Collision Attacks Against CAESAR Candidates

Forgery and Key-Recovery Against AEZ and Marble

Thomas Fuhr[1](✉), Gaëtan Leurent[2], and Valentin Suder[3]

[1] ANSSI, Paris, France
thomas.fuhr@ssi.gouv.fr
[2] Inria, Paris, France
[3] University of Waterloo, Waterloo, Canada

Abstract. In this paper we study authenticated encryption algorithms inspired by the OCB mode (Offset Codebook). These algorithms use secret offsets (masks derived from a whitening key) to turn a block cipher into a tweakable block cipher, following the XE or XEX construction.

OCB has a security proof up to $2^{n/2}$ queries, and a matching forgery attack was described by Ferguson, where the main step of the attack recovers the whitening key. In this work we study recent authenticated encryption algorithms inspired by OCB, such as Marble, AEZ, and COPA. While Ferguson's attack is not applicable to those algorithms, we show that it is still possible to recover the secret mask with birthday complexity. Recovering the secret mask easily leads to a forgery attack, but it also leads to more devastating attacks, with a key-recovery attack against Marble and AEZ v2 and v3 with birthday complexity.

For Marble, this clearly violates the security claims of full n-bit security. For AEZ, this matches the security proof, but we believe it is nonetheless a quite undesirable property that collision attacks allow to recover the master key, and more robust designs would be desirable.

Our attack against AEZ is generic and independent of the internal permutation (in particular, it still works with the full AES), but the key-recovery is specific to the key derivation used in AEZ v2 and v3. Against Marble, the forgery attack is generic, but the key-recovery exploits the structure of the E permutation (4 AES rounds). In particular, we introduce a novel cryptanalytic method to attack 3 AES rounds followed by 3 inverse AES rounds, which can be of independent interest.

Keywords: CAESAR competition · Authenticated encryption · Cryptanalysis · Marble · AEZ · PMAC · Forgery · Key-recovery

1 Introduction

The purpose of an *Authenticated Encryption* scheme is to provide both privacy and integrity with a single cryptographic algorithm. In 2014, the CAESAR competition was launched with the goal to identify good Authenticated Encryption schemes as better alternatives to current options such as AES-GCM [14]. 57

T. Iwata and J.H. Cheon (Eds.): ASIACRYPT 2015, Part II, LNCS 9453, pp. 510–532, 2015.
DOI: 10.1007/978-3-662-48800-3_21

candidates have been submitted to the CAESAR competition, and they must now be analyzed carefully.

In this paper, we provide a security analysis of the AES-based candidates Marble [5] and AEZ v3 [7]. Both designs are inspired by OCB [16], designed in 2001 by Rogaway, Bellare, Black, and Krovetz. They are built as modes of operation of a block cipher[1], using secret offsets at the input and/or output of the block cipher calls.

OCB. In OCB, a whitening key L is derived from the master key K, and the i-th message block M_i is enciphered to $C_i = E_K(M_i \oplus \gamma_i \cdot L) \oplus \gamma_i \cdot L$, where γ_i is a (Gray) counter, $\cdot$ is a finite field multiplication, and $\gamma_i \cdot L$ is the i-th offset. This design principle was later formalized as the XE and XEX construction [15], and proved to turn efficiently a secure block cipher into a secure tweakable block cipher [12]. OCB with a an n-bit block cipher is proven secure up to $2^{n/2}$ queries, and Ferguson showed a collision attack matching the bound [3]. The attack uses a long message M, so that there is a collision between two block cipher inputs:

$$M_i \oplus \gamma_i \cdot L = M_j \oplus \gamma_j \cdot L$$

The collision can be detected because $M_i \oplus C_i = M_j \oplus C_j$, and the value of L is recovered as $(\gamma_i \oplus \gamma_j)^{-1} \cdot (M_i \oplus M_j)$. When L is known, it is easy to forge messages.

Marble. Marble [5] is a CAESAR candidate by Jian Guo inspired by COPA [1]. COPA was designed in 2013, and combines OCB's offsets with an internal dependency chain in order to achieve some security in the case of nonce repetition. Marble uses two internal chains in order to prevent birthday attacks on the internal chain, and uses reduced-round AES as building blocks. Marble claims security against nonce-repetition, and against release of unverified plaintexts, but cannot hide common prefixes in case of nonce reuse (Marble is online). As opposed to most CAESAR candidates, Marble claims full 128-bit security (beyond the birthday bound). The structure of Marble can be seen in Fig. 2.

Results presented so far on Marble include a cipher-text distinguisher with complexity 2^{64}, similar to the distinguisher against the counter mode [17].

AEZ. AEZ is a CAESAR candidate designed by Hoang, Krovetz, and Rogaway. The authors define the security notion of *Robust AE*, which is the optimal security achievable when nonces are repeated, and unverified plaintexts are released. AEZ is claimed to achieve this security notion. In this paper, we focus on the current version of AEZ, AEZ v3, as proposed on the `crypto-competition` mailing list, and presented at DIAC [7]. AEZ v3 has also been accepted at Eurocrypt 2015, and presented as one of the honorable mentions for the best papers

[1] For efficiency reasons, Marble and AEZ actually use 4-rounds of AES rather than a full block cipher.

award [8]. Our result can also be applied to AEZ v2, but not to AEZ v1, because of a different key expansion.

As far as we are aware, no cryptanalysis of AEZ as been published so far.

Our Results. In this paper, we describe generic collision attacks against Marble and AEZ, allowing to recover the whitening key with about $2^{n/2}$ chosen message queries. When the whitening key is known the security offered by Marble and AEZ crumbles and we show a forgery attack using a single extra encryption query. Moreover, we extend this result to key-recovery attacks using properties of the internal permutations and/or the key scheduling.

Our results are summarized in Tables 1 and 2. The data complexity is listed in number of message blocks (16 bytes). We now detail our results on each Authenticated-Encryption with Associated-Data (AEAD) scheme.

Marble. Our attack against Marble uses queries with repeated nonces, which should be secure according to the security claims of Marble. Since Marble claims security beyond the birthday bound (allowing up to 2^n block of data), the forgery attack using collisions clearly violates the security claims. In addition, we show that if unverified plaintexts are released, *i.e.* if we can obtain plaintexts from ciphertexts without a valid tag, then we can further recover the master key K. For this attack, we build special queries so that only 3 forward AES rounds and 3 backwards AES rounds are active, and we develop a novel method to attack such a reduced cipher with only known plaintext/ciphertexts. Our attack can be build upon two different distinguishers. the first one is based on the detection of collision events, and the second one on a statistical property. In both cases, our attack requires about 2^{33} queries and its time complexity is 2^{64}; we believe this result is also of independent interest.

Following the disclosure of this attack, Guo proposed a minor modification of the specifications of Marble as version 1.2 [6]. However, our attack is still applicable to the modified version, as shown independently by ourselves and Lu [13]. Guo later decided to withdraw Marble from the CAESAR competition.

AEZ. Our analysis of AEZ *v*3 focuses on the processing of Associated Data. In particular, if AEZ is used with an empty message and no nonce, it turns into a variant of PMAC, and the security notion of Robust AE becomes the usual MAC security notion. We show how to recover the whitening key of this variant of PMAC with a collision attack (a collision attack is also possible against the standard PMAC, *e.g.* following [11]). More importantly, the key derivation of PMAC allows to recover the master key K from the whitening key.

This attack does not violate the security proof, but matches the security bound. However, collision attacks usually have a more limited impact (*e.g.* only affecting authenticity), and it seems quite unfortunate that a collision attack leads to a key-recovery. This property should probably be avoided when possible[2].

[2] In AEZ *v*4, for the second round of the competition, the designers took into account our result and modified the key derivation in order to prevent this property.

Table 1. Our results against Marble.

Attack (Sec. claim)	Data	Time
Recover L	$2^{65} \times 2$ CP	2^{64}
Forgery (2^{128})	$2^{65} \times 2$ CP	2^{64}
Key-recovery (2^{128}):		
Collision-based[a]	$2^{65} \times 2$ CP + $2^{32.6} \times 130$ CC	2^{64}
Collision-based	$2^{65} \times 2$ CP + $2^{33} \times 130$ CC	2^{64}

[a] The chosen ciphertexts use the decryption-misuse model.

