Low Data Complexity Attacks on AES

Charles Bouillaguet[1], Patrick Derbez[1], Orr Dunkelman[2],
Nathan Keller[2,⋆], and Pierre-Alain Fouque[1]

[1] Département d'Informatique
École normale supérieure
45 Rue D'Ulm
75320 Paris, France
{charles.bouillaguet, patrick.derbez, pierre-alain.fouque}@ens.fr
[2] Faculty of Mathematics and Computer Science
Weizmann Institute of Science
P.O. Box 26, Rehovot 76100, Israel
{orr.dunkelman,nathan.keller}@weizmann.ac.il

Abstract. The majority of current attacks on reduced-round variants of block ciphers seeks to maximize the number of rounds that can be broken, using less data than the entire codebook and less time than exhaustive key search. In this paper, we pursue a different approach, restricting the data available to the adversary to a few plaintext/ciphertext pairs.
We show that consideration of such attacks (which received little attention in recent years) serves an important role in assessing the security of block ciphers and of other cryptographic primitives based on block ciphers. In particular, we show that these attacks can be leveraged to more complex attacks, either on the block cipher itself or on other primitives (e.g., stream ciphers, MACs, or hash functions) that use a small number of rounds of the block cipher as one of their components.
As a case study, we consider the AES — the most widely used block cipher, whose round function is used in various cryptographic primitives. We present attacks on up to four rounds of AES that require at most 10 known/chosen plaintexts. We then apply these attacks to cryptanalyze a variant of the stream cipher LEX, and to mount a new known plaintext attack on 6-round AES.
Keywords: AES, Cryptanalysis, Side Channel Attacks, Slide Attacks, LEX.

1 Introduction

The field of block cipher design has advanced greatly in the last two decades. New strategies of designing secure block ciphers were proposed, and following the increase in computing power, designers could offer larger security margins with reduced performance penalties. As a result, practical attacks on block ciphers became extremely rare, and even "certificational attacks" (that is, attacks which

⋆ The fourth author was partially supported by the Koshland center for basic research.

are not practical but are still faster than exhaustive key search on the full version of the cipher), are not very common.[1]

This led to two approaches in the cryptanalysis community. The first is to concentrate on attacking reduced-round variants of block ciphers, where the usual goal of the adversary is to maximize the number of rounds that can be broken, using less data than the entire codebook and less time than exhaustive key search. This approach usually leads to attacks with extremely high data and time complexities, e.g., [22, 38]. The second approach is to allow the adversary more degrees of freedom in his control. Examples of this approach are attacks requiring adaptive chosen plaintext and ciphertext queries, the related-key model, the related-subkey model, and even the known-key model [13, 28, 36]. This approach allows to achieve practical complexities even against widely used block ciphers such as the AES, but the practicality of the models themselves in real-life situations can be questioned.

Attacks following each of these approaches are of great importance, as they ensure that the block ciphers are strong enough, almost independently of the way in which they are deployed. Moreover, they help to establish the security margins offered by the cipher. A block cipher which is resistant to attacks when the adversary has a strong control and almost unrestricted resources offers larger security margins than a block cipher which does not possess this resistance.

At the same time, concentrating the cryptanalytic attention only on such attacks may prove insufficient to truly understand the security of the analyzed block cipher. It seems desirable to consider also other approaches, such as restricting the resources available to the adversary in order to adhere to "real-life" scenarios. For example, one may study the maximal number of rounds that can be broken with *practical* data and time complexity (as considered in [12] with respect to the related-key model).

In this paper we pursue this direction of research, but we concentrate on another restriction of the adversary's resources. In the attacks we consider, the time complexity is not restricted (besides the natural bound of exhaustive search), but the data complexity is restricted to only a few known or chosen plaintexts.

At first glance, this scenario may seem far fetched. However, it turns out that our scenario is very natural in the context of several classes of attacks:

1. **Slide attacks [15]:** This class of attacks is especially designed against block ciphers whose rounds are very similar to each other. The main feature of slide attacks is that they are independent of the number of rounds, and thus, the common countermeasure of increasing the number of rounds is not effective against them. Since most other attack techniques can be easily undermined by adding a few rounds, this makes the slide attacks one of the most powerful attacks against modern block cipher designs. The main idea of the slide attacks is to reduce the attack on the entire block cipher to an

[1] We note that in stream ciphers and hash functions, the situation is dramatically different. Several commonly used hash functions were practically broken in recent years [43, 40], and practical attacks on new stream cipher designs appear every several months [33, 42].

attack on a single round, where the data available to the adversary is only *two* known plaintext/ciphertext pairs. Hence, the scenario considered in our paper is exactly the one faced by the adversary in the slide attack.[2]

2. **Attacks based on fixed point properties [18]**: In this class of attacks, the adversary looks for a fixed point of *some part of* the encryption process. For such a fixed point, the cipher is reduced to a smaller variant, which can be (sometimes) attacked efficiently. Since usually the number of fixed points is extremely small (e.g., one or two), the adversary's goal is to attack a reduced-round variant of the cipher given a few known plaintexts.
3. **Side channel attacks:** In this class of attacks, the adversary has access to some information on the internal states during the encryption process. Usually, due to practical restrictions, the amount of data available to the adversary is extremely low. In the case where the information available to the adversary is the full intermediate state after a few rounds, the scenario the adversary faces is exactly the one considered in our paper.[3]
4. **Building block in more complex attacks:** As we demonstrate in this paper on the example of AES, an attack on 2-round AES with two known plaintexts can be leveraged to a known plaintext attack on 6-round AES. The attack uses a meet-in-the-middle approach combined with a low probability differential. We expect that such "leveraging" attacks are applicable against other block ciphers as well.
5. **Attacks on other primitives based on the block cipher:** In recent years, many designs of stream ciphers (e.g., Sosemanuk [3]), hash functions (like Hamsi [37]), MACs (e.g., ALPHA-MAC [21]), etc., use a small number of rounds of a block cipher as one of their components. Due to the restrictions of the environments where these designs are deployed, in some of them the information available to the adversary is only a few known or chosen plaintexts, and thus the attacks we consider can be used against these primitives.

Our concentration on attacks with an extremely small data complexity affects the techniques used in the attacks. The small amount of available data makes the classical statistical attacks, such as differential and linear cryptanalysis, almost irrelevant. Moreover, it turns out that even algebraic attacks and attacks based on SAT-solvers perform quite badly when the available data is very scarce. Instead, our attacks are based on the meet-in-the-middle approach, combined with differential-type ideas and vast exploitation of the key schedule of the analyzed block cipher.[4]

[2] We note that several variants of slide attacks suggested methods to increase the amount of data available to the adversary [16, 6, 31]. However, all these methods either require the knowledge of large portion of the codebook or perform in the more creative adaptively chosen plaintext model.

[3] We note that the complementary scenario, where the adversary has access to a small part of the internal state in multiple encryptions, was studied in [24].

[4] We note that it is possible to interpret meet in the middle attacks as trying to solve a set of (nonlinear) equations by trial substitution. Some of the tools we have used

In order to make our results concrete, we chose to concentrate on a single block cipher — the AES, the *Advanced Encryption Standard* [20]. AES is a 128-bit block cipher with a variable key length (128, 192, and 256-bit keys are supported). Since its selection, AES gradually became one of the most widely used block ciphers and received a great deal of cryptanalytic attention, both during the AES process, and even more after its selection. Moreover, a single AES round, or a small number of AES rounds, serve as a component in numerous designs of stream ciphers (e.g., LEX [10]), hash functions (e.g., ECHO [2]), and MACs (e.g., ALPHA-MAC [21]). Thus, we believe that the AES is a good case study to demonstrate our techniques, which (as we believe) can be used against other block ciphers as well.

We present several attacks on up to four rounds of AES requiring up to ten chosen plaintexts. Most of the attacks are based on the meet-in-the-middle approach. Some of the attacks exploit heavily the AES key schedule, while others apply even if the subkeys in AES are replaced by independent subkeys. The attacks are summarized in Table 1.

After presenting the low data complexity attacks, we show two applications:

1. **The best known-plaintext attack on 6-round AES:** We show that one of the attacks we present on 2-round AES can be leveraged to an attack on 6-round AES, using a low-probability differential. The resulting attack is the best known attack on 6-round AES in the known-plaintext model.
2. **Attack on a variant of the stream cipher LEX:** We consider a variant of the stream cipher LEX, in which instead of extracting 32 bits of the state after every AES round, the cipher outputs the entire state after every four rounds. In this case, the core cipher is reduced to 4-round AES (without the initial key whitening), but due to the stream cipher environment, the adversary is restricted to $2^{46.3}$ *known* plaintext bytes. While 4-round AES is considered very weak due to the Square attack which can break it with only 2^8 chosen plaintexts, all the previous attacks on AES have very high data complexity when transformed to the known plaintext model, and thus cannot be applied in our scenario. Thus, a priori, it is not clear whether this variant of LEX is less secure than the original LEX (on which the best known attack requires about 2^{40} known plaintext bytes and 2^{100} encryptions).
   We show that one of our attacks on 3-round AES can be used to break this variant with only 360 bytes of keystream and time complexity of 2^{40} encryptions, thus showing that this variant is practically insecure. Furthermore, we show that even a stronger variant in which the entire state is output only every 5 rounds, is still less secure than the original LEX, since a known plaintext variant of one of our attacks on 4-round AES can be used to break it with $2^{38.5}$ bytes of keystream and time complexity of 2^{80} encryptions.

The paper is organized as follows: Section 2 gives a description of AES. The attacks on one round, two rounds, three rounds, and four rounds are given in

in this paper followed this approach, but as the actual solution of the equations is not done using algebraic means, we distinguish between the algebraic nature of them and the actual attacks.

Attack Type	Number of Rounds	Complexity		
		Data	Time	Memory
SQUARE [30]	7	$2^{127.997}$ CP	2^{120}	$2^{127.997}$
Imp. Diff. [38]	7	$2^{112.2}$ CP	$2^{117.2}$ MA	$2^{112.2}$
MitM (Collision) [32]	7	2^{32} CP	2^{128}	2^{32}
MitM (Collision) [29]	7	2^{103+k} CP	2^{129-k}	2^{103+k}
MitM (Sect. 4)	1	1 KP	2^{40}	1
MitM (Sect. 4)	1	1 KP	2^{32}	2^{24}
Diff. (Sect. 4)	1	2 KP	2^{12}	1
MitM (Sect. 5)	2	1 KP	2^{80}	1
MitM (Sect. 5)	2	1 KP	2^{64}	2^{49}
Diff. (Sect. 5)	2	2 KP	2^{48}	1
Diff. (Sect. 5)	2	2 CP	2^{28}	1
Diff. (Sect. 5)	2	3 KP	2^{32}	1
MitM. (Sect. 6)	3	1 KP	2^{120}	1
MitM. (Sect. 6)	3	1 KP	2^{104}	2^{49}
Diff. (Sect. 6)	3	2 CP	2^{32}	1
Diff. Mitm (Sect. 6)	3	9 KP	2^{40}	2^{35}
Diff. MitM (Sect. 7)	4	2 CP	2^{104}	1
Diff. MitM (Sect. 7)	4	5 CP	2^{64}	2^{68}
Diff. MitM (Sect. 7)	4	10 CP	2^{40}	2^{43}

KP — Known plaintext, CP — Chosen plaintext,
MA — Memory Accesses
Time complexity is measured in encryption units unless mentioned otherwise.

