Low-Data Complexity Attacks on AES

Charles Bouillaguet, Patrick Derbez, Orr Dunkelman, Pierre-Alain Fouque, Nathan Keller, and Vincent Rijmen

***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 argue that consideration of such attacks (which received little attention in recent years) improves our understanding of the security of block ciphers and of other cryptographic primitives based on block ciphers. In particular, 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 Advanced Encryption Standard (AES)—the most widely used block cipher. The AES round function is used in many cryptographic primitives, such as the hash functions Lane, SHAvite-3, and Vortex or the message authentication codes ALPHA-MAC, Pelican, and Marvin. We present attacks on up to four rounds of AES that require at most three known/chosen plaintexts. We then apply these attacks to cryptanalyze an AES-based stream cipher (which follows the leak extraction methodology), and to mount the best known plaintext attack on six-round AES.**

***Index Terms*—Advanced Encryption Standard (AES), cryptanalysis, reflection attacks, slide attacks.**

I. Introduction

THE field of block cipher design has advanced greatly in the last two decades. New strategies for 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 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 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., [28] and [48]. 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 [15], [35], [46]. 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 in establishing 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, it is desirable to consider also other approaches, such as restricting the resources available to the adversary in order to adhere to some of the "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 [14] with respect to the related-key model), or with a practical memory complexity (as suggested in [5]).

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.

This restriction is motivated by numerous papers, which tackle the problem of using only a few plaintext/ciphertext pairs to break reduced-round variants of a block cipher. In these papers, low-data complexity attacks are used in the following scenarios:

1) *Building block in more complex attacks on the block cipher:* Several cryptanalytic techniques leverage an attack with only a few plaintexts on a few rounds of the cipher into attacks on the entire scheme. These include slide attacks

Manuscript received December 12, 2010; revised July 07, 2011; accepted January 20, 2012. Date of publication August 01, 2012; date of current version October 16, 2012. V. Rijmen was supported in part by the Research Fund K. U. Leuven (OT/08/027), in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT programme under Contract ICT-2007-216676 ECRYPT II.

C. Bouillaguet is with the Versailles Saint-Quentin-en-Yvelines University, 78000 Versailles, France (e-mail: charles.bouillaguet@ens.fr).

P. Derbez and P.-A. Fouque are with the Département d'Informatique, École Normale Supérieure, 75005 Paris, France (e-mail: derbez@di.ens.fr; fouque@di.ens.fr).

O. Dunkelman is with the Department of Computer Science, University of Haifa, 31905 Haifa, Israel (e-mail: orr.dunkelman@weizmann.ac.il).

N. Keller is with the Department of Mathematics, Bar-Ilan University, Ramat Gan, 52900, Israel (e-mail: nathan.keller@weizmann.ac.il).

V. Rijmen is with the Department of Electrical Engineering ESAT/SCD-COSIC, University of Leuven, 3000 Leuven, Belgium, and also with the Security Department of IBBT, Kasteelpark Arenberg, B-3001 Leuven-Heverlee, Belgium (e-mail: Vincent.Rijmen@esat.kuleuven.be).

Communicated by K. Martin, Associate Editor for Complexity and Cryptography.

Digital Object Identifier 10.1109/TIT.2012.2207880

[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 [50], [54], and practical attacks on new stream cipher designs are more common [40], [53].

[17] (which employ an attack with two known plaintexts on the subcipher), reflection attacks [43] (in which only a single known plaintext is available), fixed-point attacks [23], and even some differential attacks.[2] For example, an efficient attack on 48-round KeeLoq with two chosen plaintexts is the heart of the practical attack on KeeLoq presented in [41] and attacks on four-round and eight-round GOST with at most three known plaintexts are the heart of the recent attacks on the full GOST [31], [42].[3]

In the case of AES, an attack on two-round AES with two known plaintexts which we develop in this paper was recently used as a subroutine in the best known attack on eight-round AES-192 and AES-256 [36] (see Section I-A).

2) *Attacks on primitives based on reduced-round variants of the block cipher:* Many designs of stream ciphers (e.g., Sosemanuk [4] and LEX [12]), hash functions (like Hamsi [47] and SHAvite-3 [8]), and MACs (e.g., ALPHA-MAC [26], Pelican [27], and Marvin [52]) use a small number of rounds of a block cipher as one of their components. A large portion of these primitives use the readily available AES block cipher, and more accurately, its round function, as the core of the new primitive.

Low-data complexity attacks play an important role in the analysis of such primitives, as shown by the recent attacks on ALPHA-MAC and Pelican [37], [57], [58]. These attacks leverage a low-data complexity attack on four-round AES into an attack on the full primitive.

We focus our attention at the Advanced Encryption Standard (AES) [25]. Due to the strength and efficiency of AES, several primitives were designed on top of a single AES round, or a small number of AES rounds: stream ciphers (e.g., LEX [12]), hash functions (e.g., ECHO [3] and SHAvite-3 [8]), and MACs (e.g., ALPHA-MAC [26] and Marvin [52]). Thus, AES is a natural candidate for this type of analysis.

In this paper, we present several attacks on up to four rounds of AES requiring a few plaintext/ciphertext pairs. Some of these attacks heavily exploit the AES key schedule, while others apply even if the subkeys in AES are replaced by independent subkeys. The attacks are summarized in Table I.

As an example to the possible uses of low-data complexity attacks on reduced-round AES, we present two applications.