Table 2. Our results against AEZ.

Attack	Data[a]	Time	Success probability
Key-recovery	$2^{66.6}$	1	1
Key-recovery	2^{44}	1	$2^{-45.2}$

[a] The AEZ specification requires to rotate the key after 2^{44} blocks

COPA. After the release of an early version of this paper, Lu applied the same techniques to COPA, and described an attack to recover the whitening key [13]. The main attack in this paper actually targets the associated data processing, which uses PMAC, and can be applied to PMAC. However, the impact of this result is unclear because COPA and PMAC do not claim security beyond the birthday bound, and this attack cannot be turned into a key-recovery attack.

Outline of the Paper. Since our collision attack on AEZ is much simpler than the attack against Marble, we first explain it in Sect. 2. Then we give a short description of the Marble authenticated encryption algorithm in Sect. 3. In Sect. 4, we show how to recover the whitening key L and describe our forgery attack. Finally, we demonstrate in Sect. 5 how to recover the encryption key K from decryption-misuse queries.

2 Collision Attack Against AEZ

We first explain the collision attack on AEZ and the resulting key-recovering attack.

2.1 Short Description of AEZ

For simplicity, we consider AEZ used with only associated data, without any nonce or message (the attack can easily be applied with a fixed nonce and message if required). In this case, AEZ turns into a variant of PMAC, and the security claim becomes the usual MAC security definition. A particularity of

AEZ is that it allows a vector-valued input, *i.e.* it can authenticate a sequence of strings rather than a single string.

More precisely, the MAC is computed as follow:

- The key derivation algorithm generates keys k_0, k_1 and whitening keys J, L
- Full data blocks A_i^j of the j-th string (indexed from 1) are processed as:

$$X_i^j = E_{k_0}\left(A_i^j \oplus (i \bmod 8) \cdot J \oplus 2^{\lfloor (i-1)/8 \rfloor} \cdot L \oplus 8j \cdot J\right)$$

- If the last block is partial, it is enciphered as:

$$X_i^j = E_{k_0}(A_i^j \oplus 8j \cdot J)$$

- The first block to be processed is the ciphertext extension τ (corresponding to the tag length). It is $\tau = 128$ by default.
- The tag is computed as $E_{k_1}(\bigoplus_{i,j} X_i^j)$

where E is a full or reduced-round AES. This is illustrated by Fig. 1.

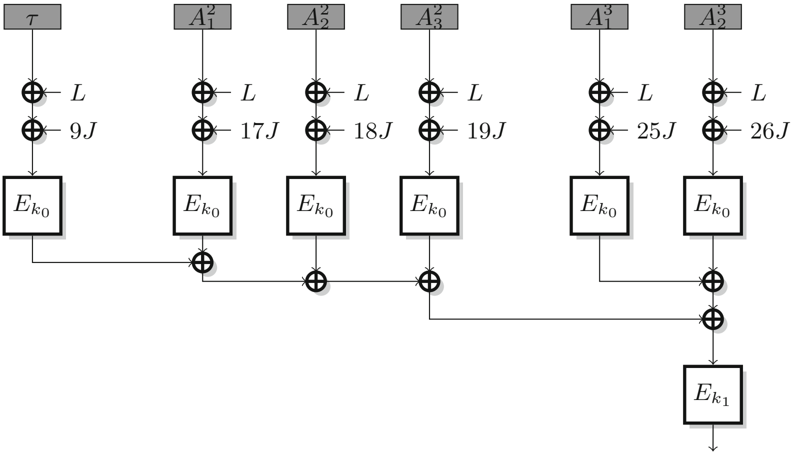


Fig. 1. AEZ used as a MAC (no message, no nonce, two AD strings).

2.2 Collision Attack on AEZ

In order to mount a collision attack against AEZ, we consider two sets of messages, with C a fixed block:

- $\mathcal{A} = \{A_x \mid x \in \{0 \dots 2^{64}-1\}\}$, with $A_x = \big(\tau; C; (C \parallel [x] \parallel 0^{64})\big)$
- $\mathcal{B} = \{B_y \mid y \in \{0 \dots 2^{64}-1\}\}$, with $B_y = \big(\tau; (C \parallel 0^{64} \parallel [y]); C\big)$

All message are made of two separate strings; message in $\mathcal{A}$ have a string of one block and a string of two blocks, while messages in $\mathcal{B}$ have a string of two blocks and a string of one block. In particular, we have:

- $\mathrm{MAC}(A_x) = E_{k1}\Big(E_{k_0}\big(\tau \oplus L \oplus 9J\big) \oplus E_{k_0}\big(C \oplus L \oplus 17J\big)$
  $\oplus E_{k_0}\big(C \oplus L \oplus 25J\big) \oplus E_{k_0}\big(([x] \parallel 0^{64}) \oplus L \oplus 26J\big)\Big)$
- $\mathrm{MAC}(B_y) = E_{k1}\Big(E_{k_0}\big(\tau \oplus L \oplus 9J\big) \oplus E_{k_0}\big(C \oplus L \oplus 17J\big)$
  $\oplus E_{k_0}\big((0^{64} \parallel [y]) \oplus L \oplus 18J\big) \oplus E_{k_0}\big(C \oplus L \oplus 25J\big)\Big)$

This leads to a simple collision attack: $\mathrm{MAC}(A_x) = \mathrm{MAC}(B_y)$ if and only if $[x] \parallel [y] = 8 \cdot J$ (where $8 = 18 \oplus 26$). With $\mathcal{A}$ and $\mathcal{B}$ of size 2^{64} as defined above, there is exactly one collision, and the collision immediately reveals the value of $J = 8^{-1} \cdot ([x] \parallel [y])$.

Key Recovery. Surprisingly, the key derivation of AEZ allows to recover the master key K from the whitening key J. More precisely, if the master key K is of length 128 bits or smaller, J is an encryption of K under a known constant C: $J = \mathrm{AES4}_C(K)$. This can easily be inverted: $K = \mathrm{AES4}_C^{-1}(J)$. We note that this is not the case in PMAC, where the whitening key is an encryption of 0 under the secret key K: $L = \mathrm{AES}_K(0)$.

This attack matches the security proof of AEZ and does not violate the security claims. However, a complete break of AEZ after the birthday bound is not expected: most schemes with birthday-bound security are more resilient and collision attacks don't allow key-recovery.

It should be mentioned that the Eurocrypt version of AEZ does not explicitly specify a key derivation algorithm, and leaves it as an open problem:

> "The key $K \in \textsc{Byte}^*$ is mapped to three 16-byte subkeys (I, J, L) using the key-derivation function (KDF) named Extract that is called at line 401. The definition of Extract is omitted from the figures and regarded as orthogonal to the rest of AEZ. See the AEZ spec for the current Extract: $\textsc{Byte}^* \to \textsc{Byte}^{48}$. In our view, it is an unresolved matter what the security properties (and even what signature) of a good KDF should be. Work has gone off in very different directions, and the area is currently the subject of a Password Hashing Competition (PHC) running concurrently with CAESAR."

Clearly, the key derivation of the AEZ v3 specification does not have the security properties of a good KDF.

Data Limit. The AEZ specification requires users to change the key after encrypting 2^{48} bytes, *i.e.* 2^{44} blocks. This prevents the attack as described above. However, we can perform the attack with smaller sets $\mathcal{A}$ and $\mathcal{B}$ of size $2^{41.4}$, with a success probability of $2^{-45.2}$. This is still much more efficient than generic attacks: with a time complexity of 2^{44}, a brute-force key search only succeeds with a probability of 2^{-84}.

3 Description of Marble

Marble is an authenticated encryption algorithm designed by Guo [5] with key-length and tag-length of both 128 bits. A plaintext and its associated data are divided into blocks of 128 bits and are then proceeded consecutively. Its internal permutation is based on a modified version of the AES block cipher. Unlike other authenticated encryption algorithms, Marble does not require a nonce.

Marble has very strong security claims: it claims to offer full 128-bit security, *i.e.* an attack should take time $T = 2^{128}$ even after large amount of data have been encrypted with the same key (up to $D = 2^{128}$). This is in contrast to many CAESAR candidates and classical modes of operations for block ciphers (*e.g.* GCM), which only offer a birthday level of security, *i.e.* the ciphers are secure as long as $T \cdot D < 2^{128}$.