Table 1. Summary of our Proposed Attacks on AES-128

Sections 4, 5, 6, 7, respectively. In Section 8 we demonstrate how one of the 2-round attacks can be leveraged to a known plaintext attack on 6-round AES. In Section 9 we apply our attacks on 3-round and 4-round AES to break variants of the stream cipher LEX. Finally, we conclude the paper in Section 10.

2 Description of AES

The Advanced Encryption Standard [20] 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

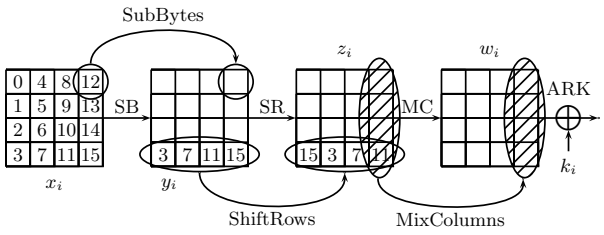


Fig. 1. An AES round

– AddRoundKey (ARK) — XORing the state with a 128-bit subkey.

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

Unlike all other works on AES, we deal with full-rounds variants, i.e., we shall not assume that the MixColumns operation is omitted from the last round. This model is the more appropriate one for the scenarios in which the attacks we consider may be applied, e.g., when the low data complexity attack is applied to a series of inner rounds in the encryption process.

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. We use the round numbers $1, \ldots, Nr$, where Nr is the number of rounds ($Nr \in \{10, 12, 14\}$). In this paper we consider only AES with 128-bit keys, and hence, AES stands for AES with 128-bit key and 10 rounds.

As we consider only AES with 128-bit key, we shall describe only its key schedule algorithm. The key schedule of the other variants can be found in [41]. The key schedule of AES-128 takes the user key and transforms it into 11 subkeys of 128 bits each. The subkey array is denoted by $W[0, \ldots, 43]$, where each word of $W[\cdot]$ consists of 32 bits. The first four 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 $i = 4, \ldots, 43$, do
  - If $i \equiv 0 \bmod 4$ then $W[i] = W[i-4] \oplus \mathit{RotByte}(SB(W[i-1])) \oplus \mathit{RCON}[i/4]$,
  - Otherwise $W[i] = W[i-1] \oplus W[i-4]$,

where $\mathit{RCON}[\cdot]$ is an array of predetermined constants, and $\mathit{RotByte}$ rotates the word by 8 bits to the right.

2.1 The Notations Used in the Paper

In our attacks we use the following notations: x_i denotes the input of round i, while y_i, w_i, and z_i denote the intermediate values after the application of SubBytes, ShiftRows, and MixColumns operations of round i, respectively. The plaintext is denoted by P, and the ciphertext is denoted by C.

We denote the subkey of round i by k_i, and the first (whitening) key by k_0, e.g., the subkey of the first round is k_1. In some cases, we are interested in interchanging the order of the MixColumns operation and the subkey addition. As these operations are linear they can be interchanged, by first XORing the data with an equivalent key and only then applying the MixColumns operation. We denote the equivalent subkey for the altered version by u_i, i.e., $u_i = MC^{-1}(k_i)$.

We denote bytes of some intermediate state x_i or a key k_i (or u_i) by integers between 0 and 15 according to their row and column, i.e., byte $x_{i,j+4\cdot\ell}$ is the byte in row j (for $j = 0, 1, 2, 3$) and column ℓ (for $\ell = 0, 1, 2, 3$) of x_i. We denote the z'th column of x_i by $x_{i,Col(z)}$, e.g., $u_{0,Col(0)} = MC^{-1}(k_{0,Col(0)})$. Similarly, by $x_{i,Col(y,z)}$ we denote columns y and z of x_i. We define two more column related sets. The first is $x_{i,SR(Col(z))}$ which is the bytes in x_i corresponding to the places after the ShiftRows operation on column z, e.g., $x_{i,SR(Col(0))}$ is composed of bytes 0,7,10,13. The second is $x_{i,SR^{-1}(Col(z))}$ which is the bytes in the positions of column z after applying the inverse ShiftRows operation.

3 Observations on the Structure of AES

In this section we present five observations on the structure of AES, that we use in our attacks. While the first three observations are well-known, the latter two are novel and far from being trivial. Hence, besides their uses in our attacks, they might be of use in future attacks on AES.

The first well-known observation considers the propagation of differences through SubBytes, which is the only non-linear operation in AES.

Observation 1 *Consider pairs $(\alpha \neq 0, \beta)$ of input/output differences for a single S-box in the SubBytes operation. For 129/256 of such pairs, the differential transition is impossible, i.e., there is no pair (x, y) such that $x \oplus y = \alpha$ and $SB(x) \oplus SB(y) = \beta$. For 126/256 of the pairs (α, β), there exist two ordered pairs (x, y) such that $x \oplus y = \alpha$ and $SB(x) \oplus SB(y) = \beta$, and for the remaining 1/256 of the pairs (α, β) there exist four ordered pairs (x, y) that satisfy the input/output differences. Moreover, the pairs (x, y) of actual input values corresponding to a given difference pattern (α, β) can be found instantly from the difference distribution table of the S-box. We recall that the time required to construct the table is 2^{16} evaluations of the S-box, and the memory required to store the table is about 2^{17} bytes.*

The second observation uses the linearity of the MixColumns operation, and follows immediately from the structure of the matrix used in MixColumns:

Observation 2 *Consider a pair (a, b) of 4-byte vectors, such that $a = MC(b)$, i.e., the input and the output of a MixColumns operation applied to one column. Denote $a = (a_0, a_1, a_2, a_3)$ and $b = (b_0, b_1, b_2, b_3)$ where a_i and b_j are byte values. The knowledge of any four out of the eight bytes $(a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3)$ is sufficient to uniquely determine the value of the remaining four bytes.*

The third observation is concerned with the key schedule of AES, and exploits the fact that most of the operations in the key schedule algorithm are linear. It allows the adversary to get relations between bytes of non-consecutive subkeys (e.g., k_r, k_{r+3} and k_{r+4}), while "skipping" the intermediate subkeys. The observation extends previous observations of the same nature made in [30, 26].

Observation 3 *Consider a series of consecutive subkeys $k_r, k_{r+1}, \ldots$, and denote $k_r = (a, b, c, d)$ and $v_r = RotBytes(SubBytes(k_{r,Col(3)})) \oplus RCON[r+1]$. Then, the subkeys $k_{r+1}, k_{r+2}, \ldots$ can be represented as linear combinations of (a, b, c, d) (the columns of k_r) and the 32-bit words $v_r, v_{r+1}, \ldots$, as shown in the following table:*

Round	$k_{Col(0)}$	$k_{Col(1)}$	$k_{Col(2)}$	$k_{Col(3)}$
r	a	b	c	d
$r+1$	$a \oplus v_r$	$a \oplus b \oplus v_r$	$a \oplus b \oplus c \oplus v_r$	$a \oplus b \oplus c \oplus d \oplus v_r$
$r+2$	$a \oplus v_r \oplus v_{r+1}$	$b \oplus v_{r+1}$	$a \oplus c \oplus v_r \oplus v_{r+1}$	$b \oplus d \oplus v_{r+1}$
$r+3$	$a \oplus v_r \oplus v_{r+1} \oplus v_{r+2}$	$a \oplus b \oplus v_r \oplus v_{r+2}$	$b \oplus c \oplus v_{r+1} \oplus v_{r+2}$	$c \oplus d \oplus v_{r+2}$
$r+4$	$a \oplus v_r \oplus v_{r+1} \oplus v_{r+2} \oplus v_{r+3}$	$b \oplus v_{r+1} \oplus v_{r+3}$	$c \oplus v_{r+2} \oplus v_{r+3}$	$d \oplus v_{r+3}$

As a result, we have the following useful relations between subkeys (for $i = 0, 1, 2, 3$):

1. $k_{r+2,i} \oplus k_{r+2,i+8} = k_{r,i+8}$,
2. $k_{r+2,i+4} \oplus k_{r+2,i+12} = k_{r,i+12}$,
3. $k_{r+2,i+4} \oplus v_{r+1,i} = k_{r,i+4}$,
4. $k_{r+4,i+12} \oplus v_{r+3,i} = k_{r,i+12}$,
5. $k_{r+3,i+12} = k_{r,i+8} \oplus k_{r,i+12} \oplus v_{r+2,i}$.

The next two observations concern a full round of AES including the AddRoundKey operation before the round itself, i.e., a sequence of ARK, SB, SR, MC, and ARK operations. Both observations show that the knowledge of a single column in one of the subkeys used in the ARK operations, along with the input and output values of the round, allows to retrieve the value of a few "unexpected" additional subkey bytes. Both observations were found by semi-automatic tools that tried to identify algebraic relations in 1-round AES.

These observations are the heart of our attacks on 1-round AES.

Assume that the input and output of this full round, denoted by P and C, respectively, are known, and denote the two subkeys used in the AddRoundKey operations by k_0 and k_1.

Observation 4 *The knowledge of P, C and the column $k_{0,Col(3)}$ allows retrieving two additional bytes of k_0, namely $k_{0,1}$ and $k_{0,8}$.*

The main idea behind this observation is that the linearity of the MixColumns operation and the almost linear key schedule of AES allows to apply the MixColumns operation to *the XOR of two columns*. The proof of the observation is presented in Appendix A.

We note that if the adversary knows also the value $k_{0,2}$, a similar strategy allows to retrieve the value $k_{0,6}$. This derivation is used in our first attack on 1-round AES with a single known plaintext in Section 4.2. We also note that similar statements hold if the adversary knows $Col(1)$ or $Col(2)$ of k_0.

We now now turn to the last observation.

Observation 5 *The knowledge of P, C, and the column $k_{1,Col(0)}$ allows to retrieve bytes $k_{0,7}$ and $k_{0,8}$ by a table look-up to a precomputed table of size 2^{24}. For each value of $k_{1,Col(0)}$, there are at most 12 values of $(k_{0,7}, k_{0,8})$, and on average a single value. The time complexity required to generate the table is 2^{32} operations.*

The proof of this observation is slightly more complex than the proof of the previous one, and is based on obtaining two nonlinear 8-bit relations involving two key bytes (given that other bytes of the key and the state are known). In such a case, we expect on average one solution to these equations (and as our test shows, in reality the maximal number of solutions is 12). Moreover, as there are 2^{32} possible systems, one can precompute the acceptable solutions and store them. The full proof is presented in Appendix A.

4 Attacks on One-Round AES

We start our analysis with the simplest case, an adversary who seeks to break one full round of AES (a sequence of ARK, SB, SR, MC, and ARK operations).

4.1 Two Known Plaintexts

If the data available to the adversary contains at least two known plaintexts, the attack takes 2^{12} encryptions.

