AES-256 (including the initial whitening subkey) have the following difference structure with probability 1:

$$(b,b,b,b||a,0,a,0)$$
$$(b,0,b,0||a,a,0,0)$$
$$(b,b,0,0||a,0,0,0)$$
$$(b,0,0,0||a,a,a,a)$$
$$(c,c,c,c||d,e,d,e),$$

where

$$c = b \oplus SB(RotByte(a)), \qquad d = a \oplus SB(c), \qquad e = d \oplus a,$$

and each row describes the differences of two additional subkeys, starting with the original key difference in the first row. This is an amazingly long trail for a key schedule which tries to be nonlinear, and it clearly indicates that this part of the design of AES-256 is seriously flawed.

To create the desired cancellations between this key characteristic and the state characteristic, we have to impose additional constraints on the choice of the two words a and b. Let α be any non-zero byte value (there are 255 ways to choose it, and each choice will lead to a different related key attack of the same complexity). By the construction of the SubBytes operation, there exists a unique byte value β such that the differential $\alpha \to \beta$ through the SubBytes operation holds with probability 2^{-6} (see [6]). Let b be the 32-bit column vector $b = MC((\beta,0,0,0)^T)$, and let a be the 32-bit column vector $a = (\alpha,0,0,0)^T$. Note that with this choice, the top three bytes of c are equal to the corresponding known values in b, and thus the only effect of the nonlinearity of the SubBytes operation on the subkey differences is to make the lowest byte of c and the four bytes of d unknown to the attacker. We denote these bytes by c_3 and (d_0,d_1,d_2,d_3).

The input difference in the state part of the characteristic is (b,b,b,b) (i.e., the same input difference b in each column). The subkey difference $\Delta(K^{-1})$ used in the whitening phase cancels the identical plaintext difference, and thus the difference $\Delta(I^0)$ is zero. The zero difference remains unchanged until the ARK operation at the end of round 0. The subkey difference $\Delta(K^0)$ inserts difference α into two bytes of the state. With probability 2^{-12}, this difference evolves through the SB operation to difference β, that is transformed through the MC operation to b, which is then cancelled with the subkey difference $\Delta(K^1)$, resulting in $\Delta(I^2)=0$. The zero difference is preserved, until the subkey difference $\Delta(K^2)$ inserts difference α into two bytes of the state. This difference is again cancelled with probability 2^{-12} with the subkey difference $\Delta(K^3)$, resulting in $\Delta(I^4)=0$. The subkey difference $\Delta(K^4)$ inserts difference α into one byte of the state, that is cancelled with the key difference $\Delta(K^5)$ with probability 2^{-6}. The zero difference is preserved again, until the subkey difference $\Delta(K^6)$ inserts difference α in four bytes of the state. With probability 2^{-24}, this difference evolves to difference (b,b,b,b) after the MC operation of round 7. Since the subkey difference $\Delta(K^7)$ is (c,c,c,c), we have $\Delta(I^8)=(f,f,f,f)$, where $f = b \oplus c = (0,0,0,b_3 \oplus c_3)^T$. This is the output difference of the differential. Overall, the differential $(b,b,b,b) \to (f,f,f,f)$ for rounds 0–7 holds with probability 2^{-54} for the subkey differences presented above. The differential characteristic is depicted in Figure 2.

In our attack we do not use this basic differential characteristic as is, but rather several "truncated" variants in which some of the differential conditions are relaxed:

1. **Main Differential.** In this differential we relax the differential conditions on three of the four active S-boxes in round 7, and leave only the condition in the SB operation in byte 0. That is, we require that the output difference of the SB operation in byte 0 is β, and do not restrict the output differences of the SB operation in bytes 4,8,12. As a result, the difference in the first column after the MC operation is b, and thus $\Delta(I^8_{0,1,2})=0$. The differences in the other columns, as well as $\Delta(I^8_3)$, are unknown. The probability of this truncated differential is 2^{-36}.
2. **Shifted Main Differential.** This truncated differential is almost identical to the previous one. The only difference is that we keep the differential condition in byte 12 of round 7, instead of byte 0.
3. **Complementary Differential for the 9-Round Attack.** In this differential we consider only rounds 0–6, and relax the differential condition in round 5. Since the input difference to round 5 is non-zero only in byte I^5_0, and there is no differential condition, the difference $\Delta(I^6)$ is in the entire first column (bytes $I^6_{0,1,2,3}$). This difference evolves to differences in all the 16 bytes in $\Delta(I^7)$, but

since the MC operation is linear, there is a total of only 256 possible differences in the four bytes of each column. The probability of this truncated differential is 2^{-24}.

4. **Differential for the 8-Round Attack.** For the sake of completeness, we also describe a simplified differential which is used later in Section 6.1 to attack an 8-round version of AES-256 with a lower complexity. In this differential we consider rounds 0–7, and relax all the differential conditions in round 7. Since the difference $\Delta(I^7)$ is non-zero only in bytes 0,4,8,12, and since in the 8-round variant of AES-256 there is no MC operation at round 7, the ciphertext difference in bytes 1,2,5,6,9,10,13,14 is known to the attacker (it is equal to the difference chosen in the respective bytes of b). This supplies the attacker with a 64-bit filtering, which will be sufficient to discard all the wrong pairs. The probability of this truncated differential is 2^{-30}.

4.1.2 The 9-Round Attack We now describe several simple properties of the round function of AES, which are exploited by our attack.

