New Birthday Attacks on Some MACs Based on Block Ciphers *

Zheng Yuan[1,2**], Wei Wang[3], Keting Jia[3],
Guangwu Xu[4], and Xiaoyun Wang[***1,3]

[1] Institute for Advanced Study, Tsinghua University, Beijing 100084, China
{zyuan, xiaoyunwang}@mail.tsinghua.edu.cn
[2] Beijing University of Posts and Telecommunications, Beijing 100876, China
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China
{weiwangsdu, ktjia}@mail.sdu.edu.cn
[4] Department of Electrical Engineering and Computer Science, University of Wisconsin-Milwaukee, USA
gxu4uwm@uwm.edu

Abstract. This paper develops several new techniques of cryptanalyzing MACs based on block ciphers, and is divided into two parts.
The first part presents new distinguishers of the MAC construction ALRED and its specific instance ALPHA-MAC based on AES. For the ALRED construction, we first describe a general distinguishing attack which leads to a forgery attack directly with the complexity of the birthday attack. A 2-round collision differential path of ALPHA-MAC is adopted to construct a new distinguisher with about $2^{65.5}$ chosen messages and $2^{65.5}$ queries. One of the most important results is to use this new distinguisher to recover the internal state, which is an equivalent subkey of ALPHA-MAC. Moreover, our distinguisher on ALRED construction can be applied to the MACs based on CBC and CFB encryption modes.
The second part describes the first impossible differential attack on MACs-PELICAN, MT-MAC-AES and PC-MAC-AES. Using the birthday attack, enough message pairs that produce the inner near-collision with some specific differences are detected, then the impossible differential attack on 4-round AES to the above mentioned MACs is performed. For PELICAN, our attack recovers its internal state, which is an equivalent subkey. For MT-MAC-AES, the attack turns out to be a subkey recovery attack directly. The complexity of the two attacks is $2^{85.5}$ chosen messages and $2^{85.5}$ queries. For PC-MAC-AES, we recover its 256-bit key with $2^{85.5}$ chosen messages and 2^{128} queries.

Key words: MAC, Birthday attack, Distinguishing attack, Forgery attack, Impossible differential cryptanalysis, AES

* Supported by the National Natural Science Foundation of China (NSFC Grant No. 60525201 and No.90604036) and 973 Project (No.2007CB807902).
** Supported by China Postdoctoral Science Foundation Funded Project (No. 20080430423).
*** To whom correspondence should be addressed.

Part I Distinguishing and Forgery Attacks on ALRED and Its AES-based Instance ALPHA-MAC [1]

1 Introduction to Part I

Message Authentication Code (MAC) is a fixed length information used to ensure data integrity and authenticity, and is widely used in network and security protocols, such as IPsec, SNMP, and SSL/TLS. A MAC algorithm takes a secret key and a message of arbitrary length as input, and outputs a short digest. MAC algorithms have been constructed using various approaches, for example, CBC-MAC [11], OMAC [12], TMAC [14], HMAC/NMAC [2], etc.

The MAC construction ALRED was introduced by Daemen and Rijmen [8]. ALRED is an iterative MAC construction using reduced block ciphers as iteration functions. The secret key, which is used as the key of the block cipher, is applied in the initialization and the finalization, respectively. The internal state is updated by consecutive injections of message blocks. ALPHA-MAC is an efficient instance of ALRED based on AES [7]. Since AES has been widely used in practice, ALPHA-MAC can be easily implemented. For the performance, ALPHA-MAC is 2.5 times faster than the popular CBC-MAC with AES.

It was proved that the ALRED construction is as strong as the underlying block cipher with respect to key recovery attacks and any forgery attacks not involving inner collisions [8]. Moreover, for ALPHA-MAC, any colliding messages of the same size have to be at least 5 blocks long. Recently, Huang et al. exploited the algebraic properties of the AES, constructed internal collisions, and found second preimages for ALPHA-MAC, under the assumption that a key or an internal state is known [10]. Biryukov et al. proposed a side-channel collision attack on ALPHA-MAC which recovered its internal state, and mounted a selective forgery attack [5].

The main contribution of this part is to present novel distinguishing attacks on the ALRED construction and ALPHA-MAC, which lead to forgery attacks directly. More importantly, the distinguishing attack on ALPHA-MAC can be applied to recover the internal state, and results in a second preimage attack.

There are two kinds of distinguishing attacks on MACs. Preneel and van Oorschot introduced a general distinguishing attack to identify iterated MACs from a random function [17]. Using the birthday paradox, the adversary can detect the internal collision by appending the same one-block message. Another kind of attacks was suggested by Kim et al., which distinguishes the cryptographic primitive embedded in a MAC construction from a random function [13]. Recently, new techniques to identify the underlying hash functions of MACs were proposed [19,20]. For example, distinguishing attacks on HMAC/NMAC-MD5 and MD5-MAC were proposed in [20]. The inner near-collisions are used in the distinguisher which reveals more information than inner collisions. In the same work, they were able to recover partial subkey of the MD5-MAC as well.

[1] by Zheng Yuan, Keting Jia, Wei Wang, and Xiaoyun Wang. See [21] for more details.

Inspired by Wang et al.'s work [19,20], we propose a new idea to detect the inner near-collision with some specific differences, which can be used to identify the cryptographic primitives embedded in MACs. Build upon this idea, two distinguishing attacks on ALRED construction and ALPHA-MAC are presented in this part. We first describe a distinguishing attack on the ALRED construction. This attack is based on the birthday attack [22] which asserts that there exists a collision differential path with some specific differences. This is an inner near-collision which can be recognized with probability 1 by appending another message pair with the same difference. Next, we present a new distinguisher for ALPHA-MAC based on a 2-round collision differential path. By combining with the specific differences in the 2-round collision differential path, we then explore a series of tricks to recover the internal state, which is an equivalent subkey. With the recovered subkey, we can obtain the second preimage of ALPHA-MAC for any given message M and its MAC value. The complexity of all the attacks of ALPHA-MAC is $2^{65.5}$ MAC queries and $2^{65.5}$ chosen messages with a success rate of 0.63. Moreover, the distinguishing attack on the ALRED construction can be applied to the MACs based on CBC and CFB encryption modes.

2 Backgrounds and Notations

In this section, we define some notations, and give brief descriptions of the ALRED construction and ALPHA-MAC.

2.1 Notations

x_i	:	the i-th message word
y_i	:	the state after the i-th iteration
C	:	the output of MAC algorithm
ΔA	:	the XOR difference of A and A'
n	:	the length of the state
l_w	:	the length of the message word
l_m	:	the length of the MAC output
$M\|N$	:	the concatenation of M and N
$\|x\|$	:	the length of a bit string x
$\lceil x \rceil$	:	the smallest integer not less than x
10^j	:	the $(j+1)$-bit sequence $(1\underbrace{00\cdots0}_{j})$

2.2 ALRED Construction

The MAC construction ALRED [8] is based on a reduced block cipher. The length of the secret key equals to that of the underlying block cipher, and the message length is a multiple of l_w bits.

Given a message $M = (x_1, x_2, \ldots, x_t)$, the ALRED construction is as follows.

1. Apply the full block cipher to the state of all-zero block, i. e., $y_0 = \mathrm{Enc}_K(0)$.

2. Perform the following iteration function f for each message word: (a) *Injection Layout*: Map the bits of the message word to an *injection input* that has the same dimensions as a sequence of r-round subkeys of the block cipher. (b) Apply a sequence of r-round block cipher function to the state, and replace the round subkeys with the injection input, i. e., $y_i = f(y_{i-1}, x_i)$, for $i = 1, 2, \ldots, t$.
3. Apply the full block cipher to the state y_t, and truncate the first l_m bits of the state as the output. The final output $C = Trunc(\mathrm{Enc}_K(y_t))$.

2.3 Alpha-MAC Algorithm

Alpha-MAC [8] is a specific instance of the Alred construction with 1-round AES as its iteration function, where $l_w = 32$. Similar to AES, the Alpha-MAC supports key length of 128, 192 or 256 bits.

The message padding method is to append a single bit '1' followed by the minimum bits of '0' such that the length of the result is a multiple of 32. For AES-128, the Injection Layout places the 4 bytes of each message word $x_i = (x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3})$ into a 4×4 array with the form:

$$\begin{pmatrix} x_{i,0} & 0 & x_{i,1} & 0 \\ 0 & 0 & 0 & 0 \\ x_{i,2} & 0 & x_{i,3} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

which acts as the corresponding 128-bit round subkey. The Alpha-MAC round function consists of the four basic transformations of AES in sequence: AddRoundKey (AK), SubBytes (SB), ShiftRows (SR), and MixColumns (MC) [7].

