Universal Forgery and Key Recovery Attacks on ELmD Authenticated Encryption Algorithm

Aslı Bay[1,⋆], Oğuzhan Ersoy[1,2] and Ferhat Karakoç[1]

[1] TÜBİTAK BİLGEM
{asli.bay,oguzhan.ersoy,ferhat.karakoc}@tubitak.gov.tr
[2] Electrical & Electronics Engineering Dept., Boğaziçi University

Abstract. In this paper, we provide a security analysis of ELmD: a block cipher based Encrypt-Linear-mix-Decrypt authentication mode. As being one of the second-round CAESAR candidate, it is claimed to provide misuse resistant against forgeries and security against block-wise adaptive adversaries as well as 128-bit security against key recovery attacks. We scrutinize ElmD in such a way that we provide universal forgery attacks as well as key recovery attacks. First, based on the collision attacks on similar structures such as Marble, AEZ, and COPA, we present universal forgery attacks. Second, by exploiting the structure of ELmD, we acquire ability to query to the block cipher used in ELmD. Finally, for one of the proposed versions of ELmD, we mount key recovery attacks reducing the effective key strength by more than 60 bits.

Key words: Authenticated encryption, CAESAR, ELmD, Forgery attack, Key recovery

1 Introduction

CAESAR competition [1] (Competition for Authenticated Encryption: Security, Applicability, and Robustness) has been announced in January 2013 aiming at fulfilling the needs of secure, efficient and robust authenticated encryption schemes. In total, 57 candidates are submitted to the competition. These schemes are released to crypto community for their security analysis and around 20 of them were eliminated in the first round of the competition in July 2015. Since then, around 30 candidates compete in the second round, and are being analyzed in terms of their security and efficiency.

ELmD is amongst the second-round CAESAR candidates designed by Datta and Nandi [5]. It is an Encrypt-Linear-mix-Decrypt block cipher

⋆ This author is financially supported by TÜBİTAK (BİDEB 2232, Project No. 115C119)

authentication mode accepting associated data, and its structure is similar to some other authenticated encryption schemes such as AES-COPA [2], Marble [10], and SHELL [12]. ELmD is fully parallelizable and online, that is, i^{th} block of ciphertext only depends on the first i blocks of plaintext. As an optional property, it provides intermediate tag verification in order to fasten verification process and to be secure against block-wise adaptive adversaries. Designers of ELmD claim that the scheme provides nonce misuse resistance against forgery attacks. According to authors' assertion, ELmD provides 62.8-bit security for integrity (forgery attacks) and for privacy (distinguishing attacks). Indeed, they claim that ELmD provides 128-bit security against key recovery attacks that we disprove by applying partial-sum [7] and Demirci-Selçuk meet-in-the-middle attacks [6] on ELmD(6,6) where 6-round AES is used as the block cipher.

Previous Results. As far as we know, ELmD has been analyzed only by Zhang and Wu [13] in terms of both integrity and privacy. Very similar to our internal state recovery, they first find internal state parameter of ELmD by birthday attack and then they provide an almost universal forgery attack with a few queries. For breaking privacy, they propose a truncated differential analysis of reduced version of ELmD (ELmD$(4, 4)$) with 2^{123} time and memory complexities. In [13], the authors consider the internal parameter L generated by only the encryption of zero with 4-round AES, i.e., $L = \text{AES}^4(0)$. However, both the usage of 4 rounds of encryption/decryption and the generation of the internal parameter L with four AES rounds in ELmD are not acceptable in the proposal. Actually, after obtaining an input and output pair of 4-round AES (i.e, $L = \text{AES}^4(0)$), it is feasible to make a meet-in-the-middle analysis to recover the secret key. Previously, similar efforts are made to other CAESAR candidates COPA [11], Marble and AEZ in [8] to find state collisions beyond the birthday bound. Indeed, for AEZ and Marble [8], this attack is used for realizing a key recovery attack.

Our Contribution: In this paper, after obtaining the internal state parameter of ELmD, we make universal forgeries with a few queries to the oracle. Furthermore, by exploiting the structure of ELmD, we are able to query decryption oracle of the block cipher in ELmD. Finally, we mount key recovery attacks on ELmD(6,6) reducing effective key strength more than 60 bits.

Outline of the rest of the paper: In Section 2, a brief description of ELmD is given. Then in Section 3, we show how to recover internal state parameter L, and present universal forgery attacks on ELmD with a few queries to the oracle. In Section 4, we introduce novel methods to gener-

ate special plaintext pairs having relation between their ciphertexts and to query to the decrytion oracle of the block cipher. By using chosen ciphertexts, in Section 5, key recovery attacks on ELmD(6,6) are presented. Section 6 concludes the paper.