1) The best known plaintext attack on six-round AES: We leverage one of our two-round attacks as a subroutine in an attack on six-round AES, which uses a low-probability differential. The resulting attack is the best known attack on six-round AES in the known-plaintext model.
2) Attack on a stream cipher which follows the Leak extraction design methodology [11]: We consider a variant of the stream cipher LEX [12], 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 four-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. We show that one of our attacks on three-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.

TABLE I
SUMMARY OF OUR PROPOSED ATTACKS ON AES-128

Attack Type	Number of Rounds	Complexity		
		Data	Time	Memory
MitM (Sect. IV)	1	1 KP	2^{40}	1
MitM (Sect. IV)	1	1 KP	2^{32}	2^{24}
Diff. (Sect. IV)	1	2 KP	2^{12}	1
MitM (Sect. V)	2	1 KP	2^{80}	1
MitM (Sect. V)	2	1 KP	2^{64}	2^{49}
Diff. (Sect. V)	2	2 KP	2^{48}	1
Diff. (Sect. V)	2	2 CP	2^{28}	1
Diff. (Sect. V)	2	3 KP	2^{32}	1
MitM. (Sect. VI)	3	1 KP	2^{120}	1
MitM. (Sect. VI)	3	1 KP	2^{104}	2^{49}
Diff. (Sect. VI)	3	2 CP	2^{32}	1
Diff. MitM (Sect. VII)	4	2 CP	2^{104}	1

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

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, recent attempts at 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.[4]

A. Follow-Up Work

After the preliminary version of this paper was written, our results were used either explicitly or implicitly in subsequent work.

1) The attacks on the MACs ALPHA-MAC and Pelican presented in [37], [57], and [58] are based on a variant of our attack against four-round AES with two chosen plaintexts using the ideas presented in Section VIII.
2) The best known attack on AES-192 and AES-256 in the single key model [36] is also inspired by the ideas presented in Section VIII.
3) Consideration of low-data complexity attacks led to the understanding that contrary to the common belief, the omission of the last round MixColumns operation in AES does affect the security of the cipher [32].
4) Our research has led to the development of an automatic tool that searches for (differential) meet in the middle attacks on reduced-round AES. This tool supported the correctness of our results, as well as introduced a few time-

[2]An example of such attack is the differential cryptanalysis of the full DES [10], where once a right pair is found, a two-round attack with two known plaintexts is used to retrieve the key.

[3]See also [17]–[19], [23], [22], and [43] for other attacks which use low-data complexity attacks as a subroutine.

[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 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.

TABLE II
SUMMARY OF SOME OF THE FOLLOW-UP RESULTS ON AES-128 REPORTED IN [20]

Attack Type	Number of Rounds	Complexity		
		Data	Time	Memory
MitM (Sect. IV)	1	1 KP	2^{32}	2^{16}
Diff. (Sect. V)	2	2 KP	2^{32}	2^{24}
Diff. (Sect. V)	2	2 CP	2^{8}	2^{8}
MitM. (Sect. VI)	3	1 KP	2^{96}	2^{72}
Diff. (Sect. VI)	3	2 CP	2^{16}	2^{8}
Diff. MitM (Sect. VII)	4	2 CP	2^{80}	2^{80}
Diff. MitM (Sect. VII)	4	4 CP	2^{32}	2^{24}

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

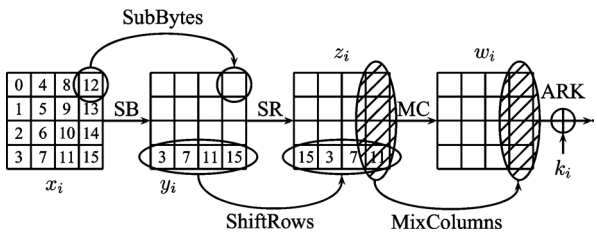


Fig. 1. AES Round.

memory tradeoffs [20], whose complexities are presented in Table II. We note that the attacks reported in [20] tend to offer better time complexities, in exchange for higher memory complexities.

B. Organization of the Paper

Section II gives a description of AES. The attacks on one round, two rounds, three rounds, and four rounds are given in Sections IV–VII, respectively. In Section VIII, we demonstrate how one of the two-round attacks can be leveraged to a known-plaintext attack on six-round AES. Finally, in Section IX, we apply our attacks on three-round and four-round AES to break a stream cipher based on the leak extraction methodology.

II. DESCRIPTION OF AES

The AES [25] 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 4×4, where each byte represents a value in $GF(2^8)$. An AES round applies four operations to the state matrix:

1) SubBytes (SB)—applying the same 8-bit to 8-bit invertible S-box 16 times in parallel on each byte of the state;
2) ShiftRows (SR)—cyclic shift of each row (the i'th row is shifted by i bytes to the left for $0 \leq i \leq 3$);
3) MixColumns (MC)—multiplication of each column by a constant 4×4 matrix over the field $GF(2^8)$;
4) AddRoundKey (ARK)—XORing the state with a 128-bit subkey.

We outline an AES round in Fig. 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 other works on AES, we mostly deal with full-rounds variants, i.e., we shall not assume that the MixColumns operation is omitted from the last round.[5] 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 ten 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 [51]. 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.

1) For $i = 4, \ldots, 43$, do:
   a) If $i \equiv 0 \bmod 4$ then $W[i] = W[i-4] \oplus RotByte(SB(W[i-1])) \oplus RCON[i/4]$;
   b) Otherwise $W[i] = W[i-1] \oplus W[i-4]$,

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

A. 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 a'th column of x_i by $x_{i,Col(a)}$, e.g., $u_{0,Col(0)} = MC^{-1}(k_{0,Col(0)})$. Similarly, by $x_{i,Col(a,b)}$ we denote columns a and b of x_i. We define two more column related sets. The first is $x_{i,SR(Col(a))}$ which is the bytes in x_i corresponding to the places after the ShiftRows operation on column a, e.g., $x_{i,SR(Col(0))}$ is composed of bytes 0,7,10,13. The second is $x_{i,SR^{-1}(Col(a))}$ which is the bytes in the positions of column a after applying the inverse ShiftRows operation.