In addition, Marble does not use nonces, and the security claim even holds if unverified plaintexts are released, *i.e.* the adversary can request the decryption of a ciphertext C without knowing a valid tag corresponding to C (decryption-misuse oracle). A few other CAESAR candidate allow the release of unverified plaintext (AEZ, POET, APE, Minalpher), but they only claim birthday security.

An overview of Marble is depicted in Fig. 2. The Marble mode of operation makes use of two 128-bit chaining variables s_1 and s_2, initialized with constants $const_1$ and $const_2$. The associated data and the plaintext are padded independently, so both resulting fields A and P can be divided into 128 bit blocks. We do not describe the padding function here, as it does not affect our attacks. We will denote a message to encrypt by $(AD \parallel M)$, where AD is a vector containing l_A 128-bit blocks of associated data and M is a vector containing l_M 128-bit blocks of plaintexts.

The internal primitive used is a modified block cipher, as intermediate values of the block are combined with the incoming chaining variables. Formally, the primitive uses 3 internal keyed permutations E_1, E_2 and E_3 and processes 128-bit blocks as follows. On input (P, s_1, s_2), (C, s_1', s_2') is defined as

$$\begin{aligned}
X &= E_{1K}(P)\\
(X', s_1') &= (3X + s_1, X + s_1)\\
Y &= E_{2K}(X')\\
(Y', s_2') &= (3Y + s_2, Y + s_2)\\
C &= E_{3K}(Y')
\end{aligned}$$

Note that additions and multiplications are performed in the binary Galois Field $\mathbb{F}_{2^{128}}$ defined by the primitive polynomial $x^{128}+x^7+x^2+x+1$. Furthermore, polynomials $\Sigma a_i X^i$ are denoted by the integers $\Sigma a_i 2^i$. Therefore, please note that additions and multiplications on such objects have to be interpreted as operations in the binary field (and not on the integer ring) and have to be handled carefully.

In the case of Marble, each one of the three permutations E_1, E_2 and E_3, is composed with 4 full-round of AES (*i.e.* SubByte, ShiftRow, MixColumn and

AddRoundKey). One can notice that no key addition is performed at the beginning of those permutations. 12 subkeys are therefore required. A 128-bit master key K is derived into 11 subkeys using the AES-128 key-schedule algorithm. The master key itself is used as the 12th subkey. For more information about the AES block cipher, we refer to [2].

The Marble encryption then works as follows. First, a mask L is derived from the key K by encrypting a constant $const_3$ (which also sets key-dependent $s_1[0], s_2[0]$). For each associated data block A_i ($i \geq 1$), a pre-whitening key is defined as $2^{i-1}3^2L$. For each plaintext block M_i, a pre-whitening key and a post-whitening key are defined as 2^iL and $2^{i-1}3L$. These blocks are processed iteratively, starting with associated data, as follows:

1. Addition (*i.e.* xor) of the pre-whitening key;
2. Application of the internal primitive;
3. For plaintext blocks, application of the post whitening key, which results in ciphertext blocks.

Finally, the tag is computed by encrypting an extra block defined as the sum of all plaintext blocks and all encrypted additional data blocks, with pre-whitening key $2^{\ell_M}7L$ and post-whitening key $2^{\ell_M-1}3L$.

4 A Universal Forgery Attack on Marble

In this section, we first describe a method to find the mask L using about 2^{65} chosen plaintext queries. Then, we use this knowledge to compute forgeries. Our attack enables to modify the associated data field in a way that affects neither the ciphertext nor the authentication tag. It can therefore be used to compute universal forgeries in a chosen plaintext setting.

4.1 Recover the Mask L

The main idea of the attack is to build a pair of message $M \neq M'$ so that the inputs to the E_1 functions are the same for both messages. This is possible if M and M' have the same total length, but the associated data and message parts have different lengths. When the inputs to E_1 collide, all the intermediate computations collide, and we can detect this event on the resulting ciphertexts. Please note that as different multiples of L are used for post-whitening, this operation is more tricky than detecting a collision on ciphertexts. In the following we use 2 blocks of AD and 1 block of message for M, but 1 block of AD and 2 blocks of message for M'.

More precisely, we encrypt messages M_α and M'_β, for different values $\alpha, \beta \in \mathbb{F}_{2^{128}}$, defined as follows (where $A \in \mathbb{F}_{2^{128}}$ is a constant value):

- $M_\alpha = (AD[1], AD[2] \parallel M[1]) = (A, 8\alpha \parallel 6\alpha)$;
- $M'_\beta = (AD[1] \parallel M'[1], M'[2]) = (A \parallel 8\beta, 6\beta)$.

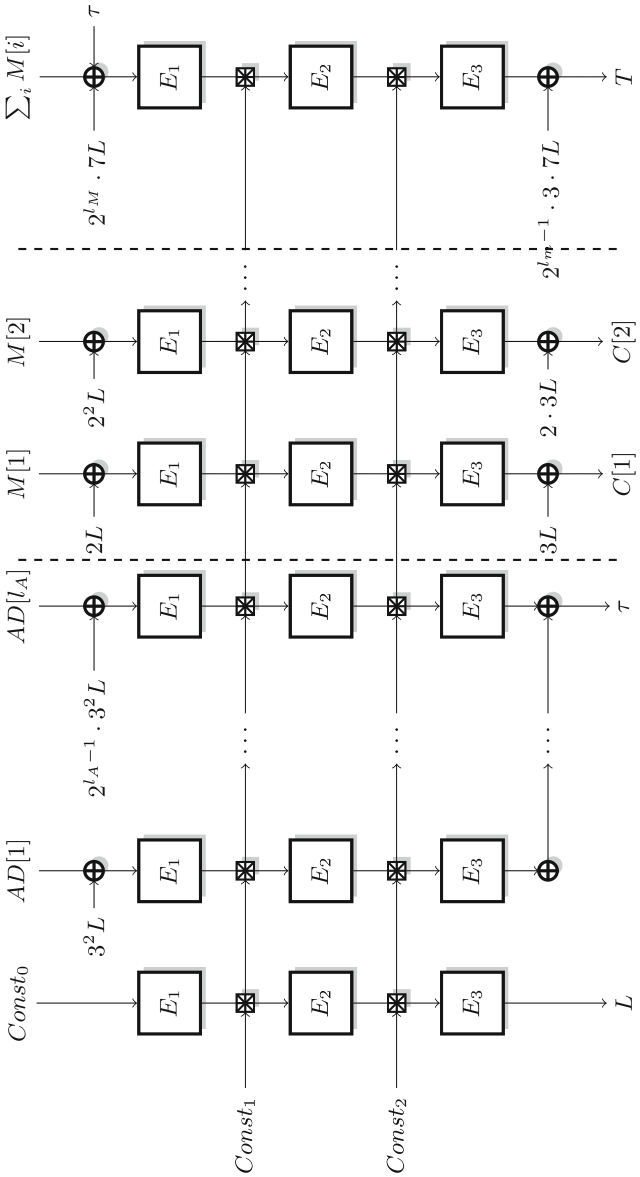


Fig. 2. General design of Marble.

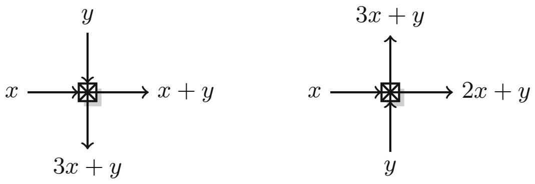


Fig. 3. The `TRANS` operation.

In the following, we consider sets of 2^{64} values α and β so that $\alpha \oplus \beta$ covers all values in $\mathbb{F}_{2^{128}}$. The inputs to the E_1 layer will be respectively (we note that $3^2 = 5$ in $\mathbb{F}_{2^{128}}$):

$$\begin{array}{llll} x_1 = A \oplus 5L & x_2 = 8\alpha \oplus 10L & x_3 = 6\alpha \oplus 2L & \text{for } M_\alpha \\ x'_1 = A \oplus 5L & x'_2 = 8\beta \oplus 2L & x'_3 = 6\beta \oplus 4L & \text{for } M'_\beta \end{array}$$

In particular, we have:

$$x_1 \oplus x'_1 = 0 \qquad x_2 \oplus x'_2 = 8(\alpha \oplus \beta \oplus L) \quad x_3 \oplus x'_3 = 6(\alpha \oplus \beta \oplus L)$$

Therefore, the inputs to E_1 collide when $\alpha \oplus \beta = L$.

