Practical Cryptanalysis of PAES

Jérémy Jean[1], Ivica Nikolić[1](✉), Yu Sasaki[2], and Lei Wang[1]

[1] Nanyang Technological University, Singapore, Singapore
{JJean,INikolic,Wang.Lei}@ntu.edu.sg
[2] NTT Secure Platform Laboratories, Tokyo, Japan
sasaki.yu@lab.ntt.co.jp

Abstract. We present two practical attacks on the CAESAR candidate PAES. The first attack is a universal forgery for any plaintext with at least 240 bytes. It works for the nonce-repeating variant of PAES and in a nutshell it is a state recovery based on solving differential equations for the S-box leaked throught the ciphertext that arise when the plaintext has a certain difference. We show that to produce the forgery based on this method the attacker needs only 2^{11} time and data. The second attack is a distinguisher for 2^{64} out of 2^{128} keys that requires negligible complexity and only one pair of known plaintext-ciphertext. The attack is based on the lack of constants in the initialization of the PAES which allows to exploit the symmetric properties of the keyless AES round. Both of our attacks contradict the security goals of PAES.

Keywords: PAES · Universal forgery · Distinguisher · Symmetric property · Authenticated encryption

1 Introduction

The CAESAR competition [2] (Competition for Authenticated Encryption: Security, Applicability, and Robustness) has started in March 2014, and its goal is to improve the understanding of the crypto community in the area of authenticated ciphers through a public competition for submitting authenticated encryption schemes that offer advantages over the widely used AES-GCM [8]. In total, 58 ciphers were submitted to the open call, and in the following three years, through security analysis and investigation of the implementations advantages, it is expected that among these ciphers, a few to be selected in a portfolio of recommended authenticated schemes that are suitable for widespread adoption.

A number of the proposed CAESAR candidates (as well as the benchmark AES-GCM) are based on the current encryption standard: the AES family of block ciphers. The reason for this is twofold. First, the AES has undergone an extensive analysis and is assumed that its security is well understood (or at least better understood compared to all of the remaining unbroken ciphers). Second,

J. Jean, I. Nikolić and L. Wang are supported by the Singapore National Research Foundation Fellowship 2012 NRF-NRFF2012-06.

A. Joux and A. Youssef (Eds.): SAC 2014, LNCS 8781, pp. 228–242, 2014.
DOI: 10.1007/978-3-319-13051-4_14

AES offers a large software implementation advantage on the latest processor through the so-called AES-NI instruction set, i.e., modern processors have dedicated instructions that allow to reduce the execution time of the AES cipher calls.

In general, the CAESAR candidates based on the AES use the block cipher in two ways: either as a whole (or a variant consisting of at least a certain number of rounds), or only its round function. The first type of candidates (OCB [6], AES-COPA [1], etc., and AES-GCM) are constructions that require calls to the full 10-round AES-128 (or at least 4-round variants with independent round keys, e.g., SHELL [11]). Usually, they are provable modes based on security reduction to the security of AES, and thus benefit from the current state-of-the-art cryptanalysis of AES-128 [4,5]. The second type uses only the AES round function and has no strict security proof, i.e., the mode is not provably secure, however, the resistance against common attacks is provided through ad-hoc techniques. Such candidates (see AEGIS [12], PAES [13], Tiaoxin-346 [9]) benefit from the good security properties and the software performance of the AES round function. They tend to use less than 10 AES round calls per message blocks, and as such are extremely fast.

Our Contributions. We provide a cryptanalysis of the CAESAR candidate PAES [13] and show two attacks that contradict the security claims given by the designers. Common for both of the attacks are the low complexity requirements and misuse of the AES round function in PAES.

The first attack targets the nonce-repeating mode of PAES (called PAES-8) and is a universal forgery attack of any plaintext with at least 240 bytes. It requires 2^{11} time and data complexity to fully recover the internal state and produce forgery. To launch the attack, we use a special differential trail that can take two different paths. By analyzing the ciphertext difference, the path is uniquely determined and allows state recovery based on the differential property of the AES S-Box. Our attack shows that a mere differential analysis (often given by providing the best differential characteristic of a construction) is insufficient for proving security in the nonce-repeating mode, even when the candidates guarantees multiple applications of AES round function.

The second attack comes in a form of a distinguisher for a class of 2^{64} weak keys among the total 2^{128} keys of PAES. We show that if the attacker can control the nonce, then a single pair of known plaintext and corresponding ciphertext is sufficient to distinguish PAES from an ideal authenticated encryption scheme. The attack relies on the initialization phase of PAES that does not use constants, while the AES round function preserves certain symmetric properties when constants are absent. The results of this paper are summarized in Table 1.

Organization of the Paper. We recall the design details of the PAES submissions in Sect. 2 and present the universal forgery attack on PAES-8 in Sect. 3. Then, in Sect. 4 we introduce the distinguisher for PAES in the context of weak keys, and we conclude the paper in Sect. 5.

Table 1. Attacks on PAES.

