ALRED Blues: New Attacks on AES-Based MACs

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

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

Abstract. The ALRED family of Message Authentication Codes (MACs) is based on three principles: Using a *keyless* block cipher in CBC mode to process the message, choosing *AES-128* as this cipher, and *reducing the effective number of rounds to 4* in order to speed up the processing. In this paper we show that each one of these principles creates significant weaknesses. More specifically, we show that any ALRED-type MAC which uses a keyless block cipher is vulnerable to new time/memory tradeoff attacks which are faster than generic tradeoff attacks on one-way functions. We then use the special properties of keyless AES to attack any number of rounds (4, 10, or a million) by forging the MAC of essentially any desired message in negligible time and space after a one-time preprocessing stage requiring 2^{96} time and negligible space. For the recommended 4-round version we show how to do the same using an improved preprocessing stage with a semi-practical time complexity of 2^{65}, which is the best one can hope for in such MAC constructions. Finally, we show that even if we replace the 4-round keyless AES by a 5-round or a 6-round version with additional secret round keys we can still compute such MACs much faster than via exhaustive search.

1 Introduction

Message Authentication Codes (MACs) are designed to compute for each input string a hard-to-forge authenticator. They combine the properties of hash functions (by dealing with arbitrarily long inputs), symmetric encryption algorithms (by using a secret shared key) and signature schemes (by dealing with forgery rather than with secrecy). The main security requirement is that even after choosing (in a possibly adaptive way) polynomially many messages m_i and obtaining their corresponding tags t_i for some unknown key k of length n, the adversary should not be able to compute the tag t for a new message m under the same key in time which is substantially smaller than the 2^n complexity of exhaustive search. The weakest and strongest flavors of this problem are called *existential forgery* where m is some message which can depend on all the previous queries, and *universal forgery* where m can be any message chosen before the adversary queries the MAC oracle.

There are many known constructions for MACs which use other cryptographic primitives as starting points. In particular, it is possible to turn any block cipher into a MAC by using it in CBC mode and defining the tag as the final ciphertext block it produces. However, block ciphers are designed with a different goal in mind (namely, to hide any information about the plaintext given the ciphertext), which can be an overkill in a MAC application in which the plaintext is actually given to the adversary. Conceivably, one can get higher speed by using weaker versions of known block ciphers in order to process each plaintext block. Such a design

will not be sufficiently secure as a stand-alone cryptosystem, but it can still be an excellent MAC which makes it very difficult to compute new tags from given tags.

The ALRED family of MACs [8] was designed in 2005 by the very experienced team of Joan Daemen and Vincent Rijmen, who developed AES in the late 1990's. They used a keyless round-reduced version of AES-128 to process the message blocks in CBC mode, and used the secret key only in order to encrypt the initial value and the final ciphertext block (in both cases using a full keyed AES). In their paper, Daemen and Rijmen proposed two instantiations of the ALRED family: *Alpha-MAC* which cuts the input into short blocks of 32 bits each and applies only one round of keyless AES to each block, and *Pelican* which uses standard sized blocks of 128 bits and applies four rounds of keyless AES to each block. Compared to the standard 10-round AES-128, both variants use only 40% of the number of rounds to process each chunk of 128 input bits, and eliminate one of the four steps (the AddRoundKey operation) in each round.

In any CBC construction in which the final MAC output is produced by a strong keyed cryptosystem, the only feasible way to predict the tag for a new message is to collide it with some previous message whose tag is already known. Such internal collisions are not likely to exist whenever we query fewer than $2^{n/2}$ messages, which led the designers of ALRED to limit the number of allowed messages in their scheme to this bound. On the other hand, once we exceed this bound, we expect to find the first colliding pair of messages, and then we can create arbitrarily many additional collisions by concatenating the same additional blocks after the two colliding messages. By querying the tag of one of these extended messages, we can safely predict that the tag of the other extended message in the pair will be the same. This provides an optimal existential forgery attack on any such scheme whose time and data complexities are $2^{n/2}$.

In this paper we develop an almost universal forgery attack on the ALRED construction, which is a much stronger form of attack. In some of our attacks, we can find in linear time and space the tag of essentially any desired message m chosen in advance, after performing a *one-time precomputation* in which we query the MAC on $2^{n/2}$ messages which are completely unrelated to m. The only sense in which this is not a universal forgery attack is that we need the ability to modify one message block in m in an easy to compute way. Our basic technique is to use the precomputation in order to recover one of the chaining values used in one of the queried messages. Since the message processing consists of a sequence of keyless permutations, we can run such a known value forwards or backwards to find all the other chaining values, and in particular the encrypted value of the IV under the secret key. Once we know this value, we can follow the evolution of the chaining value when we MAC any new message. In the case of Pelican, we can now modify just one of the 128-bit message blocks (it does not matter which) to be the XOR of two known chaining values, and force the input to the final AES encryption to be any value which was used in some previous query. Consequently, we can predict in linear time the tag of any selected message under the unknown key provided that we can modify any one of its blocks.

The best previous attack of this type on the ALRED family [17] was published at CRYPTO 2009 by Xiaoyun Wang et al. In the case of Alpha-MAC, their attack was optimal, requiring a preprocessing stage of 2^{65} time and data in order to recover one of the secret internal states. However, in the case of Pelican, the best such attack required $2^{85.5}$ time and data during the preprocessing stage, leaving a considerable gap of about one million between the upper and lower bound.[3]

[3] The results presented in this paper were communicated by us during the ASIACRYPT 2010 rump session. A recent result concerning an automatic tool for identification of cryptanalytic attacks [4], which is going to be presented at CRYPTO 2011, re-discovered some of our results, thus verifying their correctness.

Attack Type	Number of AES Rounds	Keyed or Keyless	Complexity Data	Time	Memory
Yuan et al. Imp. Diff. [17]	4	Keyless	$2^{85.5}$ CM	$2^{85.5}$	2^{64}
Diff. MitM (Sect. 6)*	4	Keyless	2^{65} CM	2^{65}	2^{64}
Imp. Diff. (App. A)	5	Keyed	2^{80} CM	2^{80}	2^{78}
Imp. Diff. (App. A)	6	Keyed	2^{110} CM	2^{110}	2^{64}
Generic (Sect. 5)	any	Keyless	2^{96} CM	2^{96}	2^{32}
Generic (Sect. 5)	any	Keyless	2^{96} ACM	2^{96}	1
Generic (Sect. 3)	any	Keyless	$2^{85.3}$ CM	$2^{85.3}$	$2^{85.3}$
Marvin (Sect. 5.3)	any	Keyless	2^{96} ACM	2^{96}	1

CM — Chosen message, ACM — Adaptively chosen message.
Time complexity is measured in MAC evaluation units.
* — This result was re-discovered in [4], see footnote 1.

Table 1. Summary of Attacks on Enhanced Variants of Pelican

In this paper we describe several new attacks on the ALRED family. Even though they do not invalidate the designers' formal security claims (which only refer to cases where the limited number of available messages makes the existence of internal collisions highly unlikely), they strongly suggest that the optimizations proposed by Daemen and Rijmen were too aggressive, and that even milder simplifications are still risky. In particular, we show that any ALRED-type MAC which uses a keyless encryption algorithm to process the messages can be attacked by an improved time/memory tradeoff attack which requires only $O(2^{2n/3})$ *online and offline* time to handle all the possible keys and messages, whereas standard tradeoff attacks on one-way functions require $O(2^{2n/3})$ online time and $O(2^n)$ offline time. We then exploit a little-known property of keyless AES to show that regardless of the number of keyless rounds we use, both Alpha-MAC and Pelican can be broken in negligible online time after a one-time preprocessing stage which requires 2^{96} time and negligible space for any given key. We then describe how to leverage almost any type of possible or impossible differential attack on reduced-round AES into an attack on the Pelican MAC construction. For the official choice of 4 rounds, we develop a new attack whose 2^{65} time complexity is optimal for such 128-bit MACs. Combined with Wang's analysis of Alpha-MAC, this completes the analysis of all the recommended variants of the ALRED family by providing tight bounds on their security. Finally, we show that even versions with a larger number of rounds are vulnerable, by breaking a 5-round version of Pelican in 2^{80} time, and by breaking a 6-round version of Pelican in 2^{110} time. Interestingly, all these attacks could be applied with the same complexities even if the designers had added an independent secret 128-bit key to each one of the rounds.