We denote the output of the E_3 layer as y_i (respectively y'_i), and the corresponding ciphertexts as $C_\alpha[1]$ (respectively $(C'_\beta[1], C'_\beta[2])$). We have:

$$C_\alpha[1] = y_3 \oplus 3L \qquad C'_\beta[1] = y'_2 \oplus 3L \qquad C'_\beta[2] = y'_3 \oplus 6L$$

In particular, if $\alpha \oplus \beta = L$, we have $x_i = x'_i$ for $i \leq 3$, therefore $y_i = y'_i$ for $i \leq 3$, and $C_\alpha[1] \oplus C'_\beta[2] = 5L$ (since $3 \oplus 6 = 5$). In order to detect this event efficiently we match the set of values $\{C_\alpha[1] \oplus 5\alpha\}$ and $\{C'_\beta[2] \oplus 5\beta\}$. When $\alpha \oplus \beta = L$, we have a match, and we can easily filter false positives using a new message pair with a different value of the constant A. The full algorithm is given by Algorithm 1, using 2^{65} short encryption queries.

4.2 An Attack Against Marble 1.2

After the first release of our attack, Guo made a minor modification to the specification of Marble [6]. Namely, the input mask for the last block of associated data is changed from $2^{i-1}3^2L$ to $2^{i-1}3^3L$. Our attacks can be adapted as follows.

The adversary needs to query an encryption oracle for messages M_α and M'_β, defined as

- $M_\alpha = (AD[1], AD[2] \parallel M[1]) = (10\alpha, 28\alpha \parallel 6\alpha)$;
- $M'_\beta = (AD[1] \parallel M'[1], M'[2]) = (10\beta \parallel 28\beta, 6\beta)$.

Using the notations of Sect. 4.1, the inputs to the E_1 layer will be :

$$\begin{array}{llll} x_1 = 10\alpha \oplus 5L & x_2 = 28\alpha \oplus 30L & x_3 = 6\alpha \oplus 2L & \text{for } M_\alpha \\ x'_1 = 10\beta \oplus 15L & x'_2 = 28\beta \oplus 2L & x'_3 = 6\beta \oplus 4L & \text{for } M'_\beta \end{array}$$

Algorithm 1. Recover L from an encryption oracle $\mathcal{E}$.

```
$H \leftarrow \varnothing$                                          ▷ $H$ is a hash table
for $\alpha \in \{0, 1, \ldots, 2^{64} - 1\}$ do
    $(C[1] \parallel T) \leftarrow \mathcal{E}(0, 8\alpha \parallel 6\alpha)$
    $H\{C[1] \oplus 5\alpha\} \leftarrow \alpha$
end for
for $\beta \in \{0, 2^{64}, \ldots, 2^{128} - 2^{64}\}$ do
    $(C'[1], C'[2] \parallel T) \leftarrow \mathcal{E}(0 \parallel 8\beta, 6\beta)$
    if $H\{C'[2] \oplus 5\beta\}$ exists then
        $\alpha \leftarrow H\{C'[2] \oplus 5\beta\}$
        $(D[1] \parallel T) \leftarrow \mathcal{E}(1, 8\alpha \parallel 6\alpha)$
        $(D'[1], D'[2] \parallel T) \leftarrow \mathcal{E}(1 \parallel 8\beta, 6\beta)$
        if $D[1] \oplus 5\alpha = D'[2] \oplus 5\beta$ then
            return $\alpha \oplus \beta$
        end if
    end if
end for
```

In particular, we have:

$$\begin{aligned} x_1 \oplus x'_1 &= 10 \cdot (\alpha \oplus \beta \oplus L), \\ x_2 \oplus x'_2 &= 28 \cdot (\alpha \oplus \beta \oplus L), \\ x_3 \oplus x'_3 &= 6 \cdot (\alpha \oplus \beta \oplus L). \end{aligned}$$

If for some (α, β), $\alpha \oplus \beta = L$, then $x_i = x'_i$ for $i = 1, 2, 3$. Then, the outputs of E_3 verify $y_1 = y'_1$, $y_2 = y'_2$ and $y_3 = y'_3$ and therefore, $C_\alpha[1] \oplus 3L = C'_\beta[2] \oplus 6L$. As $3 \oplus 6 = 5$, This can also be expressed as:

$$C_\alpha[1] \oplus 5\alpha = C'_\beta[2] \oplus 5\beta.$$

Therefore, L has to be searched among the values $(\alpha \oplus \beta)$ for which this relation holds. As for our attack on the previous version of Marble, about 2^{64} different values of both α and β are required.

4.3 Computing Forgeries on Marble Without Whitening Keys

Once we have retrieved L, we can consider a simplified description of Marble where the masks are removed, as depicted in Fig. 4. In its mask-less description, Marble possesses an interesting property as described in Fig. 5: a series of identical input blocks X has a periodic effect on the internal state.

Indeed, if we let $E_1(X) = u$, $E_2(3S_1 \oplus u) = v$ and $E_2(3S_1 \oplus 2u) = w$, it is easy to see that after encrypting 4 blocks X, the internal states S_1 and S_2 remain unchanged. Furthermore, if we use a series of 8 consecutive identical associated data blocks X, the effect on τ also cancels out. This leads to a universal forgery attack: for any associated data AD and plaintext M, the adversary computes the masked value B of a chunk of 8 identical blocks of associated data after AD

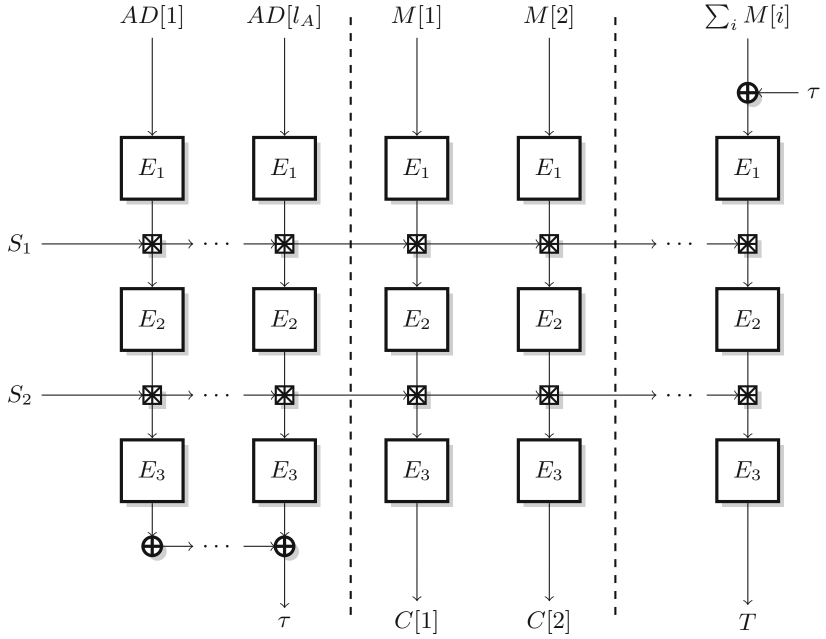


Fig. 4. Mask-less description of Marble. S_1 and S_2 are unknown key-dependent values.

and queries the encryption oracle on $((A, B) \parallel M)$. The answer $(C \parallel T)$ to that query is also a valid ciphertext for $(AD \parallel M)$, therefore the adversary can return $(C \parallel T)$ as a forgery. The attack is given as Algorithm 2.

Algorithm 2 . Compute the ciphertext $(C \parallel T)$ from $(AD \parallel M)$ using known L.

```
$B \leftarrow \left(2^l \cdot 3^2 \cdot L\right)_{l=l_A}^{l_A+7}$
$(C \parallel T) \leftarrow \mathcal{AE}_K((AD, B) \parallel M)$          ▷ Encryption oracle call
return $(C \parallel T)$
```

5 Key-Recovery Attack

We now show how to recover the master key once the mask L has been determined. In order to simplify the description of the attack, we now focus on the mask-less variant of Marble; however the full attack can easily be adapted to the full version of Marble with a known mask.