Design	Supported nonce modes	Attack	Attack mode	Key class size (out of 2^{128})	Time complexity
PAES-4	respecting	distinguisher	respecting	2^{64}	1
PAES-8	respecting+repeating	universal forgery	repeating	2^{128}	2^{11}
PAES-8	respecting+repeating	distinguisher	respecting+repeating	2^{64}	1

2 Description of PAES

The family of authenticated encryption (AE) algorithms PAES has been submitted to the ongoing CAESAR competition and consists of two concrete proposals: PAES-4 and PAES-8. As the name suggests, they both use the AES design strategy [3], and take as input a variable-length plaintext, a 128-bit key, a 128-bit nonce and produce a variable-length ciphertext and a 128-bit authentication tag. The difference between PAES-4 and PAES-8 lies in the size of the internal state, which amounts to four 128-bit blocks for the former, and eight 128-bit blocks for the latter. A functional difference between these two variants is in the mode: PAES-4 has security claims only in the nonce-respecting mode, while PAES-8 in both, the nonce-respecting and nonce-repeating modes.

To simplify the presentation, we describe only PAES-8 in the sequel, and only as authenticated encryption. The design resembles a stream cipher: it has an initialization (where the key and the nonce are loaded into the state), then it processes the input message and produces the ciphertext, and finally in the finalization it produces the tag. The internal state S has eight words $S_1, S_2, \ldots, S_8$, each of 128 bits, i.e., $|S_i| = 128, i = 1, \ldots, 8$. The state update function $StateUpdate(S, M)$ is the round transformation and uses eight keyless[1] AES-round calls (denoted further as $\mathtt{AES}_0$) to update the state as depicted in Fig. 1.

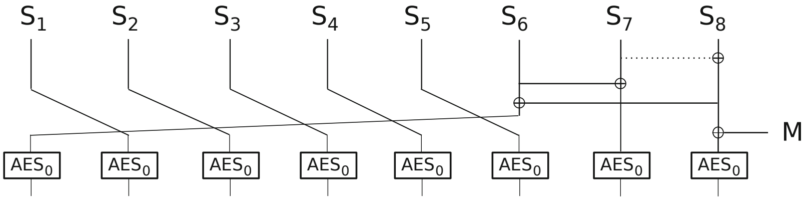

Fig. 1. The round function $StateUpdate(S, M)$. During the processing of the plaintext, the XOR from S_7 to S_8 is absent.

[1] We emphasize that all the AES calls are keyless, that is, composed of SubBytes, ShiftRows and MixColumns (but no AddRoundKey).

Initialization. The 128-bit master key K and the nonce N are loaded into the eight words of the state, the state goes through 10 rounds and at the end the key is XORed to all eight words of the state:

$$\begin{aligned}
&S_1 = K \oplus N, && S_5 = L^4(K) \oplus L^7(N)\\
&S_2 = L(K) \oplus L^3(N), && S_6 = L^5(K) \oplus L^3(N)\\
&S_3 = L^2(K) \oplus L(N), && S_7 = L^6(K) \oplus L^5(N)\\
&S_4 = L^3(K) \oplus L^2(N), && S_8 = L^7(K) \oplus L^6(N)
\end{aligned}$$

```
$for\ i = 1\ to\ 10$
    $S = StateUpdate(State, 0)$
$for\ i = 1\ to\ 8$
    $S_i = S_i \oplus K$
```

where L is the linear transformation that operates on the four 32-bit columns a, b, c, d of a 128-bit word $a||b||c||d$, and is defined as $L(a, b, c, d) = (b, c, d \oplus a, a)$. We denote L^i the i-th functional power of the transformation L, e.g., $L^2 = L \circ L$.

Processing the Plaintext. In one round, from 16-byte plaintext P_i, 16-byte ciphertext C_i is obtained with one call to the $StateUpdate$ function (see Fig. 2):

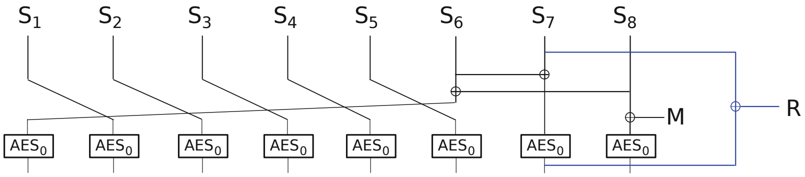

Fig. 2. One round of the encryption.

$$\begin{aligned}
&tmp = S_7\\
&StateUpdate(S, P_i)\\
&R_i = tmp \oplus S_7\\
&C_i = P_i \oplus R_i
\end{aligned}$$

Finalization and the Tag Production. Let $|M|$ be the 128-bit encoding of the message length. Then, the tag T is produced after 10 rounds of the $StateUpdate$ function where the message input is set to $|M|$:

```
$for\ i = 1\ to\ 10$
    $StateUpdate(S, |M|)$
$T = S_7 \oplus S_8$
```

Claimed Security of PAES. The claimed security of PAES is given in Table 2. We emphasize in particular that 128-bit security is claimed for the integrity of PAES in the nonce-repeating mode.

Table 2. Bits of security goals of PAES [13, Table 3.1].

Goal	**Nonce-respecting**	**Nonce-repeating**	
	PAES-4/PAES-8	PAES-4	PAES-8
Confidentiality for the plaintext	128	-	-
Integrity for the plaintext	128	-	128
Integrity for the associated data	128	-	128
Integrity for the public message number	128	-	128

3 Practical Universal Forgery Attack Against PAES-8

In this section, we show a universal forgery attack for PAES-8 in the nonce-repeating mode. The attack works for any plaintext with length of at least 240 bytes, and requires only a small time and data complexity. The steps of the attack can be summarized as follows:

1. Inject differences in two consecutive plaintext blocks such that they cancel in S_8 with a high probability.
2. The ciphertext difference after eight rounds will reveal if the cancellation in S_8 occurred and if so, it will leak information about the state bits.
3. Once the state is recovered, the tag is produced by going through the remaining of the transformations of the (now) public construction.