1. **Observation A.** Consider a pair that satisfies the main differential described in Section 4.1.1. As noted above, the difference in columns 1,2,3 after the MC operation of round 7 is unknown. However, due to the properties of the SB and MC operations, there are only 127 possible values for the difference in each of these columns. Moreover, these 127 differences assume 127 different values in each of the four bytes. As a result, if the attacker knows the difference in one byte, she can obtain immediately the difference in the other three bytes of the same column, along with a one-bit filtering (since only 127 out of the 256 byte values are possible differences). Similarly, if a pair satisfies the complementary differential described in Section 4.1.1, then if the attacker knows the difference $\Delta(I^7)$ in one byte, she can obtain immediately the difference in the other three bytes of the same column (though, without the additional filtering since in this case there are 256 possible differences).
2. **Observation B.** Given an input and an output difference to the SB operation, there are three possibilities:
   (a) With probability 1/256, there exist four pairs of actual values that satisfy both the input and the output difference.
   (b) With probability 126/256, there exist two such pairs.
   (c) With probability 129/256, there exist no such pairs, and thus the impossible input/output difference pair can be discarded immediately.

   When there exist possible pairs, they can be found immediately by a table look-up. In order to do this, the attacker prepares in advance the difference distribution table which stores for each possible input/output difference, the actual values of the pairs satisfying these differences. The time required to prepare the table is 2^{24} evaluations of SB, and the required memory is 2^{17} bytes, which are completely practical. Each look-up operation in this table allows the attacker to either discard the input/output difference pair (with probability $\approx 1/2$), or to find immediately the actual input and output values of the two bytes (with a small expected number of possibilities).
3. **Observation C.** Consider the subkey differences between the related-keys used in the attack. It turns out that the unknown difference bytes c_3, d_0, d_1, d_2, d_3 can take on only 127 out of the 256 possible values, and (except for d_3), these 127 values are known to the attacker in advance. Indeed, since by the key schedule, $c = b \oplus SB(RotByte(a))$, the attacker knows that $x = c_3 \oplus b_3$ is one of the 127 differences such that the differential $\alpha \rightarrow x$ through SB is possible. Since b_3 is known to the attacker, the 127 possible values of c_3 are also known. Similarly, since $d = a \oplus SB(c)$ and $(c_0, c_1, c_2) = (b_0, b_1, b_2)$ are known to the attacker, she can find the 127 possible values for each of the bytes d_0, d_1, d_2.

4.1.3 A Detailed Description of the Attack. We now present the actual algorithm used by the attacker to derive the key information from the given ciphertexts, along with some textual explanations:

1. **Data Generation.**
   (a) Choose 2^{37} arbitrary plaintexts P, and add to them the 2^{37} chosen plaintexts $P' = P \oplus (b, b, b, b)$. Ask the user to encrypt each one of these 2^{38} plaintexts under the two unknown keys K and $K' = K \oplus (b, b, b, b, a, 0, a, 0)$. Each one of the 2^{37} choices of P provides two different pairs of

encryption operations $(((P,K),(P',K'))$ and $((P,K'),(P',K)))$ which have the desired input difference $\Delta(P)=(b,b,b,b)$, along with the desired key difference $\Delta(K)=(b,b,b,b,a,0,a,0)$, and thus we get a total of 2^{38} such pairs from 2^{38} plaintexts using 2^{39} total time. In the sequel we treat each pair of corresponding ciphertexts (C,C') as if it is a "right pair" with respect to the main differential presented in Section 4.1.1 (i.e., assume that it satisfies all the conditions in the differential).

(b) Insert the 2^{38} ciphertext pairs into a hash table indexed by the difference in the three bytes 0,10,13. We note that if a pair is a right pair, then since $\Delta(I^8_{0,1,2})=0$, the ciphertext difference in bytes $(0,13,10)$ is equal to (d_0,d_1,d_2). Hence, the pairs are divided into 2^{24} sets according to the possible values of (d_0,d_1,d_2), and the attack is sequentially applied to each set (which contains $2^{38-24}=2^{14}$ pairs on average).

(c) Note that by Observation C, only $127^3\approx 2^{21}$ of the values (d_0,d_1,d_2) are possible, and hence the rest of the attack can be applied only to the $2^{24-3}=2^{21}$ possible sets.