2 Brief Description of ELmD

Notation: '$\oplus$': bitwise addition in modulo 2 (exclusive OR), '$\cdot$': field multiplication modulo the polynomial $p(x) = x^{128} + x^7 + x^2 + x + 1$ in $GF(2^{128})$. Also, 0^a denotes a-bit string of 0.

ELmD is a block cipher based Encrypt-Linear-mix-Decrypt authentication mode proposed by Datta and Nandi [5] for CAESAR competition. In the proposal of ELmD, AES-128 [4] is used as the block cipher where the number of rounds can be either 10 or 6. Note that 6-round AES used in ELmD includes whitening-key layer and MixColumns operation at the last round. Hence from now on, AES^{rd} denotes AES with **rd** rounds. For simplicity, E_K is also used for AES-128 in the rest of the paper. In addition, L is a key-depending mask which is generated in two ways; $L = \text{AES}^6(\text{AES}^6(0))$ when rd = 6 and $L = \text{AES}^{10}(0)$ when rd = 10.

The linear mixing function ρ takes two inputs $t, x \in \{0,1\}^{128}$ and produces two outputs $t', y \in \{0,1\}^{128}$ as follows

$$\rho(x,t) = (y,t') : \quad y = x \oplus 3 \cdot t \text{ and } t' = x \oplus 2 \cdot t.$$

Algorithm 1 Processing associated data: IV generation

Input: D, d
Output: IV
for $i = 0$ to $d-1$ **do**
    $DD_i = D_i \oplus 3 \cdot 2^i \cdot L$
    $Z_i = E_K(DD_i)$
    $(Y_i, W'_{i+1}) = \rho(Z_i, W'_i)$
end for
if $|D^*_d| = 128$ **then** $DD_d = D_d \oplus 3 \cdot 2^d \cdot L$
else $DD_d = D_d \oplus 7 \cdot 3 \cdot 2^{d-1} \cdot L$
end if
$Z_d = E_K(DD_d)$
$(Y_d, W'_{d+1}) = \rho(Z_d, W'_d)$
$IV = W'_{d+1}$

Associated data is used to generate IV (see Algorithm 1) which is an input to both encryption/decryption function of ELmD. Let pub and

param be a public message number and the parameter set, respectively, which are both 64 bits, and $D = (D_1, \ldots, D_d^*)$ be an associated data. By construction, the designers of ELmD assign $D_0 = \mathsf{pub} \| \mathsf{param}$ and $W_0' = 0$. The last block of associated data is padded as $D_d = D_d^* \| 10^*$ if $|D_d^*| \neq 128$, otherwise $D_d = D_d^*$.

ELmD has two versions, namely v1.0 and v2.0. ELmD v1.0 was modified by the generation of last message block in such a way that the XOR of previous messages added to this block. Also, **rd** is modified to ELmD(6,6) and ELmD(10,10).

Tagged ciphertext is generated as follows. Let $M = M_1 \| M_2 \| \cdots \| M_\ell^*$ be the message to be encrypted. Padding is performed as $M_\ell = (\oplus_{i=1}^{\ell-1} M_i) \oplus (M_\ell^* \| 10^*)$ if $|M_\ell^*| < 128$, otherwise $M_\ell = (\oplus_{i=1}^{\ell-1} M_i) \oplus M_\ell^*$. ELmD has an intermediate tag option if it is needed, however for the simplicity we mention only tagged ciphertext generation without producing intermediate tags ($t = 0$) in Algorithm 2. ELmD encryption including processing associated data is depicted in Figure 1.

Algorithm 2 Encryption and tag generation without producing intermediate tag ($t = 0$)

Input: ℓ, IV, $M_1, \ldots, M_\ell$, L
Output: $C_1, \ldots, C_\ell, C_{\ell+1}$
$W_0 = IV$
$M_{\ell+1} = M_\ell$
for $i = 1$ to $\ell - 1$ **do**
    $MM_i = M_i \oplus 2^{i-1} \cdot L$
    $X_i = E_K(MM_i)$
    $(Y_i, W_i) = \rho(X_i, W_{i-1})$
    $CC_i = E_K^{-1}(Y_i)$
    $C_i = CC_i \oplus 3^2 \cdot 2^{i-1} \cdot L$
end for
if $|M_\ell^*| = 128$ **then** $MM_\ell = M_\ell \oplus 2^{\ell-1} \cdot L$ and $MM_{\ell+1} = M_{\ell+1} \oplus 2^\ell \cdot L$
else $MM_\ell = M_\ell \oplus 7 \cdot 2^{\ell-2} \cdot L$ and $MM_{\ell+1} = M_{\ell+1} \oplus 7 \cdot 2^{\ell-1} \cdot L$
end if
for $i = \ell$ to $\ell + 1$ **do**
    $X_i = E_K(MM_i)$
    $(Y_i, W_i) = \rho(X_i, W_i)$
end for
$CC_\ell = E_K^{-1}(Y_\ell)$
$C_\ell = CC_\ell \oplus 3^2 \cdot 2^{\ell-1} \cdot L$
$CC_{\ell+1}^* = E_K^{-1}(Y_{\ell+1} \oplus 1)$
$C_{\ell+1}^* = CC_{\ell+1}^* \oplus 3^2 \cdot 2^\ell \cdot L$
if $|M_\ell^*| \neq 128$ **then** $C_{\ell+1} = trunc(C_{\ell+1}^*)_{|M_\ell^*|}$
else $C_{\ell+1} = C_{\ell+1}^*$
end if