The simplest attack starts by applying the $SR^{-1} \circ MC^{-1}$ to the ciphertext difference, to obtain the output differences of all the S-boxes. Since the input differences of the S-boxes are equal to the plaintext difference in the respective bytes, the adversary can consider each S-box independently, go over the 2^8 possible pairs of inputs whose difference equals the plaintext difference, and find the pairs suggesting the "correct" output difference. In each S-box, the expected number of suggested pairs is two, and each such pair gives a suggestion of one byte in the subkey k_0.[5] Thus, the adversary gets 2^{16} suggestions for the entire subkey k_0, which can be checked by trial encryption.

This attack, whose time complexity is 2^{16} encryptions, can be further improved using the relation between the subkeys k_0 and k_1. If the adversary checks the S-boxes in bytes $0, 5, 10$, and 15, she can use the $2^4 = 16$ suggestions of output

[5] We note that this step can be performed only if the differences in all S-boxes are non-zero. However, since in the known plaintext attack model it is common to assume that the plaintexts are chosen at random, it is expected that two known plaintexts have non-zero difference in all the 16 bytes with probability $(255/256)^{16} = 0.939$.

values of these S-boxes to get 16 suggestions for the column $k_{1,Col(0)}$, along with bytes $0, 5, 10, 15$ of k_0. Similarly, checking bytes $3, 4, 9, 14$ yields 16 suggestions for the column $k_{1,Col(1)}$, along with bytes $3, 4, 9, 14$ of k_0. Combining the suggestions, the adversary obtains 256 suggestions for two columns of k_1 and eight bytes of k_0. At this stage, the adversary can use the relation $k_{1,4} = k_{0,4} \oplus k_{1,0}$, which holds by the AES key schedule, as a consistency check. Only a single suggestion is expected to remain. The value of the remaining 8 bytes of k_0 can be obtained similarly by examining the other eight S-boxes. This improvement reduces the time complexity of the attack to 2^{12} S-box applications.

4.2 One Known Plaintext

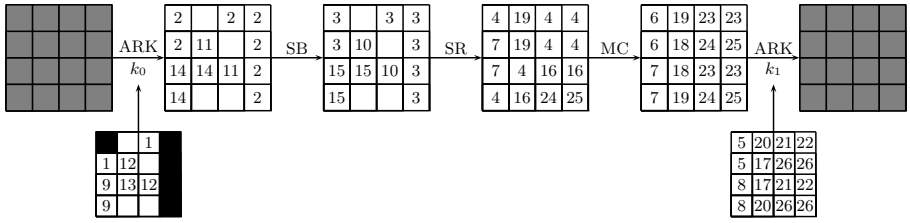
If the data available to the adversary is only a single plaintext, then the attack must use the relation between the two subkeys k_0 and k_1. (If the subkeys are independent, then the information available to the adversary is not sufficient to retrieve the key uniquely). Since any relation between the plaintext and the ciphertext involves the MixColumns operation, it seems likely that any such attack should require the guess of a full column, and thus have complexity of at least 2^{32} encryptions.[6]

We present two attacks — an attack with time complexity of 2^{40} encryptions and a negligible memory requirement, and an attack with time complexity of 2^{32} encryptions, and memory requirement of 2^{24} bytes. Thus, it seems that our attacks are close to reach the optimum for this variant.

The first attack, depicted in Figure 2, is based on Observation 4. The adversary guesses five bytes of the subkey k_0 — the column $k_{0,Col(3)}$ and an additional byte — $k_{0,0}$. The steps in the key derivation are shown in the figure, where the number in each cell denotes the step in which its value is retrieved. Steps 1 and 13 are based on Observation 4, steps $5, 9, 17, 21$, and 22 exploit the key schedule, steps $7, 19, 24$, and 25 are based on Observation 2, and the rest of the steps are performed using the application of known operations on known values.

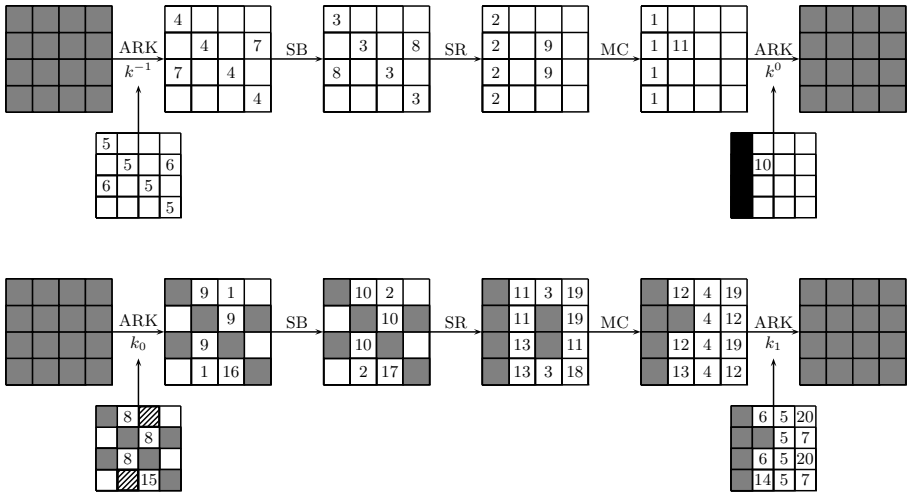
The second attack, depicted in Figure 3, is based on Observation 5. In the first phase of the attack, the adversary guesses the column $k_{1,Col(0)}$, and retrieves the value of seven additional subkey bytes. This phase is shown in the first row of the figure, where steps 6 and 10 are based on key schedule arguments, and the rest of the steps use the application of known operations on known values. The second phase of the attack, depicted in the second row of the figure, starts with retrieving the possible values of bytes $k_{0,7}$ and $k_{0,8}$ using Observation 5. Steps $6, 7, 8$, and 15 use the key schedule, steps 13 and 19 use Observation 2, and the rest of the steps follow the application of AES' operations to known values.

[6] We note that the problem of attacking one round AES without the MixColumns operation with a single known plaintext is studied in [25]. It is shown that 2^{16} encryptions are sufficient to retrieve the key.



Bytes marked by black are guessed, bytes marked by gray are known.

Fig. 2. A 2^{40} time attack on one round AES given one known plaintext



Bytes marked by gray are known (from previous steps of analysis). Bytes marked with tilted lines have at most 12 possible values by Observation 5.

Fig. 3. An attack on one round AES given one known plaintext with time complexity of 2^{32} and memory complexity of 2^{24}

5 Attacks on Two-Round AES

In this section we consider attacks on two rounds of AES, denoted by rounds 1 and 2. First we present attacks on two *full* rounds with two known plaintexts and with a single known plaintext. Then we present an improved attack with two known plaintexts that can be applied if the MixColumns operation in round 2 is omitted. This attack is used as a procedure in our attack on 6-round AES presented in Section 8.

5.1 Two Known Plaintexts

The attack with two known plaintexts, depicted in Figure 4, is based on Observation 1. As in the one-round attack with two known plaintexts, we observe that the ciphertext difference allows to retrieve the intermediate difference after the SubBytes operation of round 2. This observation is used in both phases of the attack. We also "swap" the order of the MixColumns and the AddRoundKey operations of the second round. This can be done since both operations are linear, as long as the subkey k_2 is replaced by the equivalent subkey $u_2 = MC^{-1}(k_2)$.

In the first phase of the attack, the adversary guesses bytes $0, 5, 10, 15$ of k_0, which allows him to retrieve the intermediate difference in $x_{2,Col(0)}$ (i.e., just before the SubBytes operation of round 2). Then, Observation 1 can be applied to the four S-boxes in that column, yielding their actual input/output values in both encryptions. This in turn allows obtaining $k_{1,Col(0)}$ (as the values before the ARK with k_1 are known). At this stage, the adversary tries to deduce and compute as many additional bytes as he can.

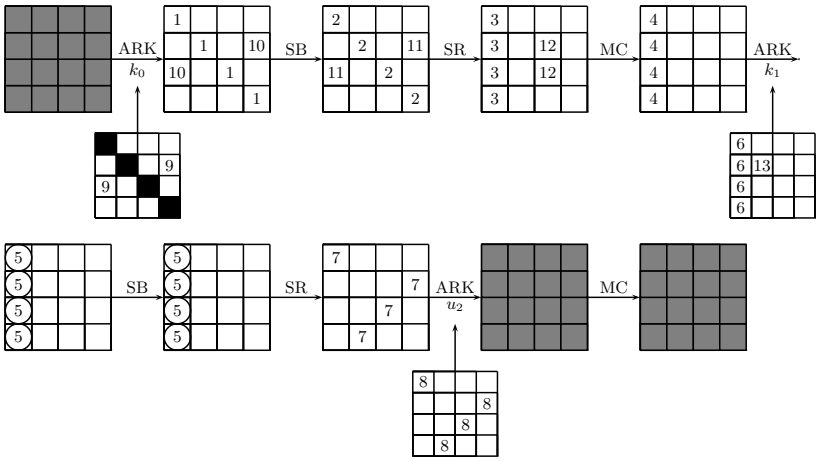
In the second phase of the attack, the adversary guesses two additional subkey bytes ($k_{0,7}$ and $k_{0,8}$) which are sufficient to retrieve the intermediate difference in $x_{2,Col(2)}$. Then, Observation 1 can be applied to the four S-boxes in $Col(2)$ of round 2.

We note that the while the first phase of the attack allows to obtain several bytes in u_2, the knowledge of these bytes cannot be combined directly with the knowledge of bytes in k_1 and k_0, since u_2 does not satisfy the equations of the key schedule algorithm. Hence, in the second phase of the attack, we obtain bytes in both u_2 and k_2 in parallel, and apply Observation 2 to the relation between k_2 and u_2, since they are the input and output of a MixColumns operation.

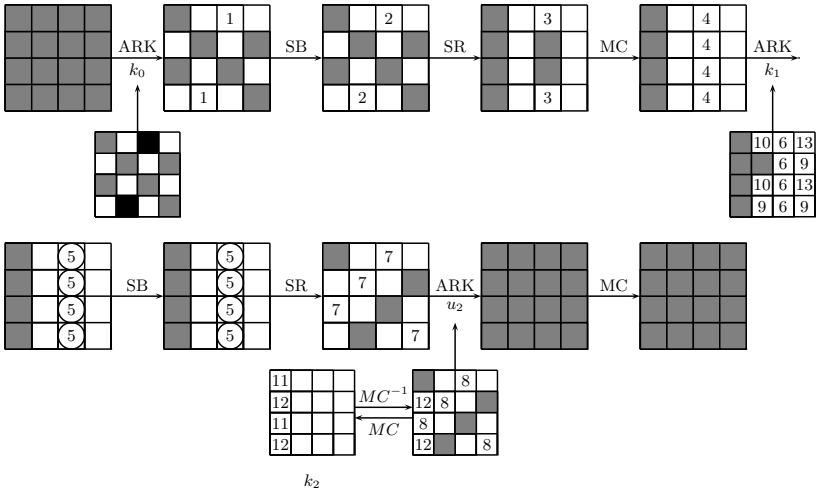
In Phase 1 of the attack, depicted in the top half of Figure 4, step 5 is based on Observation 1, steps 9 and 13 exploit the key schedule, and the rest of the steps are performed using encryption/decryption. In the second phase, depicted in the bottom half of the figure, step 5 is based on Observation 1, step 12 uses Observation 2 applied to the relation between k_2 and u_2, steps $9, 10, 11$, and 13 exploit the key schedule, and the rest of the steps are performed using encryption/decryption.