The main idea is to collect pairs of messages with a fixed difference in some internal state variables. This will allow us to attack a reduced cipher composed

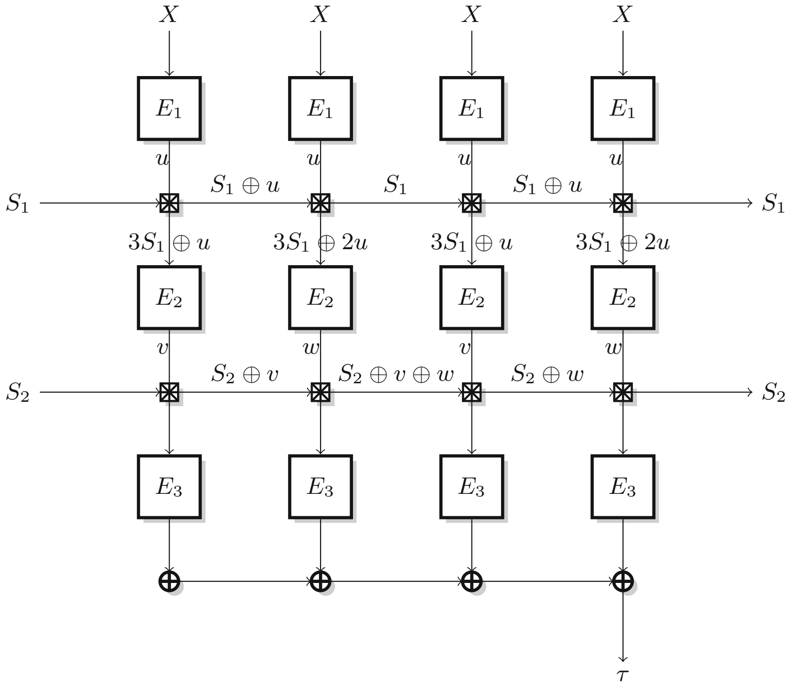


Fig. 5. Collision on the internal state of the associated data.

by 4 AES rounds followed by 4 inverse AES rounds rather than a 12-round AES (see details below). Moreover, we apply this strategy to E_1 rather that to E_3 because the whitening key of E_1 is directly derived from L. Since L is known, the first AES round of E_1 is key-independent. Therefore we can peel it off, and attack only 3 forward rounds and 3 inverse rounds. However, this requires us to use decryption queries, but we can't forge valid tags for an arbitrary ciphertext yet, so we use a decryption-misuse oracle.

5.1 Gathering Pairs

The first step is to collect pairs of plaintext blocks that have the same difference in the S_1 lane (after the permutation E_1). In order to construct such plaintexts, we build pairs of ciphertexts with specific differences and values. More precisely, we consider pairs of messages as follows (with the same associated data AD):

$$\widetilde{C}_x = (AD \parallel 0, 0^{120}, 0, 0, 0, 0, 0, 0, 0, 0, x) \qquad C_x[i] = \widetilde{C}_x[i] \oplus 2^{i-1} \cdot 3 \cdot L$$
$$\widetilde{C}'_x = (AD \parallel 1, 0^{120}, 1, 0, 0, 0, 0, 1, 1, 1, x) \qquad C'_x[i] = \widetilde{C}'_x[i] \oplus 2^{i-1} \cdot 3 \cdot L$$

where 0 and 1 are constant one-block values and x takes a different value for each pair. We decrypt these pairs and we collect the final plaintext blocks.

We now study the differences in the S_2 lane (before the permutation E_3). Using the definition of the TRANS operation as given in Fig. 3, S_2 is updated as follows during decryption:

$$S_2[i+1] = 2 \cdot S_2[i] \oplus E_3^{-1}(\widetilde{C}[i])$$

With the messages C_x and C'_x, we have

$$S_2[129] = 2^{129} S_2[0] \oplus (1 \oplus 2 \oplus \cdots \oplus 2^{128})A,$$
$$S'_2[129] = 2^{129} S_2[0] \oplus (1 \oplus 2 \oplus \cdots \oplus 2^{128})A \oplus (2^0 \oplus 2^1 \oplus 2^2 \oplus 2^7 \oplus 2^{128})(A \oplus B),$$

where $A = E_3^{-1}(0)$ and $B = E_3^{-1}(1)$. Since $2^{128} = 2^0 \oplus 2^1 \oplus 2^2 \oplus 2^7$, we have $S_2[129] = S'_2[129]$. This is shown in Fig. 6, where $\delta = A \oplus B$.

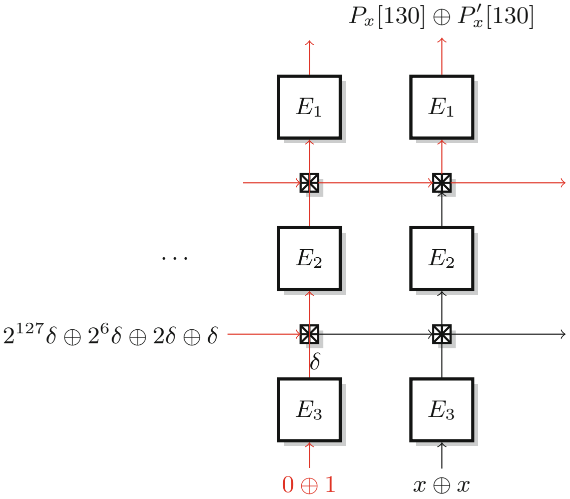

$P_x[130] \oplus P'_x[130]$
E_1
E_1
$\cdots$
E_2
E_2
$2^{127}\delta \oplus 2^6\delta \oplus 2\delta \oplus \delta$
δ
E_3
E_3
$0 \oplus 1$
$x \oplus x$

Fig. 6. Difference propagation in decryption. A red arrow means that there is a fixed unknown difference. A black arrow means that the difference is null.

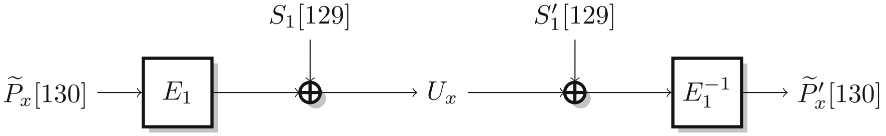
We now consider the final plaintext block given by the decryption oracle.

$$\begin{aligned}\widetilde{P}_x[130] &= P_x[130] \oplus 2^{130} \cdot L \\ &= E_1^{-1}\left(E_2^{-1}\left(E_3^{-1}(x) \oplus 3 \cdot S_2[129]\right) \oplus 3 \cdot S_1[129]\right) \\ \widetilde{P}'_x[130] &= P'_x[130] \oplus 2^{130} \cdot L \\ &= E_1^{-1}\left(E_2^{-1}\left(E_3^{-1}(x) \oplus 3 \cdot S_2[129]\right) \oplus 3 \cdot S'_1[129]\right)\end{aligned}$$

With $U_x = E_2^{-1}\left(E_3^{-1}(x) \oplus 3 \cdot S_2[129]\right)$, we have

$$\widetilde{P}_x[130] = E_1^{-1}(U_x) \oplus 3 \cdot S_1[129]$$
$$\widetilde{P}'_x[130] = E_1^{-1}(U_x) \oplus 3 \cdot S'_1[129]$$

Therefore, the pair $\widetilde{P}_x[130], \widetilde{P}'_x[130]$ can be seen as a plaintext/ciphertext pair for a cipher with 4 AES rounds, a middle key $S_1[129] \oplus S'_1[129]$, and 4 inverse AES rounds:



In addition, we can peel off the outer rounds since there is no whitening key in E_1.