2 The ALRED Construction and its Instantiations Alpha-MAC and Pelican

As described in the introduction, ALRED is a MAC construction based on an iterated block cipher. Given a secret key k whose length is equal to the key length of the block cipher, and a message m, the generation of the tag is composed of four steps:

1. **Message padding and splitting:** The message is padded with a single 1 and the minimum number of 0's so that the resulting length is a multiple of l_w bits, where l_w is a characteristic of the MAC. The padded message is then divided into blocks $m_1, m_2, \ldots, m_l$ of length l_w each.

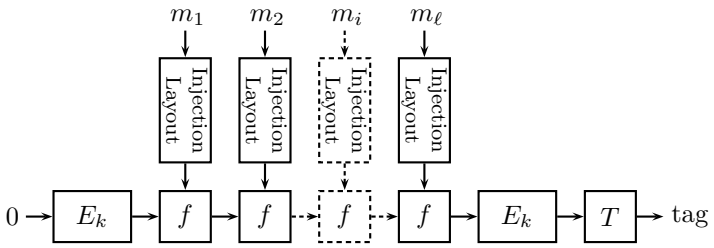

Fig. 1. The ALRED construction

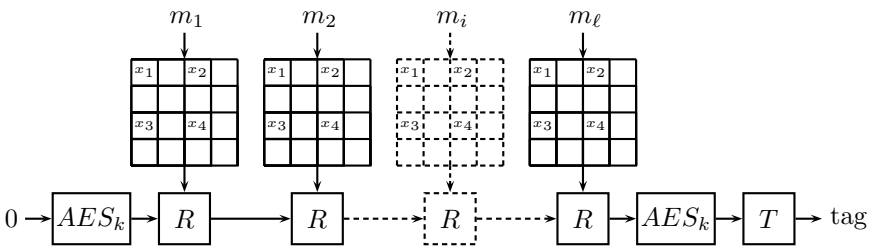

Fig. 2. The structure of Alpha-MAC

2. **State initialization:** The state is initialized with the all-zero block, and then the block cipher is applied to it, to obtain $y_0 = E_k(0)$.
3. **Chaining:** The following iteration is applied to the blocks $m_1, m_2, \ldots, m_l$ sequentially:
   - The message block m_i is mapped to an injection input Inj_i whose length is equal to the length of r round keys of the block cipher.
   - A reduced r-round variant of the block cipher is applied to the state y_{i-1}, with Inj_i replacing the round keys. The resulting state is denoted by y_i.
4. **Finalization:** The block cipher is applied to the state y_l to obtain $z' = E_k(y_l)$. Then z' is truncated to the required length to obtain the tag $z = Truncate(z')$.

The construction is illustrated in Figure 1, where f denotes a reduced r-round variant of the block cipher.

In this paper we focus on two concrete instantiations of the ALRED construction, proposed by the ALRED designers [10]. Both of them use AES [7] as the underlying block cipher, and they differ in the block length l_w, in the number r of rounds in the reduced variant of AES, and in the injection inputs used in the chaining step.

2.1 Alpha-MAC

In Alpha-MAC, the number of rounds is $r = 1$, the block length is $l_w = 32$ bits, and the injection input consists of inserting the 32 bits of the message block into four specific bytes of the round key, and inserting zero into the remaining 12 bytes. The structure of Alpha-MAC is shown in Figure 2, where R denotes a single AES round, and x_1, x_2, x_3, and x_4 denote the positions in which the message block is injected.

2.2 Pelican

In Pelican, the number of rounds is $r = 4$, the block length is $l_w = 128$ bits, and the injection input consists of using the message as the first round subkey, and zeros as the other three

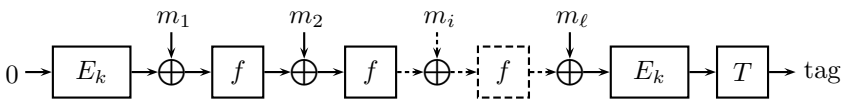

Fig. 3. The structure of Pelican

subkeys. An equivalent description, which is more convenient for our analysis, is that in the i'th iteration of the chaining step, the 128-bit message block m_i is XORed to the state y_{i-1}, and then four-round *keyless AES* (i.e., AES without the AddRoundKey operations) is applied to obtain the next state y_i. The equivalent description is shown in Figure 3, where f denotes four-round keyless AES.

3 Generic Attack on the ALRED Construction

In this section we present a generic attack on the ALRED construction, which is similar in spirit to Daemen's chosen plaintext attack [6] on the Even-Mansour encryption scheme [11] and to the attack of Coppersmith et al. [5] on MacDES. The attack is completely independent of the underlying block cipher, and of the number of rounds in its reduced variant. The only features exploited in the attack are the fact that the function applied in the chaining step is keyless, and the structure of the injection input. For the sake of simplicity, we consider first Pelican-type constructions. We present a basic attack which requires data, memory and online time complexities of slightly more than $2^{l_w/2}$, and a preprocessing of 2^{l_w} operations. Then we show a tradeoff that allows to reduce the preprocessing time, at the expense of increasing the data and time complexities. Finally, we show that a variant of the attack is applicable also to instantiations of ALRED with other classes of injection inputs, such as Alpha-MAC-type constructions.

3.1 The Basic Attack on Pelican-type Constructions

In a Pelican-type construction, we assume that the structure of the MAC is as shown in Figure 3, where E is some block cipher (possibly provably secure), and f is some keyless permutation (which is possibly perfectly random). Note that for such constructions, both E and f operate on l_w-bit blocks.

The algorithm of the basic attack is as follows.

1. **Preprocessing phase.** Choose an arbitrary non-zero l_w-bit value Δ, and perform the following operations for each l_w-bit value C:
   (a) Compute the values $P = f^{-1}(C), P^* = f^{-1}(C \oplus \Delta)$.
   (b) Check whether the first $l_w/2$ bits of $P \oplus P^*$ are zeros. If this is the case, store the pair (P, P^*) in a hash table indexed by the last $l_w/2$ bits of the difference $P \oplus P^*$.
2. **On-line phase.**
   (a) Ask for the MAC evaluation of two structures S_1 and S_2 of $2^{l_w/2}$ two-block messages each, such that:
   - In S_1, in all messages, the first $l_w/2$ bits of the block m_1 are zeros, the remaining $l_w/2$ bits of m_1 assume all the $2^{l_w/2}$ possible values, and the block m_2 is fixed to zero.
   - In S_2, the first $l_w/2$ bits of the block m_1 are zeros, the remaining $l_w/2$ bits of m_1 assume all the $2^{l_w/2}$ possible values, and the block m_2 is fixed to Δ.

(b) Insert the tags into a hash table, and search for an internal collision (i.e., collision before the final application of $E_k(\cdot)$).[4]
(c) If the messages $(m_1, m_2) \in S_1$ and $(m_1^*, m_2^*) \in S_2$ form an internal collision, look at the table prepared in the preprocessing phase, in the cell corresponding to $m_1 \oplus m_1^*$. For each pair (P, P^*) in the cell, assume that $E_k(0) = P \oplus m_1$, and verify the guess using the MAC evaluation of one of the messages in the data set.

Analysis of the algorithm The table constructed in the preprocessing phase contains all the pairs of input/output values of f in which the output difference is Δ, and the first $l_w/2$ bits of the input difference are zeros. It is expected that the number of pairs (P, P^*) in the table is close to $2^{l_w/2}$, and that for each possible value of $P \oplus P^*$, the table contains at most several pairs corresponding to that difference.

The idea behind the algorithm is that if $(m_1, m_2) \in S_1$ and $(m_1^*, m_2^*) \in S_2$ form an internal collision, then the inputs to the function f in the first step of the chaining processes of (m_1, m_2) and of (m_1^*, m_2^*) must form a pair in the pre-constructed table.

To prove this claim, denote the intermediate values in the chaining processes of (m_1, m_2) and of (m_1^*, m_2^*) by (y_0, y_1, y_2), and (y_0^*, y_1^*, y_2^*), respectively. Denote the inputs to the function f in the first step of the chaining processes by x and x^*. In order to show that the pair (x, x^*) appears in the table, we have to show that the difference between the corresponding outputs of f, which are y_1 and y_1^*, is Δ, and that the first $l_w/2$ bits of $x \oplus x^*$ are zeros.