2. **First Filtering Step.** For each set of pairs corresponding to a possible value of (d_0,d_1,d_2), perform the following operations:
   (a) Guess the byte K^8_{12} and partially decrypt the ciphertext pairs through the last round to get $\Delta(I^8_{12})$. (Note that $\Delta(K^8_{12})=d_0\oplus\alpha$ is now known to the attacker). Check whether the obtained difference $\Delta(I^8_{12})$ is possible (see Observation A). Half of the pairs are expected to pass this filtering.
   (b) Use the difference $\Delta(I^8_{12})$ to find the differences $\Delta(I^8_{13,14})$ (see Observation A). Using the ciphertext difference in bytes 9,6, find the input and output differences to the SB operation in bytes 13,14. (Note that the corresponding key differences, $\Delta(K^8_6)=d_2$ and $\Delta(K^8_9)=d_1$, are now known to the attacker). Check whether this input/output difference pair is possible, and if it is possible, retrieve the corresponding actual values of the pairs (see Observation B). This is a two-bit filtering, and hence $2^{14}\cdot 2^{-1}\cdot 2^{-2}=2^{11}$ pairs are expected to remain, and each pair suggests two pairs of actual values on average for each byte. Each such pair of actual values can be combined with the corresponding ciphertext pair to get a suggestion for the subkey bytes K^8_6 or K^8_9.
   (c) Go over all the 2^{16} possible values of $K^8_{6,9}$ and check how many times each value is suggested. Discard all the values which are suggested fewer than four times. Note that the correct value is suggested by all the right pairs, and hence it gets at least four suggestions with probability about 0.57. On the other hand, the probability that a wrong subkey is suggested at least four times is approximately $\binom{2^{13}}{4}\cdot 2^{-64}=(2/3)\cdot 2^{-16}$. Hence, less than one subkey suggestion remains on average (If no subkey suggestions remain, which happens most of the time, the set is discarded).
   (d) Discard all the pairs that lead to a "wrong" value of $K^8_{6,9}$ (as we said before, all the right pairs suggest the correct value). Only a few wrong pairs (at most four) are expected to remain, and only these pairs are considered in the rest of the attack.
3. **Second Filtering Step.** This step is actually a repetition of Step 2 with a different column. For each remaining value of (d_0,d_1,d_2,K^8_{12}) and the corresponding remaining pairs, perform the following:
   (a) Guess the subkey byte K^8_8 and partially decrypt the ciphertext pairs through the last round to obtain $\Delta(I^8_8)$. Use this difference to retrieve $\Delta(I^8_{9,10})$ (or discard the pair if the difference is impossible). Using the ciphertext difference in bytes 5,2, find the input and output differences to the SB operation in bytes 9,10. Check whether this input/output difference pair is possible, and if it is, retrieve the corresponding actual values of the pairs and use them to get suggestions for the subkey $K^8_{5,2}$. Go over all the 2^{16} possible values of $K^8_{5,2}$ and check how many times each value is suggested. Discard all the subkey values that are suggested fewer than four times.
   (b) At this stage, the probability that a wrong subkey is suggested at least four times is extremely low (about 2^{-64}), and hence it is expected that all the wrong values of $(d_0,d_1,d_2,K^8_{12},K^8_8)$ will be discarded. Thus, the attacker obtains the right pairs, and the correct values of d_0,d_1,d_2, and six of the sixteen bytes of K^8 (namely, $=K^8_{2,5,6,8,9,12}$).
4. **Retrieving the Rest of K^8.**
   (a) For each remaining pair, guess the values c_3 and d_3. Using the known input and output differences of the SB operations applied to bytes 3,7,11,15 of round 8, obtain suggestions for the subkey

bytes $K^8_{3,7,11,15}$. Discard all the values that are suggested fewer than four times. Since only four pairs are expected to remain at this stage, only the correct subkey value is likely to remain, along with the correct values of c_3 and d_3.

(b) Repeat Steps 1,2,3 of the attack with the *shifted main differential* (presented in Section 4.1.1) instead of the main differential used until this step, and guessing the subkeys $K^8_{0,4}$ instead of $K^8_{8,12}$. This retrieves the subkey bytes $K^8_{0,1,4,10,13,14}$, which completes the derivation of K^8. Note that the time complexity of this step is significantly lower than the time complexity of Steps 1,2,3, since the correct values of d_0, d_1, and d_2 are already known to the attacker.

5. **Retrieving K^7.** At this stage the attacker already knows the full value of the last subkey K^8, and hence can peel off the last round. Note that the "ciphertexts" referred to in this part are actually the outputs of round 7.

(a) For the right pairs with respect to the main differential, guess the value of $K^7_{0,4,8,12}$ and partially decrypt the ciphertexts through round 7 to obtain $\Delta(I^7_{0,4,8,12})$. Check whether $\Delta(I^7_{0,4,8,12}) = (\alpha, \alpha, \alpha, \alpha)$. If not, discard the guessed key bytes. Since all the right pairs suggest the correct subkey value, only the correct value is likely to remain. Note that since this step can be performed in each byte independently, its time complexity is negligible.

(b) Use the key schedule to obtain two possible values for each of the bytes $K^7_{13,14,15}$. (In the key schedule algorithm, the column value $K^7_{12,13,14,15}$ is the input to SB operations for which both the input and output differences are known. Hence, two suggestions for the actual value can be obtained by Observation B).

(c) This step uses the *complementary differential* presented in Section 4.1.1. Consider any subset of 2^{26} pairs of plaintexts amongst the 2^{38} pairs used in the attack (using more pairs at this stage will be a waste of time). For each pair, assume that it is a right pair with respect to the complementary differential.

i. Partially decrypt the ciphertext pairs through round 7 to obtain the differences $\Delta(I^7_{0,1,4,6,8,11})$. Note that the partial decryption is possible since the subkey bytes $K^7_{0,4,8,13,14,15}$ are already known to the attacker.

ii. Consider each of Columns 0,1,2 of $\Delta(I^7)$ separately, and check whether the difference in the two "known" bytes (e.g., bytes 0,1 in Column 0) agrees with one of the column differences which are possible for right pairs w.r.t. the complementary differential (as explained in Section 4.1.1, there is a total of 256 such differences in each column). In each of the columns, this yields an 8-bit filtering, and hence about four pairs are expected to pass to the next step.

iii. For each of the remaining pairs, use the known differences $\Delta(I^7_{0,4,8,12})$ to retrieve the full difference $\Delta(I^7)$ (see Observation A). Then, use the input and output differences to all the SB operations in round 7 to obtain two suggestions for the actual values in each byte, and use these suggestions to get two suggestions for each byte of the subkey K^7. Discard all the subkey values that are suggested fewer than three times. Since all the right pairs suggest the correct subkey value, it is easy to show that only the correct subkey values are likely to remain.

(d) At this stage the attacker knows the full values of K^7 and K^8, and hence she can find the original 256-bit key K by running the (invertible) key schedule of AES-256 backwards from the known values of these two consecutive subkeys.