[5]Following the fact that the last round of AES contains no MixColumns, it is legitimate to test the strength of reduced-round variants, such that the last round has no MixColumns. Moreover, in several cases, this has no effect on the attack's complexity whatsoever.

TABLE III
KEY RELATIONS IN AES-128

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

III. 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 nonlinear 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 nonconsecutive 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 [33] and [38].

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 Table III.

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 semiautomatic tools that tried to identify algebraic relations in one-round AES.

These observations are the heart of our attacks on one-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.

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 one-round AES with a single known plaintext in Section IV-B. We also note that similar statements hold if the adversary knows $Col(1)$ or $Col(2)$ of k_0.

We 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 experiments 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 the Appendix.

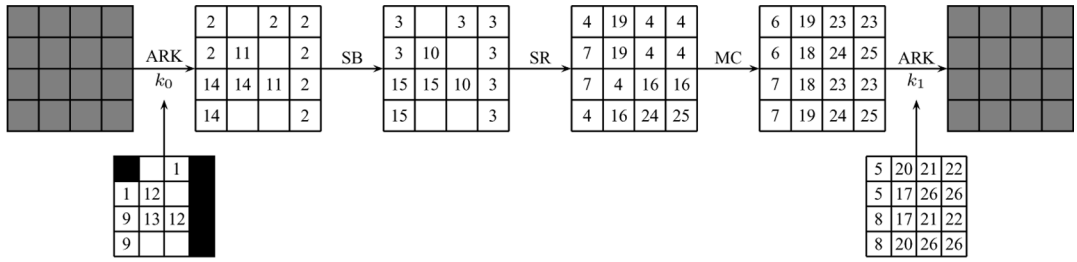


Fig. 2. 2^{40} Time attack on one-round AES with one known plaintext.

IV. 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).

A. 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.[6] 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 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.

B. 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 insufficient 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.[7]

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 Fig. 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 AES' operations on known values.

The second attack, depicted in Fig. 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.

V. Attacks on Two-Round AES

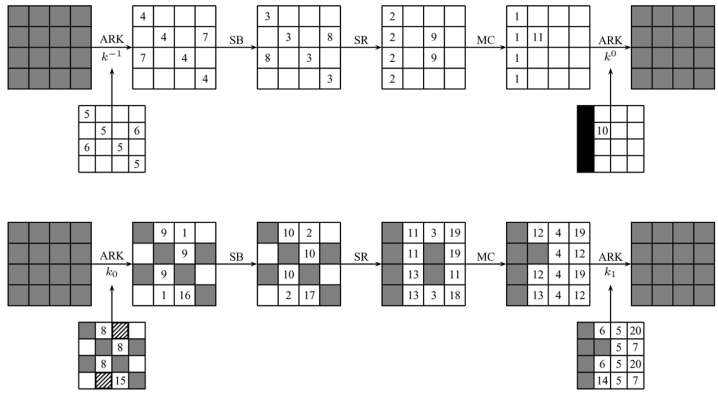
In this section, we consider attacks on two rounds of AES, denoted by rounds 1 and 2. First we present attacks on two *full* rounds with two known plaintexts and then with a single known plaintext. We follow with an improved attack with two known plaintexts that can be applied if the MixColumns operation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

A. Two Known Plaintexts

The attack with two known plaintexts, depicted in Fig. 4, is based on Observation 1. As in the one-round attack with two

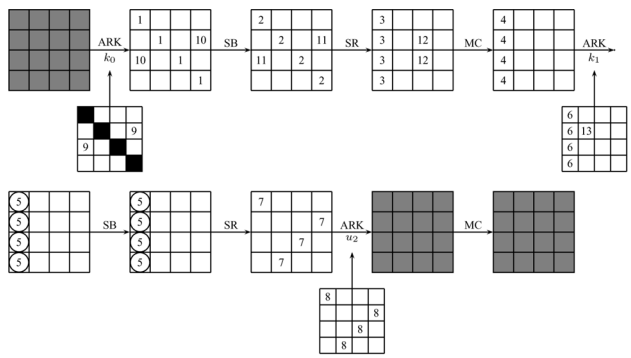
[6]We note that this step can be performed only if the differences in all S-boxes are nonzero. However, since in the known-plaintext attack model it is common to assume that the plaintexts are chosen at random, it is expected that two known plaintexts have nonzero difference in all the 16 bytes with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

[7]We note that the problem of attacking one round AES without the MixColumns operation with a single known plaintext is studied in [32]. It is shown that 2^{16} encryptions are sufficient to retrieve the key.

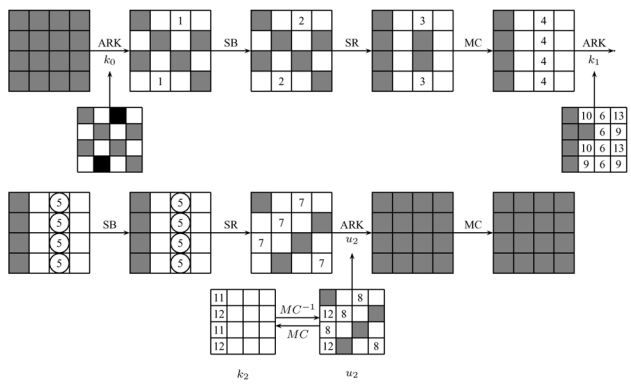


Bytes marked by gray are known (from previous steps of analysis). Bytes marked with tilted lines have at most 12 possible values by Observation 5.