This indeed follows from the structure of S_1 and S_2. On the one hand, since (m_1, m_2) and (m_1^*, m_2^*) form an internal collision, we have $y_2 = y_2^*$. Since the function $f(\cdot)$ is invertible, this implies that $y_1 \oplus y_1^* = m_2 \oplus m_2^* = \Delta$, as required. On the other hand, by definition of the ALRED construction, $y_0 = y_0^* = E_k(0)$, and thus, $x \oplus x^* = m_1 \oplus m_1^*$. By the choice of the structures, the first $l_w/2$ bits of both m_1 and m_1^* are zeros, and thus the first $l_w/2$ bits of $x \oplus x^*$ are zeros as well.

Hence, the pair (x, x^*) indeed appears in the pre-computed table, and as mentioned before, there are likely to be only a few pairs in the table with the difference $m_1 \oplus m_1^*$. Since $x = E_k(0) \oplus m_1$, each such pair yields a single suggestion for the initial state $y_0 = E_k(0)$.

The data complexity of the attack is $2^{l_w/2+1}$ chosen messages, and the time complexity is dominated by evaluating the MAC value of the messages. The memory required for the attack is $l_w \cdot 2^{l_w/2+1}$ bits. The attack succeeds for sure once an internal collision is encountered, and hence, the success probability of the attack is $1 - 1/e$ (which is the probability that the given data set during the on-line phase of the attack contains an internal collision).

3.2 Preprocessing/Data Tradeoff

If the preprocessing time available to the adversary is smaller than 2^{l_w}, she can still mount a variant of the attack, but with a higher data complexity.

Assume that the available preprocessing time is 2^{l_w-k}. In such situation, during the preprocessing phase, the adversary is able to check only a 2^{-k} fraction of the pairs $(C, C \oplus \Delta)$, and thus, the table contains only a small portion of the pairs (P, P^*) with the prescribed input and output differences. As a result, the adversary cannot assure that the pair (x, x^*) obtained from the internal collision appears in the pre-computed table. However, if the adversary will examine

[4] If the length of the tag is l_w bits then a collision in the tag value results from an internal collision with high probability. If the tag is shorter, there can be many false alarms, but the adversary can verify that a collision is internal by appending to the two messages the same block m_3 and checking whether the new tags also collide. For the sake of simplicity, we assume in the sequel that the tag length is l_w bits.

2^k random internal collisions, then with probability of $1 - 1/e$, one of them would appear in the table and allow to retrieve the initial state $E_k(0)$.

Formally, the attack algorithm is similar to the original attack, with the following changes:

1. In the preprocessing phase, the adversary checks $2^{l_w - k}$ pairs of the form $(C, C \oplus \Delta)$, and stores the corresponding plaintext pair (P, P^*) in the table if the first $(l_w - k)/2$ bits of $P \oplus P^*$ are zeros. The table is indexed by the $(l_w + k)/2$ last bits of $P \oplus P^*$.
2. In the on-line phase, in both structures, only the $(l_w - k)/2$ first bits of m_1 are fixed to zeros, while the remaining $(l_w + k)/2$ bits assume all the $2^{(l_w+k)/2}$ possible values. Thus, the size of the structures is increased to $2^{(l_w+k)/2}$, and the adversary expects to obtain 2^k internal collisions. For each collision, the adversary checks the pre-computed table like in the basic attack, and it is expected that in one of the internal collisions, the pair (x, x^*) exists in the table and allows to retrieve $E_k(0)$.

The preprocessing phase of the attack requires $2^{l_w - k}$ operations, and the data and time complexities are $2^{(l_w+k)/2+1}$. The memory requirement is $l_w \cdot 2^{(l_w+k)/2+1}$ bits.
Since such attack is possible for any value of k, this yields the tradeoff curve:

$$PD^2 = 2^{2l_w},$$

where P is the preprocessing time, and D is the data complexity. In particular, the adversary can mount an attack with overall complexity of $2^{2l_w/3}$, without using any additional preprocessing time.

3.3 Variants of the Attack Applicable to Other ALRED Constructions

In this section we present a variant of the generic attack applicable to constructions of the type of Alpha-MAC. Following the same ideas, one can construct attacks on other ALRED constructions as well, where the exact structure of the attack depends on the *injection input* used in the attacked construction.

In the attack we assume that the MAC has the structure described in Figure 2, where R is any keyless function (which may be provably secure). The difference between this structure and Pelican-type structures considered in Section 3.1 is that in Alpha-MAC, the adversary can control only 32 of the 128 "subkey" bits inserted before the keyless function, while the remaining 96 bits are forced to be zeros. As a result, the adversary is forced to use structures of size at most 2^{32}, and hence, 2^{64} pairs of structures are required in order to obtain an internal collision.

Apart from this difference, the attack is essentially similar to the attack on Pelican-type structures. Let the *injection positions* be the four byte positions in which the message is injected into the state (i.e., the positions denoted by x_1, x_2, x_3, x_4 in Figure 2). In the preprocessing phase, the adversary chooses a non-zero difference Δ such that only the injection positions in Δ may be non-zero. Then, she constructs the table like in the original attack, but stores only pairs (P, P^*) in which $P \oplus P^*$ is non-zero only in the injection positions. In the online phase of the attack, the adversary considers 2^{64} pairs of structures of 2^{32} four-block messages each, such that in each pair of structures, the first two blocks are fixed to the same value (m_1, m_2) in both structures, the fourth blocks are fixed to two values whose difference is Δ, and the third blocks assume all the 2^{32} possible values. (The first two blocks are required in order to supply 2^{64} different pairs of structures.) The adversary looks for an internal collision, and by a table lookup she finds a few suggestions to the value of the intermediate state after the insertion of m_3. Then she rolls back the process to obtain suggestions for $E_k(0)$, and checks them using the already obtained data.

The data, time, and memory complexities of the attack are slightly above 2^{96}. Like in the attack on Pelican-type constructions, the preprocessing complexity can be reduced at the expense of increasing the data complexity. However, this time the tradeoff curve is worse, since the adversary cannot increase the size of the structures, and thus a decrease of factor 2^k in the preprocessing time leads to an increase of 2^k in the data complexity.[5] Hence, the resulting curve is:

$$PD = 2^{224}.$$

In general, if the size of the state is l_b (and thus R operates on l_b-bit blocks), and the adversary can control s out of the l_b bits inserted into the state, then the possible values on the tradeoff curve are:

$$\begin{cases} PD^2 = 2^{2l_b}, & 2^{l_b/2} \leq D \leq 2^s, \\ PD = 2^{2l_b - s}, & \max(2^s, 2^{l_b - s}) \leq D \leq 2^{l_b}. \end{cases}$$

We note that in the specific case of Alpha-MAC, where the function f is 1-round keyless AES, the attack procedure has to be altered a bit, since as shown in [8], due to the structure of AES, a difference which is non-zero only in the injection positions cannot lead through a single AES round to a difference which is also non-zero only in the injection positions. Actually, at least 4 AES rounds are needed in order to make such transition possible. In order to overcome this problem, the adversary considers 7-block messages, where the fourth, fifth, and sixth blocks are fixed to zero in all messages. After making this change, the attack described above applies to the original Alpha-MAC. Moreover, the attack can be improved (in the specific case of Alpha-MAC) using the fact that the mixing in one-round AES is weak. The resulting attack has data and time complexities of slightly more than 2^{64}, with a preprocessing of 2^{128}. However, this attack is inferior to the attack on Alpha-MAC presented in [17], and thus we omit the details here.

4 Leveraging Differential-type Attacks on the Underlying Reduced Block Cipher to an Attack on the Full ALRED

In Section 3, we presented a generic attack on the ALRED construction which exploits the fact that the function f used in the chaining step is keyless. We showed that the attack is applicable even if f has no cryptographic weaknesses and behaves as a random function. In this section we present an attack which can be applied whenever the function f is weak with respect to some differential-type attack, which is often the case for reduced-round block ciphers (like 4-round AES suggested by the ALRED designers). While the generic attacks described in the previous section apply only if f is keyless, this attack can be applied even if f is keyed.

Due to space constraints, we consider in this section only Pelican-type constructions. The applicability of the attack to other ALRED constructions is discussed in Appendix A.

4.1 The Generic Leveraging Procedure for Pelican-type Constructions

The main observation used in the attack is that two-block internal collisions in a Pelican-type scheme can be viewed as input/output pairs for the block cipher f, where the unknown initial state $E_k(0)$ is viewed as an initial whitening key of f. In order to understand the observation, we consider an equivalent representation of the first two steps of a Pelican-type construction, presented in Figure 4. In terms of the equivalent representation, given a pair of two-block messages (m_1, m_2) and (m_1^*, m_2^*), we treat (m_1, m_1^*) as an input pair for the block cipher f. If

[5] Note that in order to obtain more than 2^{64} pairs of structures, the adversary has to append another block in the beginning of the messages.