3.1 Differential Trail and Detection of Difference Cancellation

The differential trail used in the attack is given in Fig. 3. We inject difference $\Delta\alpha$ in the plaintext P_0, and try to cancel it with another difference $\Delta\beta$ in the plaintext P_1. Interestingly, this type of trail has been discussed by the designers of PAES (see [13, Figure 4.3]), however, they focused on the standard case of propagating the difference through eight rounds and tried to predict it. On the other hand, we use a different approach: our goal is not to predict the difference after eight rounds, but only to detect if the initial differences in $\Delta\alpha$ and $\Delta\beta$ have canceled. In Fig. 3, the trail can take two paths:

1. The differences $\Delta\alpha$ and $\Delta\beta$ cancel, thus only the words with bold lines are active,
2. The differences $\Delta\alpha$ and $\Delta\beta$ do not cancel and there are additional active words depicted with red lines.

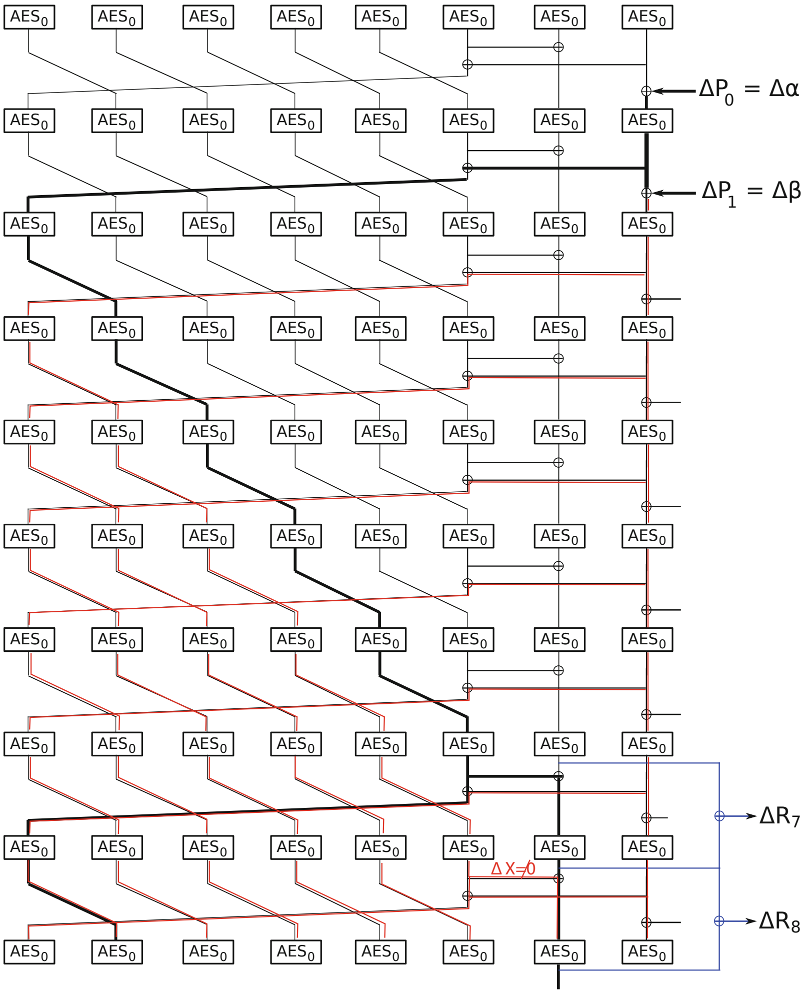

Fig. 3. Differential trail used in the attack. The black bold lines denote active state words. The red lines denote active words when $\Delta\alpha$ and $\Delta\beta$ do not cancel in S_8.

We further show how to choose optimal $\Delta\alpha$ and $\Delta\beta$ and how to detect the cancellation.

Choosing Plaintext Differences $\boldsymbol{\Delta\alpha}$ and $\boldsymbol{\Delta\beta}$. For an arbitrary difference $\Delta\alpha$ in the plaintext P_0, the difference $\Delta\beta$ in the plaintext P_1 should be chosen such that it will cancel $\Delta\alpha$ and thus will avoid activating the state S_8. Therefore, $\Delta\alpha$ and $\Delta\beta$ are chosen so that the cancellation can occur with a high probability – this happens when $\Delta\alpha$ has only one active byte. Let α and β be the input and output difference transition of the S-Box, i.e., α changes to β with a probability 2^{-6}. Then, $\Delta\alpha$ and $\Delta\beta$ are defined as

$$\Delta\alpha = (\alpha, 0, 0, 0,\ 0, 0, 0, 0,\ 0, 0, 0, 0,\ 0, 0, 0, 0),$$
$$\Delta\beta = \texttt{MixColumns} \circ \texttt{ShiftRows}(\beta, 0, 0, 0,\ 0, 0, 0, 0,\ 0, 0, 0, 0,\ 0, 0, 0, 0),$$

and thus $\Delta\alpha$ after $\texttt{AES}_0$ will change to $\Delta\beta$ with probability 2^{-6}. We note that the difference α can be located in any of the 16 bytes of the state.