The Complexity of the Attack. The data complexity is 2^{38} chosen plaintexts (composed of 2^{37} arbitrary plaintexts P along with their 2^{37} companions P'). It is easy to see that the most time-consuming step of the attack is Step 2(c), that takes $2^{21} \cdot 2^8 \cdot 2^{16} = 2^{45}$ simple table lookup operations. To make a fair comparison, we have to remember that each 9-round AES-256 encryption requires $9 \times 16 = 144 \approx 2^7$ SB operations, and thus the 2^{45} simple operations required to carry out the complete attack are likely to be faster than the 2^{39} encryption operations required to prepare all the ciphertexts. The RAM memory requirements of the attack are negligible.[2] The probability of success is about 57% (required for the success of Step 2(c)), but can be made arbitrarily high by using additional pairs.

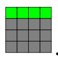
[2] The data can be stored on a hard disk, as we need to write it once, and read it once, while operating each time on a small amount of plaintext/ciphertext pairs which easily fit 1 MByte of memory.

4.2 A Related-Subkey Attack

If we relax the conditions on the key relation, a more efficient attack can be obtained. Recall that the XOR condition we imposed on the key in the previous attack directly implied the same XOR condition on the subkeys K^{-1} and K^0 which were used in the whitening phase and in the first round. In the new attack, we impose a XOR condition on the two subkeys K^0 and K^1 used in the first and second round. Let us define $\Delta(K^0) = (b, b, b, b)$ and $\Delta(K^1) = (a, 0, a, 0)$, where a and b are the same as in the previous attack. The key difference $\Delta(K)$ can then be defined by running the key schedule backwards as (f, a, a, a, b, b, b, b), where f is a full-column unknown difference.

A differential characteristic for 9 rounds, based on this key difference, is depicted in Figure 3 in the center. It contains 13 active S-boxes in the state. The input of the first four S-boxes and the output of the last four S-boxes are not specified, and the other S-boxes yield the desired values with probability 2^{-6}. Therefore, the plaintext difference ΔP is specified in 9 bytes, and the ciphertext difference ΔC is specified in 12 bytes.

The best attack runs in the chosen-ciphertext scenario as follows:

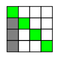
1. Prepare two structures of 2^{31} ciphertexts each, whose active bytes are located in row 0, and where byte 0 takes on only 128 possible values. The constants differ according to ΔC .
2. Decrypt the structures with K and K', respectively.
3. Select all the plaintext pairs that satisfy ΔP.
4. Every candidate pair proposes two candidates for each of six key bytes (K_4^{-1}), (K_8^{-1}), (K_{12}^{-1}), $(K_{12}^0), (K_{14}^0), (K_{15}^0)$.

Let us compute the number of right pairs. Of 2^{62} possible pairs, about 2^{30} pairs survive the 32-bit filter in the last round, producing a zero difference in I^7. The five S-boxes, whose inputs are specified, reduce the number of right pairs to one. The single right pair can be detected because the fixed 72-bit plaintext difference filters out all the 2^{62} wrong ciphertext pairs. The plaintext difference in bytes 1, 2, 3, 4, 8, 12 proposes two candidates for each one of the corresponding key bytes. As a result, we can recover 56 bits of the key using only 2^{32} time and 2^{32} chosen ciphertexts.

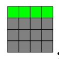
5 Attacks On 10 Round Variants of AES-256

In this section we describe two attacks on the 10-round variant of AES-256. Both of them are based on a key relation which is defined by imposing a fixed difference on two consecutive subkeys. We also have to start from an odd round so that the 10-round attack is run on rounds 1–10 of AES-256. We define $\Delta(K^2) = (b, b, b, b)$ and $\Delta(K^3) = (a, 0, a, 0)$, where a and b are the same values defined in the previous attacks. As a result, column 0 of $\Delta(K^1)$ and byte 0 of $\Delta(K^0)$ are not known to the attacker (see Figure 3, right).

5.1 Chosen-plaintext attack

A differential characteristic for 10 rounds, based on this key difference, contains 17 active S-boxes in the state. Compared to the 9-round differential characteristic, the last 7 rounds remain the same, and the input of three of the four active S-boxes in the second round is restricted. Therefore, 8 S-boxes behave as expected with probability 2^{-6}. The plaintext difference $\Delta(P)$ is specified in 12 bytes. .

The algorithm used by the attacker is as follows:

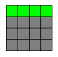
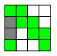
1. Prepare 2^{16} structures of 2^{32} plaintexts each, whose active bytes are located on the main diagonal. The constants differ according to $\Delta(P)$ and so that the total number of distinct plaintexts is 2^{48}.
2. Encrypt all the structures under both K and K'.
3. Select all the ciphertexts pairs that satisfy $\Delta(C)$ , and whose plaintexts belong to the same structure.

4. Every candidate pair proposes two candidates for each of six key bytes (K_4^{-1}), (K_8^{-1}), (K_{12}^{-1}), $(K_{12}^0), (K_{14}^0), (K_{15}^0)$.

Let us compute the number of right pairs. Of 2^{80} possible pairs about $2^{80-24} = 2^{56}$ pairs have zero difference in bytes 1,2,3 of I^2. The four S-boxes in round 2 can be used as a 26-bit filter, so 2^{30} pairs come out of the first two rounds. The next five S-boxes, whose inputs are specified, reduce the expected number of right pairs to one. The single right pair can be detected because the fixed 96-bit plaintext difference filters out all the 2^{80} candidate ciphertext pairs. As a result, we get a distinguisher whose total complexity is 2^{48} data, 2^{49} time, and 2^{33} memory.