ELmD decrypts and verifies a given tagged ciphertext pair in three steps. First of all, IV is produced by using pub, param, and D as in Algorithm 1. Afterwards, the tagged ciphertext is decrypted as an inversion of Algorithm 2, and then tag is verified when $M_{\ell+1} = M_{\ell}$. Once the tag is verified, plaintext is released otherwise $\perp$ is returned.

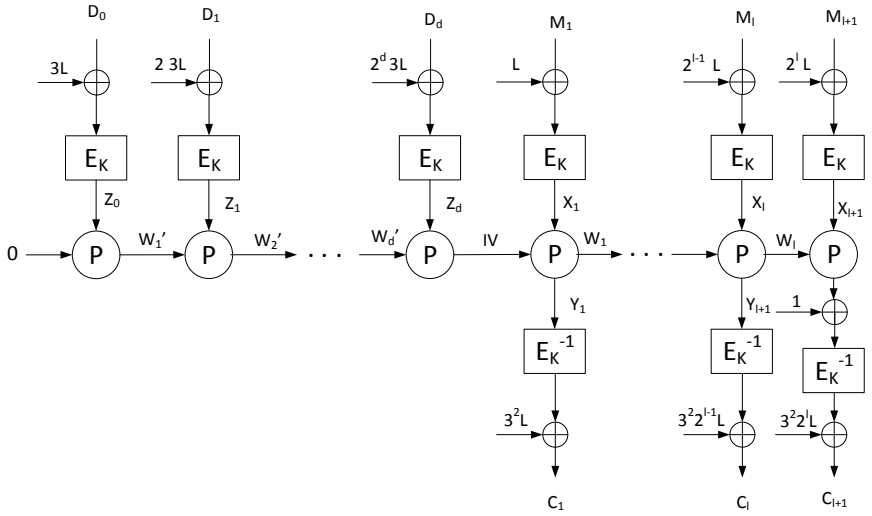

Fig. 1. Processing associated data and the generation of tagged ciphertext in ELmD.

3 Universal Forgery Attack on ELmD

In this section, we present universal forgery attacks on ELmD. First, we recover ELmD state L by collision search of ciphertexts. Using L, we can make universal forgery attack on ELmD. Before going into details, we briefly describe the two main forgery models:

- **Existential Forgery** is the generation of a valid ciphertext and tag pair for an unspecified message which is not previously queried to an oracle.
- **Universal Forgery** is the generation of ciphertext and tag pair for a given message which is not previously queried to an oracle.

3.1 Recovering Internal State Parameter L

Similar to state recovery attacks of COPA and Marble [11, 8], we recover ELmD state L by collision search of ciphertexts which has approximate complexity 2^{65} due to birthday attack as follows.

For a fixed D_0, let $(D, M) = (D_1, M_1) = (\alpha, M)$ and $(D', M') = (D'_1, M'_1) = (\beta, M)$ be two set of message pairs including associated data where α and β take all possible values from the set $\{0, 1, \ldots, 2^{64} - 1\}$ and α is an incomplete block and β is complete, i.e., $|\alpha| = 64$ and $|\beta| = 128$. Here, we aim to exploit different parameter mask additions to the last blocks of associated data when the block is incomplete. Also, we pick α and β such that $(\alpha \| 10^{63}) \oplus \beta$ scans all values in $\mathbb{F}_{2^{128}}$.

After message pairs are queried, we search a collision in the first ciphertexts C_1 and C'_1, i.e., $C_1 = C'_1$. According to the birthday attack, around $2 \cdot 2^{64}$ message pairs is enough to construct a collision. This collision implies that messages' corresponding IV values are equal, i.e., $IV = IV'$. As we use the same D_0 for two messages implying the same internal chaining value ($W'_1 = W''_1$), we obtain $DD_1 = DD'_1$ (see Figure 2). We recover L by solving

$$D'_1 \oplus 3 \cdot 7 \cdot L = D_1 \oplus 3 \cdot 2 \cdot L, \tag{1}$$

since L is the only unknown in the equation, where $D_1 = \alpha \| 10^{63}$ and $D'_1 = \beta$.

3.2 Forgery

Once we recover L, we can make universal forgery attacks on ELmD by making a few queries to the oracle.