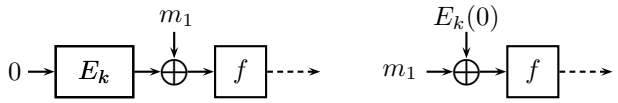

Fig. 4. The original structure of Pelican (on the left) and the equivalent structure

(m_1, m_2) and (m_1^*, m_2^*) form an internal collision, then we know that the difference between the corresponding outputs of f must be $m_2 \oplus m_2^*$, since f is a permutation and thus the only way to create a zero difference after the second f is to have a zero difference before it.

Hence, even if the function f is keyed, any local collision provides the adversary with a pair of input/output values for f, in which the actual inputs (i.e., m_1 and m_1^*) and the output difference (i.e., $m_2 \oplus m_2^*$) are known to the adversary. This makes it possible to leverage several classes of differential-type attacks on f to attacks on the MAC construction.

Specifically, the usual structure of differential-type attacks (both ordinary differential attacks and impossible differential attacks) is as follows:

1. The analyzed cipher E is considered as a cascade $E = E_1 \circ E' \circ E_0$, where E' is the "core" of the cipher, and E_0 and E_1 are either empty or consist of a few rounds.
2. The adversary finds a *differential-type distinguisher* for E', showing that if a pair of inputs of E' has some prescribed difference α, then the corresponding outputs of E' have difference β with an unexpected probability (either high in differential cryptanalysis, or zero in an impossible differential attack).
3. The adversary guesses (all or part of) the subkeys used in E_0 and E_1, performs partial encryption/decryption, and checks whether the prediction of the distinguisher on E' holds. This makes it possible to retrieve the value of the guessed subkeys, provided there is a sufficiently large number of input/output pairs for E.

In our situation, assuming that f is keyed, the adversary cannot perform partial decryption on the outputs of the block cipher f, since she knows only the output difference, but not the actual values. However, if in the differential-type attack, the subcipher E_1 in the subdivision of f into $E_1 \circ E' \circ E_0$ is empty, then the attack can be applied directly to the input/output pairs obtained from internal collisions, allowing to retrieve the internal state $E_k(0)$ (which serves as the subkey used in E_0).

Therefore, any differential-type attack with no rounds after the distinguisher can be leveraged directly into an attack on the MAC construction. The data and time complexities of the attack *given the internal collisions* are the same as the complexities of the original attack on f.

We note that the total complexity of the attack on the MAC is expected to be much higher than the complexity of the attack on the block cipher due to the need to obtain internal collisions by the birthday paradox at the starting point of the attack. Generally, a differential-type attack using 2^l chosen plaintext pairs is transformed to an attack on the MAC requiring at least $2^{(l+l_b)/2}$ messages, where l_b is the block size of f, since only this amount of messages is sufficient for obtaining 2^l internal collisions by the birthday paradox. Moreover, in actual attacks it may be desirable to tweak the differential-type attack before leveraging it to an attack on the MAC, in order to allow using larger data structures in the attack, and thus reduce the amount of data required for obtaining the internal collisions. Such tweak is demonstrated in the attacks on enhanced Pelican presented in the appendix.

The general technique outlined above makes it possible to use known impossible differential attacks on reduced-round AES in order to break even stronger variants of Pelican, in which 4-round keyless AES is replaced by 5-round or 6-round AES with AddRoundKey operations

(possibly with independent subkeys). The data and time complexities of the resulting 5-round attack are less than 2^{80}. We note that the best previously known attack on Pelican, by Yuan et al. [17], which is also an impossible differential attack, has data and time complexities of $2^{85.5}$, and thus, our new attack breaks a stronger variant of Pelican with a lower complexity. Due to space constraints, the detailed description of the attacks is given in Appendix A.

5 Generic Attacks on ALRED Based on Keyless AES

In this section we concentrate on the suggestion of the ALRED designers to use keyless AES as the internal block cipher in the construction. We show that while the choice of AES seems natural, the keyless variant of AES is vulnerable to new types of generic attacks, which are independent of the number of rounds. In particular, we present generic attacks on generalized Pelican and Alpha-MAC in which the internal encryption is a full 10-round AES, or even stronger versions of AES with any number of rounds, with a fixed data complexity of 2^{97} messages and an extremely small memory complexity. While the data complexity of the attacks is higher than that of the generic attacks presented in Section 3, the extremely low memory requirement and its applicability to ALRED variants in which the adversary can control only part of the state makes these attacks more attractive. Furthermore, we show that these attacks apply also to the MAC Marvin proposed in [15, 16], which seems to be immune to the attacks presented in the previous sections, as well as to the attacks presented by Yuan et al. [17].

5.1 The Basic Generic Attack on Pelican-type Constructions

The attack is based on a simple observation on the structure of keyless AES, first presented in [13]. We note that this observation was also applied in [3] to attack the AES-based hash function Lesamnta.

Observation 1 (Le et al.) *Consider a single round of keyless AES, i.e., a sequence of the operations SubBytes, ShiftRows, MixColumns. Denote the states before and after the round by (x, y, z, w) and $(x', y', z', w') = F(x, y, z, w)$, respectively, where each of the variables denotes a column 32-bit vector. Then we have the following:*

$$\text{If } F(x, y, z, w) = (x', y', z', w'), \qquad \text{then} \qquad F(y, z, w, x) = (y', z', w', x').$$

In particular, for the input (x, y, x, y) with two repeated columns, $F(x, y, x, y)$ also has repeated columns of the form (x', y', x', y').

It is clear that while in ordinary AES, the special form (x, y, x, y) is destroyed by the AddRoundKey operation, in keyless AES this special form is preserved through an arbitrarily large number of rounds.

Using the observation, we can mount the following simple attack on generalized Pelican with any number of rounds of keyless AES as the internal block cipher.

1. For each of the 2^{64} possible 128-bit blocks of the form $m_1 = (x, y, 0, 0)$, do the following:
   (a) Ask for the MAC evaluation of a pair of structures S_1 and S_2 of 2^{32} two-block messages each, such that:
      i. In S_1, the first block is fixed to m_1, and in S_2, the first block is fixed to $m_1 \oplus (1, 1, 1, 1)$.[6]

[6] Note that the first blocks in the two structures must differ, since otherwise there will be no collision between the structures as keyless AES is a permutation.

ii. In each of the structures, the second block assumes 2^{32} random values of the form (x', y', x', y').

(b) Insert the tags into a hash table and search for a collision between the two structures. If a collision is found, deduce that the initial state is one of the 2^{64} possible values of the form $(z, w, z, w) \oplus m_1$, for some z, w. If not, discard the guess of m_1.

2. For the single expected value of m_1, that remains, find the values of z, w by exhaustive search.

The idea behind the algorithm is the following: If $E_k(0)$ is of the form $m_1 \oplus (z, w, z, w)$, then $E_k(0) \oplus m_1$ and $E_k(0) \oplus m_1 \oplus (1, 1, 1, 1)$ are both of the special form (z, w, z, w) (i.e., have two repeated columns). Since all the values of the second block m_2 are also of the special form, the corresponding y_2 values also have two repeated columns (independently of the number of rounds of keyless AES). The space of such special values is of size 2^{64}. On the other hand, the number of messages in each structure is 2^{32}, and thus the number of pairs $(m_1, m_2), (m_1^*, m_2^*)$, where $(m_1, m_2) \in S_1$ and $(m_1^*, m_2^*) \in S_2$, is 2^{64}. For all these 2^{64} pairs, the corresponding y_2 values reside in the space of special values which is of size 2^{64}, and hence it is expected that there exists an internal collision $(m_1, m_2), (m_1^*, m_2^*)$, where $(m_1, m_2) \in S_1$ and $(m_1^*, m_2^*) \in S_2$.

On the other hand, if $E_k(0)$ is not of the form $m_1 \oplus (z, w, z, w)$, then $E_k(0) \oplus m_1$ and $E_k(0) \oplus m_1 \oplus (1, 1, 1, 1)$ are not of the special form, and thus, there is no restriction on the y_2 values. Therefore, for the 2^{64} pairs of the form $(m_1, m_2), (m_1^*, m_2^*)$, where $(m_1, m_2) \in S_1$ and $(m_1^*, m_2^*) \in S_2$, the corresponding y_2 values reside in a space of size 2^{128}, and hence the probability that these pairs contain an internal collision is extremely low.