2.4 Related Works

Our work is related to two types of attacks in the literature. They are the general distinguishing-R attack on all iterated MACs proposed by Preneel and van Oorschot [17], and the distinguishing-H attack on HMAC/NMAC-MD5 and MD5-MAC introduced by Wang et al. [20].

Preneel et al. proposed a general forgery attack on iterated MACs by the birthday paradox, which is applicable to all iterated MACs, such as CBC-MAC. Their technique detects all the colliding pairs among $2^{(n+1)/2}$ known text-MAC pairs by the birthday paradox, where n is the bit length of the chaining variable. For each searched collision, i. e., $MAC(K, M) = MAC(K, M')$, a one-block message N is appended to identify whether it is an internal collision by comparing $MAC(K, M||N)$ and $MAC(K, M'||N)$. If an internal collision is found, then a forgery is created since the MACs of $M\|N'$ and $M'\|N'$ are the same. However, this method cannot be used to distinguish the cryptographic primitives embedded in the MAC.

Wang et al. introduced another interesting idea which can distinguish HMAC/NMAC-MD5 without the related-key setting. They also implemented a partial

key recovery attack on MD5-MAC. The main strategy of the distinguishing attack is as follows: The adversary first collects enough two-block message pairs $(M\|N, M'\|N)$ to guarantee the appearance of an expected internal near-collision in the first iteration, then detects such a near-collision by changing the second block with enough messages N'. Once the expected inner near-collision is identified, the MAC is known to be based on MD5. The core of the attack is to detect an inner near-collision instead of a collision.

3 Distinguishing and Forgery Attacks on MAC Construction ALRED

This section presents distinguishing and forgery attacks on ALRED construction. Enlightened by the idea of Wang et al., we can detect a proper output difference as an inner near-collision by the birthday paradox. When the MAC construction is ALRED rather than a random function, this kind of inner near-collision can be detected with probability 1 by substituting the last different message pair with another message pair having the same difference. Based on this detected inner near-collision, a forgery attack can be constructed immediately.

3.1 Distinguishing Attack on ALRED Construction

The iteration part of ALRED construction is based on the r-round block cipher, where the r-round subkeys are substituted by the injection input. The core of our distinguisher is to detect Δy_{j-1}, which is the output difference of $(j-1)$-th iteration. According to the operation between the injection input and the state involved in the iteration function f, the message word difference Δx_j may extinguish Δy_{j-1}, and lead to a collision at the final output. The form of the difference depends on the operation between the injection input and the state; e. g., for ALRED based on IDEA or RC6, the operation is modular addition, while for some others, it is XOR. Without loss of generality, we neglect Injection Layout map, and only consider the round number $r = 1$ and the XOR operation between the message word and the state.

As shown in Fig. 1, there is an inner near-collision after round $(j-1)$. When $\Delta x_j = \Delta y_{j-1}$, there will be an internal collision $x_j \oplus y_{j-1} = x'_j \oplus y'_{j-1}$, which can be propagated to the output. If the construction is ALRED, we replace the (x_j, x'_j) with a different $(\overline{x_j}, \overline{x'_j})$, where $\Delta\overline{x_j} = \Delta x_j$, the collision still occurs. According to this property, the distinguisher is constructed as follows:

1. Randomly choose a structure $T = \{M^i | M^i = (x_1^i, x_2^i, \ldots, x_t^i)\}$ composed of $2^{(n+1)/2}$ different messages, and query their corresponding MAC values C^i.
2. By the birthday paradox, search a collision $C^a = C^b$, the corresponding messages are M^a and M^b.
3. Counting backwards, suppose that (x_j^a, x_j^b) is the first unmatched pairs of words in (M^a, M^b), i. e., $x_j^a \neq x_j^b$, $M^a = (x_1^a, \ldots, x_j^a, x_{j+1}, \ldots, x_t)$, and $M^b = (x_1^b, \ldots, x_j^b, , x_{j+1}, \ldots, x_t)$. Replace (x_j^a, x_j^b) with another $(\overline{x_j^a}, \overline{x_j^b})$,

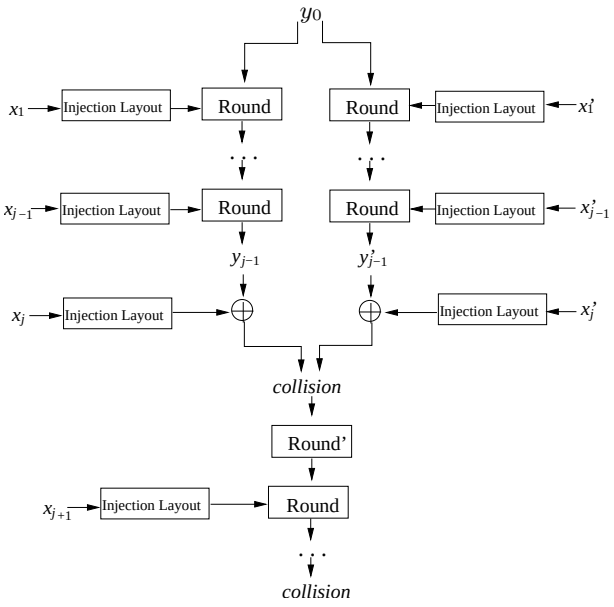

Fig. 1. The Distinguisher with XOR Operation

where $\overline{x_j^a} \oplus \overline{x_j^b} = x_j^a \oplus x_j^b$. Query the MACs with $(\overline{M^a}, \overline{M^b})$, where $\overline{M^a} = (x_1^a, \ldots, x_{j-1}^a, \overline{x_j^a})$ and $\overline{M^b} = (x_1^b, \ldots, x_{j-1}^b, \overline{x_j^b})$.

- If $\overline{C^a} = \overline{C^b}$, we conclude that the MAC is ALRED construction.
- Otherwise, it is a random function.

Note that t should be large enough to guarantee the existence of an inner near-collision at round $(j-1)$, where $t \geq 6$.

This attack requires about $2^{(n+1)/2}$ chosen messages, and works with probability 0.63 by the birthday paradox.

Remark 1. For MACs based on the block ciphers with $r \geq 2$, such as CBC-MAC, OMAC, TMAC, etc., the iteration function is $H_i = f(H_{i-1}, x_i) = E_K(H_{i-1} \oplus x_i)$. Therefore, with a little modification, the above attack is applied to these situations. Besides, our method also works for the MACs based on CFB mode, i.e., $H_i = f(H_{i-1}, x_i) = E_K(H_{i-1}) \oplus x_i$.

3.2 Forgery Attack on ALRED Construction

Once the inner near-collision is identified, we can replace message words by another pair with the same difference to achieve a new collision pair. Hence, the forgery attack is easily realized with the same complexity and success rate as the distinguishing attack. To be more specific, let (M^a, M^b) be the colliding pair

detected in the above distinguishing attack. We query the MAC oracle with $\widetilde{M^a}$, where $\widetilde{M^a} = (x_1^a, \ldots, x_{j-1}^a, \widetilde{x_j^a}, s)$, and s is an arbitrary message string. We can get a MAC forgery of the message $\widetilde{M^b} = (x_1^b, \ldots, x_{j-1}^b, \widetilde{x_j^a} \oplus \Delta x_j, s)$.

4 Recovering the Equivalent Subkey of Alpha-MAC

It is remarked that the above distinguisher can be utilized to distinguish the Alpha-MAC from a random function. However, we introduce a new distinguisher in this section, where the expected collision implies an inner near-collision with some specific differences. With this distinguisher, we can recover an internal state, which is an equivalent subkey, i. e., the state y_0 (See Fig. 1).

4.1 Some Important Properties of Alpha-MAC

This section introduces a 2-round collision differential path of Alpha-MAC, and summarizes some useful facts based on it. The 2-round differential path will be used to recover the internal state in Section 4.3.

For $i = 1, \ldots, t$, denote

$$\begin{pmatrix} y_{i-1,0} & y_{i-1,1} & y_{i-1,2} & y_{i-1,3} \\ y_{i-1,4} & y_{i-1,5} & y_{i-1,6} & y_{i-1,7} \\ y_{i-1,8} & y_{i-1,9} & y_{i-1,10} & y_{i-1,11} \\ y_{i-1,12} & y_{i-1,13} & y_{i-1,14} & y_{i-1,15} \end{pmatrix} \oplus \begin{pmatrix} x_{i,0} & 0 & x_{i,1} & 0 \\ 0 & 0 & 0 & 0 \\ x_{i,2} & 0 & x_{i,3} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\text{SB}} \begin{pmatrix} z_{i,0} & z_{i,1} & z_{i,2} & z_{i,3} \\ z_{i,4} & z_{i,5} & z_{i,6} & z_{i,7} \\ z_{i,8} & z_{i,9} & z_{i,10} & z_{i,11} \\ z_{i,12} & z_{i,13} & z_{i,14} & z_{i,15} \end{pmatrix},$$

where y_{i-1} is the output of round $(i-1)$, and $(x_{i,0}, 0, x_{i,1}, 0, 0, 0, 0, 0, x_{i,2}, 0, x_{i,3}, 0, 0, 0, 0, 0)$ is the injection input to round i which acts as the round subkeys. Assume that M=(x_1, x_2, …, x_{t-1}, x_t) and M' =(x'_1, x'_2, …, x'_{t-1}, x'_t) follow the 2-round collision differential path as depicted in Fig. 2.