The time complexity of the attack is determined by the fact that 6 subkey bytes are guessed, and for each guess a few simple analysis steps are performed. Hence, the time complexity of the attack is 2^{48}.

A Three Known Plaintext Variant We note that if the adversary is given *three* known plaintexts, the time complexity can be reduced to 2^{32} encryptions. In order to achieve the reduction, the adversary applies the first phase of the attack twice (for the pairs (P_1, P_2) and (P_1, P_3)), and uses the values of $k_{1,Col(0)}$ retrieved in that phase for a consistency check. Since for the correct guess of bytes $0, 5, 10, 15$ of k_0, both pairs suggest the same value of the four bytes of k_1, and for an incorrect guess, the two pairs suggest the same value only with probability 2^{-32}, this allows to discard most of the wrong guesses. Then, the adversary performs the second phase of the attack only for the remaining guesses, and



The difference in the bytes marked is guessed. The bytes marked by 5 are found using the known input and output differences.



The difference in the bytes marked in black is guessed. The bytes marked by 5 are found using the known input and output differences. The bytes marked by 12 are found using the relation between the keys k_2 and u_2.

Fig. 4. The attack with two known plaintexts on two round AES

thus the time complexity of the attack is dominated by the first step, whose complexity is 2^{32} encryptions.

A Two Chosen Plaintext Variant If the adversary is given two *chosen* plaintexts, then the time complexity can be reduced to 2^{28} encryptions. In order to achieve this reduction, the adversary asks for the encryption of two plaintexts

which differ only in four bytes composing one column. In this case, at the end of round 1, there are exactly 127 possible differences in each column. For each such difference, the adversary can apply Observation 1 to the four S-boxes of the column, and obtain one suggestion on average for the actual values after the SubBytes operation of round 2. Combining the values obtained from all four columns, the adversary gets about 2^{28} suggestions for the entire state after the SubBytes operation of round 2, and each such suggestion yields a suggestion of the subkey k_2. Thus, the time complexity of the attack is 2^{28} encryptions.

5.2 One Known Plaintext

It is also possible to attack 2-round AES using a single known plaintext. The attack, depicted in Figure 5, is based mainly on Observation 3 and on many simpler key schedule considerations.

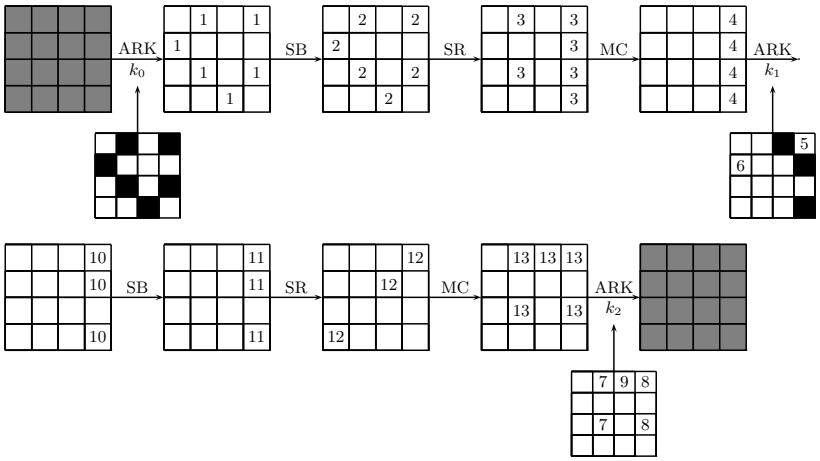
In the first phase of the attack, the adversary guesses nine subkey bytes (marked in black in the upper part of the figure). Step 7 uses Observation 3(3), step 8 uses Observation 3(2), steps $5, 6$, and 9 use the key schedule, and the remaining steps are computed using the AES algorithm.

In the second phase of the attack, the adversary guesses one state byte (marked in black in the lower part of the figure). Steps 9 and 13 are based on Observation 3(1), steps $1, 5, 29$, and 32 use Observation 2, steps 3,7,8,10–12, and 21–24 use the key schedule, and the remaining steps are performed by applying AES' operations on known values.

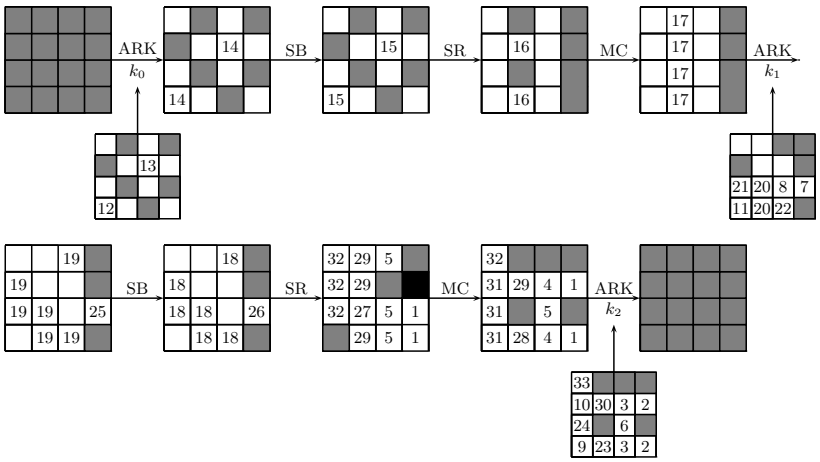
The time complexity of the attack is determined by the amount of bytes which are guessed. Namely, as the adversary guesses 10 bytes, the time complexity of the attack is 2^{80} encryptions.

A Time-Memory Tradeoff The time complexity can be reduced at the expense of enlarging the memory complexity, using non-linear equations and a precomputed table as in Observation 5. In order to achieve this reduction, the adversary performs the following precomputation: Let bytes 4 and 14 of k_0 be denoted by b and c.[7] It is possible to represent all the bytes found during the attack procedures in terms of b, c, the plaintext, the ciphertext, and the other 8 key bytes which are guessed in the original attack procedure. At the end of the deduction procedure, after a suggestion for the full subkey k_2 (in terms of b and c) is obtained, the adversary decrypts the ciphertext through the last round and obtains a suggestion for bytes 4 and 5 of k_1. These bytes can be used as a consistency check, as they can be retrieved independently by the key schedule algorithm, using the suggestion of k_2. This consistency check supplies two non-linear equations in b and c, and it turns out that the equations are of the

[7] We note that in this subsection we use notations which are somewhat different from the notations in the rest of the paper, in order to be consistent with the reference [23], in which this improvement is described in detail.



The value of the bytes marked in black is guessed. The bytes marked by 7 and 8 are found using Observation 3.



The bytes marked is black are guessed. The bytes marked by 9 and 13 are found using Observation 3(1).

Fig. 5. The attack with one known plaintext on two round AES

following form:

$$\begin{aligned} a_5 =& f_0(b, c, a_0, a_1, a_2, a_3, a_4), \\ a_7 =& f_1(b, c, a_0, a_1, a_2, a_4, a_6), \end{aligned} \tag{1}$$

where f_0 and f_1 are fixed known functions, and $a_0, a_1, \ldots, a_7$ are one-byte parameters depending on the plaintext, the ciphertext, and the eight additional subkey bytes guessed in the original attack.[8]

Hence, it is possible to compute in advance the values of (b, c) corresponding to each value of $(a_0, a_1, \ldots, a_7)$, and store them in a table.

In the online phase of the attack, the adversary guesses only 8 subkey bytes (instead of 10), computes the values of $(a_0, a_1, \ldots, a_7)$, and uses the table in order to retrieve b and c. The rest of the attack is similar to the original attack.

The time complexity of the resulting attack is reduced to 2^{64}, but on the other hand, the attack requires 2^{65} bytes of memory.

The memory requirement can be further reduced to 2^{49} bytes by observing that the knowledge of a_1, a_4, and the six subkey bytes $k_{0,6}, k_{0,11}, k_{0,12}, k_{1,8}, k_{1,13}, k_{1,15}$ allows to deduce the value of the two remaining subkey bytes guessed in the modified attack. Using this observation, the attack procedure can be slightly changed as follows: The adversary starts with guessing the values of a_1 and a_4, and prepares the table for the given value of a_1, a_4. In the online phase of the attack, the adversary guesses the six subkey bytes $k_{0,6}, k_{0,11}, k_{0,12}, k_{1,8}, k_{1,13}, k_{1,15}$, deduces the value of the two additional required subkey bytes, and performs the original attack. This change reduces the memory complexity to 2^{49} bytes (since the table is constructed according to 6 byte parameters instead of 8),[9] while the time complexity remains unchanged at 2^{64}.

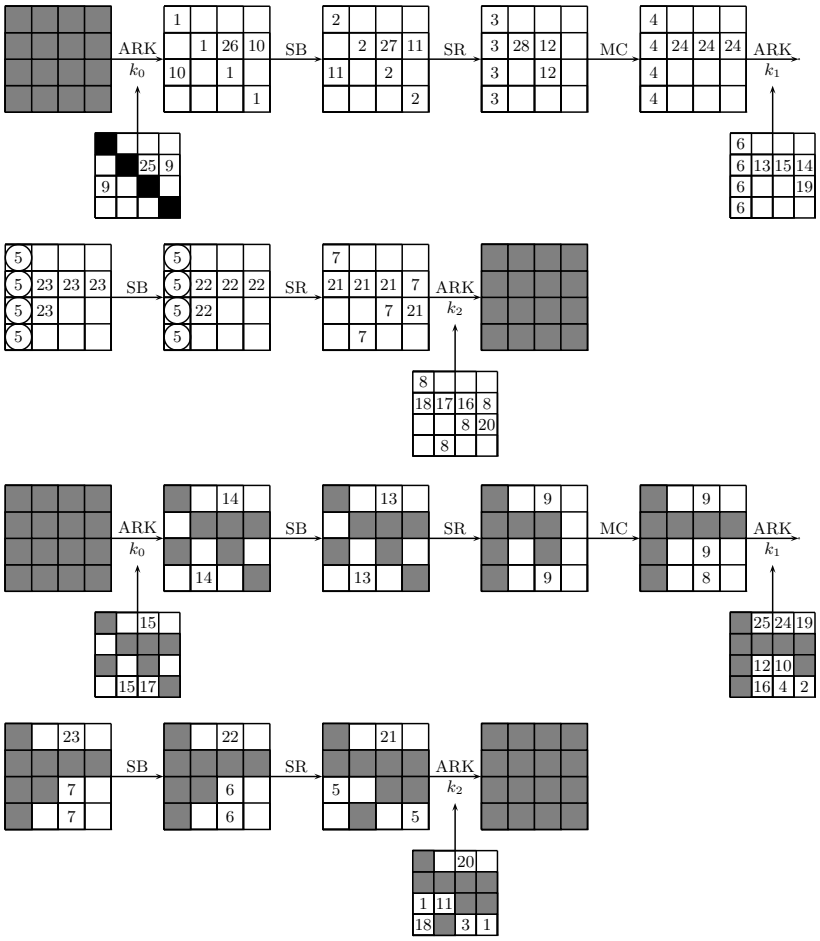
5.3 Improved Attack When the Second MixColumns is Omitted

In Section 8 we present a differential attack on 6-round AES which uses as a subroutine a 2-round attack on AES. In the attack scenario, the two rounds attacked in the subroutine are the last two rounds of AES, i.e., a full round and a round without the MixColumns operation. In this section we present an improved variant of the attack with two known plaintexts presented above that applies in this scenario. We note that this attack gives another evidence to the claim made in [25] that the omission of the the last MixColumns operation in AES reduces the security of the cipher.