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



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.

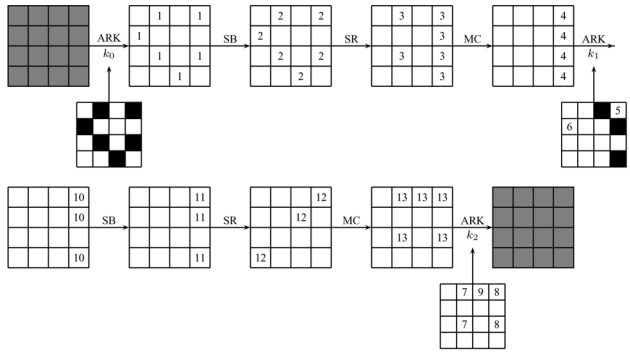
Fig. 4. Attack with two known plaintexts on two-round AES.

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 $u_2 = MC^{-1}(k_2)$.

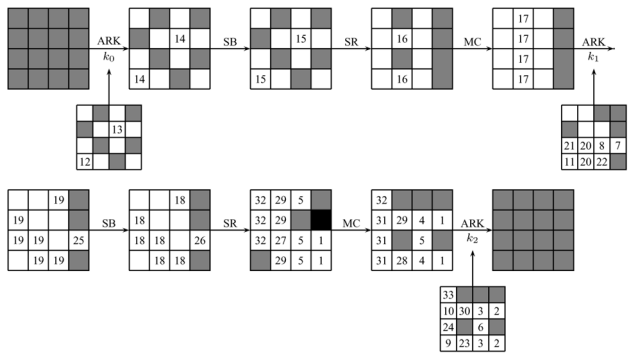
In the first phase of the attack, the adversary guesses bytes 0, 5, 10, 15 of k_0, which allows him to retrieve the intermediate difference in $x_{2,Col(0)}$ (i.e., just before the SubBytes operation of round 2). Then, Observation 1 can be applied to the four S-boxes in that column, yielding their actual 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 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 algo-



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. Attack with one known plaintext on two-round AES.

rithm. 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 Fig. 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 six 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}.

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

2) 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.

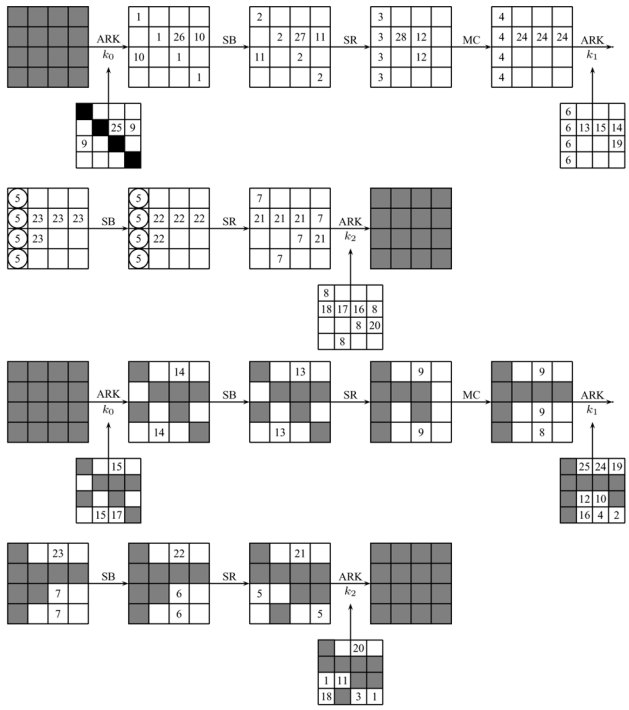
B. One Known Plaintext

It is also possible to attack two-round AES using a single known plaintext. The attack, depicted in Fig. 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



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. Attack on two rounds of AES (Without MixColumns in the second round) using two known plaintexts.

guesses 10 bytes, the time complexity of the attack is 2^{80} encryptions.

1) Time-Memory Tradeoff: The time complexity can be reduced at the expense of enlarging the memory complexity, using nonlinear 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.[8] 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 eight 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 nonlinear equations in b and c, and it turns out that the equations are of the 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.[9]

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 eight subkey bytes (instead of ten) 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.

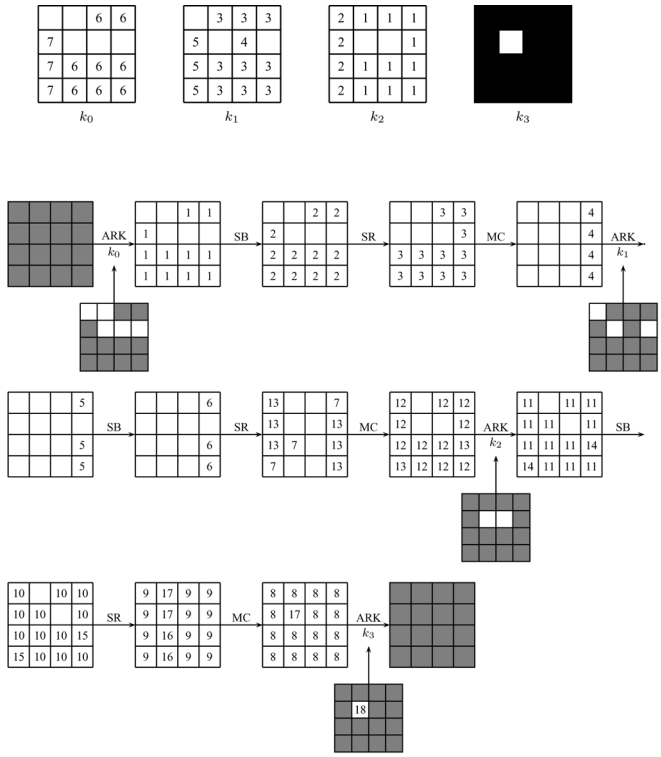
The time complexity of the resulting attack is reduced to 2^{64}, but on the other hand, the attack requires 2^{65} bytes of memory.