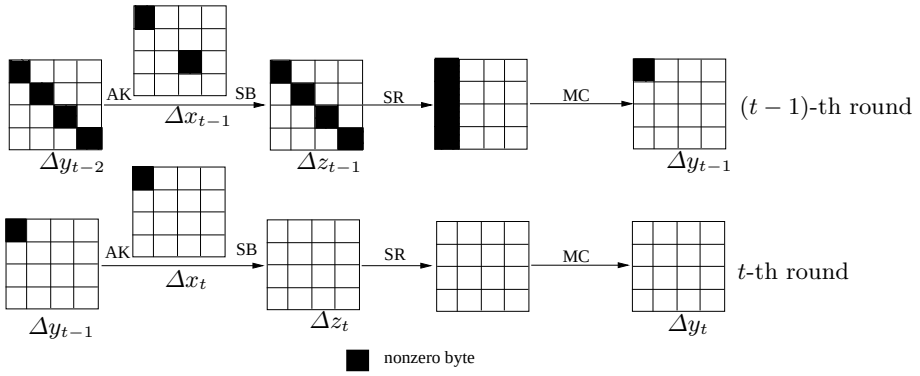

Fig. 2. 2-Round Collision Differential Path

From the differential path, we can see that there is only one nonzero byte in Δy_{t-1}, which equals to $\Delta x_{t,0}$. Because MC is a linear transformation, and SR has no impact on the value of difference, we can compute the output differences of four S-boxes in the $(t-1)$-th round from $\Delta x_{t,0}$:

$$(\Delta z_{t-1,0}, \Delta z_{t-1,5}, \Delta z_{t-1,10}, \Delta z_{t-1,15})^T = MC^{-1}(\Delta x_{t,0}, 0, 0, 0)^T. \quad (1)$$

Since the branch number of MC transformation in AES is 5 [7], there are four nonzero bytes in Δz_{t-1}, they satisfy the difference structure in Fig. 2. Given the 2-round collision differential path in Fig. 2, we have the following facts:

Fact 1. Given two messages $M=(x_1, x_2, \ldots, x_{t-1}, x_t)$ and $M' = (x'_1, x'_2, \ldots, x'_{t-1}, x'_t)$ that follow the 2-round collision differential path, where $\Delta x_t=(\Delta x_{t,0}, 0, 0, 0)$, and $\Delta x_{t-1}=(\Delta x_{t-1,0}, 0, 0, \Delta x_{t-1,3})$, there exists an *Algorithm* $\boldsymbol{A}_1$ to find another different message pair $\overline{M} =(x_1, x_2, \ldots, \overline{x_{t-1}}, x_t)$ and $\overline{M'} =(x'_1, \overline{x'_2}, \ldots, \overline{x'_{t-1}}, x'_t)$ satisfying the 2-round collision differential path. Here $(\overline{x_{t-1}}, \overline{x'_{t-1}})$ is obtained by only replacing $(x_{t-1,0}, x'_{t-1,0})$ with different $(\overline{x_{t-1,0}}, \overline{x'_{t-1,0}})$. The complexity of the algorithm is about 2^9 queries and 2^9 chosen messages.

Proof. Since only $(x_{t-1,0}, x'_{t-1,0})$ changes, all bytes in Δz_{t-1} remain the same except $\Delta z_{t-1,0}$, where $\Delta z_{t-1,0} = S(y_{t-2,0} \oplus x_{t-1,0}) \oplus S(y'_{t-2,0} \oplus x'_{t-1,0})$. Thus, M and M' collide if and only if $S(y_{t-2,0} \oplus \overline{x_{t-1,0}}) \oplus S(y'_{t-2,0} \oplus \overline{x'_{t-1,0}}) = \Delta z_{t-1,0}$. From the distribution table of the S-box in AES, we observe that, there are 2^7 pairs corresponding to each output difference on average. Hence, each randomly chosen pair $(x_{t-1,0}, x'_{t-1,0})$ leads to the expected output difference $\Delta z_{t-1,0}$ with probability $2^7/2^{15} = 2^{-8}$. So *Algorithm* $\boldsymbol{A}_1$ needs about 2^8 chosen message pairs $(\overline{M}, \overline{M'})$ and 2^9 corresponding MACs to find the message pair $(\overline{M}, \overline{M'})$ which follows the 2-round collision differential path. □

Fact 1 will be used in the new distinguishing attack to identify the ALPHA-MAC from a random function. And the following Fact can recover two bytes of the unknown internal state y_{t-2} corresponding to the nonzero bytes of Δx_{t-1}.

Fact 2. Given two messages $M=(x_1, x_2, \ldots, x_{t-1}, x_t)$ and $M' = (x'_1, x'_2, \ldots, x'_{t-1}, x'_t)$ that follow the 2-round collision differential path, where $\Delta x_t=(\Delta x_{t,0}, 0, 0, 0)$, and $\Delta x_{t-1}=(\Delta x_{t-1,0}, 0, 0, \Delta x_{t-1,3})$, there exists an *Algorithm* $\boldsymbol{A}_2$ to recover $(y_{t-2,0}, y'_{t-2,0})$ with about 2^{16} XOR operations and 2^9 chosen messages.

Proof. Algorithm $\boldsymbol{A}_2$ is described as follows.

1. Call *Algorithm* $\boldsymbol{A}_1$ to find another message pair $\overline{M} =(x_1, x_2, \ldots, \overline{x_{t-1}}, x_t)$ and $\overline{M'} =(x'_1, x'_2, \ldots, \overline{x'_{t-1}}, x'_t)$ which produce a 2-round collision differential path in Fig. 3. Here $(\overline{x_{t-1}}, \overline{x'_{t-1}})$ are only different at byte position 0 from (x_{t-1}, x'_{t-1}).
2. Compute $z_{t-1,0}$ from Eq. (1), guess all 2^{16} possibilities of $(y_{t-2,0}, y'_{t-2,0})$, and check if the following two equations hold.

$$S(y_{t-2,0} \oplus x_{t-1,0}) \oplus S(y'_{t-2,0} \oplus x'_{t-1,0}) = \Delta z_{t-1,0}, \quad (2)$$

$$S(y_{t-2,0} \oplus \overline{x_{t-1,0}}) \oplus S(y'_{t-2,0} \oplus \overline{x'_{t-1,0}}) = \Delta z_{t-1,0}. \quad (3)$$

3. If there is only one solution $(y_{t-2,0}, y'_{t-2,0})$ satisfying Eq. (2) and (3) among 2^{16} guesses, outputs $(y_{t-2,0}, y'_{t-2,0})$.
   Otherwise, repeat steps 1 and 2 until only one solution is left.

It is obvious that, the time complexity of *Algorithm* $\boldsymbol{A}_2$ is about 2^{16} XOR operations and 2^9 chosen messages. □

Fact 3. Given two messages $M=(x_1, x_2, \ldots, x_{t-1}, x_t)$ and $M' = (x'_1, x'_2, \ldots, x'_{t-1}, x'_t)$ that follow the 2-round collision differential path, where $\Delta x_t=(\Delta x_{t,0}, 0, 0, 0)$, and $\Delta x_{t-1}=(\Delta x_{t-1,0}, 0, 0, \Delta x_{t-1,3})$, there exists an *Algorithm* $\boldsymbol{A}_3$ to recover $(y_{t-2,10}, y'_{t-2,10})$ with about 2^{16} XOR operations and 2^9 queries.

Proof. The proof of Fact 3 is similar to that of Fact 2. We only need to replace $(x_{t-1,10}, x'_{t-1,10})$ by different $(\overline{x_{t-1,10}}, \overline{x'_{t-1,10}})$. □

4.2 Distinguishing Attack on Alpha-MAC

Similar to the distinguisher for Alred construction, the new distinguisher on Alpha-MAC is based on the identification of an inner near-collision Δy_{t-1} as shown in Fig. 2. By the birthday paradox, such an inner near-collision exists, and can be detected by the new distinguisher. From *Algorithms* $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$, we can recover two bytes of the internal state y_{t-2}. Moreover, we explore a series of tricks to recover more bytes of the internal state y_{t-3}, and further recover y_0. It is noted that $y_0 = Enc_K(0)$ equals to a subkey used in the secret prefix method.