5.2 Chosen-ciphertext attack

In this attack we relax the input to one of the S-boxes in round 2. As a result, the plaintext difference is specified in 8 bytes only, so a chosen-ciphertext attack is more practical in terms of the data requirements. Our best attack runs as follows:

1. Prepare 2^{12} structures of 2^{32} ciphertexts each, whose active bytes are located in row 0, and decrypt all the texts with K. The constants differ according to $\Delta(P)$ and so that the total number of distinct plaintexts would be 2^{44}.
2. Apply to each one of these ciphertexts the difference $\Delta(C)$  and decrypt all of them with the related key K'.
3. Select all the plaintext pairs that satisfy ΔP .
4. Every candidate pair proposes two candidates for each of five key bytes $(K_{12}^9), (K_0^{10})$, (K_4^{10}), (K_8^{10}), (K_{12}^{10}).

Let us compute the number of right pairs. Of 2^{76} possible pairs about 2^{44} pairs have zero difference in I^9. The seven S-boxes with input restrictions in rounds 2–10 provide a 42-bit filter, which reduces the number of right pairs to $2^{44-42} = 4$. The plaintext difference is specified in 64 bits, so $2^{76-64} = 2^{12}$ pairs come out of the last filter.

The false pairs are filtered at the bottom of the differential. We guess $\Delta(K_{12}^9)$ and thus derive the full $\Delta(K^{10})$. Then each candidate pair proposes sixteen 32-bit key candidates, or 2^{16} candidates in total. The probability that four wrong pairs propose the same values is 2^{-32}, and is still very low when we combine all the guesses. As a result, we get two candidates for each of the five key bytes, which provide 42 bits of information about the key. The total complexity of this attack is 2^{44} data, 2^{45} time, and 2^{33} memory.

5.3 Comments on Figures 2 and 3

Figures 2 and 3 depict differential characteristics that are used in the related-key attacks. Figure 3 deals with related-subkey attacks, when the differentials coincide in the last six rounds. For simplicity, we depict the common part of the related-subkey differentials only once for the 8-round differential characteristic. The other related-subkey differentials differ in the top rounds only, so we omit the 6.5-round part with seven active S-boxes.

In Figure 3 we also show how the data we start with is structured, and how the number of right pairs decreases during the encryption process. In the beginning of the attack, n STR above $k\,T$ stands for n structures with k texts each. Similarly, $k\,P$ stands for k right pairs, and n-bit F stands for an n-bit filter.

Colors. We extensively use colors while depicting differentials. We use the following color scheme in order to visually demonstrate the textual explanations:

🟩	Arbitrary difference		
🟪	Fixed S-box input difference		
🟦	Fixed S-box output difference		
🟥	Key schedule: the S-box application to 🟪		
▨▨▨▨	MixColumns expansion of 🟦	▦▦▦▦	Key schedule: the S-box application to ▨▨▨▨

6 Attacks On Other Variants of AES-256

6.1 A Related-Key Attack on 8 rounds

6.1.1 Distinguishing Attack The basic distinguishing attack in this case uses the simplified 8-round attack presented in Section 4.1.1. The attack itself is very simple: the attacker asks for the encryption of 2^{30} pairs of plaintexts with the input and key differences of the differential. For each pair, he checks whether the difference in bytes 1,5,9,13 of the ciphertext is equal to the known value of b_1, and whether the difference in bytes 2,6,10,14 of the ciphertext is equal to the known value of b_2. Since this is a 64-bit filter, for a random permutation all the pairs are likely to be discarded, while for an 8-round AES-256, one pair will remain with probability $1 - 1/e \approx 0.63$.

This efficient distinguishing attack was verified experimentally. We sampled 100 pairs of related keys (for a specific value of α and its corresponding β). For each such pair, we took 2^{32} random pairs with input difference (b, b, b, b), and encrypted them under the related keys. As the probability of the 7-round differential characteristic is 2^{-30}, the expected number of right pairs in each experiment is 4, where the actual number is distributed like a Poisson random variable with a mean value of 4.

In the 100 tests we performed, we encountered 10 tests with one right pair, 18 with two right pairs, 10 with three, 28 with four, 18 with five, 6 with six, eight with 7, one with 8, and even one with 12 right pairs. These values are very close to the expected values for sampling a Poisson random variable. We compare our results and the expected values in Table 1.

Right Pairs	0	1	2	3	4	5	6	7	8	9	10	11	12
Theory	1.8	7.3	14.7	19.5	19.5	15.6	10.4	6.0	3.0	1.3	0.5	0.2	0.06
Experiment 1 (random plaintexts)	0	10	18	10	28	18	6	8	1	0	0	0	1
Experiment 2 (counter mode)	1	3	17	19	23	18	4	4	8	2	1	0	0

Table 1. The Number of Right Pairs in 100 Experiments

This distinguisher can be used for key recovery by considering the remaining pair(s), guessing the subkey bytes $K^7_{0,4,8,12}$ and checking whether the difference $\Delta(I^7_{0,4,8,12})$ is equal to $(\alpha, \alpha, \alpha, \alpha)$, as predicted by the differential characteristic.

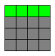
It is worth mentioning that we did a second experiment, where the plaintexts where chosen in a counter mode manner. In each experiment, we picked at random two related keys and two IVs satisfying the input difference, and tried 2^{32} consecutive counters. In the 100 experiments, we encountered a distribution which follows the same distribution, thus proving that these attacks can be applied even when counter mode is used. The exact distribution is given in Table 1 as Experiment 2.

The data and time complexities of the attack can be reduced to 2^{29} by using simple structures instead of pairs of plaintexts. We omit the details of this small optimization.