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 six byte parameters instead of eight),[10] while the time complexity remains unchanged at 2^{64}.

[8]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 [29], in which this improvement is described in detail.

[9]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 [29].

[10]We note that as in the one-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 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. Attack on three rounds of AES using one known plaintext.

C. Improved Attack When the Second MixColumns is Omitted

In Section VIII, we present a differential attack on six-round AES which uses as a subroutine a two-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 [32] that the omission of the last MixColumns operation in AES reduces the security of the cipher.

The attack, presented in Fig. 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 presented in Section V-A 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.

VI. 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 very time-consuming attack with a *single known* plaintext. Finally, for the sake of completeness, we present a meet-in-the-middle attack with nine *known* plaintexts which is used in Section IX.

A. 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 V-A. 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.

B. One Known Plaintext

The attack with a single known plaintext, depicted in Fig. 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.

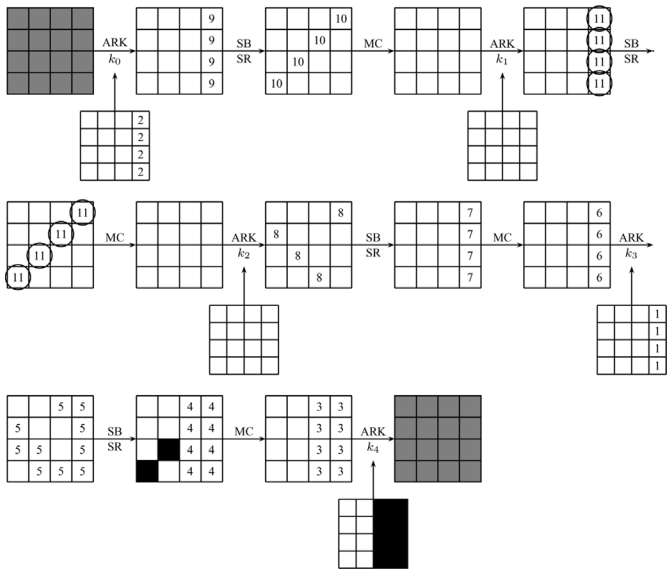


Fig. 8. First phase of the attack on four rounds of AES using two chosen plaintexts. Step 11 is relatively complex and is explained in the text.

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 nonlinear 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 two-round attack presented in Section V-B, and the obtained equations are quite cumbersome, we do not present the improvement here and refer the reader to [29].

C. Nine Known Plaintexts

The attack with nine 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.

VII. Attacks on Four-Round AES

In this section, we consider attacks on four-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.

A. Two Chosen Plaintexts

This attack 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 Fig. 8, the adversary guesses eight 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 the application of AES' operations to known values.

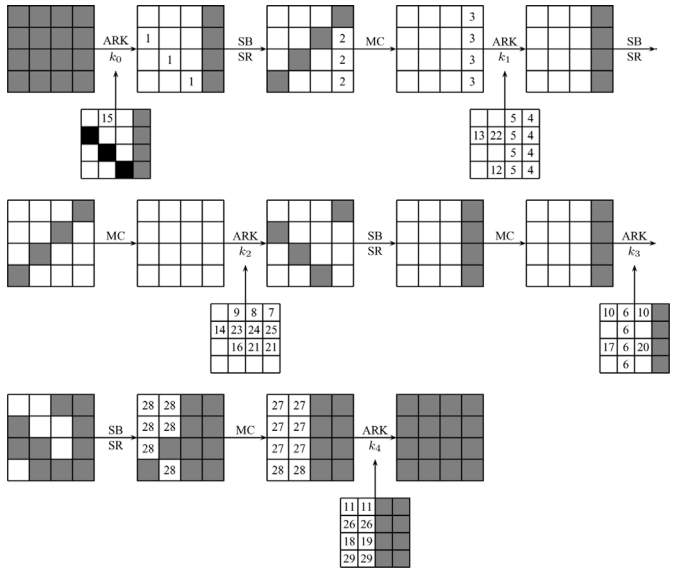


Fig. 9. Second phase of the attack on four rounds of AES using two chosen plaintexts. Step 7 is based on observation 3(5).

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 nonzero 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.

For the sake of completeness, we present also attacks with ten and five chosen plaintexts which are used in Section IX.