It is claimed that an extinguishing differential in Alpha-MAC spans at least 5 message words, and given the state value y_{i-1}, the map from the sequence of four message words $(x_i, x_{i+1}, x_{i+2}, x_{i+3})$ to the state value before the $(i+4)$-th iteration is a bijection [8]. Hence, we choose a structure composed of $2^{64.5}$ messages with t-word length, where t is required to be bigger than or equal to 6 in order to guarantee the map from $(x_1, \ldots, x_{t-1})$ to y_{t-1} is a random function, and to ensure the existence of an inner near-collision. It is recommended to choose $t = 9$.

Given a fixed word difference $(\eta, 0, 0, 0)$, construct two structures as follows:

$$T_1 = \{M^a = (x_1^a, x_2^a, \ldots, x_{t-1}^a, x_t)\},$$
$$T_2 = \{M^b = (x_1^b, x_2^b, \ldots, x_{t-1}^b, x_t \oplus (\eta, 0, 0, 0))\},$$

where the message words (x_i^a, x_i^b) $(i = 1, 2, \ldots, t-2)$, $(x_{t-1,0}^a, x_{t-1,3}^a)$ of x_{t-1}^a, and $(x_{t-1,0}^b, x_{t-1,3}^b)$ of x_{t-1}^b are randomly chosen, and other bytes of (x_{t-1}^a, x_{t-1}^b) are fixed, i. e., we choose Δx_{t-1} and Δx_t as shown in Fig. 2. The distinguisher works in the following manner:

1. Query the MAC with all the $2^{65.5}$ messages in structure T_1 and T_2, and obtain the corresponding MACs.
2. Search for (M^a, M^b) such that $C^a = C^b$ by the birthday attack, where $M^a \in T_1$, $M^b \in T_2$. Randomly choose another different pair $(\overline{M^a}, \overline{M^b})$,

where $\overline{M^a} = (x_1^a, \ldots, x_{t-1}^a, \overline{x_t^a})$, $\overline{M^b} = (x_1^b, \ldots, x_{t-1}^b, \overline{x_t^b})$, $\Delta\overline{x_t} = \Delta x_t$. Query the MAC with the new message pair $(\overline{M^a}, \overline{M^b})$. If $(\overline{M^a}, \overline{M^b})$ is a collision, we conclude that the MAC is ALRED-MAC, and go to step 3. Otherwise, the MAC is a random function.

3. Randomly choose 2^8 different $(\overline{x_{t-1,0}^a}, \overline{x_{t-1,0}^b})$ to replace $(x_{t-1,0}^a, x_{t-1,0}^b)$. Query the MACs of the new messages. Check whether there is at least one collision among them. If a collision appears, the ALRED construction is claimed as the ALPHA-MAC. Otherwise, it is other ALRED MAC instance.

Complexity Evaluation. Step 1 takes $2^{65.5}$ MAC queries. There are only 2 queries and $2^{65.5}$ table look-ups with $2^{65.5}$ entries in step 2, and 2^8 MAC queries in step 3. Thus, the total complexity is dominated by step 1, which is about $2^{65.5}$ MAC queries and $2^{65.5}$ chosen messages.

Success Rate. A collision between the two structures occurs with probability 0.63 from the birthday paradox, which is also the success rate of our attack.

4.3 Internal State Recovery of ALPHA-MAC

In this section, we recover the internal state y_0 combining the new distinguisher discussed above with some new tricks. Once the ALRED construction is identified as the ALPHA-MAC, we obtain a message pair (M^a, M^b), which follows the 2-round collision differential path (See Fig. 2).

Denote $M^a = (x_1^a, x_2^a, \ldots, x_{t-1}^a, x_t^a)$ and $M^b = (x_1^b, x_2^b, \ldots, x_{t-1}^b, x_t^b)$. The process of the internal state recovery attack is depicted in Fig. 3, where '$*$' denotes the difference that can be computed, '?' stands for the unknown difference, and '0' means zero difference. The details of the recovery attack are as follows:

$$\Delta y_{t-3} = \begin{pmatrix} * & ? & * & ? \\ ? & * & ? & * \\ * & ? & * & ? \\ ? & * & ? & * \end{pmatrix}$$

$$\xleftarrow{AK^{-1}\ SB^{-1}} \Delta z_{t-2} = \begin{pmatrix} * & ? & * & ? \\ ? & * & ? & * \\ * & ? & * & ? \\ ? & * & ? & * \end{pmatrix} \xleftarrow{SR^{-1}\ MC^{-1}} \Delta y_{t-2} = \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & ? & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & ? \end{pmatrix}$$

$$\xleftarrow{AK^{-1}\ SB^{-1}} \Delta z_{t-1} = \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \end{pmatrix} \xleftarrow{SR^{-1}\ MC^{-1}} \Delta y_{t-1} = \begin{pmatrix} \Delta x_{t,0} & 0 & 0 & 0 \\ 0 & & 0 & 0 & 0 \\ 0 & & 0 & 0 & 0 \\ 0 & & 0 & 0 & 0 \end{pmatrix}$$

Fig. 3. Recovering the Internal State

1. **Recovering** $(y^a_{t-2,0}, y^b_{t-2,0}, y^a_{t-2,10}, y^b_{t-2,10})$**.**
   By Algorithms $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$, the corresponding bytes $(y^a_{t-2,0}, y^b_{t-2,0}, y^a_{t-2,10}, y^b_{t-2,10})$ can be recovered directly.
   Next, let us explore more techniques to recover more bytes of the internal states y_{t-2} and y_{t-3}.
2. **Recovering** $(y^a_{t-3,0}, y^b_{t-3,0}, y^a_{t-3,2}, y^b_{t-3,2}, y^a_{t-3,8}, y^b_{t-3,8}, y^a_{t-3,10}, y^b_{t-3,10})$**.**
   Applying MC^{-1} and SR^{-1} to the state y_{t-2}, we obtain the values $(\Delta z_{t-2,0}, \Delta z_{t-2,5}, \Delta z_{t-2,10}, \Delta z_{t-2,15}, \Delta z_{t-2,2}, \Delta z_{t-2,7}, \Delta z_{t-2,8}, \Delta z_{t-2,13})$ from the following two equations:

$$(\Delta z_{t-2,0}, \Delta z_{t-2,5}, \Delta z_{t-2,10}, \Delta z_{t-2,15})^T = MC^{-1}(\Delta y_{t-2,0}, 0, 0, 0)^T, \quad (4)$$
$$(\Delta z_{t-2,2}, \Delta z_{t-2,7}, \Delta z_{t-2,8}, \Delta z_{t-2,13})^T = MC^{-1}(0, 0, \Delta y_{t-2,10}, 0)^T. \quad (5)$$

   (a) Recovering $(y^a_{t-3,0}, y^b_{t-3,0})$.
      i. Look up the differential distribution table of AES S-box, and obtain about 2^8 possible values $(y_{t-3,0}, y'_{t-3,0})$ satisfying

$$S(y_{t-3,0} \oplus x^a_{t-2,0}) \oplus S(y'_{t-3,0} \oplus x^b_{t-2,0}) = \Delta z_{t-2,0}, \quad (6)$$

      The correct $(y^a_{t-3,0}, y^b_{t-3,0})$ must be among the 2^8 possible $(y_{t-3,0}, y'_{t-3,0})$.
      ii. Detect the correct $(y^a_{t-3,0}, y^b_{t-3,0})$ from the following fact.