6.2 A Related-Subkey Attack on 8 Rounds

In this attack we consider AES-256 reduced to 8 rounds and starting from an odd round. We take the differential for the related-subkey attack on 9 rounds, cut the first round, and relax the input of the first two active S-boxes (Figure 3, left). As a result, the plaintext difference ΔP is specified in 8 bytes, and the ciphertext difference ΔC is specified in 12 bytes. There are three active S-boxes such that both their input and output differences are fixed.

The best attack runs in the chosen-ciphertext scenario as follows:

1. Prepare two structures of about $2^{25.5}$ ciphertexts, where bytes in row 0 are active, but do not run through all their possible values. The constants differ according to ΔC .
2. Encrypt structures with K and K', respectively.
3. Detect all the plaintext pairs that satisfy ΔP.
4. Every candidate pair proposes two candidates for each one of six key bytes (K_4^{-1}), (K_8^{-1}), (K_{12}^{-1}), $(K_{12}^0), (K_{14}^0), (K_{15}^0)$.

Let us compute the number of right pairs. Of 2^{51} possible pairs, about 2^{19} pairs survive the 32-bit filter in the last round, producing a zero difference in I^7. The three S-boxes, whose inputs are specified, reduce the number of right pairs to two. All the 2^{61} wrong pairs are discarded by the 64-bit plaintext difference filter.

The two right pairs provide information on the five bytes of the last subkeys (see the attack on 9 rounds), which are recovered after we guess $\Delta(K_0^8)$. As a result, we recover 35 bits of the subkey with $2^{26.5}$ time, data, and memory complexity.

6.3 Related-Subkey Attacks on 11 Rounds

The related-subkey differentials can be extended in several ways to 11 rounds. However, the best attacks we get are beyond the practical 2^{56} bound, so we only briefly sketch the underlying ideas.

6.3.1 Differential

6.3.1 Differential An 11-round differential (rounds 1–11) is obtained by adding one round to the bottom of the 10-round related-subkey differential (Figure 3, right). The final round is then similar to the last round of the 9-round related-key differential 4.1.1. We are flexible in the number of active S-boxes in round 10 whose output differences are restricted. As we fix more S-boxes, we get a better ciphertext filter, but a lower overall probability of the differential. We can also change the parity of rounds and start from round 0 instead of round 1; this makes the scenario more practical but adds one more unknown byte difference to the plaintext.

6.3.2 Attacks The following attacks can be applied to 11-round AES-256:

1. Start from an odd round and restrict the output difference of three active S-boxes in round 10. The data and time complexity would be about 2^{70}, and the last steps are done with non-trivial key ranking. About 50 bits of the subkey are recovered.
2. Restrict one more active S-box. Then the data and time complexities increase to 2^{76}, but it is easier to discard the wrong pairs.
3. Start from an even round (as in the original AES) and restrict three S-boxes in round 9 (former 10). The data complexity would be about 2^{70}, and the time complexity about 2^{75}.
4. Start from an even round and restrict two S-boxes. The data complexity drops to 2^{63}, but the time complexity increases to 2^{90} due to complicated filtering and key ranking.

7 Additional Results

We conclude with several additional results on reduced-round AES-256.

7.1 Tradeoffs Between the Complexity and the Number of Keys.

We can reduce the time complexity of the 8-round attack from 2^{29} to 2^{21} by using a larger number of related keys, arranged in a structure. We observe that the basic differential can be applied with $a = (\alpha, 0, 0, 0)^T$ and $b = MC((\beta, 0, 0, 0)^T)$ for any choice of α, β. For a random pair (a, b) of this form, the probability of the differential used in the 8-round attack (see Section 6.1) is 2^{-36} on average (more precisely, for 2^{-8} of the pairs the probability is 2^{-30}, for almost half of the pairs the probability is 2^{-35}, and for the rest the probability is zero). Let $S = \{MC((\beta, 0, 0, 0)^T) : \beta \in \{0, 1\}^8\}$. We consider structures of 2^{16} related keys, such that in each structure the keys are of the form $K \oplus (b, b, b, b, a, 0, a, 0)$, where $b \in S$ and $a = (\alpha, 0, 0, 0)^T$ for $\alpha \in \{0, 1\}^8$, and a, b go over all the possible values. The plaintexts are also slightly changed: in each structure, we start with a fixed plaintext P encrypted under the key K, and the modified plaintext encrypted under the key $K \oplus (b, b, b, b, a, 0, a, 0)$ is $P \oplus (b, b, b, b)$. Each such structure can be used to construct 2^{31} pairs, and hence 32 structures are sufficient for obtaining a right pair with probability 63%. Then, the attack proceeds like the 8-round attack presented in Section 6.1. In total, the attack requires 2^{13} plaintexts, each encrypted under 2^8 different keys. The total number of related-keys involved in the attack is 2^{16}, and the total time complexity of the attack is equivalent to 2^{21} encryptions.

7.2 The Choice of the Data and the Key Differences

The data and key differences in all the differentials we use depend on two byte values α and β, such that the differential $\alpha \rightarrow \beta$ through the SB transformation holds with probability 2^{-6}. For each value of α there exists a unique such value β, and vice versa. There are no other restrictions on α and β in our attacks, and in fact it is even possible to use other values of α and β for which the differential $\alpha \rightarrow \beta$ through the SB transformation holds with probability 2^{-7}. However, such choices will lead to slightly less efficient attacks.