B. Ten Chosen Plaintexts

The attack with ten chosen plaintexts is similar to the three-round attack with nine known plaintexts presented in Section VI-C. 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 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 $2^{80} \times 2^{-72} = 2^8$ key suggestions are expected to remain. By repeating the procedure with the three other columns (from the ciphertext side), the adversary

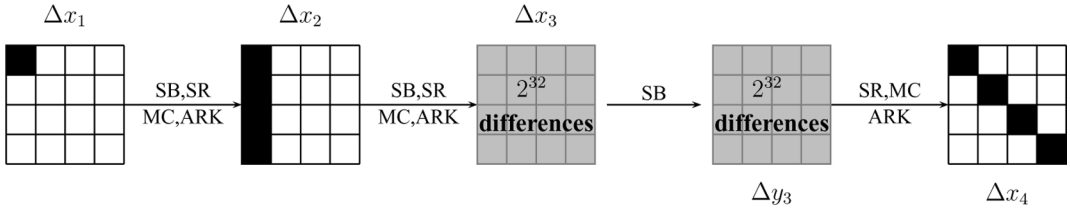


Fig. 10. Three-round truncated differential used in our attack.

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.

C. 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.

VIII. Differential Attack on Six-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 [55], that any four-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 five rounds of AES. Furthermore, the best currently known differential attack on AES-128 is on only four rounds, and all known attacks on five 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 two-round AES presented in Section V-C can be leveraged to a differential attack on six-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 six-round AES in the known-plaintext model. While our attack 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.[11]

Our six-round attack is based on the following three-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 Fig. 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 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.
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 V-C on rounds 5–6.[12]

Since the time complexity of the two-round attack presented in Section V-C is 2^{32} encryptions, the overall complexity of the attack is $2^{24} \times 2^{64} \times 2^{32} = 2^{120}$ 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., [21]).

[11]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 [32] that the omission of the last round MixColumns affects the security of AES.

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

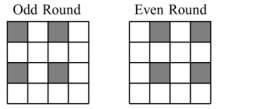


Fig. 11. State bytes which compose the output in odd and even rounds of LEX.

IX. Attack on a Stream Cipher That Follows the Leak Extraction Methodology

In [11], Biryukov introduced 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 exposed as part of the key stream.

Along with the methodology, Biryukov introduced a specific instantiation called LEX which uses AES as the underlying block cipher [12]. In the initialization step of LEX, the publicly known IV is encrypted by AES[13] 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 Fig. 11. After 500 iterations, 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 [12]). 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 considered a very promising candidate and selected to the third (and final) phase of evaluation. However, it was not 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 [33] 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 [34].

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 four-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 four-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 exposed every five 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 the knowledge of the key). Thus, the attack on three-round AES with nine known plaintexts, described in Section VI-C, applies directly to this variant and allows to break it with only 360 bytes of keystream (obtained from nine AES invocations), time complexity of 2^{40} encryptions, and 2^{43} bytes of memory.

Similarly, if the full state is output every five 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 VII-C. A set of $2^{49.5}$ plaintexts picked at random is expected to contain $2^{98} \times 2^{-96} = 4$ 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 four-round AES with ten chosen plaintexts presented in Section VII. 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.

Appendix

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 $k_{0,8}$.

[13]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.

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

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

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 (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 Fig. 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

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

2) Note that $k_{1,5}$ can be computed following the procedure shown in Fig. 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)–(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 Δ_1, Δ_2, Δ_3, and Δ_4 are one-byte parameters depending on the plaintext, the ciphertext, and the guessed subkey bytes.[14]

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 (9). Use these values to update a table of solutions of (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
   a) *Building the table:* For all (a, b) pairs:
      i) Compute Δ_3 (following the first part of (9)).
      ii) For every Δ_4 determine Δ_2 (following the second part of (9)).
      iii) Store in a table (a, b) as the solution of $(\Delta_2, \Delta_3, \Delta_4)$.
   b) *Using the table:*
      i) Guess any 3 bytes of $k_{1,Col(0)}$.

[14]We note that the exact values of the parameters are given in [29] by linearly combining two equations (namely, Equation (0) of [29, p. 11] and three times Equation (1) of [29, p. 11]).

ii) Compute the remaining byte under the assumption that Δ_1 is correct.
iii) 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 [29]. ■

ACKNOWLEDGMENT

We would like to thank Adi Shamir for our lengthy discussions and his suggestions for improvement of this paper. We would also like to thank the anonymous referees for their insightful comments. The support of the Clore Fellowship given to Orr Dunkelman, and the Koshland center for basic research support of Nathan Keller, are greatly acknowledged.