Detecting the Cancellation Between $\boldsymbol{\Delta\alpha}$ and $\boldsymbol{\Delta\beta}$. We can detect if the cancellation occurred by observing the differences in the ciphertexts C_i (or equivalently, the difference in the key streams R_i) after eight rounds. There are two cases:

- **Cancellation occurred.** From the trail on Fig. 3, it follows that the difference $\Delta R_8 \oplus \Delta R_7$ is obtained when ΔR_7 goes through one $\texttt{AES}_0$ round. It means that the difference in each of the 16 bytes of ΔR_7 can be matched through the S-Box with the corresponding differences in the bytes of $\texttt{ShiftRows}^{-1} \circ \texttt{MixColumns}^{-1}(\Delta R_8 \oplus \Delta R_7)$. We note that the probability of matching is one.
- **Cancellation did not occurred.** If the cancellation did not occur, then there are additional state words with differences (depicted with red lines in Fig. 3). In this case, $\Delta R_8 \oplus \Delta R_7$ is obtained when $\Delta R_7 \oplus \Delta X$ (where ΔX is the non-zero difference in S_6) goes through $\texttt{AES}_0$. In contrast to the above case, now ΔR_7 and $\texttt{ShiftRows}^{-1} \circ \texttt{MixColumns}^{-1}(\Delta R_8 \oplus \Delta R_7)$ can be matched through the S-Box only with some probability lower than one.

Two randomly chosen differences can be matched through the S-Box with a probability $127/256 \approx 2^{-1}$. Without loss of generality, we can assume that ΔX is active in all 16 bytes[2]. Therefore, when $\Delta\alpha$ and $\Delta\beta$ cancel, the probability of a 16-byte match is 1, however, when they do not cancel, then the probability drops to 2^{-16}. As a result, we can easily distinguish the above two cases, by analyzing ΔR_7 and ΔR_8.

The same distinguishing method can be applied to 4 additional rounds (see Fig. 4). This way, we can increase the probability of distinguishing the two cases, and end up with a very low probability of matching differences through S-Boxes in the case when $\Delta\alpha$ and $\Delta\beta$ do not cancel. As we apply it to five rounds, the probability becomes $2^{-5\cdot 16} = 2^{-80}$.

3.2 Recovery of State Words

Assume that $\Delta\alpha$ and $\Delta\beta$ have canceled (as demonstrated above, we can single out the case when they cancel). It means that we have the input difference ΔR_7 and the output difference $\Delta R_8 \oplus \Delta R_7$ of an active $\texttt{AES}_0$ for the word S_7, i.e., $\texttt{SubBytes}(\Delta R_7) = \texttt{ShiftRows}^{-1} \circ \texttt{MixColumns}^{-1}(\Delta R_8 \oplus \Delta R_7)$. As in S_7, all 16 bytes are active (with a probability very close to 1), we can easily find the values of the individual bytes by the well-known method of solving 16 differential

[2] The difference ΔX is produced after some initial difference goes through multiple $\texttt{AES}$ rounds, thus we can assume ΔX is a random 16-byte difference. As a result, the probability that in ΔX all 16 bytes are active is $(1 - 1/256)^{16} \approx 1$.

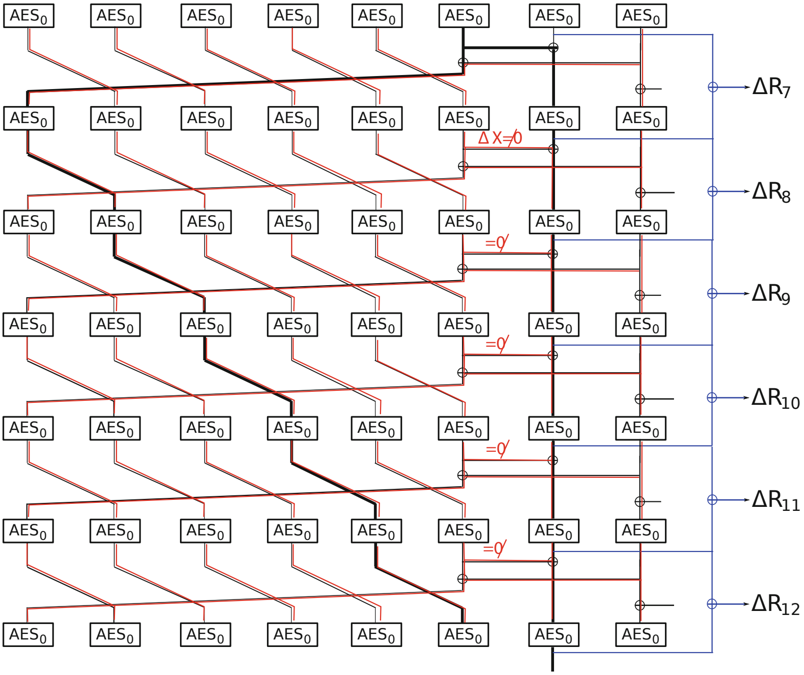

Fig. 4. Extending the previous trail for 4 additional rounds.

equations of the form $\mathsf{S}(x \oplus \Delta_{input}) \oplus \mathsf{S}(x) = \Delta_{output}$ that come from the system using S-Box S. Each such equation on average has two solutions, because if x is a solution, then $x \oplus \Delta_{input}$ is also a solution. To find a single solution for each byte, we repeat once the recovery for different $\Delta\alpha$ and $\Delta\beta$. As a result, we can recover the value of S_7 at round 8 of the encryption.