The attack so far recovers 64 bits of $E_k(0)$, and the rest of the bits can be found by exhaustive key search. The data complexity of the attack is 2^{97} two-block messages, the time complexity is 2^{97} MAC evaluations, and the memory requirement is only 2^{32} 128-bit blocks.

5.2 A Memoryless Variant of the Attack

If the adversary is allowed to ask for the MAC evaluation of adaptively chosen messages, the memory requirement of the attack can be further reduced to a few cells of memory, using Floyd's cycle finding algorithm [12].

The attack algorithm is as follows:

1. For each of the 2^{64} possible 128-bit blocks of the form $m_1 = (x', y', 0, 0)$, do the following:
   (a) Ask for the MAC evaluation of a sequence of adaptively chosen two-block messages, defined as follows:
   - $m^0 = (m_1, 0)$.
   - For $k > 0$, if $MAC(m^{k-1}) = (x, y, z, w)$, then $m^k = (m_1 \oplus (x, y, x, y), (z, w, z, w))$.

   (b) Use Floyd's cycle finding algorithm to find a cycle in the sequence of tag values. If the algorithm does not terminate within 2^{32} steps, discard the guess of m_1. If the algorithm terminates, deduce that $E_k(0)$ is of the form $(z', w', z', w') \oplus m_1$ for some z', w'.
2. Only one value of m_1 is expected to survive. For this value, find z', w' by exhaustive search on their 2^{64} possible values.

The idea behind this algorithm is the same as the idea behind the basic algorithm. If $E_k(0)$ is of the form $(z', w', z', w') \oplus m_1$, then all the tag values in the generated sequence lie in a smaller space of size 2^{64}, and thus Floyd's algorithm is expected to terminate in about 2^{32} steps. If $E_k(0)$ is not of the desired form, then the tag values of the sequence lie in the entire space of size 2^{128}, and thus no short cycle is expected.

The data and time complexities of the algorithm are 2^{96}, and the memory requirement is only a few memory cells.

5.3 Application to the Marvin MAC

Another ALRED construction that can be attacked using the generic algorithm described above is the Marvin MAC designed by Simplício et al. [15, 16]. The main difference between Marvin and Pelican is the chaining step, where in Marvin, each block m_i of the message is encrypted independently to

$$F_i(m_i) = AES4(m_i \oplus y_0 \cdot c_i),$$

where y_0 is the value after the initialization, c_i are a series of known constants, $\cdot$ denotes multiplication over $GF(2^n)$, and $AES4$ denotes a 4-round keyless variant of AES. Then the values $F_i(m_i)$ are XORed together to obtain

$$y_\ell = F_1(m_1) \oplus \ldots \oplus F_\ell(m_\ell),$$

and the finalization is essentially the same as in Pelican. See [15] for the exact description of Marvin algorithm.

It seems that in Marvin, the simplest form of an internal collision an adversary can obtain is achieved by altering the values of two message words m_i, m_j and hoping that the differences generated in $F_i(m_i)$ and $F_j(m_j)$ will cancel each other. Such an internal collision provides the adversary with two input/output pairs for 4-round AES, in which the input differences are known, and it is known that the output differences are equal (but it is not known what is this output difference). This information is insufficient neither for the attacks presented in the previous sections nor for the attacks presented by Yuan et al. [17] and thus, it is unclear whether these attacks can be applied to Marvin.

On the other hand, it can be easily seen that the generic attack presented above applies to Marvin (up to a small modification). Indeed, note that if the XOR difference between the two halves of the value y_0 (denoted in [15] by R) is known, then the differences between the halves of $y_0 \cdot c_i$ are known as well. Hence, for each possible guess of the difference between the halves of y_0, the adversary can construct a pool of 2^{32} two-block messages (m_1, m_2) with the appropriate differences between the halves of the two blocks, such that if the guess is correct, the two halves of $m_1 \oplus y_0 \cdot c_1$ and of $m_2 \oplus y_0 \cdot c_2$ are equal. Then, the attack can be completed as in the basic generic attack on Pelican-type structures above. The memoryless variant of the attack can be modified similarly.

Thus, our generic attack on Marvin allows to recover the initial value y_0 (denoted by R in [15]) and thus to perform almost universal forgery of the MAC with data and time complexity of 2^{96} and only a few memory cells. The attacks extend immediately without any change, even if the number of keyless AES rounds is increased.

We note that these attacks do not violate the security claims of Marvin, as these ensure security only as long as the number of queries is below the birthday bound.

5.4 Attacks on Other ALRED Constructions

Unlike the attacks presented in the previous sections, this generic attack performs similarly even if the adversary can control only part of the state bytes. As an example, we show the changes required in order to adapt the basic attack to Alpha-MAC with an arbitrary number of keyless AES rounds as the internal block cipher.

- At the beginning of the attack, instead of considering all the possible values m_1 of the form $(x, y, 0, 0)$, the adversary considers arbitrary two-block values (m_1, m_2). (Note that the size of each block in Alpha-MAC is 32 bits).

– For each value of (m_1, m_2), the adversary considers a single structure of four-block messages, such that the first two blocks in all messages are fixed to (m_1, m_2), and each of the blocks m_3, m_4 assumes all the 2^{16} possible 32-bit values (x_1, x_2, x_3, x_4) in which $x_1 = x_3$ and $x_2 = x_4$.
– If a collision is found, the adversary deduces that the intermediate state after the insertion of m_2 is of the form (z, w, z, w), guesses the values of z and w, rolls the MAC evaluation back to obtain $E_k(0)$, and checks the obtained guess against the MAC evaluations existing in the data set.

The memoryless attack can be modified in a similar way.

6 A Dedicated Attack on Pelican

In this section we consider the original Pelican proposal, which is the specific instantiation of ALRED described in [10]. We show that using a simple differential-type attack, the adversary can recover the initial state $E_k(0)$ with data and time complexities of 2^{65} – essentially the complexity required for detecting a single internal collision by the birthday paradox, which is likely to be the smallest possible complexity of any generic attack on a MAC construction.

The attack is based on a simple differential property of S-boxes:

Observation 2 *Consider an S-box S, and pairs $(\alpha \neq 0, \beta)$ of input/output differences for the S-box. For each such pair, let $S_{\alpha,\beta}$ denote the set of pairs (x, y) such that $x \oplus y = \alpha$ and $S(x) \oplus S(y) = \beta$. Then the expected cardinality of the set $S_{\alpha,\beta}$ is 1, and the elements of all sets $S_{\alpha,\beta}$ can be enumerated efficiently using the pre-computed difference distribution table of the S-box. For an n-by-n bit S-box, the time required to construct the table is 2^{2n} evaluations of the S-box, and the memory required to store the table is about 2^{2n+1} n-bit words.*

In the attack, the adversary searches for a two-block internal collision $(m_1, m_2), (m_1^*, m_2^*)$, in which the input and output differences to the keyless 4-round AES (which are $m_1 \oplus m_1^*$ and $m_2 \oplus m_2^*$, respectively) have a "nice form". Specifically, the adversary requires that $m_1 \oplus m_1^*$ is zero in bytes $1, 2, 6, 7, 8, 11, 12, 13$, and that $m_2 \oplus m_2^*$ has a fixed non-zero difference in byte 0 and zero difference in the remaining bytes.[7] (Such differences can be easily obtained by an appropriate choice of structures, as we show in the attack algorithm below.)

Figure 5 presents the propagation of differences through the 4-round AES encryption, given the chosen input and output differences (denoted by ΔX_1 and ΔX_5, respectively). As shown in the figure, the input difference to round 2 (denoted by ΔX_2) is non-zero only in the first two columns, and thus there are at most 2^{64} possible differences at the state before the SubBytes operation of round 3 (denoted by ΔX_3). On the other hand, the input difference to round 4 (denoted by ΔX_4) is non-zero only in four bytes, and thus there are at most 2^{32} possible differences in the state after the SubBytes operation of round 3 (denoted by $\Delta X_{3(SB)}$). For each of the $2^{64} \cdot 2^{32} = 2^{96}$ combinations of differences, the adversary can apply Observation 2 to the SubBytes operation of round 3 (which consists of 16 8-bit S-boxes applied in parallel), and obtain a single suggestion on average for the actual state at the input of round 3. Then she can run the MAC evaluation backwards to obtain a single suggestion for the initial state $E_k(0)$.