Fact 4. For each possible $(y_{t-3,0}, y'_{t-3,0})$, set $x_{t-2} = (x_{t-2,0}, x^a_{t-2,1}, x^a_{t-2,2}, x^a_{t-2,3})$ and $x'_{t-2} = (x'_{t-2,0}, x^b_{t-2,1}, x^b_{t-2,2}, x^b_{t-2,3})$, where

$$x_{t-2,0} = x^b_{t-2,0} \oplus y_{t-3,0} \oplus y'_{t-3,0},$$
$$x'_{t-2,0} = x^a_{t-2,0} \oplus y_{t-3,0} \oplus y'_{t-3,0}.$$

Suppose $\overline{\Delta x_{t-1}} = x_{t-1} \oplus x'_{t-1}$ and $\Delta x_{t-1} = x^a_{t-1} \oplus x^b_{t-1}$.
Select 2^8 different word pairs (x_{t-1}, x'_{t-1}) such that

$$\overline{\Delta x_{t-1,0}} \neq \Delta x_{t-1,0} \text{ and } \overline{\Delta x_{t-1,i}} = \Delta x_{t-1,i} \ (i = 1, 2, 3).$$

Query the MAC with 2^8 message pairs (M, M'), where

$$M = \{x^a_1, \ldots, x^a_{t-3}, x_{t-2}, x_{t-1}, x^a_t\} \text{ and } M' = \{x^b_1, \ldots, x^b_{t-3}, x'_{t-2}, x'_{t-1}, x^b_t\}.$$

- If one collision is found among 2^8 message pairs (M, M'), the correct $(y^a_{t-3,0}, y^b_{t-3,0}) = (y_{t-3,0}, y'_{t-3,0})$.
- Otherwise, the $(y_{t-3,0}, y'_{t-3,0})$ is not correct.

Proof. If $(y_{t-3,0}, y'_{t-3,0}) = (y^a_{t-3,0}, y^b_{t-3,0})$, the two inputs to S-boxes are

$$x_{t-2,0} \oplus y^a_{t-3,0} = x^b_{t-2,0} \oplus y^a_{t-3,0} \oplus y^b_{t-3,0} \oplus y^a_{t-3,0} = x^b_{t-2,0} \oplus y^b_{t-3,0},$$
$$x'_{t-2,0} \oplus y^b_{t-3,0} = x^a_{t-2,0} \oplus y^a_{t-3,0} \oplus y^b_{t-3,0} \oplus y^b_{t-3,0} = x^a_{t-2,0} \oplus y^a_{t-3,0},$$

respectively, and the corresponding outputs are

$$S(x_{t-2,0} \oplus y_{t-3,0}) = z_{t-2,0} = z^b_{t-2,0},\ S(x'_{t-2,0} \oplus y'_{t-3,0}) = z'_{t-2,0} = z^a_{t-2,0},$$

i. e., $\overline{\Delta z_{t-2,0}} = \Delta z_{t-2,0}$, which implies that $\overline{\Delta z_{t-2}} = \Delta z_{t-2}$.
It is noted that, the byte $z_{t-2,0}$ only affects four bytes $(y_{t-2,0}, y_{t-2,4}, y_{t-2,8}, y_{t-2,12})$, which means that the 2nd to 4th columns of $\overline{\Delta y_{t-1}}$ are the same as Δy_{t-1}. Therefore, $y_t = y'_t$ if and only if

$$S(y_{t-2,0} \oplus x_{t-1,0}) \oplus S(y'_{t-2,0} \oplus x'_{t-1,0}) = \Delta z_{t-1,0}. \tag{7}$$

There exists one collision among 2^8 different pair (M, M') on average. If $(y_{t-3,0}, y'_{t-3,0}) \neq (y^a_{t-3,0}, y^b_{t-3,0})$, the two inputs to the S-box are

$$x_{t-2,0} \oplus y^a_{t-3,0} = x^b_{t-2,0} \oplus y_{t-3,0} \oplus y'_{t-3,0} \oplus y^a_{t-3,0},$$
$$x'_{t-2,0} \oplus y^b_{t-3,0} = x^a_{t-2,0} \oplus y_{t-3,0} \oplus y'_{t-3,0} \oplus y^b_{t-3,0}.$$

The equation $\overline{\Delta z_{t-2,0}} = \Delta z_{t-2,0}$ holds with probability 2^{-8}.
Thus, to guarantee a collision occur, it is required that (i) $\overline{\Delta y_{t-2,4}} = 0$, $\overline{\Delta y_{t-2,8}} = 0$ and $\overline{\Delta y_{t-2,12}} = 0$ when $\overline{\Delta z_{t-2}} \neq \Delta z_{t-2}$, or (ii) Eq. (6) holds when $\overline{\Delta z_{t-2}} = \Delta z_{t-2}$. Among 2^8 different message pairs (M, M'), there is a collision with probability $2^{-24} \times 2^8 = 2^{-16}$ for the first case, and the probability is at most 2^{-8} for the second. □

(b) In a similar manner, the values $(y^a_{t-3,2}, y^b_{t-3,2})$, $(y^a_{t-3,8}, y^b_{t-3,8})$ and $(y^a_{t-3,10}, y^b_{t-3,10})$ can be each filtered by 2^8 message pairs.

3. **Recovering** $(y^a_{t-3,5}, y^b_{t-3,5}, y^a_{t-3,7}, y^b_{t-3,7}, y^a_{t-3,13}, y^b_{t-3,13}, y^a_{t-3,15}, y^b_{t-3,15})$**.**
Compute the correct $(y^a_{t-3,5}, y^a_{t-3,15}, y^b_{t-3,5}, y^a_{t-3,15})$ by

$$\Delta z_{t-2,5} = S(y^a_{t-3,5}) \oplus S(y^b_{t-3,5}),$$
$$\Delta z_{t-2,15} = S(y^a_{t-3,15}) \oplus S(y^b_{t-3,15}),$$
$$y^a_{t-2,0} = 3S(y^a_{t-3,0} \oplus x^a_{t-2,0}) \oplus 2S(y^a_{t-3,5}) \oplus S(y^a_{t-3,10} \oplus x^a_{t-2,3}) \oplus S(y^a_{t-3,15}),$$
$$y^b_{t-2,0} = 3S(y^b_{t-3,0} \oplus x^b_{t-2,0}) \oplus 2S(y^b_{t-3,5}) \oplus S(y^b_{t-3,10} \oplus x^b_{t-2,3}) \oplus S(y^b_{t-3,15}).$$

Similarly, obtain the correct $(y^a_{t-3,7}, y^a_{t-3,13}, y^b_{t-3,7}, y^a_{t-3,13})$ from

$$\Delta z_{t-2,7} = S(y^a_{t-3,7}) \oplus S(y^b_{t-3,7}),$$
$$\Delta z_{t-2,13} = S(y^a_{t-3,13}) \oplus S(y^b_{t-3,13}),$$
$$y^a_{t-2,10} = S(y^a_{t-3,2} \oplus x^a_{t-2,1}) \oplus S(y^a_{t-3,7}) \oplus 3S(y^a_{t-3,8} \oplus x^a_{t-2,2}) \oplus 2S(y^a_{t-3,13}),$$
$$y^b_{t-2,10} = S(y^b_{t-3,2} \oplus x^b_{t-2,1}) \oplus S(y^b_{t-3,7}) \oplus 3S(y^b_{t-3,8} \oplus x^b_{t-2,2}) \oplus 2S(y^b_{t-3,13}).$$

4. **Recovering the internal state** y_0**.**
Guess all the 2^{64} possibilities of the rest 8 bytes of y^a_{t-3}. Take $(x^a_{t-3}, \ldots, x^a_1)$ as the decryption subkey, and obtain 2^{64} y_0. For each y_0, compute the corresponding y^b_{t-3} with $(x^b_1, \ldots, x^b_{t-3})$ to filter out the wrong guesses.
If there are more than one y_0 left, using the distinguisher to get another colliding pair, and repeat the whole recovery attack until there is only one value left. Two colliding pairs are enough to sieve the right y_0.

Until now, the recovery attack on the internal state y_0 is completed.

Complexity Evaluation. The complexity of this attack is dominated by the distinguishing attack and the final exhaustive search, which is about $2^{65.5}$ queries and $2^{65.5}$ chosen messages.

Second Preimages for ALPHA**-MAC.** Once the internal state y_0 is recovered, the second preimages can be found by Huang et al.'s attack [10], and a selective forgery attack can be performed as in [5].

5 Conclusion

In this part, the distinguishing and forgery attacks on the ALRED construction and its specific instance ALPHA-MAC are presented. The complexity of the attacks is dominated by the birthday attack, far less than the exhaustive search. Our contribution is to detect inner near-collisions with specific differences rather than collisions, from which more information can be obtained. Especially for ALPHA-MAC, combining with the distinguishing attack, we explore a series of tricks to recover the internal state, which equals to an equivalent subkey. This leads to the second preimage attack on ALPHA-MAC. It is remarked that the distinguishing and forgery attacks on ALRED construction are also applicable to the MACs based on CBC and CFB encryption modes.

Part II Impossible Differential Cryptanalysis of PELICAN, MT-MAC-AES and PC-MAC-AES[2]

6 Introduction to Part II

Besides the MACs introduced in Part I, there are several others based on reduced block ciphers, such as PELICAN [9], MT-MAC-AES and PC-MAC-AES [15], and all of them take the 4-round AES [3] as the iteration function.