By using this considerable freedom in the choice of α and β, the attacker can try to achieve several goals:

1. **Reducing the Hamming Weight of the Data and Key Differences.** Since in actual attacks the key difference is likely to be caused by applying physical faults to the encryption device, it is desirable to make the required changes as small as possible. The minimal possible Hamming Weight of the key difference is 24 bits, and it is obtained by taking $\alpha = 5, \beta = 8$ or $\alpha = A, \beta = 4$.
2. **Restricting the Plaintext Bytes to ASCII Characters.** Recall that in ASCII characters, the MSB in each byte is zero. Hence, if the plaintext difference used in the attack has value zero in the MSB of each byte, this increases the probability that if the initial plaintext consists of ASCII characters, the "modified" plaintext will also consist of ASCII characters. Such difference can be obtained by fixing the two MSBs of β to be zeros (e.g., $\beta = 8$).
3. **Adapting the Plaintext Difference to Numeric Characters.** By choosing $\beta = 01_x$, the plaintext difference in all its bytes is only in the two LSBs. As a result, we can choose numeric plaintexts whose modified versions also contain only numeric characters.

As a final comment, consider the case of an AES cryptosystem which is used in counter mode. Its plaintexts are defined by a fixed prefix, followed by a 64-bit counter. When we XOR our fixed difference to a sequence of 2^t such consecutive ciphertexts, we get another sequence of 2^t plaintexts which has the same structure (but with a different fixed prefix and a different counting order). Consequently, instead of repeatedly forcing the cryptosystem to encrypt our 2^t chosen plaintexts, we can just force it to start from two chosen starting points, and let the natural counting process in this mode of operation generate all the other ciphertexts we need in our attack.

8 Summary of the Attacks

All the attacks allow us to recover some key bytes (we expect that they can be extended to full key-recovery staying within the same complexity). We summarize our efforts in Table 2, where by Time we mean the number of AES encryptions, by Data we mean the number of different plaintexts or ciphertexts we use, by Memory we mean the size of our tables and counting arrays, CP stands for Chosen Plaintexts, and CC stands for Chosen ciphertexts.

Rounds	Scenario	Time	Data	Memory	Result
8	Key Diff.–CP	2^{31}	2^{31}	2	Distinguisher
8	Subkey Diff.– CC	$2^{26.5}$	$2^{26.5}$	$2^{26.5}$	35 subkey bits
9	Key Diff. – CP	2^{39}	2^{38}	2^{32}	Full key
9	Subkey Diff.-CC	2^{32}	2^{32}	2^{32}	56 key bits
10	Subkey Diff.-CP	2^{49}	2^{48}	2^{33}	Distinguisher
10	Subkey Diff.-CC	2^{45}	2^{44}	2^{33}	35 subkey bits

Table 2. Summary of attacks on AES-256.

9 Conclusions and Open Problems

This paper continues the rapid deterioration in the security of AES which took place during the last few months. The main problem seems to be the key schedule of AES-256, which is "not of industrial strength": It does not mix the initial key sufficiently, it is too linear, and as a result it has unusually long key differentials of probability 1. In addition, the similarity between the key schedule and the data encryption in AES makes it possible to repeatedly cancel data differences with corresponding key differences over many rounds. Ironically, the new attacks work best against AES-256 (which was supposed to be the strongest member of the AES family), and do not currently seem to work against AES-128.

The attacks described in this paper clearly have a practical time complexity. The number of chosen plaintexts they need is comparable to this time complexity, and can be considered practical when the attacker has the encryption device in his possession (e.g., in the form of a smart card). The most problematic aspect of the attack is its reliance on related keys, which is not universally accepted as a practical attack model. However, we believe that it is important to consider such attacks for several reasons: First of all, resistance against the largest possible variety of attacks should be an essential part of the certification process for new ciphers, even if these attacks do not seem to pose any immediate risk. In addition, AES was specifically designed to resist such attacks (see [5]), and its failure to do so raises serious doubts about its overall security. Finally, implementors should be aware of the possibility that such attacks exist, since they may become practical in some particular modes of operation (e.g., when the block cipher is employed in a MAC which uses the chosen input blocks as keys), or when some key bits can be flipped with a laser beam in a fault attack. The most disturbing aspect of the new attacks is that AES can no longer be considered as a safe black box construction, which can be dropped into any security application with little thought about how it is used.

The main problems left open by this paper are:

1. What are the best cryptanalytic attacks on other hybrid AES-like schemes which combine different numbers of rounds with different key sizes?
2. In particular, is there any attack on AES-128 which is faster than exhaustive search?
3. What are all the combinations of parameters which can be attacked with practical complexity?
4. Can we reduce the number of chosen plaintexts or ciphertexts which are required by the attacks described in this paper?
5. Can we modify the attack on 10 round AES-256 from the related subkey model into the more standard related key model?

References

1. Eli Biham. New types of cryptanalytic attacks using related keys. *J. Cryptology*, 7(4):229–246, 1994.
2. Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 507–525. Springer, 2005.
3. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256, 2009, available at `http://eprint.iacr.org/2009/317.pdf`.

4. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and related-key attack on the full AES-256. In *CRYPTO'09*, LNCS. Springer, 2009. to appear.
5. Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. NIST AES proposal, 1998.
6. Joan Daemen and Vincent Rijmen. *The Design of Rijndael. AES — the Advanced Encryption Standard.* Springer, 2002.
7. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of rijndael. In Bruce Schneier, editor, *FSE*, volume 1978 of *LNCS*, pages 213–230. Springer, 2000.
8. Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of rc4. In *Selected Areas in Cryptography*, volume 2259 of *LNCS*, pages 1–24. Springer, 2001.
9. Jongsung Kim, Seokhie Hong, and Bart Preneel. Related-key rectangle attacks on reduced AES-192 and AES-256. In *FSE'07*, volume 4593 of *LNCS*, pages 225–241. Springer, 2007.
10. National Security Agency (NSA). *National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information*, June 2003, available at `http://www.cnss.gov/Assets/pdf/cnssp_15_fs.pdf`.
11. Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. Breaking 104 bit wep in less than 60 seconds. In *WISA*, volume 4867 of *LNCS*, pages 188–202. Springer, 2007.
12. Gene Tsudik and Els Van Herreweghen. On simple and secure key distribution. In *ACM Conference on Computer and Communications Security*, pages 49–57, 1993.