Using the very same method, we can recover S_7 at rounds 9, 10, 11 and 12. For instance, for round 9, the input (resp. output) difference of $\mathtt{AES}_0$ is $\Delta R_7 \oplus \Delta R_8$ (resp. $\Delta R_7 \oplus \Delta R_8 \oplus \Delta R_9$). With the knowledge of the values of 5 consecutive S_7, we can uniquely recover the values of S_6, S_5, S_4, S_3 at round 8 by simple computation using those words. Let S_u^{vR} be the u-th variable of the state for round v. For instance, $S_6^{8\mathrm{R}}$ is computed by $S_7^{8\mathrm{R}} \oplus \mathtt{AES}_0^{-1}(S_7^{9\mathrm{R}})$.

We can recover two more S_7 words (of additional 2 rounds) if we shift the round where we apply the difference $\Delta\alpha$ and instead to P_0 we introduce $\Delta\alpha$ at P_2 and $\Delta\beta$ at P_3. Hence, we will have the values of S_7 for 7 consecutive rounds.

The state word S_8 is different compared to the remaining seven words and it is not possible to recover it by using the above method. Nevertheless, we can still recover S_8 at round 0 of the encryption based on the differences $\Delta\alpha$ and $\Delta\beta$, i.e., we can recover the active byte where the difference $\Delta\alpha$ is non-zero. By repeating the recovery with 16 different positions of active bytes, we can deduce the whole state word S_8 at round 0. As S_8 does not take feedback from any other word (but the plaintext), we can easily find the value of S_8 at any round, including our target round 8. That is, with the knowledge of S_7 of seven consecutive rounds (8, 9,...14) which can be deduced as shown above, and S_8 at round 8, we can recover the full state at round 8.

3.3 The Attack

We now present the universal forgery attack. The goal of the attack is to produce a tag of an arbitrary plaintext. In our case, the attack works as long as the length of the plaintext is at least 16 blocks (240 bytes). Our forgery is based on a state recovery, i.e., if at some round the whole state is known, then the tag can easily be produced by performing the remaining operations of the finalization, and therefore it can be produced offline.

Let $P_0, P_1, \ldots, P_{14}$ be the first 15 blocks of the plaintext. Then, the forgery can be described with the following algorithm:

```
1. Query the first 15 plaintext blocks of the target ($P_0 \| P_1 \| \cdots \| P_{14}$), and obtain
   the key stream $R_0, R_1, \cdots, R_{14}$.
2. FOR $position = 1$ to 16 DO
3.      FOR $i = 1$ to $2^7$ DO
4.           Choose 1-byte difference $\Delta\alpha^i$ with active byte at $position$ and
   find the corresponding $\Delta\beta^i$.
5.           Query ($P_0 \oplus \Delta\alpha^i \| P_1 \oplus \Delta\beta^i \| P_2 \| \cdots \| P_{14}$) and obtain the key stream
   $R^i_0, \cdots, R^i_{14}$.
6.           Check if the difference $R_7 \oplus R^i_7$ can result in $R_7 \oplus R^i_7 \oplus R_8 \oplus R^i_8$
   by $\mathtt{AES}_0$.
7.           Check the same property for additional 4 rounds.
8.           Save the pairs that pass all the above checks.
9.      END FOR
10.     Recover the byte at $position$ of the state word $S_8$ at round 0
11. END FOR
12. Recover $S_7$ at rounds 8,9,10,11,12
13. FOR $i = 1$ to $2^7$ DO
14.     Choose 1-byte difference $\Delta\alpha^i$ and find the corresponding $\Delta\beta^i$.
15.     Query ($P_0 \| P_1 \| P_2 \oplus \Delta\alpha^i \| P_3 \oplus \Delta\beta^i \| P_4 \| \cdots \| P_{14}$) and obtain the key
   stream $R^i_0, \cdots, R^i_{14}$.
16.     Check if the difference $R_9 \oplus R^i_9$ can result in $R_9 \oplus R^i_9 \oplus R_{10} \oplus R^i_{10}$ by
   $\mathtt{AES}_0$.
17.     Check the same property for next 4 additional rounds.
18.     Save the pairs that pass all the above checks.
19. END FOR
20. Recover $S_7$ at rounds 13 and 14.
21. Deduce all the state words at round 8.
22. Go through the remaining of the transformations and produce the tag.
```

The first loop is used to recover S_8, and to recover five S_7, and the second to recover the remaining two S_7. Note, each of the loops (the inner loop of the first loop) will produce two pairs, as the probability of the trail in the top ($\Delta\alpha$ will be canceled by $\Delta\beta$) is 2^{-6}. In case no good trails with probability 2^{-6} exist, the attacker can switch to ones with probability 2^{-7} and run the loops 2^8 times. Furthermore, as we have seen from the previous analysis, a probability of false positives is very low (around 2^{-80}).

From the algorithm, it follows that the time complexity of the attack is $16 \cdot 2^7 + 2^7 \approx 2^{11}$ computations. The data complexity is similar and comes in a form of chosen plaintexts. To solve efficiently the differential equations, the attack needs about 2^{16} bytes in memory.

4 Practical Distinguisher for a Weak-Key Class of PAES

We continue our analysis by presenting a distinguisher for a class of 2^{64} weak keys (out of 2^{128} keys) in PAES-8. The distinguisher requires negligible time complexity and only a single pair of known plaintext-ciphertext and a chosen nonce. It exploits the lack of constants in the design and the symmetric properties of the keyless AES round function. Although we give the distinguisher for PAES-8, we note that a similar attack is applicable to the nonce-respecting mode PAES-4.