PELICAN is an optimized version of ALPHA-MAC, which was proposed by Daemen and Rijmen. It generates the MAC value in a CBC-like manner. The side-channel collision attack on ALPHA-MAC works for PELICAN, too [5]. Minematsu and Tsunoo also proposed two provably secure MAC constructions, MT-MAC and PC-MAC, which make use of the provably secure almost universal hash functions (AU_2). The MT-MAC uses differentially uniform permutations such as 4-round AES with independent keys in a Wegman-Carter binary tree. However, it is not memory efficient. A modified version PC-MAC, which is based on a CBC-like AU_2 hash PCH (Periodic CBC Hash), was suggested.

Inspired by recent MAC cryptanalysis techniques of Wang et al. [19,20] and the methods introduced in Part I [21], we observe that the impossible differential

[2] by Wei Wang, Xiaoyun Wang, and Guangwu Xu. See [18] for the full version.

[3] The MT-MAC-AES and PC-MAC-AES take the *simplified* 4-round AES described in Section 7.2.

attack can be extended to MACs provided that enough inner near-collisions with specific differences are detected.

Impossible differential attack [3] is one of the widely used cryptanalytic techniques on block ciphers. It is a sieving attack which focuses on a differential path with probability 0. If a pair of messages is encrypted or decrypted to an impossible difference under some trial key, one can filter out this trial key from the key space. Thus, the correct key is found by eliminating all the wrong keys which lead to a contradiction. For MAC algorithms, the secret key is usually replaced by the internal state. It seems that, the impossible differential attack is hard to work with MAC algorithms, due to the fact that the internal state values as well as their differences, are concealed by the final full encryption or complex keyed iterations. However, the recent techniques based on the birthday attack overcome this obstacle. One can recognize the inner near-collisions with some specific differences, hence the impossible differential attack can be performed with the detected inner near-collisions.

Taking 4-round AES as a building block, we are able to recover its secret state utilizing a 3-round impossible differential characteristic. For PELICAN, the secret subkey is replaced by the internal state, thus we can recover its internal state with $2^{85.5}$ chosen messages and $2^{85.5}$ queries. This attack can be further extended to a subkey recovery attack on MT-MAC-AES with the same complexities. For PC-MAC-AES, we recover its two secret keys separately once the internal state is sieved, with $2^{85.5}$ chosen messages and 2^{128} queries. We emphasis that our results do not contradict to any security proof associated with the designs. Due to space limitations, we only present attacks on PELICAN and PC-MAC-AES, while the attack on MT-MAC-AES appears in [18].

7 Backgrounds

Beside the notations defined in Part I, we will use the following notations in this part: let z_i^I denote the input of the i-th AES round, while z_i^B, z_i^R, z_i^M and z_i^O denote the intermediate values after the application of SB, SR, MC and AK of the i-th AES round, respectively. z_i is exhibited as a 4×4 two dimensional array of bytes indexed as:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Next, we give brief descriptions of PELICAN and PC-MAC-AES.

7.1 PELICAN Algorithm

PELICAN is a specific instance of ALRED construction taking 4-round AES as a building block [9]. It supports 128-bit block and 128/160/192/224/256-bit key.

PELICAN takes a message of arbitrary length as input, and outputs a MAC value with length up to 128-bit.

To construct the MAC, let us pad a message M of any length to a multiple of 128-bit by appending a single bit '1' followed by the minimum bits of '0', and split the padded message into 128-bit words $(x_1, x_2, \ldots, x_b)$. The PELICAN MAC function works as follows:

1. **Initialization:** Fill the 128-bit *state* with zeros, and encrypt the zero state with AES encryption, i. e., $y_0 = E_K(0)$, where E is the AES, and K is the secret key.
2. **Chaining:** XOR the first message word x_1 to the state, i. e., $y_1 = y_0 \oplus x_1$. For each message word x_i $(i = 2, \ldots, b)$, perform an iteration operation: $y_i = f(y_{i-1}) \oplus x_i$, where f consists of 4-round AES with the round subkeys set to 0.
3. **Finalization:** Apply the full AES to the state, and take the first l_m bits of the state as the MAC value of M. The final output C is $C = Trunc(E_K(y_m))$.

7.2 PC-MAC-AES

PC-MAC is a provably secure MAC construction proposed in [15]. It is composed of an n-bit block cipher E_K, and an n-bit auxiliary keyed permutation G_U. Two secret keys are required, one for the block cipher and the other for making the block cipher tweakable.

For $i = 1, 2, \ldots, s$, let F_i be an n-bit random function. Suppose $x = (x_1, x_2, \ldots, x_{s+1})$, we first define the chaining function:

$$Ch[F_1, \ldots, F_s](x) = x_{s+1} \oplus F_s(x_s \oplus F_{s-1}(\cdots F_2(x_2 \oplus F_1(x_1)) \cdots)),$$

which is used iteratively when the input is longer than $(s+1)$ blocks, and terminates as soon as the last input block is XORed. The PCH is defined as follows:

Definition 1 (Periodic CBC Hash (PCH) [15]).
Let E_K be an n-bit block cipher. For $d \geq 0$, let $G = (G_1, \ldots, G_d)$ be the sequence of keyed auxiliary permutation, where for G_i $(i = 1, \ldots, d)$, the subkey involved in G is U_i. We assume that $(K_1^{XOR}, \ldots, K_{d-1}^{XOR})$ are (d-1) n-bit subkeys. The Periodic CBC Hash is defined as:

$$PCH_d[E_K, G] = Ch[E_K, G_1, G_2^{\oplus K_1^{XOR}}, \ldots, G_d^{\oplus K_{d-1}^{XOR}}].$$

Here, $G_i^{\oplus K_{i-1}^{XOR}}(\alpha) = G_i(\alpha \oplus K_{i-1}^{XOR})$ $(i = 2, \ldots, d)$, where α is an n-bit variable. $PCH_d[E_K, G]$ terminates as soon as the last input block is XORed.

The next is the description of the PC-MAC$_d[E_K|G_U]$ construction.

– **Preprocessing:**
  - Compute $U = (U_1, \ldots, U_d)$, which is the first dl bits of $E_K(0 \oplus L), \ldots, E_K(\hat{a} \oplus L)$. Here, K, L are the secret keys, and $\hat{a} = \lceil dl/n \rceil$.

  - Compute $K^{\mathrm{XOR}}_{j-\hat{a}+1} = E_K(j \oplus L)$, for $j = \hat{a}, \ldots, \hat{a}+d-2$.
- **MAC Computation:** For a message M with arbitrary length,

$$C = \begin{cases} E_K(PCH_d[E_K, G](M) \oplus L \cdot u) & if\ |M| \mod n = 0, \\ E_K(PCH_d[E_K, G](M \| 10^t) \oplus L \cdot u^2) & if\ |M| \mod n = n-t-1, \end{cases}$$

  where u is an element of $GF(2^n)$ that is not 0 or 1, and $L \cdot u$ is the multiplication of L and u on $GF(2^n)$.

The authors of [15] recommended to implement block cipher E_K with the AES-128, and the permutation G_U with the *simplified* 4-round AES, where the transformations of each round perform in the order of AK, SB, SR and MC, and the addition of the first round key and the last diffusion layer are omitted. We call this AES-based instance PC-MAC-AES.

8 Main Idea of the Impossible Differential Cryptanalysis

Similar to the cryptanalysis of block ciphers, to implement an impossible differential attack on MACs, we need to find an impossible differential path first. Then collect many structures of chosen messages, query MACs with them, and sieve the message pairs with the required intermediate differences. For each sieved pair, discard the wrong subkeys (or internal states) which cause the partial encryption and decryption to match the impossible differential path. Finally, after enough pairs are analyzed, only the correct subkey (or internal state) is left.

8.1 Three-Round Impossible Differential Property of AES

For AES, several 4-round impossible differential paths have been found in literature, e. g. [1,4,16]. However, we note that, among the MAC algorithms presented in the previous section, the 4-round AES is taken as a building block. Thus, we focus on the reduced AES and only need a 3-round impossible differential path.

The 3-round impossible differential path is stated as follows.

Property 1 (Impossible Differential Path of 3-round AES). For 3-round AES, given an input pair $(z_2^I, z_2^{I\prime})$ whose components equal in all except six bytes indexed by $(0,1,5,8,12,13)$ (or $(0,1,4,5,9,12)$, $(0,4,5,8,9,13)$, $(1,4,8,9,12,13)$), the difference of the output pair $(z_4^O, z_4^{O\prime})$ can not have exactly one nonzero byte.

The correctness of Property 1 can be easily proved, and Fig. 4 illustrates the impossible differential path for the case of $(0,1,5,8,12,13)$.