The attack, presented in Figure 6, consists of two phases. In the first phase, the first 13 steps are identical to the first 13 steps of the attack on two full rounds

[8] Since the values of $a_0, a_1, \ldots, a_7$ are very cumbersome, we do not present them in this paper, and refer the reader to [23].

[9] We note that as in the 1-round attack, the expected number of solutions in each entry of the table is 1. However, there are some small variations (as some entries are expected to contain more than one solution). This can be dealt with by using a simple memory encoding where each entry contains a zero bit to indicate no solution, and a 1 bit otherwise. Then, each solution is encoded in 16 bits, where a "0" bit is appended if this is the last solution, and a "1" bit is appended to suggest an additional solution. The increased memory consumption is by 2^{45} bytes, and searching in this data structure is expected to be extremely efficient on average, as we can have a good estimation on where to start looking for a required entry.



The two phases of the attack on two round AES without MixColumns at the second round. Bytes marked in black are guessed, and bytes marked in gray are known at this phase of the attack.

Fig. 6. The attack on two rounds of AES without the second MixColumns using two known plaintexts

presented in Section 5.1 above. Steps 14–20 and 25 exploit the key schedule, and the rest of the steps apply AES' operations to known values.

The second phase uses Observation 3 and simpler key schedule observations. Step 1 uses Observation 3(1,2), step 20 uses Observation 3(1), step 9 uses Observation 2, steps 2–4,11,12,16–19, and 25 use the AES key schedule, and the remaining steps are performed using partial encryption or decryption.

6 Attacks on Three-Round AES

In this section we consider attacks on three rounds of AES, denoted by rounds 1–3. First we present a simple attack with two *chosen* plaintexts, then we present a bit more complex meet-in-the-middle attack with 9 *known* plaintexts, and finally we present a very time-consuming attack with a *single known* plaintext.

6.1 Two Chosen Plaintexts

The attack with two chosen plaintexts is similar to the two-round attack with two chosen plaintexts, presented at the end of Section 5.1. As before, we observe that the ciphertext difference allows to retrieve the intermediate difference after the SubBytes operation of round 3. In the attack, the adversary asks for the encryption of two plaintexts which differ only in one byte. In this case, at the end of round 2, there are at most 256 possible differences in each column. For each such difference, the adversary can apply Observation 1 to the four S-boxes of the column, and obtain one suggestion on average for the actual values after the SubBytes operation of round 3. Combining the values obtained from all four columns, the adversary gets at most 2^{32} suggestions for the entire state after the SubBytes operation of round 3, and each such suggestion yields a suggestion for the subkey k_3. Thus, the time complexity of the attack is 2^{32} encryptions.

6.2 Nine Known Plaintexts

The attack with 9 known plaintexts uses a combination between the differential approach and the standard meet-in-the-middle approach. The adversary guesses subkey material in k_0 and k_3, and obtains a consistency check on the intermediate difference after the ShiftRows operation of round 2.

Concretely, denote the intermediate values in byte 0 after the ShiftRows operation of round 2 by $X_1, X_2, \ldots, X_9$. In the first phase of the attack, the adversary guesses bytes $0, 7, 10, 13$ of the equivalent subkey u_3 and partially decrypts the ciphertexts through the last round (obtaining the actual values in $x_{3,Col(0)}$). Then, using the linearity of the MixColumns operation, the adversary computes the differences $X_1 \oplus X_2, X_1 \oplus X_3, \ldots, X_1 \oplus X_9$, and stores their concatenation (a 64-bit vector) in a hash table. In the second phase of the attack, the adversary guesses bytes $0, 5, 10, 15$ of k_0 and byte $k_{1,0}$ and by partial encryption of the plaintexts, obtains the values of $X_1 \oplus X_2, X_1 \oplus X_3, \ldots, X_1 \oplus X_9$, and checks whether their concatenation appears in the hash table. This consistency check is a 64-bit filtering, and thus only $2^{72} \cdot 2^{-64} = 2^8$ key suggestions are expected to remain. By repeating the procedure with the three other columns, the adversary obtains about 2^{32} suggestions for the full subkey k_0 (along with many other subkey bytes), which can be checked by exhaustive key search. The time complexity of the attack is about 2^{40} encryptions, and the memory requirement is 2^{35} bytes of memory.

6.3 One Known Plaintext

The attack with a single known plaintext, depicted in Figure 7, combines the meet-in-the-middle approach with key schedule observations. The attack consists of two phases.

In the first phase, shown in the top part of the figure, the adversary guesses 15 subkey bytes, and uses key schedule considerations to deduce numerous additional subkey bytes in the four subkeys k_0, k_1, k_2, and k_3. Step 4 of the deduction uses Observation 3(1), and the other steps use the key schedule algorithm directly.

The second phase, shown in the bottom part of the figure, is the meet-of-the-middle part of the attack. Using the known subkey bytes, the adversary partially encrypts the plaintext and decrypts the ciphertext and obtains sufficient information in order to apply Observation 2 to the MixColumns operations of rounds 2 and 3. Steps 13 and 17 of this part use Observation 2, and the other steps use AES' operations and the knowledge obtained in previous steps.

Since the adversary guesses 15 key bytes, the time complexity of the attack is 2^{120} encryptions. As in the single-plaintext attacks on one-round and two-round AES, the adversary can reduce the time complexity at the expense of enlarging the memory requirement, using non-linear equations and a precomputed table. The time complexity of the resulting attack is 2^{104} encryptions, and the memory requirement is 2^{49} bytes. Since the technique is similar to the improvement of the 2-round attack presented in Section 5.2, and the obtained equations are quite cumbersome, we do not present the improvement here and refer the reader to [23].

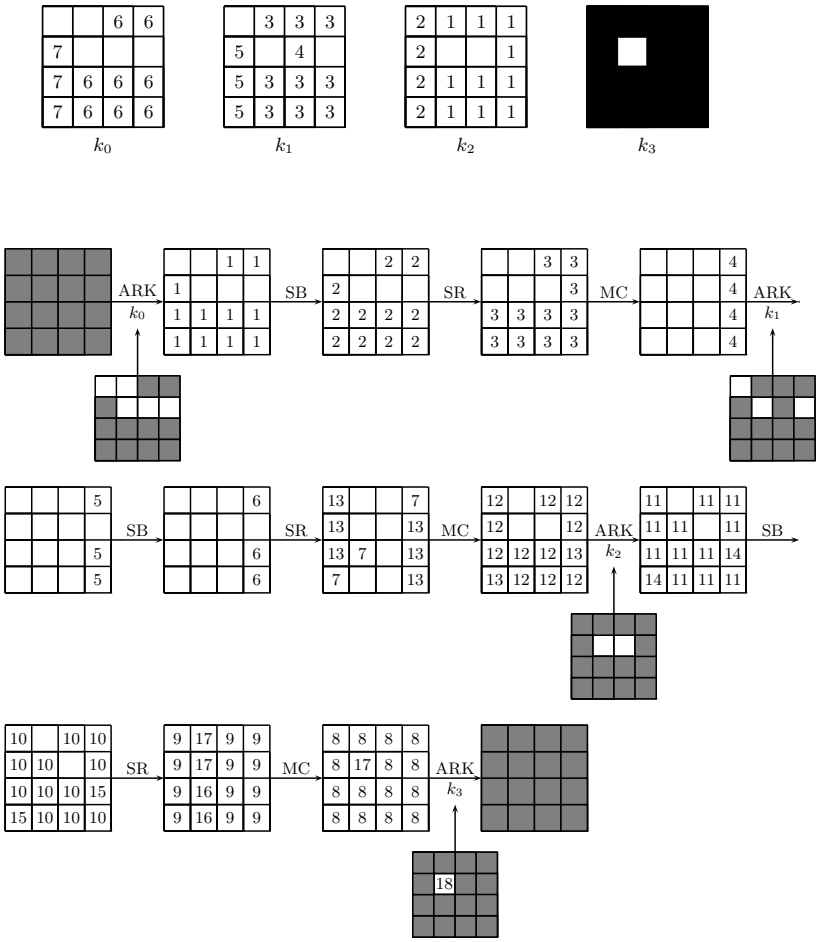
7 Attacks on Four-Round AES

In this section we consider attacks on 4-round AES. Unfortunately, we did not succeed in finding *known plaintext* attacks which require only a few plaintexts, and thus we present only chosen plaintext attacks. We note that these attacks can be transformed into known plaintext attacks using the standard birthday-based transformations, but these usually result in a high data complexity.

7.1 Ten Chosen Plaintexts

The attack with 10 chosen plaintexts is similar to the 3-round attack with 9 known plaintexts presented in Section 6.2. The adversary asks for the encryption of ten plaintexts which differ only in bytes 0,5,10,15. Then she guesses subkey material in the subkeys k_0, k_1 and the equivalent subkeys u_3 and u_4, and obtains a consistency check on some intermediate difference after the MixColumns operation of round 2.

Let the intermediate values in byte 0 after the MixColumns operation of round 2 be $X_1, X_2, \ldots, X_{10}$. In the first phase of the attack, the adversary guesses bytes $0, 7, 10, 13$ of the equivalent subkey u_4 and byte 0 of the equivalent subkey



The two steps of the attack on three round AES with a single known plaintext. The first diagram shows the order of deducing subkeys using the key relations, and the second shows the deduction of the remaining subkey bytes. Guessed bytes are marked by black, and known bytes are marked by gray.

Fig. 7. The attack on three rounds of AES using one known plaintext

u_3 and partially decrypts the ciphertexts through the last two rounds obtaining the actual values in the byte $x_{3,0}$. (Note that reversing the order of the MixColumns and AddRoundKey operations in the two last rounds allows to obtain this intermediate value by guessing only 40 subkey bits). Then, using the linearity of the AddRoundKey operation, the adversary computes the differences $X_1 \oplus X_2, X_1 \oplus X_3, \ldots, X_1 \oplus X_{10}$, and stores their concatenation (a 72-bit vector) in a hash table.

In the second phase of the attack, the adversary guesses bytes $0, 5, 10, 15$ of k_0 and the byte $k_{1,0}$. By the structure of the chosen plaintexts, this allows to compute the differences between pairs of intermediate values $w_{2,Col(0)}$ (since the actual values in byte 0 before the MixColumns operation of round 2 are known by partial encryption, and the difference in bytes $1, 2, 3$ is zero). Thus, the adversary obtains the values of $X_1 \oplus X_2, X_1 \oplus X_3, \ldots, X_1 \oplus X_{10}$, and checks whether their concatenation appears in the hash table. This consistency check is a 72-bit filtering, and thus only $2^{80} \cdot 2^{-72} = 2^8$ key suggestions are expected to remain. By repeating the procedure with the three other columns (from the ciphertext side), the adversary obtains about 2^{32} suggestions for the full equivalent subkey u_4 (along with many other subkey bytes), which can be checked by exhaustive key search. The time complexity of the attack is about 2^{40} encryptions, and the memory requirement is about 2^{43} bytes of memory.