4.1 Symmetric Properties of the AES Round Function

We first recall the known symmetric property of the AES round function [7]. Namely, if a state is symmetric in the sense that its two halves are equal, then the keyless round function $\mathtt{AES}_0$ of the AES maintains this property. We recall the property of [7] using block matrices, and we introduce the following more general notations:

$$U(A,B) = \left(\begin{array}{c|c} A & A \\ \hline B & B \end{array}\right), \qquad V(A,B) = \left(\begin{array}{c|c} A & B \\ \hline B & A \end{array}\right), \qquad W(A,B) = \left(\begin{array}{c|c} A & B \\ \hline A & B \end{array}\right).$$

Additionally, we denote by $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{W}$ the associated sets respectively for all possible values of the 2×2 block matrices A and B. Finally, we denote M the constant MDS matrix used in the AES round function, and observe that:

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} = \left(\begin{array}{c|c} M_1 & M_2 \\ \hline M_2 & M_1 \end{array}\right) = V(M_1, M_2) \in \mathcal{V}.$$

Property 1. Let $S \in \mathcal{U}$. Then, $\mathtt{AES}_0(S) \in \mathcal{U}$.

Proof. Let $S = U(A,B) \in \mathcal{U}$, and write the bytes in S as:

$$\left(\begin{array}{c|c} A & A \\ \hline B & B \end{array}\right) = \left(\begin{array}{cc|cc} x_0 & x_4 & x_0 & x_4 \\ x_1 & x_5 & x_1 & x_5 \\ \hline x_2 & x_6 & x_2 & x_6 \\ x_3 & x_7 & x_3 & x_7 \end{array}\right).$$

As the SubBytes operation applies the same bijection to all the bytes in the state, we ignore it here as it obviously preserves the structure. After the ShiftRows operation, the state becomes

$$\left(\begin{array}{cc|cc} x_0 & x_4 & x_0 & x_4 \\ x_5 & x_1 & x_5 & x_1 \\ \hline x_2 & x_6 & x_2 & x_6 \\ x_7 & x_3 & x_7 & x_3 \end{array}\right) \overset{\text{def}}{=} \left(\begin{array}{c|c} A' & A' \\ \hline B' & B' \end{array}\right),$$

thus it still belongs to $\mathcal{U}$. Then, the MixColumns operation results in:

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \times \begin{pmatrix} x_0 & x_4 & x_0 & x_4 \\ x_5 & x_1 & x_5 & x_1 \\ x_2 & x_6 & x_2 & x_6 \\ x_7 & x_3 & x_7 & x_3 \end{pmatrix} = \left(\begin{array}{c|c} M_1 & M_2 \\ \hline M_2 & M_1 \end{array}\right) \times \left(\begin{array}{c|c} A' & A' \\ \hline B' & B' \end{array}\right)$$

$$= \left(\begin{array}{c|c} M_1A' \oplus M_2B' & M_1A' \oplus M_2B' \\ \hline M_2A' \oplus M_1B' & M_2A' \oplus M_1B' \end{array}\right) \overset{\text{def}}{=} \left(\begin{array}{c|c} A'' & A'' \\ \hline B'' & B'' \end{array}\right) \in \mathcal{U}.$$

□

Property 2. Let $S \in \mathcal{W}$. Then, $\texttt{AES}_0(S) \in \mathcal{V}$, and $\texttt{AES}_0(\texttt{AES}_0(S)) \in \mathcal{W}$.

Proof. Let $S = W(A, B) \in \mathcal{W}$, and write the bytes in S as:

$$\left(\begin{array}{c|c} A & B \\ \hline A & B \end{array}\right) = \left(\begin{array}{cc|cc} x_0 & x_2 & x_4 & x_6 \\ x_1 & x_3 & x_5 & x_7 \\ \hline x_0 & x_2 & x_4 & x_6 \\ x_1 & x_3 & x_5 & x_7 \end{array}\right).$$

Again, we ignore the SubBytes operation as the applied bijection preserves the structure of the internal states. However, after the ShiftRows operation the state becomes:

$$\left(\begin{array}{cc|cc} x_0 & x_2 & x_4 & x_6 \\ x_3 & x_5 & x_7 & x_1 \\ \hline x_4 & x_6 & x_0 & x_2 \\ x_7 & x_1 & x_3 & x_5 \end{array}\right) \overset{\text{def}}{=} \left(\begin{array}{c|c} A' & B' \\ \hline B' & A' \end{array}\right) \in \mathcal{V},$$

which is transformed by the subsequent MixColumns transformation into the state:

$$\left(\begin{array}{c|c} M_1 & M_2 \\ \hline M_2 & M_1 \end{array}\right) \times \left(\begin{array}{c|c} A' & B' \\ \hline B' & A' \end{array}\right) = \left(\begin{array}{c|c} M_1A' \oplus M_2B' & M_1B' \oplus M_2A' \\ \hline M_2A' \oplus M_1B' & M_2B' \oplus M_1A' \end{array}\right)$$

$$\overset{\text{def}}{=} \left(\begin{array}{c|c} A'' & B'' \\ \hline B'' & A'' \end{array}\right) \in \mathcal{V}.$$