In a naive implementation, the time complexity of the attack is about 2^{96} MAC evaluations. However, the time complexity can be reduced to 2^{65} MAC evaluations by using a few pre-computed tables, as described in the next paragraph.

Consider a fragment of the 4-round AES, consisting of round 3 and the SubBytes operation of round 4, i.e., a sequence of SubBytes, ShiftRows, MixColumns, SubBytes operations. In this

[7] The numbering of the bytes in an AES state which we use here is defined in the appendix.

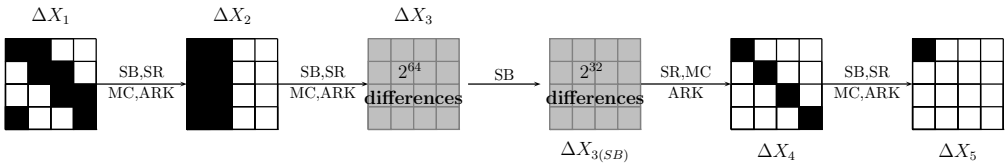

Fig. 5. A difference pattern in 4-round AES. White cells denote zero difference, and black cells denote arbitrary difference.

fragment, the encryption can be viewed as four 32-bit to 32-bit encryptions acting in parallel. (The first of these encryptions transforms bytes $0, 5, 10, 15$ at the input to round 3 to bytes $0, 1, 2, 3$ after the SubBytes of round 4, and the other encryptions are constructed similarly.) The adversary can treat each such encryption as a super S-box mapping 32 bits to 32 bits, and apply to it Observation 2. Specifically, she can construct in advance the differential distribution tables of these super S-boxes, and then in the on-line phase of the attack, deduce the actual values in the input to round 3 from the input and output differences of the super S-boxes. Note that in our attack, the difference after the SubBytes operation of round 4 is fixed and known in advance to the adversary. (This difference is $MixColumns^{-1}(m_2 \oplus m_2^*)$.) Thus, the adversary has to keep in the pre-computed tables only the entries which correspond to that fixed output difference. Therefore, the pre-computed tables require less than 2^{40} bits (or 128 gigabytes) of memory. The time required to construct the tables is 2^{66} operations, but it can be easily reduced to 2^{42} operations using the specific form of the fixed output difference.

We are ready now to present the attack algorithm.

1. **Detecting an internal collision of a specific form:**
   (a) Ask for the MAC evaluation of two structures S_1 and S_2 of 2^{64} two-block messages each, such that:
      – The first blocks in all messages in both structures have a fixed value in bytes $1, 2, 6, 7, 8, 11, 12, 13$ (the same value for all messages in both structures), and assume all 2^{64} possible values in the remaining 8 bytes.
      – The second blocks in all messages in S_1 are fixed to a constant value m_2 and the second blocks of all messages in S_2 are fixed to a constant value m_2^*, where the difference $m_2 \oplus m_2^*$ is non-zero only in byte 0.
   (b) Insert the output tags into a hash table and search for an internal collision $(m_1, m_2), (m_1^*, m_2^*)$, such that $(m_1, m_2) \in S_1$ and $(m_1^*, m_2^*) \in S_2$.
2. **Retrieving the internal state:**
   (a) Given an internal collision, for each of the 2^{64} possible input differences to round 3, access the pre-computed tables and find the corresponding input values to round 3. Decrypt these values through rounds 2 and 1, and check whether the difference in the input to round 1 is $m_1 \oplus m_1^*$. If no, discard the guess of the input difference to round 3. Since this is a 64-bit filtering, it is expected that only a few guesses of the difference remain.
   (b) For the remaining guesses of the input difference of round 3, use the input to round 1 in the first of the two colliding blocks (which equals $E_k(0) \oplus m_1$) to obtain a unique suggestion for $E_k(0)$.

The data complexity of the attack is 2^{65} chosen two-block messages, and its time complexity is dominated by the evaluation of the MAC on the two structures of 2^{64} messages each. The memory requirement of the attack is 2^{64} 128-bit blocks.

We note that using a similar strategy, one can attack generalized variants of Pelican with up to seven keyless AES rounds as the internal block cipher. However, since the complexity of these attacks is higher than that of the generic attack presented in Section 3, we do not present them here.

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. Eli Biham and Nathan Keller, *Cryptanalysis of Reduced Variants of Rijndael*, unpublished manuscript, 1999.
3. Charles Bouillaguet, Orr Dunkelman, Gaëtan Leurent, and Pierre-Alain Fouque, *Another Look at Complementation Properties*, proceedings of Fast Software Encryption 2010, Lecture Notes in Computer Science 6147, pp. 347–364, Springer, 2010.
4. Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque, *Automatic Search of Attacks on Round-Reduced AES and Applications*, accepted to CRYPTO 2011. To appear in Lecture Notes in Computer Science.
5. Don Coppersmith, Lars R. Knudsen, and Chris J. Mitchell, *Key Recovery and Forgery Attacks on the MacDES MAC Algorithm*, Advances in Cryptography, proceedings of CRYPTO 2000, Lecture Notes in Computer Science 1880, pp. 184–196, springer, 2000.
6. Joan Daemen, *Limitations of the Even-Mansour Construction*, proceedings of Asiacrypt 1991, Lecture Notes in Computer Science 739, pp. 495–498, Springer, 1991.
7. Joan Daemen and Vincent Rijmen, *The design of Rijndael: AES — the Advanced Encryption Standard*, Springer, 2002.
8. 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.
9. Joan Daemen and Vincent Rijmen, *The Pelican MAC Function*, IACR ePrint report 2005/088.
10. Joan Daemen and Vincent Rijmen, *Refinements of the Alred construction and MAC security claims*, IET Information Security **4(3)**, pp. 149–157, 2010.
11. Shimon Even and Yishai Mansour, *A Construction of a Pseudorandom Cipher from Single Pseudorandom Permutation*, Journal of Cryptology **10(3)**, pp. 151–162, 1997.
12. Donald Knuth, *The Art of Computer Programming*, 2nd Edition, Vol. 2, pp. 7, Addison-Wesley, 1981.
13. Tri Van Le, Rüdiger Sparr, Ralph Wernsdorf, and Yvo Desmedt, *Complementation-like and Cyclic Properties of AES Round Functions*, proceedings of the 4th AES conference, Lecture Notes in Computer Science 3373, pp. 128–141, Springer, 2004.
14. 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.
15. Marcos A. Simplício Jr., Pedro d'Aquino F. F. S. Barbuda, Paulo S. L. M. Barreto, Tereza Cristina M. B. Carvalho, and Cintia B. Margi, *The MARVIN message authentication code and the LETTER-SOUP authenticated encryption scheme*, Security and Communication Networks **2(2)**, pp. 165–180, 2009.
16. Marcos A. Simplício Jr., Paulo S. L. M. Barreto, and Tereza Cristina M. B. Carvalho, *Revisiting the Security of the Alred Design*, proceedings of Information Security Conference (ISC) 2010, Lecture Notes in Computer Science 6531, pp. 69–83, Springer, 2011.
17. Zheng Yuan, Wei Wang, Keting Jia, Guangwu Xu, and Xiaoyun Wang, *New Birthday Attacks on Some MACs Based on Block Ciphers*, Advances in Cryptology, proceedings of CRYPTO 2009, Lecture Notes in Computer Science 5677, pp. 209–230, Springer, 2009.

A Using Impossible Differential Attacks on Reduced-round AES to Attack 5-round and 6-round Pelican

As an example of the general technique outlined in Section 4, we show how to use known impossible differential attacks on reduced-round AES in order to break even stronger variants of Pelican, in which 4-round keyless AES is replaced by 5-round or 6-round AES with AddRoundKey operations (possibly with independent subkeys). The data and time complexities of the 5-round attack are less than 2^{80}. We note that the best previously known attack on Pelican, by Yuan et al. [17], which is also an impossible differential attack, has data and time complexities of $2^{85.5}$, and thus, our new attack breaks a stronger variant of Pelican with a lower complexity. A variant of our 5-round attack can be used to break the original 4-round Pelican, with the same data and time complexities of 2^{80}. We omit its detailed description, as it is inferior to another dedicated attack on the original Pelican presented in Section 6, which is based on a different idea.