5.2 Extracting the Key

We must now extract the key of a reduced cipher with 3 AES rounds, and 3 inverse AES rounds. First, we notice that the middle ShiftRow and MixColumn operations can be removed, if we transform the middle key. In a basic description, the operations in the middle are ShiftRow, MixColumn, AddKey, then XORing the constant $S_1[129] \oplus S'_1[129]$, AddKey, InverseMixColumn, and InverseShiftRow. Instead we move the (unknown) constant addition before ShiftRow, using the linearity of ShiftRow and MixColumn, so that ShiftRow, MixColumn and AddKey cancel out with AddKey, InverseMixColumn and InverseShiftRow. We denote the addition of the modified constant as AddDeltaS, and its value as $\delta_S = \mathsf{InverseShiftRow}(\mathsf{InverseMixColumn}(S_1[129] \oplus S'_1[129]))$. The middle rounds are then reduced to byte-wise operations: AddRoundKey, SubByte, AddDeltaS, InverseSubByte, AddRoundKey. This can be seen as a key-dependent Sbox layer. These transformations are summarised on Fig. 7.

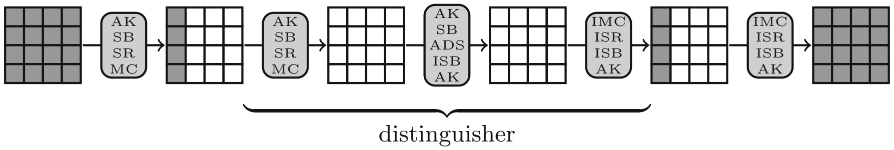


Fig. 7. Reduced cipher composed of 3 AES rounds, the addition of δ_S and 3 inverse AES rounds. The distinguisher covers the middle part of this cipher.

The first step of our attack is to guess a diagonal of the first round key, which allows to compute a column after the first round and before the last round. Next we focus on the middle rounds. The middle rounds have only one MixColumn operation, and one InverseMixColumn without byte shuffling in between. Therefore it can be seen as four parallel 32-bit functions, acting on each diagonal (similar to the Super-SBox technique [4]). Note that if the key guess is wrong, the resulting function can not be decomposed into 4 parallel functions. For each function, 1 input byte and 1 output byte are known. We describe below two different distinguishers for the middle rounds, that lead to key recovery attack with similar complexities. The first one is based on a rare event that can easily be detected, the second one relies on the detection of a statistical bias in the generic case.

Collision-based Distinguisher. For our first distinguisher, we focus on the constant δ_S. We only know that δ_S is non-zero on the full state. Considering that it is distributed uniformly on non-zero constants, it cancels one of the diagonals[3] with probability $(2^{96}-1)/(2^{128}-1) \approx 2^{-32}$. Then, an average of 2^{30} different choices of AD are necessary to reach a value of δ_S that cancels on one of the diagonals. Let us consider that it occurs on the first diagonal (w.l.o.g.), which contains bytes $0, 7, 10, 13$. Then, the value of these bytes collide before and after the AddDeltaS operation. Then, the values of the first column of the block (bytes $0, 1, 2, 3$) are not affected by the middle rounds. If we continue the decryption process towards both ends of the modified version of the AES, the collision passes through the InverseMixColumn operation. After undoing the ShiftRow, SubByte and textsfAddKey operation, we notice that the values of bytes $0, 5, 10, 15$ are equal at the beginning and at the end of the middle part of the cipher.

For each choice of AD, we then generate 3 (plaintext-ciphertext) pairs $(\widetilde{P}_x, \widetilde{P}'_x)$ for 3 values of x. Then, we proceed as follows.

In each of the 2^{30} sets, we guess separately the 32 bits on each of the 4 anti-diagonals[4] of the first round key. This enables to compute one full column of the state before and after the middle rounds, for each value of x. For each byte b_i in this column, we store a list L_i of the key values such that the input byte and the output byte of the middle rounds are equal for each x.

Then we consider the first diagonal before and after the middle rounds. It contains bytes 0, 5, 10 and 15 of the block. Remember that the diagonals contain the inputs and outputs of 4 independent functions F_i. From the 4 lists of partial keys $L_j, j = 0, 5, 10, 15$, we can build all the keys such that the input of F_i collides with the output for each value of x. Using the known plaintexts and ciphertexts for the full cipher, we can try all these keys as candidates. Then, we repeat the whole process with the other three diagonals.

We now explain why this attack works.

Filtering Keys. Following the analysis above, the right key can be decomposed into 4 partial keys covering each diagonal of the block. If δ_S cancels on one of the columns, then the partial values of the right key will appear on the four corresponding lists L_i, and the full key will be among the combination of elements of the four lists. Therefore, the right key will be detected by our algorithm.

False Positives. For each wrong partial 32-bit key, it is stored in the corresponding list L_j if the input and output of F_i collide on byte j, for each of the 3 values of x. This occurs with probability 2^{-24}, if we consider the input and output

[3] defined as the images of columns by the ShiftRow operation.

[4] defined as the pre-images of columns by the ShiftRow operation.

byte computed for F_i as independent. Therefore, we have on average one false positive of each of the 4 diagonals of the key. Considering that the number of false positives are independent for each diagonal of the key, there are on average $(2^8)^4 = 2^{32}$ keys to try, for each of the 4 diagonals and each of the 2^{30} sets of values. The expected number of key candidates is marginally increased to $(2^8 + 1)^4 \approx 2^{32}$ when the difference δ_S cancels on the diagonal, as each set of partial keys at least contains the right key.
$\delta_S \neq 0$ on column i. As above, each wrong key guess is stored with probability 2^{-24}, which leads to $(2^8)^4$ false positives on average, that are discarded by exhaustive search.

Summary of the Attack. We focus on the key recovery attack on the mask-less version of Marble. In the decryption-misuse setting, it requires the decryption of 6×2^{30} ciphertexts composed of 130 blocks of plaintext and 1 block of associated data, which correspond to 2^{30} sets of 3 pairs. To build the lists of partial keys, one has to perform $6 \times 1/4$ of an AES round for each partial key guess, leading to a total of 3×2^{31} AES rounds, for each set and each diagonal. The average complexity of this step for the full attack is then 3×2^{63} AES rounds. The most time-consuming part of the attack is the exhaustive search among the remaining candidates, which requires 2^{64} AES encryptions on average (2^{32} per column and per set).

Linear Cryptanalysis. The method described in Sect. 5.1 leads to the knowledge of plaintext-ciphertext pairs for a cipher that consists of 3 AES rounds, a key addition and 3 inverse AES rounds. The adversary therefore targets a cipher with a reduced number of rounds, in a known plaintext setting. Using linear cryptanalysis therefore seems a natural idea. As shown above, one can guess 4 key bytes, which leads to the knowledge of 4 input and 4 output bytes of the inner 4 rounds of this cipher.

In [9], Keliher and Sui demonstrate that the maximum expected linear probability over 2 AES rounds is about $LP \approx 1.638 \times 2^{-28}$. In our case, we can concatenate a linear trail with its inverse. When averaging over the possible values of the key and of the intermediate difference δ_S, the maximum expected probability for a 4-round characteristic would be about $LP^2 \approx 1.342 \times 2^{-55}$. This number also gives an estimation of the amount of data required for the attack to work. Even by taking into account the possible bias due to the linear hull effect, the complexity of the linear attack is expected to be far higher than the one suggested by the experiments below.

A refinement of the linear attack consists in noticing that between the two middle rounds, each byte of the block is affected only by a key byte and a byte of δ_S, but not by other bytes of the block. Therefore, the two middle Sbox layers could be expressed as one layer of 8-bit key-dependent Sboxes, leading to trails with 6 active Sboxes instead of 10. Nevertheless, the best linear trail would then depend on the unknown value of δ_S, which would make it hard to exploit. Instead, we use the following statistical distinguisher.

Statistical Distinguisher. Intuitively, if we have many partial input/output pairs, we should detect some correlation between the inputs and output. Indeed, when the key guess is wrong, the function composing the distinguisher behaves as a 128-bit permutation instead of the parallel application of four 32-bit functions. Hence, the input and output bytes are less correlated. We focus on a property that does not require to know in advance which values are correlated, and works for any function based on (four 32-bit) parallel permutations.