After applying a second keyless AES round, we get:

$$\left(\begin{array}{c|c} A'' & B'' \\ \hline B'' & A'' \end{array}\right) = \left(\begin{array}{cc|cc} y_0 & y_2 & y_4 & y_6 \\ y_1 & y_3 & y_5 & y_7 \\ \hline y_4 & y_6 & y_0 & y_2 \\ y_5 & y_7 & y_1 & y_3 \end{array}\right) \xrightarrow{\textsf{SR}} \left(\begin{array}{cc|cc} y_0 & y_2 & y_4 & y_6 \\ y_3 & y_5 & y_7 & y_1 \\ \hline y_0 & y_2 & y_4 & y_6 \\ y_3 & y_5 & y_7 & y_1 \end{array}\right) \overset{\text{def}}{=} \left(\begin{array}{c|c} A''' & B''' \\ \hline A''' & B''' \end{array}\right) \in \mathcal{W},$$

and by the MixColumns:

$$\left(\begin{array}{c|c} M_1 & M_2 \\ \hline M_2 & M_1 \end{array}\right) \times \left(\begin{array}{c|c} A''' & B''' \\ \hline A''' & B''' \end{array}\right) = \left(\begin{array}{c|c} M_1 A''' \oplus M_2 A''' & M_1 B''' \oplus M_2 B''' \\ \hline M_2 A''' \oplus M_1 A''' & M_2 B''' \oplus M_1 B''' \end{array}\right)$$

$$\stackrel{\text{def}}{=} \left(\begin{array}{c|c} A'''' & B'''' \\ \hline A'''' & B'''' \end{array}\right) \in \mathcal{W},$$

which concludes the proof. □

Finally, we can represent the action of the keyless AES round function AES_0 on the three sets $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{W}$ as follows on Fig. 5.

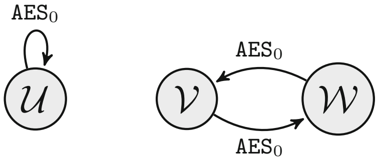

Fig. 5. Action of AES_0 of the symmetrical states from $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{W}$.

4.2 Symmetric Properties of the PAES Transformations

Along with AES_0, PAES uses a few more transformations, in particular, the XOR and the linear transformation L. We investigate here how these two transformations preserve the class belongings.

Property 3. Let $\mathcal{X}$ be either $\mathcal{U}$, $\mathcal{V}$ or $\mathcal{W}$, and let $S_1, S_2 \in \mathcal{X}$. Then, $S_1 \oplus S_2 \in \mathcal{X}$.

Proof. Let $S_1 = U(A_1, B_1), S_2 = U(A_2, B_2) \in \mathcal{U}$. Then:

$$S_1 \oplus S_2 = \left(\begin{array}{c|c} A_1 & A_1 \\ \hline B_1 & B_1 \end{array}\right) \oplus \left(\begin{array}{c|c} A_2 & A_2 \\ \hline B_2 & B_2 \end{array}\right) = \left(\begin{array}{c|c} A_1 \oplus A_2 & A_1 \oplus A_2 \\ \hline B_1 \oplus B_2 & B_1 \oplus B_2 \end{array}\right) \in \mathcal{U}.$$

The cases for $\mathcal{V}$ and $\mathcal{W}$ can be proven similarly. □

Property 4. Let $S \in \mathcal{W}$. Then, $L(S) \in \mathcal{W}$.

Proof. Let $S = W(A, B) \in \mathcal{W}$, and write the bytes in S as:

$$S = \left(\begin{array}{c|c} A & B \\ \hline A & B \end{array}\right) = \left(\begin{array}{cc|cc} x_0 & x_2 & x_4 & x_6 \\ x_1 & x_3 & x_5 & x_7 \\ \hline x_0 & x_2 & x_4 & x_6 \\ x_1 & x_3 & x_5 & x_7 \end{array}\right).$$

Then:

$$L(S) = L\left(\begin{array}{cc|cc} x_0 & x_2 & x_4 & x_6 \\ x_1 & x_3 & x_5 & x_7 \\ \hline x_0 & x_2 & x_4 & x_6 \\ x_1 & x_3 & x_5 & x_7 \end{array}\right) = \left(\begin{array}{cc|cc} x_2 & x_4 & x_6 \oplus x_0 & x_0 \\ x_3 & x_5 & x_7 \oplus x_1 & x_1 \\ \hline x_2 & x_4 & x_6 \oplus x_0 & x_0 \\ x_3 & x_5 & x_7 \oplus x_1 & x_1 \end{array}\right) \in \mathcal{W}.$$

□

4.3 The Distinguisher

To distinguish PAES, we use the first ciphertext C_0 produced during the encryption of an arbitrary plaintext P_0 with a secret key $K \in \mathcal{W}$ and nonce $N \in \mathcal{W}$. The key K can be any of such 2^{64} keys (the first two rows equal to the second two rows), and the same structure holds for the nonce N.

We first inspect how the state words $S_1, S_2, \ldots, S_8$ change the class belongings (either $\mathcal{W}$ or $\mathcal{V}$) from the very first to the last steps of the initialization phase:

- $K, N \in \mathcal{W}$. By Properties 3 and 4 $S_1, S_2, \ldots, S_8 \in \mathcal{W}$ after the initial assignments in the initialization.
- After the first update. By Property 3, the XORs do not change the class belongings, thus each S_6, S_7, S_8 stay in $\mathcal{W}$ after the XORs at the top of the *StateUpdate*. Further, according to the Property 2, $\mathtt{AES}_0$ changes the class from $\mathcal{W}$ to $\mathcal{V}$. Consequently, at the end of the first update, $S_i \in \mathcal{V}, i = 1, \ldots, 8$.
- The second update is similar to the previous one, but this time the class of S_i changes to $\mathcal{W}$.
- ...
- After the tenth update. The classes of all S_i are $\mathcal{W}$.
- After the XORs of the key. As each S_i is in $\mathcal{W}$ and the key is in $\mathcal{W}$, by Property 3, it follows that each S_i will be in $\mathcal{W}$.