In the description of the attacks on AES (both in this appendix and in Section 6), we use the following standard notations. Each state during the AES encryption is represented by a 4-by-4 byte matrix, and the entries of the matrix are numbered by $0, 1, \ldots, 15$, such that the j-th entry in the i-th row (for $0 \leq i, j \leq 3$) is numbered by $i + 4j$. The four operations applied in each round – SubBytes, ShiftRows, MixColumns, and AddRoundKey – are denoted by SB, SR, MC, and ARK, respectively. The rounds are numbered $1, 2, \ldots, 10$. The subkey used in the r-th round is denoted by k_r, and the initial whitening key is denoted by k_0.

A.1 Impossible Differential Attacks on 5-round and 6-round AES

Impossible differential attacks on reduced-round AES with 128-bit keys were extensively studied in the last decade, and several attacks on 5,6, and 7 rounds were presented (see [14] for a summary of the attacks).

The simplest attack, by Biham and Keller [2] on 5-round AES, does not use any rounds after the distinguisher, and thus can be applied directly to a 5-round Pelican using the generic technique described in Section 4.1. The most advanced attacks on 7-round AES (e.g., by Bahrak and Aref [1]) cannot be leveraged directly since they analyze rounds on both sides of the distinguisher, but a reduced 6-round variant of the attacks, which drops the round after the distinguisher, can be leveraged directly.

However, in order to enable the adversary to use larger structures and thus reduce the data complexity of the attacks on the MAC, we present slightly different impossible differential attacks on 5-round and 6-round AES, which we will then apply to 5-round and 6-round Pelican.

Our attacks are based on a 3-round impossible differential of AES which is similar to (but not identical with) the impossible differentials used in all previous attacks. The differential, depicted in Figure 6, asserts that if the difference in the input to round i of (either keyed or un-keyed) AES is zero in bytes $0, 5, 10, 15$ (regardless of the difference in the rest of the bytes), then the difference in the input to round $i + 3$ cannot be non-zero only in byte 0.

Indeed, consider a pair (P, P^*) of inputs to the i-th round of AES, such that the difference $P \oplus P^*$ is zero in bytes $0, 5, 10, 15$. By the structure of AES, the corresponding intermediate difference in the input to round $i + 1$ is zero everywhere in the first column.

On the other hand, if the difference in the input to round $i + 3$ is non-zero in byte 0 and zero in all other bytes, then the input difference to round $i + 2$ is non-zero in bytes $0, 5, 10, 15$ and zero in the other bytes. Consequently, the difference in the input to round $i + 1$ is non-zero in all bytes, contradicting the forward direction.

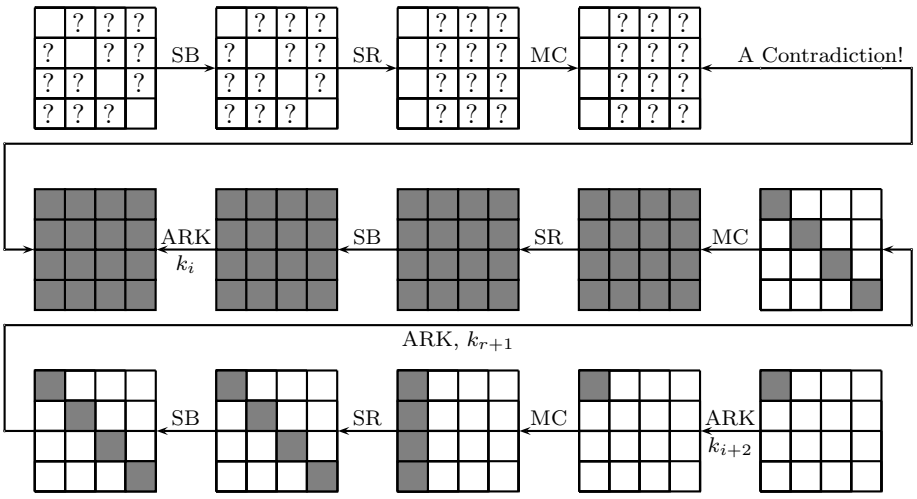

Fig. 6. An Impossible Differential of 3-round AES. White cells denote zero difference, gray cells denote non-zero difference, and "?" denotes arbitrary difference.

Similar impossible differentials hold if the zero input difference is placed at any of the four sets of bytes: $\{0, 5, 10, 15\}$, $\{1, 6, 11, 12\}$, $\{2, 7, 8, 13\}$, and $\{3, 4, 9, 14\}$, and if the output is replaced by any single-byte difference.

A standard way to use this impossible differential to break 5-round AES, is to perform the following three-step attack procedure (see, e.g., [2] for a similar attack):

1. Ask for the encryption of a structure of plaintexts in which the values of several bytes are constant, and the other bytes assume different values (these bytes are called "active").
2. Consider only those ciphertext pairs within the structure in which the difference in the input to round 5 is in a single byte (using a hash table, these pairs can be found instantly).
3. Guess the part of the first subkey which are used in the active bytes, and for each guess and for each pair, check whether the difference in the input to round 2 is zero in one of the four sets of bytes: $\{0, 5, 10, 15\}$, $\{1, 6, 11, 12\}$, $\{2, 7, 8, 13\}$, and $\{3, 4, 9, 14\}$. If so, discard the key guess.

In order to extend the attack to 6-round AES, the adversary can add one round in the beginning, guess relevant subkey material in the first two subkeys, and check whether the input to round 3 is zero in one of the four sets: $\{0, 5, 10, 15\}$, $\{1, 6, 11, 12\}$, $\{2, 7, 8, 13\}$, and $\{3, 4, 9, 14\}$. We omit the details of the attacks here, since they are essentially the same as the attacks on Pelican we present below.

A.2 Leveraging the attacks to 5-round and 6-round Pelican

We start with an attack on generalized Pelican, in which the underlying reduced block cipher is enhanced from 4-round keyless AES to 5-round AES including the AddRoundKey operations, with independent subkeys to each one of the 5 rounds.

Attack on 5-round Pelican The algorithm of the attack on the generalized 5-round Pelican is as follows:

1. **Detecting internal collisions:**
   (a) Ask for the MAC evaluation of two structures S_1 and S_2 of 2^{78} two-block messages each, such that:
      – The first blocks in all messages in both structures have a fixed value in bytes $1, 6, 11, 12, 13, 14$ (i.e., the same value for all messages in both structures), and assume 2^{79} different values in the remaining 10 bytes in the two structures.
      – The second blocks in all messages in S_1 are fixed to a constant value m_2 and the second blocks of all messages in S_2 are fixed to a constant value m_2^*, such that $MixColumns^{-1}(m_2 \oplus m_2^*)$ is non-zero only in byte 0.
   (b) Insert the tags into a hash table and search for internal collisions $(m_1, m_2), (m_1^*, m_2^*)$, such that $(m_1, m_2) \in S_1$ and $(m_1^*, m_2^*) \in S_2$.
2. **Attacking the internal reduced block cipher:** For each internal collision and for each guess of bytes $0, 2, 3, 4, 5, 7, 8, 9, 10, 15$ of the internal state $E_k(0)$,[8] partially encrypt the pair $(m_1 \oplus E_k(0), m_1^* \oplus E_k(0))$ through the first round of AES, and check whether the input difference to round 2 is zero in one of the four sets of bytes $\{0, 5, 10, 15\}$, $\{1, 6, 11, 12\}$, $\{2, 7, 8, 13\}$, and $\{3, 4, 9, 14\}$. If this is the case, discard the guess of $E_k(0)$.

Analysis of the attack The time complexity of the first phase of the attack (i.e., detecting internal collisions) is 2^{79} MAC evaluations, and its memory complexity is 2^{78} 128-bit blocks. The data is expected to contain $2^{78} \cdot 2^{78} \cdot 2^{-128} = 2^{28}$ internal collisions of the form $(m_1, m_2), (m_1^*, m_2^*)$, where $(m_1, m_2) \in S_1$ and $(m_1^*, m_2^*) \in S_2$.