8.2 Message Pairs Collection Phase

In the cryptanalysis of block ciphers, we can collect the message pairs available to the impossible differential attack directly according to the output differences and chosen message differences. While for MACs mentioned above, we have to explore new techniques to collect such message pairs since the 4-round AES is

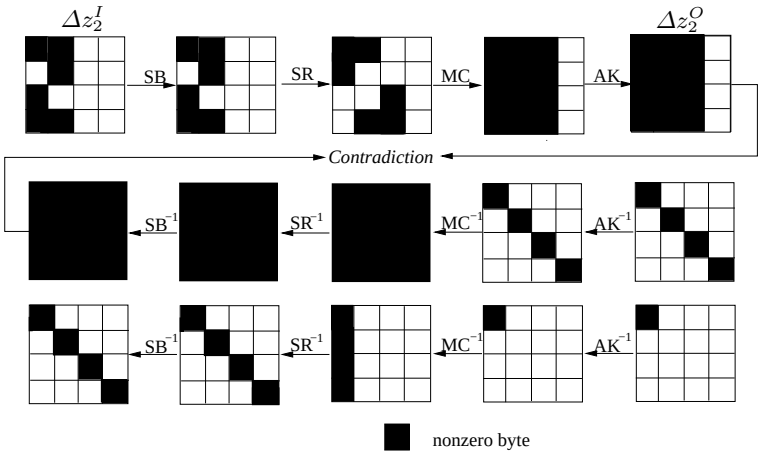

Fig. 4. 3-Round Impossible Differential Path of AES

used as a chaining or auxiliary permutation function, whose output is concealed by the final full AES encryption, as well as the output difference.

To get over this obstacle, we take advantage of the idea described in [19,20,21]. First, randomly choose two structures of messages, with the message differences of some specific forms. One example is that there is only one nonzero byte in the difference of the last word. The concrete structures are constructed based on the concrete MAC constructions. Second, utilize the birthday attack to search collisions between the two structures. Finally, once enough message collisions are collected, we can sieved the correct subkeys in the similar manner as the impossible differential cryptanalysis of block ciphers. The details of collecting collision pairs will be given in the next section.

9 Impossible Differential Cryptanalysis of PELICAN and PC-MAC-AES

In this section, we present the impossible differential attacks based on the 3-round impossible differential path proposed in Section 8.

9.1 Internal State Recovery of PELICAN

This subsection describes the internal state recovery attack on PELICAN with one additional round at the beginning of the 3-round impossible differential path. The recovery of the internal state results in the derivation of an equivalent subkey, i. e., the state $y_0 = E_K(0)$. We depict the PELICAN algorithm with two message words in Fig. 5 for simplicity.

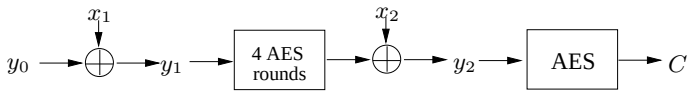

Fig. 5. PELICAN ($b = 2$)

We first consider the situation that there is no truncation at the final output, i. e., $l_m = 128$. From Fig. 5, we can see that a collision at C indicates a collision at y_2 since the final AES encryption is a permutation. Because

$$y_2 = AES^{4r}(y_1) \oplus x_2,$$

where AES^{4r} stands for the 4-round AES, the inner collision at y_2 happens if and only if

$$AES^{4r}(y_1) \oplus x_2 = AES^{4r}(y_1') \oplus x_2',$$

which yields the output difference of 4-round AES

$$AES^{4r}(y_1) \oplus AES^{4r}(y_1') = x_2 \oplus x_2'. \tag{8}$$

If there is truncation at the final output, i. e., $l_m < 128$, then we need to distinguish the collision caused by inner collision, which means to detect the message pairs leading to $y_2 = y_2'$. Suppose $(x_1 \| x_2, x_1' \| x_2')$ is a collision. Query the MAC with $(x_1 \| x_2', x_1' \| x_2)$. If they still collide, we conclude that the pair $(x_1 \| x_2, x_1' \| x_2')$ satisfies $y_2 = y_2'$, i. e., Eq. (8).

It is clear that, once an inner collision is detected, we can deduce the information of the output difference of the inner 4-round AES from Δx_2, and apply the impossible differential cryptanalysis. Therefore, it is essential to collect enough message pairs which cause inner collisions at y_2.

Message Pairs Collection Phase

We sieve the message pairs resulting in the inner collisions as follows.

1. Construct two structures, each has 2^{64} two-word messages. Randomly choose $(x_{1,2}, \ldots, x_{1,14})$, which are the bytes of the first word x_1 indexed by $(2, 3, 4, 7, 8, 9, 13, 14)$, and set the corresponding bytes of x_1' with the same values; randomly choose two 128-bit message words x_2 and x_2', with only one nonzero byte in $\Delta x_2 = x_2 \oplus x_2'$. The two structures are

$$S_1 = \{(x_1, x_2) | (x_{1,0}, x_{1,1}, x_{1,5}, x_{1,6}, x_{1,10}, x_{1,11}, x_{1,12}, x_{1,15}) \in \{0,1\}^{64}\},$$
$$S_2 = \{(x_1', x_2') | (x_{1,0}', x_{1,1}', x_{1,5}', x_{1,6}', x_{1,10}', x_{1,11}', x_{1,12}', x_{1,15}') \in \{0,1\}^{64}\}.$$

   It is noted that the difference Δx_1 is zero at bytes indexed by $(2, 3, 4, 7, 8, 9, 13, 14)$, where $\Delta x_1 = x_1 \oplus x_1'$.
2. Query MAC on the two structures, and search collisions between the corresponding MAC values of the two structures by the birthday attack.

- If there is no truncation at the final output, the corresponding colliding message pairs cause inner collisions at y_2.
- Else, for all collected colliding pairs $(x_1\|x_2, x_1'\|x_2')$, query the MAC on $(x_1\|x_2', x_1'\|x_2)$. If still collide, $(x_1\|x_2, x_1'\|x_2')$ must be an inner collision.

Since there are 2^{64} elements in each structure, and the internal state is 128-bit, one inner collision is expected to be found with probability 2^{-1}. Repeat the message pairs collection phase by choosing different $(x_{1,2}, x_{1,3}, x_{1,4}, x_{1,7}, x_{1,8}, x_{1,9}, x_{1,13}, x_{1,14})$, one inner collision pair is expected to be obtained. This means that, we can get one useful pair with about $2 \times 2 \times 2^{64} = 2^{66}$ chosen messages. To obtain 2^a colliding pairs, $2^a \times 2^{66} = 2^{a+66}$ chosen messages are required. Thus, the time complexity is 2^{a+66} queries.

For each collected pair, there is only one nonzero byte in Δz_4^O since there is only one nonzero byte in Δx_2, where $z_4^O = AES^{4r}(y_1)$. The input to the 4-round AES, y_1, equals to $x_1 \oplus y_0$, and the round subkeys are set to zero, so y_0 can be regarded as the subkey XORed before the first round, and is recovered in a similar manner as the impossible differential cryptanalysis of AES.

Internal State Recovery Phase

We can recover 8 bytes of y_0 at position $(0, 1, 5, 6, 10, 11, 12, 15)$ by exhaustive search directly (See Fig. 6).

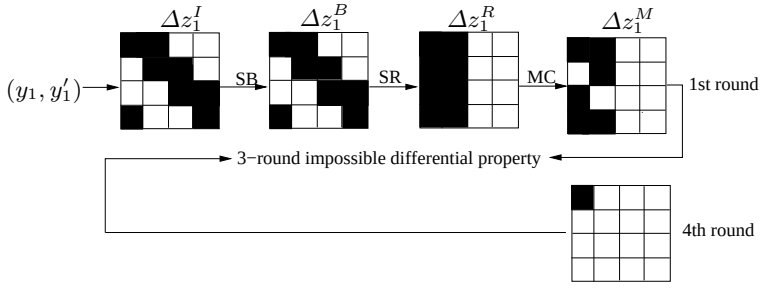

Fig. 6. Internal State Recovery of PELICAN

1. Initialize a list L to store the 2^{64} possible values $(y_{0,0}, y_{0,1}, y_{0,5}, y_{0,6}, y_{0,10}, y_{0,11}, y_{0,12}, y_{0,15})$.
2. For each of the 2^a valid pairs, perform partial encryption with each element in L, and obtain the first two columns of z_1^M and $z_1^{M\prime}$, respectively. From the fact that Δx_1 is zero at bytes $(2, 3, 4, 7, 8, 9, 13, 14)$, we can deduce that the last two columns of Δz_1^M are zero. Thus, if Δz_1^M is in the form of Δz_2^I as described in Property 1, the corresponding 8 bytes of y_0 must be wrong, because of Property 1. We delete it from the list L.
   After all pairs are processed, we expect that there is only one element in the list L, which is the correct one.