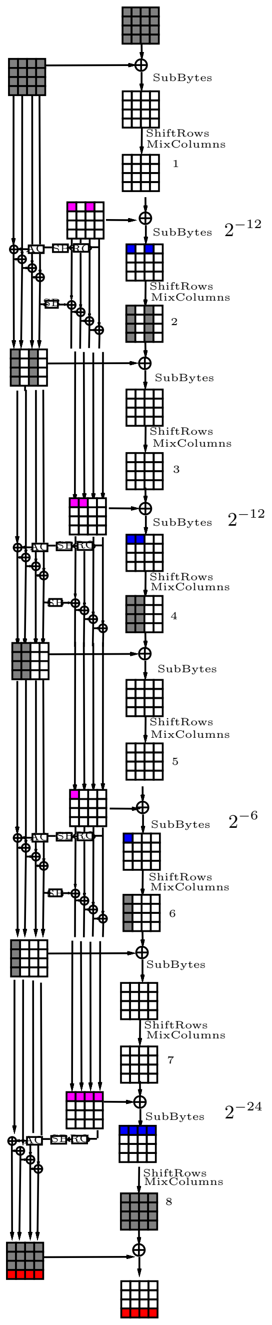


Fig. 2. Related-key differential for 8-round AES-256.

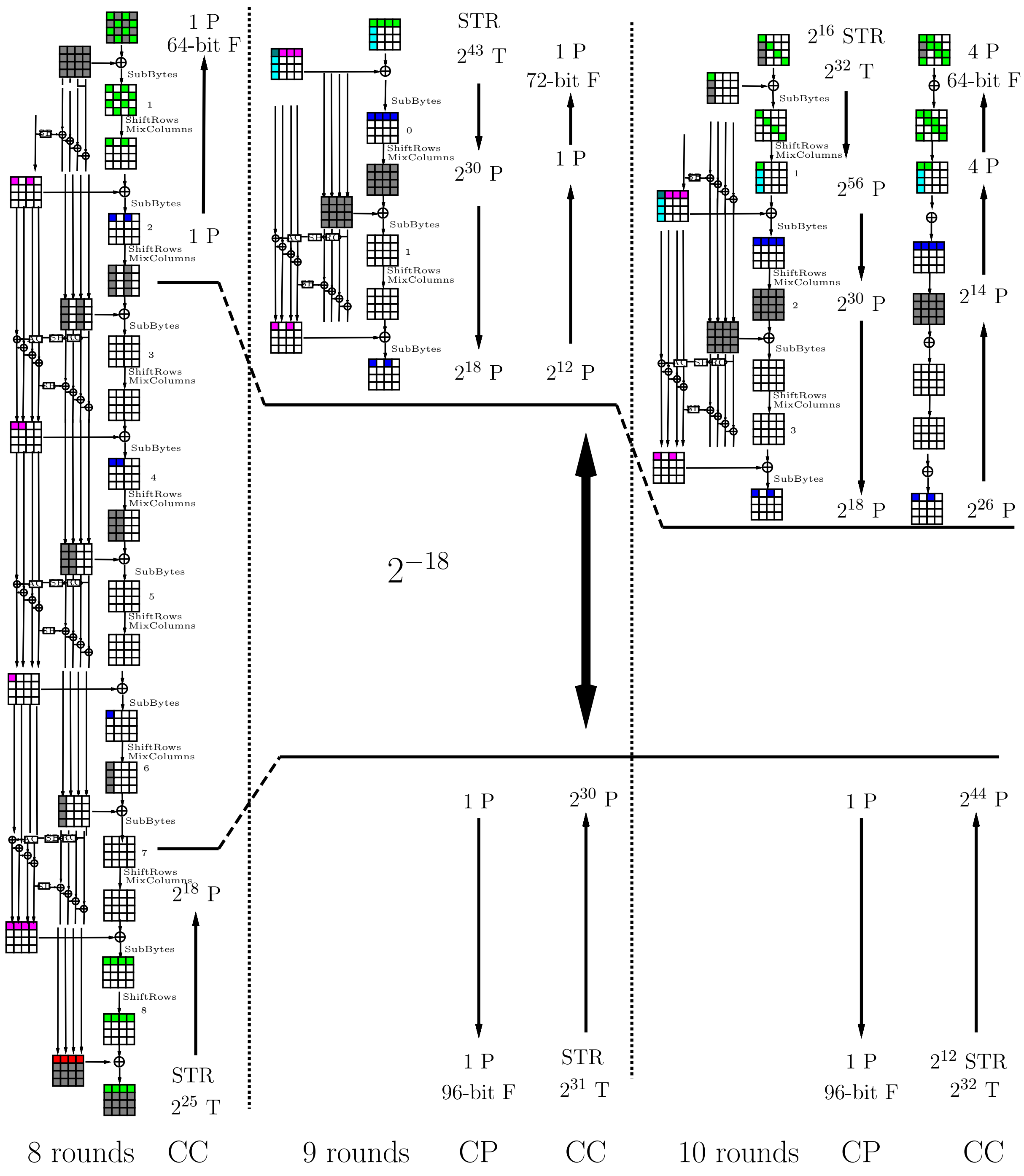


Fig. 3. Related-key attacks on 8-, 9- and 10-round AES-256.