In the second phase of the attack, for each internal collision, the adversary guesses 80 bits of $E_k(0)$ and checks whether the difference between the intermediate states in the input to round 2 of AES is zero in one of the four sets of bytes $\{0, 5, 10, 15\}$, $\{1, 6, 11, 12\}$, $\{2, 7, 8, 13\}$, and $\{3, 4, 9, 14\}$. Since the difference in the entire Column 3 in the input to round 2 is zero (independently of the guessed subkey values), the probability that a subkey guess is discarded is $4 \cdot 2^{-24} = 2^{-22}$. Hence, the expected number of remaining subkey guesses is $2^{80} \cdot (1 - 2^{-22})^{2^{28}} \approx 2^{80} \cdot e^{-64} \approx 2^{-12}$, i.e., only the correct guess is expected to remain.

The time complexity of a naive application of this step is $2^{80} \cdot 2^{28} = 2^{108}$ partial encryptions. However, this complexity can be reduced significantly by noting that for each internal collision, the adversary can perform the partial encryption in each of the three active columns independently. Specifically, the adversary can perform a two-step procedure:

1. For each of the 2^{28} internal collisions, the adversary computes all the values of bytes $3, 4, 9$ of $E_k(0)$ that lead to zero difference in byte 5 in the input to round 2, and stores them in a table.
2. For each guess of bytes $0, 2, 5, 7, 8, 10, 15$ of $E_k(0)$:
   (a) The adversary goes over all the internal collisions and checks whether the difference in bytes 0 and 10 in the input to round 2 is zero. Only $2^{28} \cdot 2^{-16} = 2^{12}$ internal collisions are expected to pass this filtering.

[8] Note that if f contains an AddRoundKey operation at the beginning, with a whitening key k_0, then the adversary should guess an equivalent key $E_k(0) \oplus k_0$ instead of $E_k(0)$. In this case, the attack makes it possible to retrieve only the value $E_k(0) \oplus k_0$, rather than $E_k(0)$. However, this value is sufficient for mounting the forgery attacks on the MAC described in the introduction. For the sake of simplicity, we assume in the sequel that f does not contain a whitening key.

(b) The adversary considers a list of all possible values of bytes $3, 4, 9$ of $E_k(0)$, and for each remaining internal collision, she discards the values of $E_k(0)$ that lead to zero difference in byte 5 in the input to round 2 (using the table computed in the first step). If all the possible values of bytes $3, 4, 9$ of $E_k(0)$ are discarded, the adversary discards the guess of bytes $0, 2, 5, 7, 8, 10, 15$ of $E_k(0)$ made in the beginning of the step.

This procedure makes it possible to discard all guesses of $E_k(0)$ which lead to zero difference in bytes $\{0, 5, 10, 15\}$ in the input to round 2, and by repeating it four times, the adversary can discard also key guesses which lead to zero difference in one of the sets $\{1, 6, 11, 12\}$, $\{2, 7, 8, 13\}$, and $\{3, 4, 9, 14\}$. The time complexity of this procedure is $2^{28} \cdot 2^{58} = 2^{86}$ simple operations, which are dominated by the MAC evaluations performed in the first phase of the attack (since evaluating the full MAC is much slower than a single memory access). The overall memory complexity of this step is $2^{28} \cdot 2^{16} = 2^{44}$ 128-bit blocks, but *random accesses* are made only to much smaller lists of size at most 2^{28} 128-bit blocks.

After the 80 bits of $E_k(0)$ are found, the adversary repeats the procedure with another set of active bytes in the first round (and another set of chosen messages) to obtain the rest of $E_k(0)$. The subkeys used in the AddRoundKey operations of AES can be found in a similar way by attacking a 4-round (or even smaller) variant of AES using known cryptanalytic techniques.

The overall data complexity of the attack on 5-round Pelican is 2^{80} chosen two-block messages, the time complexity is 2^{80} MAC evaluations, and the memory complexity is 2^{78} 128-bit blocks. Note that this is better than the best previously known attack due to Yuan et al. [17], which attacks 4-round Pelican in $2^{85.5}$ data and time.

Attack on 6-round Pelican A similar attack can be applied to a generalized variant of Pelican where the internal block cipher is 6-round AES, even when AddRoundKey operations with independent subkeys are used in each of these rounds.

In the first phase of the attack, the adversary considers pairs of structures S_1, S_2 of 2^{64} two-block messages each, such that in the first block, 8 bytes which form two shifted columns (e.g., bytes $0, 3, 4, 5, 9, 10, 14, 15$) are fixed to the same value in both structures and the remaining 8 bytes assume all 2^{64} possible values. The second blocks are the same as in the 5-round attack, and for each pair of structures, we expect to find a single collision which is detected in the same way as in the 5-round attack.

In the second phase of the attack, the adversary guesses the eight bytes of $E_k(0)$ that correspond to the active bytes in the data (in our example above, these are bytes $1, 2, 6, 7, 8, 11, 12, 13$), and for each internal collision, she partially encrypts the pair $(m_1 \oplus E_k(0), m_1^* \oplus E_k(0))$ through the first round of AES. The analysis of the internal collision is continued only if the intermediate difference before the MixColumns operation in round 2 of AES is non-zero only in the first three columns. In such cases, the difference is non-zero only in two bytes in each of these columns (in our example these are bytes $2, 3, 5, 6, 8, 9$). The adversary guesses the value of these six bytes in the subkey used in the AddRoundKey operation in round 1 of AES, partially encrypts the pair of messages through round 2, and checks whether the difference in the input to round 3 is zero in one of the four sets of bytes $\{0, 5, 10, 15\}$, $\{1, 6, 11, 12\}$, $\{2, 7, 8, 13\}$, and $\{3, 4, 9, 14\}$. If yes, the 14-byte subkey guess is discarded. For a single examined internal collision and a fixed subkey guess, the probability that the guess is discarded is $4 \cdot 2^{-16} \cdot 2^{-24} = 2^{-38}$. Therefore, in order to discard most of the subkey suggestions, the adversary can examine 2^{44} internal collisions, so that the expected number of remaining subkey suggestions is $2^{112} \cdot e^{-64} \approx 2^{19.7}$. The attack is completed by examining another data set (with other active bytes) and comparing the subkey suggestions in the overlapping subkey bytes.

Since each pair of structures is expected to contain a single internal collision, the data complexity of the attack is $2 \cdot 2^{44} \cdot 2^{65} = 2^{110}$ chosen messages. The time complexity is dominated

by the MAC evaluations of these chosen messages (since the second part of the attack can be performed column-wise, like in the 5-round attack), and the memory requirement is 2^{64} 128-bit blocks. (There is no need to store a list of the 2^{112} possible subkey guesses, since the guesses of the 8 active bytes in $E_k(0)$ can be checked sequentially, and for each of them, the adversary can keep a list of the values of the 6 active bytes in the second subkey that have to be discarded.)

It seems that it will be hard to extend this type of attack to 7-round Pelican, as such attack would be roughly equivalent to an impossible differential attack on 8-round AES-128, which seems out of reach with current techniques.

A.3 Applicability of the Leveraging Technique to Other ALRED Constructions

As in the case of the generic attack presented in Section 3, the ability to leverage a differential-type attack on the reduced block cipher into an attack on the MAC depends on the structure of the injection input. In the leveraging attack the adversary has two additional difficulties:

- First, the attack can be applied only if the active bytes in the input and output differences required in the attack on the block cipher are included among the bytes which can be controlled by the adversary.
- Second, while in the generic attack, the adversary can obtain multiple structures by appending blocks at the beginning of the message, in the leveraging attack this is impossible. Indeed, in the equivalent structure of the MAC used in the leveraging attack, the intermediate state before the insertion of the message m_1 is considered as a subkey of the block cipher, which the adversary wants to retrieve. If another block m_0 is appended before m_1 and in different pairs of structures the values of m_0 are different (as is the case of the generic attack on Alpha-MAC-type constructions), then for different internal collisions, the "subkey" that the adversary tries to retrieve will be different! Of course, this results in failure of the attack, since the adversary needs several input/output pairs of the block cipher encrypted under *the same* key in order to retrieve the key.

As a result, it seems that the leveraging attack is possible only if the number of bits the adversary can control is at least $(l_b + l)/2$, where l_b is the block size of f and 2^l is the number of pairs required by the differential-type attack on f. In particular, this attack is not applicable to generalized Alpha-MAC, in which the original 1-round AES is replaced by a stronger (and keyed) reduced block cipher.[9]

[9] We note that on the actual version of Alpha-MAC, a variant of the attack can be applied, since the adversary can append a block m_0 at the beginning of the message, and overcome the difference between the internal states before the insertion of m_1 by using the weak diffusion of 1-round AES. However, since the resulting attack is slower than the attack on Alpha-MAC presented in [17], we omit it.