7.2 Five Chosen Plaintexts

If only five chosen plaintexts are available to the adversary, she can perform a variant of the attack described above, at the expense of enlarging the time and memory complexities. The plaintexts are chosen as before, but more key material is guessed: from the ciphertext side, the adversary guesses bytes $0, 7, 10, 13$ of u_4 and bytes $0, 1, 2, 3$ of u_3, and from the plaintext side, the adversary guesses bytes $0, 5, 10, 15$ of k_0 and bytes $0, 1, 2, 3$ of k_1. This allows to get a consistency check on the intermediate difference at the end of round 2 in bytes $0, 5, 10, 15$ (instead of only byte 0), and thus, the four pairs which can be extracted from the data supply a 128-bit filtering and only the correct key suggestion is expected to remain. Finally, the adversary repeats the attack procedure with three other columns from the ciphertext side, and obtains a single suggestion (or a few suggestions) for the full equivalent subkey u_4. The time complexity of the attack is about 2^{64} encryptions, and the memory requirement is about 2^{68} bytes.

7.3 Two Chosen Plaintexts

The attack with two chosen plaintexts is the most complex attack in this paper, and uses a combination of Observations 1, 2, and 3 with differential techniques. The adversary asks for the encryption of two plaintexts which differ only in byte 3, which assures that the difference in the state z_2 (i.e., just before MixColumns of round 2) is zero in all bytes except for one byte in each column (specifically, except for bytes $3, 6, 9, 12$).

In the first phase of the attack, presented in Figure 8, the adversary guesses 8 subkey bytes — $k_{3,Col(2,3)}$, and two state bytes — $z_{4,3}$ and $z_{4,6}$. Note that a single guess is sufficient to obtain the state bytes in both encryptions, since their difference can be computed by applying the inverse of MixColumns to the ciphertext difference. Step 1 follows directly from the key schedule algorithm, step 2 is based on Observation 3(4), and steps 3–10 are performed by application of AES' operations to known values.

Step 11 is more complex, and actually consists of three steps.

1. First, consider the MixColumns operation of round 2. By the choice of the plaintext difference, the input difference of that MixColumns in three bytes of each column is zero. On the other hand, by step 8, the output difference of that MixColumns is known in one byte in each column. Thus, the adversary can apply Observation 2 to the MixColumns operation (in all columns) with respect to differences, and obtain the entire difference in the states z_2 and w_2. This allows the adversary to obtain the difference in those bytes of the state z_2 numbered 11.
2. Second, consider the MixColumns operation of round 1, and concentrate on $Col(3)$. By the choice of the plaintext difference, the only byte with non-zero difference in $z_{1,Col(3)}$ is byte 12, and the difference in that byte is known from step 10. Thus, the output difference of that MixColumns is also known, which means that the difference in the bytes numbered 11 in the state x_2 is known to the adversary.
3. Finally, by combination of the two steps above, the input and output differences of the SubBytes operation in bytes $0, 1, 2, 3$ of round 2 are known to the adversary. This allows to apply Observation 1 to these S-boxes and obtain the actual input and output values.

In the second phase of the attack, the adversary guesses three additional subkey bytes — $k_{0,1}, k_{0,6}$, and $k_{0,11}$, then retrieves many additional subkey bytes using key schedule considerations, and finally applies Observation 2 to the MixColumns operation in Columns $0, 1$ of round 4. Step 7 is the most complex one and uses Observation 3(5), step 28 uses Observation 2, step 6 uses Observation 3(2), steps $15, 16$, and 22 use Observation 3(3), step 26 uses Observation 3(1,2), steps 5,8–14,17–21, and 23–25 are performed by direct application of the key schedule algorithm, and the remaining steps are application of AES' operations to known values.

Since the adversary guesses 13 key bytes, the time complexity of the attack is 2^{104} encryptions.

8 Differential Attack on 6-Round AES

The design of AES follows the *wide trail* design strategy, which assures that the probability of differentials is extremely low, even for only four rounds of the cipher. For example, it was proved in [44], that any 4-round differential of AES has probability of at most 2^{-110}. Hence, it is widely believed that no regular differential attack can be mounted on more than 5 rounds of AES. Furthermore, the best currently known differential attack on AES-128 is on only four rounds, and all known attacks on 5 and more rounds use "more sophisticated" techniques like impossible differentials, boomerangs, or Squares.

In this section we show that the low data complexity attack on 2-round AES presented in Section 5.3 can be leveraged to a differential attack on 6-round AES. Although the data complexity of the resulting attack is high, the data complexity of its *known plaintext* variant is still smaller than the data complexity of the best known attack on 6-round AES in the known plaintext model. While our attack

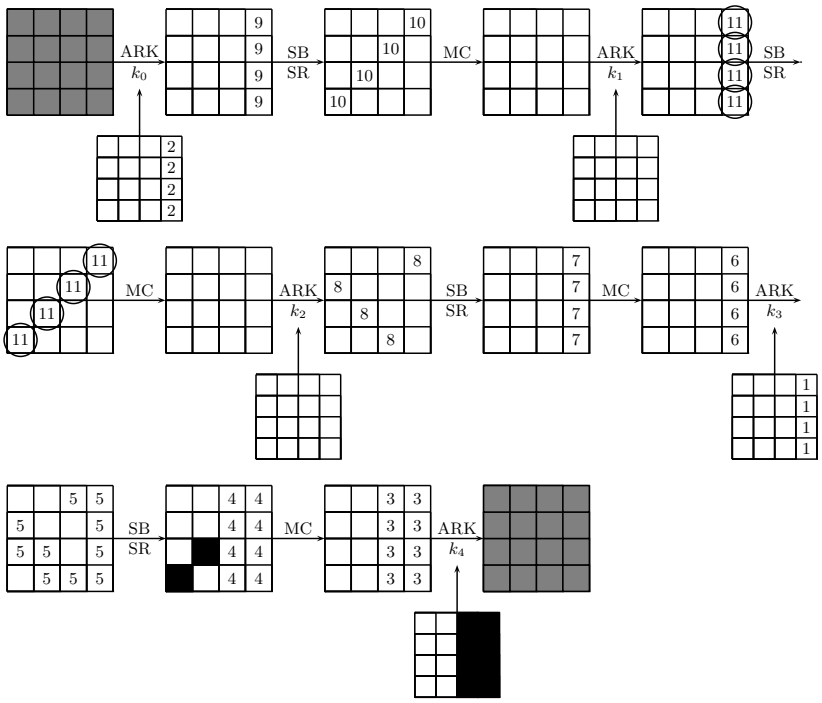


Fig. 8. The first phase of the attack on four rounds of AES using two chosen plaintexts. Step 11 is relatively complex and is explained in the text.

certainly does not threaten the security of AES, it shows that its security with respect to conventional differential attacks is lower than expected before.

As in most published attacks on reduced-round variants of AES, we assume that the MixColumns operation in the last round is omitted, like in the full AES.[10]

Our 6-round attack is based on the following 3-round truncated differential: The input difference in all bytes except for byte 0 is zero, and the output difference in all bytes except for bytes $0, 5, 10, 15$ is zero. We depict the differential in Figure 10. A pair satisfying the input and output requirements of the differential in rounds 2–4 is called a right pair.

Consider a right pair (P, P'). By the structure of AES, the intermediate difference at the input of round 3 is zero in all bytes except for $0, 1, 2, 3$. Thus, there are at most 2^{32} possible differences in the input of the SubBytes operation of round 4. On the other hand, since the difference at the output of round 4 is zero in all bytes except for $0, 5, 10, 15$, there are only 2^{32} possible differences after

[10] We were not able to extend the attack to the case where the last MixColumns operation is not omitted. This gives another evidence to the claim made in [25] that the omission of the last round MixColumns affects the security of AES.

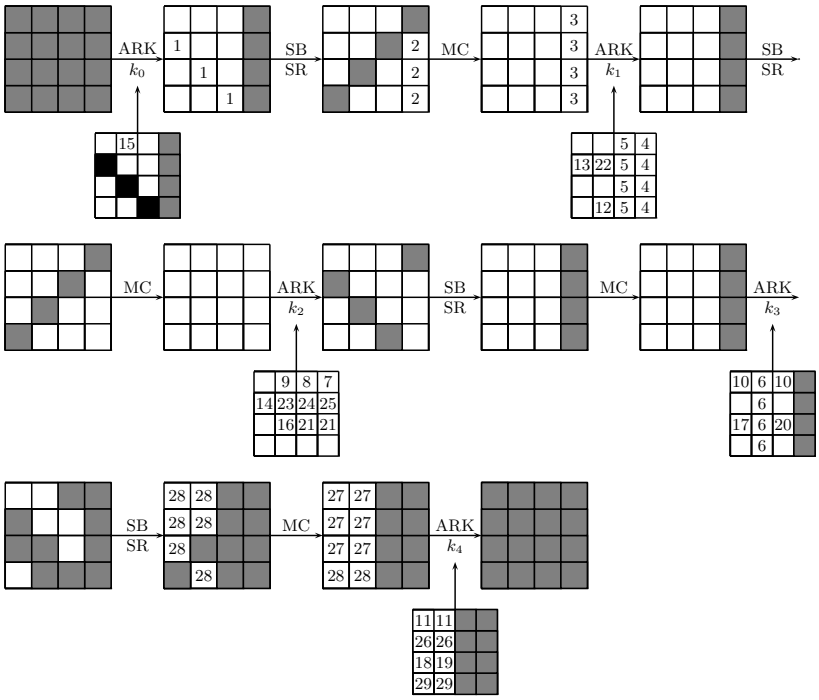


Fig. 9. The second phase of the attack on four rounds of AES using two chosen plaintexts. Step 7 is based on Observation 3(5).

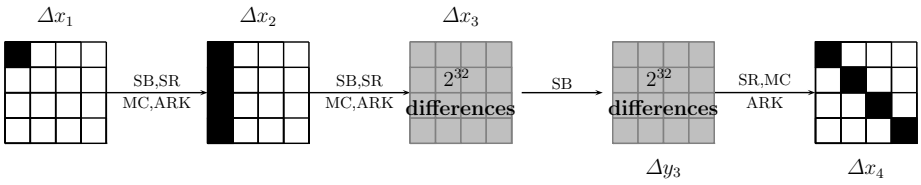
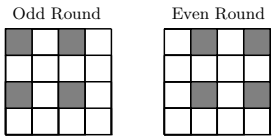


Fig. 10. The 3-Round Truncated Differential Used in Our Attack

the SubBytes operation of round 4. Note that by Observation 1, the input and output differences of a SubBytes operation yield a single suggestion (on average) for the actual values. Therefore, if (P, P') is a right pair, then there are only 2^{64} possibilities of the corresponding actual values after the SubBytes of round 4 (or equivalently, for the actual values after the MixColumns operation of round 4).