REFERENCES

[1] B. Bahrak and M. R. Aref, "A novel impossible differential cryptanalysis of AES," presented at the presented at the Western Eur. Workshop Res. Cryptol., Bochum, Germany, 2007.
[2] B. Bahrak and M. R. Aref, "Impossible differential attack on 7-round AES-128," *IET (IEE) J. Inf. Security*, vol. 2, no. 2, pp. 28–32, 2008.
[3] R. Benadjila, O. Billet, H. Gilbert, G. Macario-Rat, T. Peyrin, M. Robshaw, and Y. Seurin, "SHA-3 Proposal: ECHO (version 1.5)," SHA-3 submission, 2009.
[4] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert, "Sosemanuk, a Fast Software-Oriented Stream Cipher," eSTREAM submission. , 2005.
[5] D. J. Bernstein, "Understanding brute force," in *Proc. ECRYPT STVL Workshop Symmetric Key Encryption (SKEW)*, 2005 [Online]. Available: http://www.ecrypt.eu.org/stream/papersdir/036.pdf
[6] E. Biham, A. Biryukov, and A. Shamir, "Miss in the middle attacks on IDEA and Khufu," in *Proc. Fast Softw. Encrypt.*, 1999, vol. LNCS-1636, pp. 124–138.
[7] E. Biham, A. Biryukov, and A. Shamir, "Cryptanalysis of skipjack reduced to 31 rounds," in *Proc. EUROCRYPT*, 1999, vol. LNCS-1592, pp. 12–23.
[8] E. Biham and O. Dunkelman, "The SHAvite-3 Hash Function," SHA-3 submission 2009.
[9] E. Biham and N. Keller, Cryptanalysis of Reduced Variants of Rijndael, unpublished, 1999.
[10] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*. New York: Springer, 1993.
[11] A. Biryukov, "The design of a stream cipher LEX," in *Proc. Sel. Areas Cryptogr.*, 2007, vol. LNCS-4356, pp. 67–75.
[12] A. Biryukov, "A new 128-bit key stream cipher LEX," ECRYPT Stream Cipher Project Report 2005/013 [Online]. Available: http://www.ecrypt.eu.org/stream
[13] A. Biryukov, "The Tweak for LEX-128, LEX-192, LEX-256," ECRYPT Stream Cipher Project Report 2006/037 [Online]. Available: http://www.ecrypt.eu.org/stream
[14] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich, and A. Shamir, "Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds," in *Proc. EUROCRYPT*, 2010, vol. LNCS-6110, pp. 299–319.
[15] A. Biryukov and D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256," in *Proc. ASIACRYPT*, 2009, vol. LNCS-5912, pp. 1–18.
[16] A. Biryukov, D. Khovratovich, and I. Nikolic, "Distinguisher and related-key attack on the full AES-256," in *Proc. CRYPTO*, 2009, vol. LNCS-5677, pp. 231–249.
[17] A. Biryukov and D. Wagner, "Slide attacks," in *Proc. Fast Softw. Encrypt.*, 1999, vol. LNCS-1636, pp. 245–259.
[18] A. Biryukov and D. Wagner, "Advanced slide attacks," in *Proc. EUROCRYPT*, 2000, vol. LNCS-1807, pp. 586–606.
[19] A. Bogdanov, Cryptanalysis of the KeeLoq Block Cipher IACR ePrint report 2007/055.
[20] C. Bouillaguet, P. Derbez, and P.-A. Fouque, "Automatic search of attacks on round-reduced AES and applications," in *Proc. CRYPTO*, 2011, vol. LNCS-6841, pp. 169–187.
[21] J. H. Cheon, M. Kim, K. Kim, J.-Y. Lee, and S. Kang, "Improved impossible differential cryptanalysis of Rijndael and Crypton," in *Proc. Inf. Security Cryptol.*, 2002, vol. LNCS-2288, pp. 39–49.
[22] N. T. Courtois, Security evaluation of GOST 28147-89 in view of international standardisation IACR ePrint report 2011/211.
[23] N. T. Courtois, G. V. Bard, and D. Wagner, "Algebraic and slide attacks on KeeLoq," in *Proc. Fast Softw. Encrypt.*, 2008, vol. LNCS-5086, pp. 97–115.
[24] J. Daemen and V. Rijmen, "AES proposal: Rijndael," in NIST AES Proposal, 1998.
[25] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard*. New York: Springer, 2002.
[26] J. Daemen and V. Rijmen, "A new MAC construction ALRED and a specific instance, ALPHA-MAC," in *Proc. Fast Softw. Encrypt.*, 2005, vol. LNCS-3557, pp. 1–17.
[27] J. Daemen and V. Rijmen, The Pelican MAC Function IACR ePrint report 2005/088.
[28] H. Demirci and A. A. Selçuk, "A meet-in-the-middle attack on 8-round AES," in *Proc. Fast Softw. Encrypt.*, 2008, vol. LNCS-5086, pp. 116–126.
[29] P. Derbez, "Rapport de Stage," report of Internship at École Normale Supérieure Sep. 2010.
[30] I. Dinur and A. Shamir, Side channel cube attacks on block ciphers IACR ePrint report 2009/127.
[31] I. Dinur, O. Dunkelman, and A. Shamir, Improved attacks on full GOST IACR ePrint report 2011/558.
[32] O. Dunkelman and N. Keller, "The effects of the omission of last round's mixcolumns on AES," *Inf. Process. Lett.*, vol. 110, no. 8–9, pp. 304–308, 2010.
[33] O. Dunkelman and N. Keller, "A new attack on the LEX stream cipher," in *Proc. ASIACRYPT*, 2008, vol. LNCS-5350, pp. 539–556.
[34] O. Dunkelman and N. Keller, "Cryptanalysis of the stream cipher LEX," *Designs, Codes, Cryptogr.*, 2010, to be published.
[35] O. Dunkelman, N. Keller, and A. Shamir, "A practical-time attack on the A5/3 cryptosystem used in third generation GSM telephony," in *Proc. CRYPTO*, 2010, vol. LNCS-6223, pp. 393–410.
[36] O. Dunkelman, N. Keller, and A. Shamir, "Improved single-key attacks on 8-round AES-192 and AES-256," in *Proc. ASIACRYPT*, 2010, vol. LNCS-6477, pp. 158–176.
[37] O. Dunkelman, N. Keller, and A. Shamir, ALRED blues: New attacks on AES-based MACs IACR ePrint report 2011/095.
[38] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved cryptanalysis of Rijndael," in *Proc. Fast Softw. Encrypt.*, 2001, vol. LNCS-1978, pp. 213–230.
[39] H. Gilbert and M. Minier, "A collision attack on 7 rounds of Rijndael," in *Proc. 3rd AES Candidate Conf.*, New York, 2000, pp. 230–241.
[40] M. Hell and T. Johansson, "Breaking the F-FCSR-H stream cipher in real time," in *Proc. ASIACRYPT*, 2008, vol. LNCS-5350, pp. 557–569.
[41] S. Indesteege, N. Keller, O. Dunkelman, E. Biham, and B. Preneel, "A practical attack on KeeLoq," in *Proc. EUROCRYPT*, 2008, vol. LNCS-4965, pp. 1–18.
[42] T. Isobe, "A single-key attack on the full GOST block cipher," in *Proc. Fast Softw. Encrypt.*, 2011, vol. LNCS-6733, pp. 290–305.
[43] O. Kara and C. Manap, "A new class of weak keys for blowfish," in *Proc. Fast Softw. Encrypt.*, 2007, vol. LNCS-4593, pp. 167–180.
[44] D. Khovratovich, A. Biryukov, and I. Nikolic, "Speeding up collision search for byte-oriented hash functions," in *Proc. Cryptographers Track, RSA Conf.*, 2009, vol. LNCS-5473, pp. 164–181.
[45] J. Kim, S. Hong, and B. Preneel, "Related-key rectangle attacks on reduced AES-192 and AES-256," in *Proc. Fast Softw. Encrypt.*, 2007, vol. LNCS-4593, pp. 225–241.
[46] L. R. Knudsen and V. Rijmen, "Known-key distinguishers for some block ciphers," in *Proc. ASIACRYPT*, 2007, vol. LNCS-4833, pp. 315–324.
[47] Ö. Küçük, "The Hash Function Hamsi," SHA-3 submission, 2009.
[48] J. Lu, O. Dunkelman, N. Keller, and J. Kim, "New impossible differential attacks on AES," in *Proc. INDOCRYPT*, 2008, vol. LNCS-5365, pp. 279–293.