We now focus on the production of the ciphertext C_0. Obviously, $tmp = S_7 = W(A_1, B_1) \in \mathcal{W}$ and after the application of the *StateUpdate*, $S_7 = V(A_2, B_2) \in \mathcal{V}$ by Property 2. Thus, from the definition of the ciphertext $C_0 = P_0 \oplus tmp \oplus S_7$, we get:

$$C_0 \oplus P_0 = \left(\begin{array}{c|c} A_1 & B_1 \\ \hline A_1 & B_1 \end{array}\right) \oplus \left(\begin{array}{c|c} A_2 & B_2 \\ \hline B_2 & A_2 \end{array}\right) = \left(\begin{array}{c|c} A_1 \oplus A_2 & B_1 \oplus B_2 \\ \hline A_1 \oplus B_2 & B_1 \oplus A_2 \end{array}\right) = \left(\begin{array}{c|c} X & Z \\ \hline Y & T \end{array}\right).$$

Obviously $X \oplus Y \oplus Z \oplus T = 0$, hence the xor of the four 32-bit blocks of the first ciphertext and plaintext must result in a zero block. Therefore, we have a distinguisher which requires negligible complexity and only a single block of plaintext/ciphertexts to distinguish PAES when instantiated with any of the 2^{64} keys and nonces from the class $\mathcal{W}$. We note that our computer simulation confirmed the correctness of the distinguisher.

5 Conclusion

We have shown two practical attacks on the CAESAR candidate PAES: a universal forgery attack and a distinguisher, which contradict the security claims of this authenticated encryption scheme.

Our analysis gives insights into possible misuses of the AES round function. Although this transformation per se provides excellent resistance against differential and linear attacks (once it has been iterated several times), by no means it is sufficient proof of security against all attacks. The designs based on the round function that does not apply any constants, as we have seen on the example of our distinguisher and the chosen-key rotational distinguisher [10] of PAES, are susceptible to attacks that exploit the symmetry of the AES transformations. Consequently, using random constants in such designs should be taken as a requirement to destroy those symmetric behaviors. Furthermore, as our forgery attack shows, evaluating the differential properties in a straightforward manner (providing the best in terms of probability differential characteristic), does not guarantee security against differential attacks in the nonce-repeating mode.

We would also like to emphasize the importance of the technique used in the forgery attack on the nonce-repeating mode. Due to the mode and the attack framework, there is no need to provide a valid tag at the beginning of the attack (forgery or state recovery). Hence the attacker can focus only on finding a differential characteristic that will leak differences in state words sufficient for recovery based on solving differential equations. The characteristic does not necessarily need to hold with a high probability, but for the forgery on PAES this was required in the first two rounds only because there was an alternative path that does not permit state recovery. In general, the probability of the characteristic is irrelevant, however, it is important for the characteristic to leak input and output differences of non-linear operations which subsequently will be used to recover the state bits. We believe that this technique (improved or modified variants) can be a valuable approach for cryptanalysis of other CAESAR submissions and authenticated encryption schemes.

Acknowledgment. We would like to thank the anonymous reviewers for their detailed feedback and comments.

References

1. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K., Compute, D.: AES-COPA v1. Submitted to the CAESAR competition, March 2014
2. Bernstein, D.: CAESAR Competition. http://competitions.cr.yp.to/caesar.html
3. Daemen, J., Rijmen, V.: The Design of Rijndael: - The Advanced Encryption Standard. Springer, New York (2002)
4. Derbez, P., Fouque, P.A., Jean, J.: Improved key recovery attacks on reduced-round AES in the single-key setting. IACR Cryptology ePrint Archive 2012, 477 (2012)
5. Derbez, P., Fouque, P.-A., Jean, J.: Improved key recovery attacks on reduced-round AES in the single-key setting. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 371–387. Springer, Heidelberg (2013)
6. Krovetz, T., Rogaway, P.: OCB v1. Submitted to the CAESAR competition, March 2014
7. Van Le, T., Sparr, R., Wernsdorf, R., Desmedt, Y.G.: Complementation-like and cyclic properties of AES round functions. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 128–141. Springer, Heidelberg (2005)
8. McGrew, D., Viega, J.: The Galois/Counter mode of operation (GCM). Submission to NIST (2004). http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf

9. Nikolić, I.: Tiaoxin-346 v1. Submitted to the CAESAR competition, March 2014
10. Saarinen, M.J.O.: PAES and rotations, March 2014. https://groups.google.com/forum/#!topic/crypto-competitions/vRmJdRQBzOo
11. Wang, L.: SHELL v1. Submitted to the CAESAR competition, March 2014
12. Wu, H., Preneel, B.: AEGIS v1. Submitted to the CAESAR competition, March 2014
13. Ye, D., Wang, P., Hu, L., Wang, L., Xie, Y., Sun, S., Wang, P.: PAES v1. Submitted to the CAESAR competition, March 2014