This observation allows to mount the following known plaintext attack:

1. Ask for the encryption of $2^{108.5}$ plaintexts P_i under the unknown key, and denote the corresponding ciphertexts by C_i.



The gray bytes are the output bytes.

Fig. 11. State Bytes which Compose the Output in Odd and Even Rounds of LEX

2. Insert (P_i, C_i) into a hash table indexed according to bytes 1–4,6–9,11–14 of P_i and bytes 1–6,8,9,11,12,14,15 of C_i, and consider only the colliding pairs in the hash table (which are the only pairs which may be right). The number of remaining pairs is $2^{216} \cdot 2^{-192} = 2^{24}$.
3. For each of the remaining pairs, assume that it is a right pair, and for each of the 2^{64} possible actual values after the MixColumns operation of round 4, apply the attack presented in Section 5.3 on rounds 5–6.[11]

Since the time complexity of the 2-round attack presented in Section 5.3 is 2^{32} encryptions, the overall complexity of the attack is $2^{24} \cdot 2^{64} \cdot 2^{32} = 2^{120}$ encryptions. The data complexity of the attack is $2^{108.5}$ known plaintexts, which is smaller than the data complexities of the previously known attacks in the known plaintext model (see, e.g., [17]).

9 An Attack on Modified LEX

LEX is a stream cipher presented by Biryukov in [9], as an example of the *leak extraction* methodology of stream cipher design. In this methodology, a block cipher is used in an OFB mode of operation, where after each *round* of the cipher, some part of the intermediate encryption value is output as part of the key stream. Lex itself uses the AES as the block cipher.

In the initialization step of LEX, the publicly known IV is encrypted by AES[12] under the secret key K to obtain $S = AES_K(IV)$. Then, S is repeatedly encrypted in the OFB mode of operation under K, where during the execution of each encryption, 32 bits of the internal state are leaked in each round. These state bits compose the key stream of LEX. The state bytes used in the key stream are shown in Figure 11. After 500 encryptions, another IV is chosen, and the process is repeated. After 2^{32} different IVs, the secret key is replaced.

LEX was submitted to the eSTREAM competition (see [10]). Due to its high speed (2.5 times faster than AES), fast key initialization phase (a single AES encryption), and expected security (based on the security of AES), LEX was

[11] Note that the 2-round attack requires that the difference in the four bytes $x_{2,Col(0)}$ in the attacked variant is non-zero, and this condition is indeed satisfied in our attack (for the state x_6 which corresponds to x_2 in the two-round attack).

[12] Actually, LEX uses a tweaked version of AES where the AddRoundKey before the first round is omitted, and the MixColumns operation of the last round is present.

considered a very promising candidate and selected to the third (and final) phase of evaluation. However, it wasn't selected to the final portfolio of eSTREAM due to an attack with data complexity of $2^{36.3}$ bytes of key stream and time complexity of 2^{112} encryptions presented in [26] a few weeks before the end of the eSTREAM competition. The complexity of the best published attack on LEX is about 2^{40} bytes of keystream and 2^{100} encryptions [27].

In order to study the *leak extraction* design methodology, we consider a modified variant of LEX, in which instead of extracting 32 bits of the state after every AES round, the cipher outputs the entire state after every four rounds. In this case, the core cipher is reduced to 4-round AES (without the initial key whitening), but due to the stream cipher environment, the adversary is restricted to $2^{46.3}$ *known* plaintext bytes (as after that amount of key material, the stream cipher is rekeyed). It turns out that while 4-round AES is considered very weak due to the Square attack which can break it with only 2^8 chosen plaintexts, all the previously known attacks on AES have very high data complexity when transformed to the known plaintext model, and thus cannot be applied in our scenario. Thus, a priori, it is not clear whether this variant of LEX is less secure than the original LEX.

However, using the attacks on reduced-round AES presented in the previous sections, we can show that this variant is practically insecure, and that even a stronger variant in which the full state is output every 5 rounds is still less secure than the original LEX.

Note that four-round AES without the initial key whitening step is equivalent to three full AES rounds (since the adversary can encrypt all the plaintexts through the first round without knowledge of the key). Thus, the attack on 3-round AES with nine known plaintexts, described in Section 6.2, applies directly to this variant and allows to break it with only 360 bytes of keystream (obtained from 9 AES invocations), time complexity of 2^{40} encryptions, and 2^{43} bytes of memory.

Similarly, if the full state is output every 5 rounds, then the underlying cipher is equivalent to four full AES rounds. In this case, our low complexity attacks perform only in the chosen plaintext model, and thus we use transformation to the known plaintext model, based on collecting a sufficient number of known plaintexts, until enough pairs with the required input difference are encountered.

We consider the attack with five chosen plaintexts presented in Section 7.2. A set of $2^{49.5}$ plaintexts picked at random is expected to contain $2^{98} \cdot 2^{-96} = 4$ pairs with zero difference in the 12 input bytes required in the chosen plaintext attack, and thus the attack can be applied with data complexity of $2^{54.5}$ keystream bytes, time complexity of 2^{64} encryptions and memory requirement of 2^{68} bytes.

It is possible to reduce the data complexity of this attack to $2^{33.5}$ known plaintexts at the expense of enlarging the time complexity, by slightly changing the underlying chosen plaintext attack. Instead of considering only pairs of plaintexts with zero difference in 12 bytes of the state, the adversary uses pairs with zero difference in only eight bytes: $2, 3, 4, 7, 8, 9, 13$, and 14. The adversary guesses bytes $0, 1, 5, 6, 10, 11, 12, 15$ of k_0 and bytes $0, 15$ of k_1, and obtains the

intermediate difference in $w_{2,Col(0)}$ (i.e., at the output of round 2). On the other hand, the guess of bytes $0, 7, 10, 13$ of u_4 and byte 0 of u_3 is sufficient to obtain the intermediate difference in byte 0 at the end of round 2, which can be used as a consistency check. The rest of the attack is similar to the chosen plaintext attack on 4-round AES with ten chosen plaintexts presented in Section 7. Since the filtering is on 8 bits, 15 pairs with the required input difference are sufficient to discard most of the wrong key guesses. Thus, the data complexity is $2^{33.5}$ known plaintexts, or $2^{38.5}$ bytes of keystream. The memory requirement is 2^{44} bytes, and the time complexity is 2^{80} encryptions.

10 Summary and Conclusions

In this paper we considered low data complexity attacks on reduced-round variants of AES. We presented several attacks on up to four rounds of AES given at most 10 known (or chosen) plaintexts, and showed how to leverage such attacks to more complex attacks on variants of AES with more rounds and on other primitives based on AES (such as modified variants of the stream cipher LEX). In particular, we showed that the security margin of AES with respect to conventional differential attacks is smaller than considered before.

We believe that a big safety margin with respect to attacks in which the adversary has limited resources (which are more similar to the scenarios in practical attacks) is an important design criterion for block ciphers and other cryptographic primitives. Thus, attacks with limited resources should be considered particularly, in parallel with the "usual" line of research which does not restrict the resources of the adversary and requires only complexity lower than that of exhaustive key search.

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. Ryad Benadjila, Olivier Billet, Henri Gilbert, Gilles Macario-Rat, Thomas Peyrin, Matt Robshaw, and Yannick Seurin, *SHA-3 Proposal: ECHO (version 1.5)*, SHA-3 submission, 2009.
3. Come Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier, Thomas Pornin, and Hervé Sibert, *Sosemanuk, a fast software-oriented stream cipher*, eSTREAM submission, 2005.
4. Eli Biham, Alex Biryukov, and Adi Shamir, *Miss in the Middle Attacks on IDEA and Khufu*, proceedings of Fast Software Encryption 1999, Lecture Notes in Computer Science 1636, pp. 124–138, Springer, 1999.
5. Eli Biham, Alex Biryukov, and Adi Shamir, *Cryptanalysis of Skipjack Reduced to 31 Rounds*, Advances in Cryptology, proceedings of EUROCRYPT 1999, Lecture Notes in Computer Science 1592, pp. 12–23, Springer, 1999.

6. Eli Biham, Orr Dunkelman, and Nathan Keller, *Improved Slide Attacks*, proceedings of Fast Software Encryption 2007, Lecture Notes in Computer Science 4593, pp. 153–166, Springer, 2007.
7. Eli Biham and Nathan Keller, *Cryptanalysis of Reduced Variants of Rijndael*, unpublished manuscript, 1999.
8. Eli Biham and Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer, 1993.
9. Alex Biryukov, *The Design of a Stream Cipher LEX*, Proceedings of Selected Areas in Cryptography 2006, Lecture Notes in Computer Science 4356, pp. 67–75, Springer, 2007.
10. Alex Biryukov, *A New 128-bit Key Stream Cipher LEX*, ECRYPT stream cipher project report 2005/013. Available at http://www.ecrypt.eu.org/stream.
11. Alex Biryukov, *The Tweak for LEX-128, LEX-192, LEX-256*, ECRYPT stream cipher project report 2006/037. Available at http://www.ecrypt.eu.org/stream.
12. 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 Science 6110, pp. 299–319, Springer, 2010.
13. 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.
14. 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.
15. Alex Biryukov and David Wagner, *Slide Attacks*, proceedings of Fast Software Encryption 1999, Lecture Notes in Computer Science 1636, pp. 245–259, Springer, 1999.
16. Alex Biryukov and David Wagner, *Advanced Slide Attacks*, Advances in Cryptology, proceedings of EUROCRYPT 2000, Lecture Notes in Computer Science 1807, pp. 586–606, Springer, 2000.
17. 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.
18. Nicolas T. Courtois, Gregory V. Bard, and David Wagner, *Algebraic and Slide Attacks on KeeLoq*, proceedings of Fast Software Encryption 2008, Lecture Notes in Computer Science 5086, pp. 97–115, Springer, 2008.
19. Joan Daemen and Vincent Rijmen, *AES Proposal: Rijndael*, NIST AES proposal, 1998.
20. Joan Daemen and Vincent Rijmen, *The design of Rijndael: AES — the Advanced Encryption Standard*, Springer, 2002.
21. Joan Daemen and Vincent Rijmen, *A New MAC Construction ALRED and a Specific Instance, ALPHA-MAC*, proceedings of Fast Software Encryption 2005, Lecture Notes in Computer Science 3557, pp. 1–17, Springer, 2005.
22. 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.
23. Patrick Derbez, *Rapport de Stage*, report of internship at École Normale Supérieure, September 2010.