[49] S. Lucks, "Attacking seven rounds of Rijndael under 192-bit and 256-bit keys," in *Proc. 3rd AES Candidate Conf.*, New York, 2000, pp. 215–229.
[50] S. Manuel and T. Peyrin, "Collisions on SHA-0 in one hour," in *Proc. Fast Softw. Encrypt.*, 2008, vol. LNCS-5086, pp. 16–35.
[51] Advanced Encryption Standard US National Institute of Standards and Technology, 2001, , Federal Information Processing Standards Publications Number 197.
[52] M. A. Simplício Jr., P. d'Aquino, F. F. S. Barbuda, P. S. L. M. Barreto, T. C. M. B. Carvalho, and C. B. Margi, "The MARVIN message authentication code and the LETTERSOUP authenticated encryption scheme," *Security Commun. Netw.*, vol. 2, no. 2, pp. 165–180, 2009.
[53] P. Stankovski, M. Hell, and T. Johansson, "An efficient state recovery attack on X-FCSR-256," in *Proc. Fast Softw. Encrypt.*, 2009, vol. LNCS-5665, pp. 23–37.
[54] M. Stevens, Fast collision attack on MD5 2006, IACR ePrint report 2006/104.
[55] S. Park, S. H. Sung, S. Chee, E.-J. Yoon, and J. Lim, "On the security of Rijndael-like structures against differential and linear cryptanalysis," in *Proc. ASIACRYPT*, 2002, vol. LNCS-2501, pp. 176–191.
[56] R. C.-W. Phan, "Impossible differential cryptanalysis of 7-round advanced encryption standard (AES)," *Inf. Process. Lett.*, vol. 91, no. 1, pp. 33–38, 2004.
[57] S. Wu, M. Wang, and Z. Yuan, A flaw in the internal state recovery attack on ALPHA-MAC IACR ePrint report 2010/160.
[58] Z. Yuan, W. Wang, K. Jia, G. Xu, and X. Wang, "New birthday attacks on some MACs based on block ciphers," in *Proc. CRYPTO*, 2009, vol. LNCS-5677, pp. 209–230.
[59] W. Zhang, W. Wu, and D. Feng, "New results on impossible differential cryptanalysis of reduced AES," in *Proc. Int. Conf. Inf. Security, Cryptology*, 2007, vol. LNCS-4817, pp. 239–250.
[60] W. Zhang, W. Wu, L. Zhang, and D. Feng, "Improved related-key impossible differential attacks on reduced-round AES-192," in *Proc. Sel. Areas Cryptogr.*, 2007, vol. LNCS-4356, pp. 15–27.

Charles Bouillaguet is a postdoc in the cryptology team of the Versailles-Saint-Quentin university. His interests cover cryptanalysis of both secret-key and public-key schemes.

Patrick Derbez is a Ph.D. student under the supervision of Pierre-Alain Fouque in ENS. He is mainly interested in low-data cryptanalysis of (reduced versions of) the AES.

Orr Dunkelman is currently a faculty member at the Computer Science Department of the University of Haifa in Israel. His research interests cover symmetric key design and analysis, computer security and privacy.

Pierre-Alain Fouque is an assistant professor at École normale supérieure in Paris. His main interests are cryptanalysis of multivariate schemes and symmetric primitives as well as side-channel attacks.

Nathan Keller is a faculty member at the Mathematics Department of the Bar Ilan University, Ramat Gan, Israel. His main research interests are Probabilistic Combinatorics and Cryptography.

Vincent Rijmen (senior member) is full professor at the University of Leuven, Dept. ESAT/IBBT (Belgium) and part-time professor at the Graz University of Technology (Austria). His research interests include design and cryptanalysis of ciphers and cryptographic hash functions. He is co-designer of the Advanced Encryption Standard (AES) and the ISO/IEC standard hash function Whirlpool.