For each known plaintext/ciphertext, we partially encrypt/decrypt one round on a specific diagonal and we denote one known input/output byte of the distinguisher by (α, β) respectively. It is possible to take into account the four known input/output pairs, but the distinguisher presented below works with only one position and is easier to explain. We use 2^{16} counters $c_{\alpha,\beta}$ to count how many times each pair (α, β) occurs with D available data. If the key guess is correct, there should be some correlation between α and β, which results in a higher value for some counters (and lower values for the other counters). In order to detect this effect, we compute the sample variance s^2 of the 2^{16} counters:

$$s^2 = 2^{-16} \sum_{\alpha,\beta} (c_{\alpha,\beta} - \overline{c})^2, \quad \text{where } \overline{c} = 2^{-16} \sum_{\alpha,\beta} c_{\alpha,\beta}.$$

We expect that s^2 is higher when the key guess is correct, because of the correlation between α and β. For a wrong key guess, the computation between α and β can not be decomposed into 4 parallel functions, and this correlation should vanish. The resulting attack is described by Algorithm 3.

Algorithm 3. Recover the key of a reduced AES (3 direct rounds and 3 inverse rounds)

```
Input: Plaintext/ciphertext pairs $(P, C)$
  for $0 \leq K < 2^{32}$ do                                   ▷ Partial key guess
      Initialize $c_{\alpha,\beta} = 0$
      for all $(P, C)$ do
          Compute $\alpha, \beta$
          $c_{\alpha,\beta} \leftarrow c_{\alpha,\beta} + 1$
      end for
      $\overline{c} \leftarrow 2^{-16} \sum_{\alpha,\beta} c_{\alpha,\beta}$
      $s^2[K] \leftarrow 2^{-16} \sum_{\alpha,\beta} (c_{\alpha,\beta} - \overline{c})^2$
  end for
  return $\arg\max_K s^2[K]$
```

In order to analyze this algorithm, we model the counters using random variables $C_{\alpha,\beta}$, and the sample variance as S^2 for a wrong key guess, and S_*^2 for the right key. Our goal is to show that when D is large enough, we have $\Pr[S^2 > S_*^2]$ negligible, *i.e.* the correct key is ranked first.

Wrong key Guess. We know that starting from α, if we revert the initial round with the wrong key, then compute three rounds forward with the correct keys, add δ_S, compute three round backwards with the correct keys, and finally one round forward with the wrong key, we reach a state with β. Therefore, α and β are partial inputs/outputs of a 128-bit permutation.

If we model this function by a random 128-bit permutation, the number of data matching a pair (α, β), in images and pre-images of this 128-bit function, follows an hypergeometric distribution. Indeed, for each input which first byte has value α, the output is drawn uniformly without replacement among all the possible outputs of the function. The success is determined by whether the first byte of the output equals β. The number of draws is 2^{120}, and there are 2^{120} success cases among 2^{128} possible values.

(α, β) occurs with D data, knowing that the probability of success is $p = 2^{120}/2^{128} = 2^{-8}$. Let us call this variable $X_{\alpha,\beta}$. Hence we have

$$\begin{aligned} \mathrm{E}[X_{\alpha,\beta}] &= \qquad\qquad (2^{120})^2/2^{128} \qquad\qquad = 2^{112} \\ \mathrm{Var}[X_{\alpha,\beta}] &= (2^{120})^2/2^{128} \times (1-2^{-8})^2/(1-2^{-128}) \approx 2^{112} - 2^{105}. \end{aligned}$$

Next we study $Y_{\alpha,u}$ the number of times each value α, u is reached with D samples, for each possible value u of the remaining 15 bytes of the input of F. The $Y_{\alpha,u}$ follow a multinomial distribution, with:

$$\begin{aligned} \mathrm{E}[Y_{\alpha,u}] &= 2^{-128}D, \\ \mathrm{Var}[Y_{\alpha,u}] &= 2^{-128}(1-2^{-128})D, \\ \mathrm{Var}[Y_{\alpha,u}, Y_{\alpha',u'}] &= -2^{-256}D. \end{aligned}$$

Let us denote by $S_{\alpha,\beta}$ the set of values u such that $F(\alpha, u) = (\beta, v)$ for some v. It contains exactly $X_{\alpha,\beta}$ elements. The counters $C_{\alpha,\beta}$ can then be expressed as

$$C_{\alpha,\beta} = \sum_{u \in S_{\alpha,\beta}} Y_{\alpha,u}.$$

The variables $Y_{\alpha,u}$ all follow the same distribution. From the law of total variance, we have:

$$\mathrm{Var}[C_{\alpha,\beta}] = \mathrm{E}_{X_{\alpha,\beta}}\left(\mathrm{Var}\left[\sum_{u \in S_{\alpha,\beta}} Y_{\alpha,u} | X_{\alpha,\beta}\right]\right) + \mathrm{Var}_{X_{\alpha,\beta}}\left(\mathrm{E}\left[\sum_{u \in S_{\alpha,\beta}} Y_{\alpha,u} | X_{\alpha,\beta}\right]\right)$$

After expanding the sums and reordering the terms to make variances and covariances of the $Y_{\alpha,u}$ appear, we get:

$$\begin{aligned} \mathrm{Var}[C_{\alpha,\beta}] &= \mathrm{E}\left[X_{\alpha,\beta} \mathrm{Var}[Y_{\alpha,u}] + (X_{\alpha,\beta}^2 - X_{\alpha,\beta}) \mathrm{Var}[Y_{\alpha,u}, Y_{\alpha,u'}]\right] + \mathrm{Var}\left[X_{\alpha,\beta} \mathrm{E}[Y_{\alpha,u}]\right] \\ &= \mathrm{E}(X_{\alpha,\beta}) \mathrm{Var}(Y_{\alpha,u}) + \mathrm{E}[X_{\alpha,\beta}^2 - X_{\alpha,\beta}] \mathrm{Var}(Y_{\alpha,u}, Y_{\alpha,u'}) + \mathrm{E}[Y_{\alpha,u}]^2 \mathrm{Var}[X_{\alpha,\beta}] \\ &= \mathrm{E}(X_{\alpha,\beta}) \mathrm{Var}(Y_{\alpha,u}) + \big(\mathrm{Var}[X_{\alpha,\beta}] + \mathrm{E}[X_{\alpha,\beta}]^2 - \mathrm{E}[X_{\alpha,\beta}]\big) \mathrm{Var}(Y_{\alpha,u}, Y_{\alpha,u'}) \\ &\quad + \mathrm{E}[Y_{\alpha,u}]^2 \mathrm{Var}[X_{\alpha,\beta}] \end{aligned}$$

We have numeric expressions for each term of this expression, therefore we can compute the following approximation:

$$\mathrm{Var}[C_{\alpha,\beta}] \approx 2^{-16}D + 2^{-144}D^2.$$

Correct Key. Let us now assume that the key guess is correct, *i.e.* the pairs (α, β) are valid partial input/output but of a 32-bit function this time. We can then reapply the above analysis by adjusting the parameters to fit the 32-bit function. In this case, $X_{\alpha,\beta}$ denotes the number of partial values of the data matching the pair (α, β) in the right column. In that case, we have an hypergeometric distribution with 2^{24} draws without replacement from a set of 2^{32} values, among which 2^{24} define a success event.

Therefore, we have

$$\begin{aligned} \mathrm{E}[X_{\alpha,\beta}] &= (2^{24})^2/2^{32} &&= 2^{16} \\ \mathrm{Var}[X_{\alpha,\beta}] &= (2^{24})^2/2^{32} \times (1-2^{-8})^2/(1-2^{-32}) &&\approx 2^{16} - 2^9. \end{aligned}$$

Similarly, we can define variables $Y_{\alpha,u}$ as the number of times a given input of the 32-bit function F is reached among D samples, drawn uniformly. As in the previous case, the $Y_{\alpha,u}$ follow a multinomial distribution, with:

$$\begin{aligned} \mathrm{E}[Y_{\alpha,u}] &= 2^{-32}D, \\ \mathrm{Var}[Y_{\alpha,u}] &= 2^{-32}(1-2^{-32})D, \\ \mathrm{Var}[Y_{\alpha,u}, Y_{\alpha',u'}] &= -2^{-64}D. \end{aligned}$$

The same formula can be used to compute the variance of the counters $C_{\alpha,\beta}$. We get approximately:

$$\mathrm{Var}[C_{\alpha,\beta}] \approx 2^{-16}D + 2^{-48}D^2.$$

Distinguisher. We obtain an efficient distinguisher with $D = 2^{32}$: for a wrong key guess, the variance of the counter is about 2^{16}, but it is about 2^{17} for the right key. In order to evaluate the probability that the correct key is ranked first, we must evaluate how far the sample variance s^2 is from the true variance $\mathrm{Var}[C_{\alpha,\beta}]$.