24. Itai Dinur and Adi Shamir, *Side Channel Cube Attacks on Block Ciphers*, IACR ePrint report 2009/127.
25. Orr Dunkelman and Nathan Keller, *The Effects of the Omission of Last Round's MixColumns on AES*, Information Processing Letters, Vol. 110, Number 8–9, pp. 304–308, Elsevier, 2010.
26. Orr Dunkelman and Nathan Keller, *A New Attack on the LEX Stream Cipher*, Advances in Cryptology, proceedings of ASIACRYPT 2008, Lecture Notes in Computer Science 5350, pp. 539–556, Springer, 2008.
27. Orr Dunkelman and Nathan Keller, *Cryptanalysis of the Stream Cipher LEX*, submitted to Designs, Codes and Cryptography, 2010.
28. Orr Dunkelman, Nathan Keller, and Adi Shamir, *A Practical-Time Attack on the A5/3 Cryptosystem Used in Third Generation GSM Telephony*, Advances in Cryptology, proceedings of CRYPTO 2010, Lecture Notes in Computer Science 6223, pp. 393–410, Springer, 2010.
29. Orr Dunkelman, Nathan Keller, and Adi Shamir, *Improved Single-Key Attacks on 8-round AES-192 and AES-256*, accepted to ASIACRYPT 2010.
30. 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.
31. Soichi Furuya, *Slide Attacks with a Known-Plaintext Cryptanalysis*, proceedings of Information and Communication Security 2001, Lecture Notes in Computer Science 2288, pp. 214–225, Springer, 2002.
32. 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.
33. Martin Hell and Thomas Johansson, *Breaking the F-FCSR-H Stream Cipher in Real Time*, Advances in Cryptology, proceedings of ASIACRYPT 2008, Lecture Notes in Computer Science 5350, pp. 557–569, Springer, 2008.
34. Dmitry Khovratovich, Alex Biryukov, and Ivica Nikolic, *Speeding up Collision Search for Byte-Oriented Hash Functions*, proceedings of CT-RSA 2009, Lecture Notes in Computer Science 5473, pp. 164–181, Springer, 2009.
35. 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.
36. Lars R. Knudsen and Vincent Rijmen, *Known-Key Distinguishers for Some Block Ciphers*, Advances in Cryptology, proceedings of ASIACRYPT 2007, Lecture Notes in Computer Science 4833, pp. 315–324, Springer, 2007.
37. Özgul Küçük, *The Hash Function Hamsi*, SHA-3 submission, 2009.
38. 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, 2008.
39. 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.
40. Stéphane Manuel and Thomas Peyrin, *Collisions on SHA-0 in One Hour*, proceedings of Fast Software Encryption 2008, Lecture Notes in Computer Science 5086, pp. 16–35, Springer, 2008.
41. US National Institute of Standards and Technology, *Advanced Encryption Standard*, Federal Information Processing Standards Publications Number 197, 2001.

42. Paul Stankovski, Martin Hell, and Thomas Johansson, *An Efficient State Recovery Attack on X-FCSR-256*, proceedings of Fast Software Encryption 2009, Lecture Notes in Computer Science 5665, pp. 23–37, Springer, 2009.
43. Marc Stevens, *Fast Collision Attack on MD5*, IACR ePrint report 2006/104, 2006.
44. Sangwoo Park, Soo Hak Sung, Seongtaek Chee, E-Joong Yoon, and Jongin Lim, *On the Security of Rijndael-Like Structures against Differential and Linear Cryptanalysis*, Advances in Cryptology, proceedings of ASIACRYPT 2002, Lecture Notes in Computer Science 2501, pp. 176–191, Springer, 2002.
45. 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.
46. 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.
47. 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 Proofs of Observations

In this appendix we present the proof of Observations 4 and 5. Throughout the appendix, we consider one full AES round (including the key whitening at the beginning), and assume that the input and the output of this full round, denoted by P and C, respectively, are known to the adversary.

We denote the two subkeys used in the AddRoundKey operations by k_0 and k_1, and the states after the first ARK, the SB, the SR, and the MC operations by x, y, z, and w, respectively (here we omit the round counters as only one round is involved).

Observation 4 *The knowledge of P, C and the column $k_{0,Col(3)}$ allows retrieving two additional bytes of k_0, namely $k_{0,1}$ and $k_{0,8}$.*

Proof. The derivation of the two additional bytes is performed as follows:

1. Consider the 32-bit value $w_{Col(2)} \oplus w_{Col(3)}$. By the key schedule, we have:

$$\begin{aligned} w_{Col(2)} \oplus w_{Col(3)} =& \Big(C_{Col(2)} \oplus C_{Col(3)}\Big) \oplus \Big(K_{1,Col(2)} \oplus k_{1,Col(3)}\Big) \\ =& \Big(C_{Col(2)} \oplus C_{Col(3)}\Big) \oplus k_{0,Col(3)}. \end{aligned} \tag{2}$$

2. Following the linearity of MixColumns, we obtain that:

$$\begin{aligned} z_{Col(2)} \oplus z_{Col(3)} =& MC^{-1}(w_{Col(2)}) \oplus MC^{-1}(w_{Col(3)}) \\ =& MC^{-1}\Big(w_{Col(2)} \oplus w_{Col(3)}\Big). \end{aligned} \tag{3}$$

3. It is possible to compute z_{12} using the knowledge of $k_{0,12}$ as:

$$z_{12} = SR(SB(P_{12} \oplus k_{0,12})). \tag{4}$$

4. Then, the value of z_8 can be retrieved by combining Equations (2), (3) and (4).
5. From z_8 it is possible to compute the subkey byte $k_{0,8}$ using:

$$k_{0,8} = P_8 \oplus x_8 = P_8 \oplus SB^{-1}(SR^{-1}(z_8)).$$

6. It is possible to use a similar procedure to obtain the value z_{13}, and retrieve the subkey byte $k_{0,1}$ as:

$$k_{0,1} = P_1 \oplus x_1 = P_1 \oplus SB^{-1}(SR^{-1}(z_{13})).$$

Observation 5 *The knowledge of P, C, and the column $k_{1,Col(0)}$ allows to retrieve bytes $k_{0,7}$ and $k_{0,8}$ by a look-up into a precomputed table of size 2^{24}. For each value of $k_{1,Col(0)}$, there are at most 12 values of $(k_{0,7}, k_{0,8})$, and on average a single value. The time complexity required to construct the table is 2^{32} operations.*

Proof. First, we note that the knowledge of the column $k_{1,Col(0)}$ along with the plaintext and the ciphertext allows to retrieve one additional byte of k_1 and six bytes of k_0, as shown in Figure 3.

We denote $a = y_8$ and $b = y_7$, and express several other bytes in terms of a, b, and known bytes (from the plaintext, the ciphertext and $k_{1,Col(0)}$). Our goal is to obtain a system of two equations in a and b, and to solve it using a precomputed table. This system is constructed as follows:

1. Note that $k_{1,Col(2)}$ can be expressed as:

$$\begin{aligned} k_{1,Col(2)} &= C_{Col(2)} \oplus w_{Col(2)} = C_{Col(2)} \oplus MC(z_{Col(2)}) \\ &= C_{Col(2)} \oplus MC\Big(a, SB(P_{13} \oplus k_{0,13}), SB(P_2 \oplus k_{0,2}), b\Big). \end{aligned} \tag{5}$$

2. Note that $k_{1,5}$ can be computed following the procedure shown in Figure 3, and the remaining three bytes of $k_{1,Col(1)}$ can be expressed as:

$$\begin{aligned} k_{1,4} =& k_{1,8} \oplus SB^{-1}(a) \oplus P_8, \\ k_{1,6} =& k_{1,10} \oplus k_{0,10}, \\ k_{1,7} =& k_{1,3} \oplus S^{-1}(b) \oplus P_7. \end{aligned} \tag{6}$$

3. Consider the two bytes z_4, z_5. By the key schedule algorithm:

$$\begin{aligned} z_4 =& SB(P_4 \oplus k_{0,4}) = SB\Big(P_4 \oplus k_{1,0} \oplus k_{1,4}\Big) \\ =& SB\Big(P_4 \oplus k_{1,0} \oplus k_{1,8} \oplus SB^{-1}(a) \oplus P_8\Big), \\ z_5 =& SB(P_9 \oplus k_{0,9}) = SB(P_9 \oplus k_{1,9} \oplus k_{1,5}). \end{aligned} \tag{7}$$

4. On the other hand, given $w_{Col(1)}$ (which is known from the ciphertext C and the key $k_{1,Col(1)}$), the column $z_{Col(1)}$ can be derived also in a different way:

$$z_{Col(1)} = MC^{-1}(w_{Col(1)}) = MC^{-1}\Big(C_{Col(1)} \oplus k_{1,Col(1)}\Big). \tag{8}$$

5. Equations (6), (7), and (8) can be combined to get a system of two equations in the unknowns a, b, and the known plaintext, ciphertext, and bytes of $k_{1,Col(1)}$. These equations can be written in the form:

$$\begin{aligned} \Delta_1 =& f_1(a,b) \oplus SB(f_2(a,b) + \Delta_3), \\ \Delta_2 =& f_3(a,b) \oplus SB(f_4(a,b) + \Delta_4), \end{aligned} \tag{9}$$

where f_1, f_2, f_3, and f_4 are fixed known functions, and $\Delta_1, \Delta_2, \Delta_3$, and Δ_4 are one-byte parameters depending on the plaintext, the ciphertext, and the guessed subkey bytes.[13]

System (9) allows to perform the following two-phase procedure:

1. In the precomputation phase, for each of the 2^{32} possible values of $(a, b, \Delta_3\Delta_4)$, compute Δ_1 and Δ_2 using Equation (9). Use these values to update a table of solutions of Equation (9) indexed by $(\Delta_1, \Delta_2, \Delta_3, \Delta_4)$ where each entry is a list of possible values of (a, b). It turns out that the maximal number of solutions (a, b) is 12, and the average number is 1. Hence, we can construct the table in 2^{32} time and 2^{32} memory.
2. In the online phase, once the values of P, C are known, and for each guess of $k_{1,Col(1)}$, compute the value $(\Delta_1, \Delta_2, \Delta_3\Delta_4)$, and obtain from the precomputed table the corresponding values of (a, b). Then deduce the subkey bytes $k_{0,7}$ and $k_{0,8}$ using the equations:

$$\begin{aligned} k_{0,8} =& P_8 \oplus x_8 = P_8 \oplus SB^{-1}(a), \\ k_{0,7} =& P_7 \oplus x_7 = P_7 \oplus SB^{-1}(b). \end{aligned} \tag{10}$$

We can reduce the 2^{32} memory required for storing the solutions to only 2^{24}, by observing that one of the bytes of $k_{1,Col(0)}$ occurs only once in Δ_1. We can then enumerate Δ_1 instead of enumerating this key byte, as the key byte can be uniquely deduced from Δ_1. By doing so, we can deal only with the equations relevant to this specific Δ_1 in each step of the attack.

1. For each Δ_1 do
   - **Building the table:** For all (a, b) pairs:
     (a) Compute Δ_3 (following the first part of Equation 9).
     (b) For every Δ_4 determine Δ_2 (following the second part of Equation 9).
     (c) Store in a table (a, b) as the solution of $(\Delta_2, \Delta_3, \Delta_4)$.

[13] We note that the exact values of the parameters are given in [23] by linearly combining two equations (namely, Equation 0 of [23, p. 11] and three times Equation 1 of [23, p. 11]).

– **Using the table:**
  (a) Guess any 3 bytes of $k_{1,Col(0)}$.
  (b) Compute the remaining byte under the assumption that Δ_1 is correct.
  (c) Apply the previous attack as before (as the full $k_{1,Col(0)}$ is known).

The size of the table is only 2^{24} (there are 2^{24} tuples of the form (a, b, Δ_4)), and we expect on average one value in each entry. To conclude, the running time of the attack remains 2^{32} operations, while the memory complexity is reduced to 2^{24}. For the full description see [23].