For random $(y_{0,0}, y_{0,1}, y_{0,5}, y_{0,6}, y_{0,10}, y_{0,11}, y_{0,12}, y_{0,15})$, the probability that Δz_1^M has the impossible form is $4 \cdot 2^{-16} = 2^{-14}$, since for the two zero bytes in the first two columns, there are 4 possible positions. Therefore, for each collected pair, we can filter out $2^{64} \cdot 2^{-14} = 2^{50}$ wrong $(y_{0,0}, y_{0,1}, y_{0,5}, y_{0,6}, y_{0,10}, y_{0,11}, y_{0,12}, y_{0,15})$, and one wrong value remains in list L with probability $1 - \frac{2^{50}}{2^{64}}$. After analyzing all 2^a pairs, the expected number of wrong elements left in L should satisfy

$$2^{64} \cdot (1 - \frac{2^{50}}{2^{64}})^{2^a} < 1.$$

This relation holds if we take $a = 2^{19.5}$.

In this manner, we can recover 8 bytes of the internal state y_0, and the other 8 bytes can be recovered in a similar way.

Complexity Estimation. For the message pairs collection phase, the data complexity is $2^{a+66} = 2^{85.5}$ chosen messages, and the time complexity is $2^{85.5}$ queries. For the internal state recovery phase, the time complexity is at most $2^{19.5} \cdot 2^{64} = 2^{83.5}$ one-round encryptions since there are $2^{19.5}$ collected pairs. Therefore, the total complexity is dominated by the message pairs collection phase, which is about $2^{85.5}$ queries and $2^{85.5}$ messages.

Selective Forgery Attack. Once the attacker obtains the value of the internal state y_0, he has full control of the internal state, and can create arbitrary colliding messages by calculating a proper 128-bit injection at the end.

9.2 Key Recovery Attack on PC-MAC-AES

The situation becomes a little different when it comes to PC-MAC-AES, where the simplified 4-round AES is applied after the second block, and there are two secret keys (K, L) involved in the MAC computation. We can use the divide-and-conquer technique to recover the two secret keys. The PC-MAC-AES with three message words is illustrated in Fig. 7.

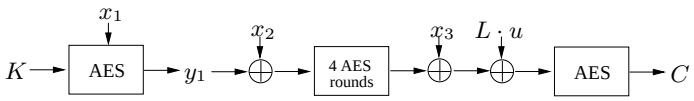

Fig. 7. PC-MAC-AES with Three Message Words

We proceed the key recovery attack according to the following procedure.

1. Construct two structures by prepending a fixed x_1 to each message of structures S_1 and S_2 given in Section 9.1. Randomly choose x_1, set the bytes at $(2, 3, 4, 7, 8, 9, 13, 14)$ of x_2 and x_2' to the same values, and choose two 128-bit

message blocks x_3 and x'_3 with only one nonzero byte in Δx_3. The following are the two structures, each has 2^{64} elements:

$$S'_1 = \{(x_1, x_2, x_3) \mid (x_{2,0}, x_{2,1}, x_{2,5}, x_{2,6}, x_{2,10}, x_{2,11}, x_{2,12}, x_{2,15}) \in \{0,1\}^{64}\},$$
$$S'_2 = \{(x_1, x'_2, x'_3) \mid (x'_{2,0}, x'_{2,1}, x'_{2,5}, x'_{2,6}, x'_{2,10}, x'_{2,11}, x'_{2,12}, x'_{2,15}) \in \{0,1\}^{64}\}.$$

2. Recover the value y_1 using the internal state recovery attack presented in Section 9.1. It is noted that, x_1 is unchanged when we choose different structures to collect enough colliding pairs.
3. Since $y_1 = E_K(x_1)$, K is recovered by exhaustive search directly.
4. When K is recovered, exhaustively search 2^{128} possibilities of L, and only the correct one is suggested by the MAC value C.

Complexity Estimation. The data complexity is the same as the internal state recovery attack on PELICAN, which is about $2^{85.5}$ chosen messages, and the time complexity is dominated by the exhaustive search of the secret key, which is about 2^{128} queries, much lower than the 2^{256} security bound.

We note that even two keys are involved in PC-MAC-AES, the security of the algorithm does not get enhanced.

10 Conclusion

In this part, we adopt the techniques of detecting the inner near-collisions with some specific differences [19,20,21] to implement impossible differential cryptanalysis on PELICAN, MT-MAC-AES and PC-MAC-AES, and all of them take the 4-round AES as the iteration function. Based on a 3-round impossible differential path of AES, we can recover the internal state of PELICAN, which is an equivalent subkey, and the recovery leads to a selective forgery attack. The data complexity is $2^{85.5}$ chosen messages, and the time complexity is $2^{85.5}$ queries. This attack is applicable to MT-MAC-AES and PC-MAC-AES directly. For MT-MAC-AES, it turns to be a subkey recovery attack with the same complexity. For PC-MAC-AES, we can deduce the two secret keys separately with 2^{128} queries and $2^{85.5}$ chosen messages. Our attacks have a complexity greater than the birthday paradox, so they are not covered by the designers proofs.

Acknowledgments. We would like to thank the anonymous reviewers for their helpful comments on the two parts.

References

1. Bahrak, B., Aref, M.R.: Impossible Differential Attack on Seven-Round AES-128. IET Information Security, 2(2), 28–32 (2008)
2. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)

3. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
4. Biham, E., Keller, N.: Cryptanalysis of Reduced Variants of Rijndael. 3rd AES Conference (2000)
5. Biryukov, A., Bogdanov, A., Khovratovich, D., Kasper, T.: Collision Attacks on AES-Based MAC: Alpha-MAC. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 166–180. Springer, Heidelberg (2007)
6. Boesgaard, M., Christensen, T., Zenner, E.: Badger - A Fast and Provably Secure MAC. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 176–191. Springer, Heidelberg (2005)
7. Daemen, J., Rijmen, V.: AES Proposal : Rijndael. The First Advanced Encryption Standard Candidate Conference. NIST AES Proposal (1998)
8. Daemen, J., Rijmen, V.: A New MAC Construction Alred and A Specific Instance Alpha-MAC. In: Gilber, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 1–17. Springer, Heidelberg (2005)
9. Daemen, J., Rijmen, V.: The Pelican MAC Function. IACR ePrint Archive, http://eprint.iacr.org/2005/088 (2005)
10. Huang, J., Seberry, J., Susilo, W.: On the Internal Structure of Alpha-MAC. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 271–285. Springer, Heidelberg (2006)
11. ISO/IEC 9797-1, Information technology - Security Techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using A Block Cipher, ISO 1999.
12. Iwata, T., Kurosawa, K.: OMAC: One-Key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)
13. Kim, J., Biryukov, A., Preneel, B., Hong, S.: On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0, and SHA-1. In: Prisco, R. D., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
14. Kurosawa, K., Iwata, T.: TMAC: Two-Key CBC-MAC. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 265–273. Springer, Heidelberg (2003)
15. Minematsu, K., Tsunoom, Y.: Provably Secure MACs from Differentially-Uniform Permutations and AES-Based Implementations. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 226–241. Springer, Heidelberg (2006)
16. Phan, R. C.-W.: Impossible Differential Cryptanalysis of 7-round Advanced Encryption Standard (AES). Information Processing Letters, 91(1), 33–38 (2004)
17. Preneel, B., van Oorschot, P.: MDx-MAC and Building Fast MACs from Hash Functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
18. Wang, W., Wang, X., Xu, G.: Impossible Differential Cryptanalysis of Pelican, MT-MAC-AES and PC-MAC-AES, Cryptology ePrint Archive, Report 2009/005, http://eprint.iacr.org/2009/005.
19. Wang, X., Wang, W., Jia, K., Wang, M.: New Distinguishing Attack on MAC using Secret-Prefix Method. FSE 2009. To appear.
20. Wang, X., Yu, H., Wang, W., Zhang, H., Zhan, T.: Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC. In: Joux, A. (ed.) EUROCRYPT 2009, LNCS, vol. 5479, pp. 121–133, Springer, Heidelberg (2009)
21. Yuan, Z., Jia, K., Wang, W., Wang, X.: Distinguishing and Forgery Attacks on Alred and Its AES-based Instance Alpha-MAC. Cryptology ePrint Archive, Report 2008/516, http://eprint.iacr.org/2008/516 (2008)
22. Yuval, G.: How to Swindle Rabin. Cryptologia, 3, 187–189 (1979)