For a wrong key guess, if we use a single counter and repeat the experiment with 2^{16} independent sets of D plaintexts, each counter $C_{\alpha,\beta}$ can be approximated by a binomial distribution with parameters D and $p = 2^{-16}$. If we approximate them as a normal distribution with parameters $\mu = 2^{-16}D$ and $\sigma = \sqrt{2^{-16}(1-2^{16})D}$, we know that the distribution of the sample variance S^2 for a wrong key guess follows a χ^2 distribution [10, Proposition 2.11]:

$$S^2 \sim \frac{\sigma^2}{(n-1)}\chi^2_{n-1} \sim 2^{-32}D\chi^2_{2^{16}-1}$$

In particular, we have $\Pr[S^2 > 2^{16} + 2^{12}] < 2^{-90}$, therefore we don't expect that the sample variance of a wrong key is above $2^{16} + 2^{12}$. In practice, we use a single set of D plaintexts, and we evaluate the sample variance of the 2^{16} counters; our experiments show that the distribution is close to a χ^2 distribution (the maximum value of s^2 with 2^{16} samples was $2^{16} + 1420$).

For the right key, we don't have an analytic expression of the distribution of the sample variance, but we can perform experiments. Our experiments show that with very high probability $S_*^2 > 2^{16} + 2^{12}$, as seen in Fig. 8. Of our 2^{16} experiments, the minimum value of s_*^2 was $102795 \approx 2^{16} + 2^{15}$. Using $D = 2^{32}$, we have a large margin and we expect the attack to work with significantly less data, but recovering L will be the bottleneck of the attack.

While this attack does not use any property of the parallel 32-bit function, we expect that it can be improved in the specific case of AES rounds. In particular, we notice a small peak around 3×2^{16} in Fig. 8, which is due to zero bytes in δ_s, and it should be possible to take advantage of this.

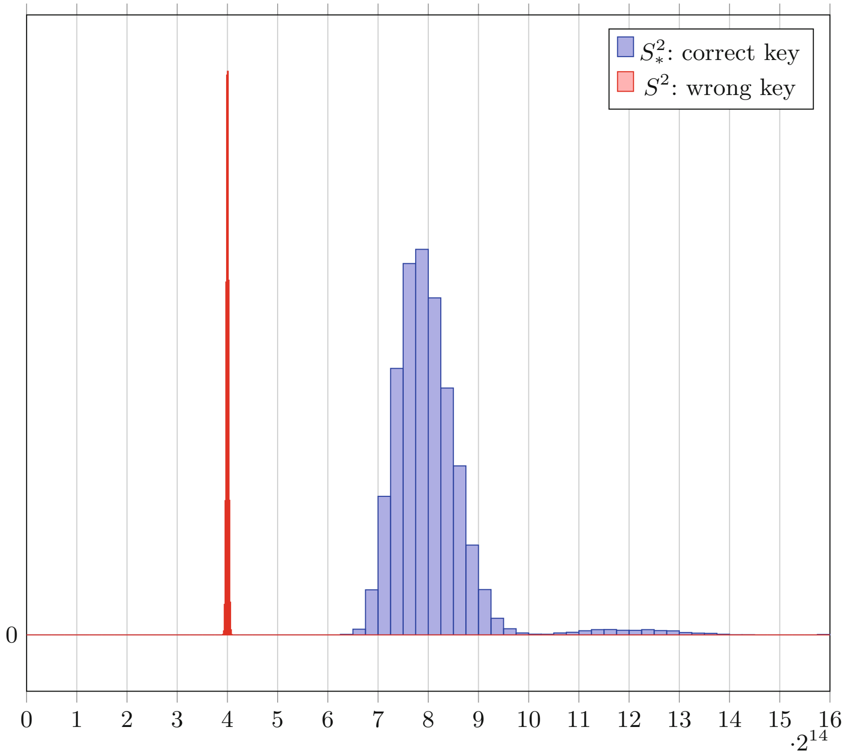


Fig. 8. Experimental results: distribution of the sample variance S^2 and S_*^2 with $D = 2^{32}$ (2^{16} experiments with random keys).

6 Conclusion

Our results show that collision attacks can have a strong impact on the security of authenticated encryption schemes. It seems that extracting the whitening key using collisions is possible in many OCB-based designs, and this can sometimes be extended into a full key-recovery attack.

On AEZ, we show how to recover the whitening key, and we point out that the key derivation of AEZ v2 and v3 has the unfortunate property that the master key can easily be recovered from the whitening key. This allows a complete break after the birthday bound. Even with a limit on the amount of data processed with a single key, this still gives an attack with a higher success probability than generic attacks. While this does not violate the security proof of AEZ, we believe it would be better to avoid this property.

Our results on Marble show that it does not provide the security features claimed by the author, *i.e.* beyond birthday bound security and decryption-misuse resistance. We note that Marble still offers a stronger security than many CAESAR candidates and classical modes of operations when using nonces (or unique AD). Once usage requirements are relaxed, our results also show that the security of Marble is similar to the security of other misuse resistant CAESAR candidates (*e.g.* APE-128, POET, AEZ, Minalpher) but it collapses badly after the birthday bound under a decryption-misuse setting, even leading to a full key recovery.

It seems that adding one extra operation on the state between the processing of the associated data and of the message would avoid our attacks, but a thorough analysis would be necessary to ensure that the resulting construction is secure. As our attack heavily relies on the fact that S_1 and S_2 keep the same values on two different plaintexts, one could xor a constant block (for example, 16 bytes encoding the byte positions in the block, $(0, 1, \ldots, 15)$) on S_1 and S_2 after processing the associated data.

In addition, our key-recovery attack of Marble suggests that 4 AES rounds in the E functions are insufficient if the adversary can find a shortcut to target two E functions instead of three. In particular, this suggest that a deeper investigation of the security of AEZ with a modified key schedule would be interesting.

Up to our knowledge, the statistical distinguisher presented to recover the encryption key of a reduced-round AES, has never been used before. Although it permits to attack few rounds, it seems that it is more efficient than a classical linear attack. We believe that it is sufficient enough for this kind of distinguisher to benefit from further research.

References

1. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and authenticated online ciphers. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 424–443. Springer, Heidelberg (2013)
2. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002)
3. Ferguson, N.: Collision attacks on OCB. Comments to NIST (2002). http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/General_Comments/papers/Ferguson.pdf

4. Gilbert, H., Peyrin, T.: Super-Sbox cryptanalysis: improved attacks for AES-like permutations. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 365–383. Springer, Heidelberg (2010)
5. Guo, J.: Marble Specification Version 1.0. Submission to the CAESAR competition, March 2014
6. Guo, J.: Marble Specification version 1.2 (January 2015), posted on the `crypto-competition` mailing list
7. Hoang, V.T., Krovetz, T., Rogaway, P.: AEZ v3: Authenticated-Encryption by Enciphering. In: DIAC 2014: Directions in Authenticated Ciphers, Santa Barbara, August 2014
8. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 15–44. Springer, Heidelberg (2015)
9. Keliher, L., Sui, J.: Exact maximum expected differential and linear probability for two-round advanced encryption standard. IET Inf. Secur. **1**(2), 53–57 (2007)
10. Knight, K.: Mathematical Statistics. Chapman & Hall, New York (1999)
11. Lee, C.-H., Kim, J.-S., Sung, J., Hong, S.H., Lee, S.-J.: Forgery and key recovery attacks on PMAC and mitchell's TMAC variant. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 421–431. Springer, Heidelberg (2006)
12. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. J. Cryptology **24**(3), 588–613 (2011)
13. Lu, J.: On the Security of the COPA and Marble Authenticated Encryption Algorithms against (Almost) Universal Forgery Attack, February 2015
14. NIST: Advanced Encryption Standard (AES) (November 2001), federal Information Processing Standards Publication FIPS 197
15. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer, Heidelberg (2004)
16. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: Reiter, M.K., Samarati, P. (eds.) CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6–8, 2001. pp. 196–205. ACM (2001)
17. Sasaki, Y.: Universal Forgery on POET and Ciphertext Distinguisher on POET and Marble (July 2014), posted on the `crypto-competition` mailing list