A Universal Forgery Attack Let $(D, M) = (D_1, \ldots, D_{d-1}, D_d, M_1, \ldots, M_{\ell-1}, M_\ell)$ be targeted associated data and message pair with assigned $D_0 = \mathsf{pub} \| \mathsf{param}$, where $|D_d| = 128$. Compute D'_d such that $D'_d \| 10^* = D_d \oplus 2^d \cdot 3L \oplus 7 \cdot 2^{d-1} \cdot 3L$ and $|D'_d| < 128$. Note that because of the padding rule, we can always obtain D'_d with $|D'_d| < 128$.

Query $(D', M) = (D_1, \ldots, D_{d-1}, D'_d, M_1, \ldots, M_{\ell-1}, M_\ell)$ with the same D_0 and obtain the corresponding ciphertext and tag pair as $(\widetilde{C}, \widetilde{T})$. Due to the choice of associated data, D and D' produce the same IV. Hence, the corresponding ciphertext and tag pair (C, T) of (D, M) is equal to that of (D', M), i.e., $(C, T) = (\widetilde{C}, \widetilde{T})$. Note that the same attack also works for $|D_d| < 128$ case. In a similar manner, a $|D'_d| = 128$ block can

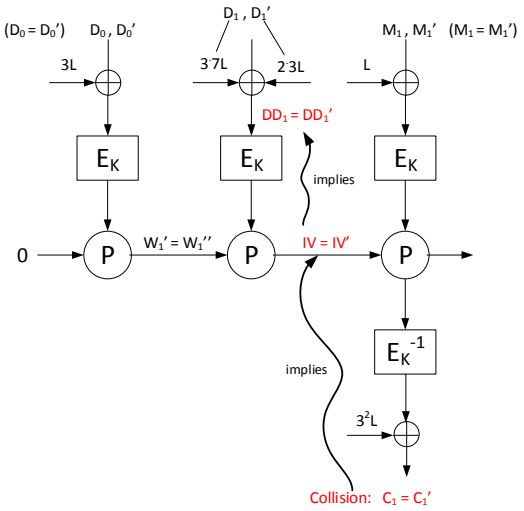

Fig. 2. Recovering L by finding a collision in $(t = 0)$

be chosen where $D'_d = D_d \| 10^* \oplus 2^d \cdot 3L \oplus 7 \cdot 2^{d-1} \cdot 3L$, and the rest of the attack is the same. Therefore, this forgery attack works for any associated data and message pair.

Another Universal Forgery Attack Here we present another forgery for the same (D_0, D, M) triple using only completed blocks. First, query $M_1 = D_0 \oplus 3L \oplus L$ without D, and obtain C_1. Then, query (D', M) such that $D'_0 = D_0$, $D'_1 = C_1 \oplus 3^2L \oplus 2 \cdot 3L$, $D'_{i+2} = D_i \oplus 2^i \cdot 3L \oplus 2^{i+2} \cdot 3L$ for $i = 0, 1, \ldots, d$ and obtain ciphertext C and tag T. It can be seen that this (C, T) pair is also valid for (D, M).

Note that this forgery attack introduces an important ability of generating a pair of plaintexts such that one of the corresponding ciphertext is half of the other one. These related plaintext pairs are explained in Section 4, and used for key recovery in Section 5.

Forgery of Intermediate tags (when $t \neq 0$) In the proposal of ELmD, the authors state that "*When intermediate tags are used i.e. $t \neq 0$, if the forger can compute a valid intermediate tag such that the ciphertext up to that is not identical to any of previous ciphertexts then the forger succeeds*". Once L is known, we can make a universal forgery attack for the version of ELmD with intermediate tags. Without any further details,

it can be seen that the previously given forgery attacks also applies when $t \neq 0$. Because both attacks only uses the associated data.

4 Exploiting the Structure of ELmD

In this section, we explore the block cipher used in ELmD by exploiting the general structure of the authenticated encryption algorithm where the bottom function is the decryption mode of the upper one. First, using the recovered L value, we can obtain two types of plaintext pairs:

1. For any P_1 and μ, (P_1, P_2) pair such that $\mu \cdot E(P_1) = E(P_2)$.
2. For any Δ, (Q_1, Q_2) pair such that $E(Q_1) = E(Q_2) \oplus \Delta$.

Using these special plaintext pairs, we can obtain plaintext and corresponding ciphertext pairs of the encryption block cipher $E_K(\cdot)$ or AES^{rd}. Especially, we can query any ciphertext to the decryption mode of the cipher.

Following attacks are mostly explained for the maskless version of ELmD. Since we know the L value, we can easily switch from (D, M, C) triple to (DD, MM, CC) triple and vice versa, where $D_i = DD_i \oplus 2^{i-1} \cdot 3L$, $M_i = MM_i \oplus 2^{i-1}L$ and $C_i = CC_i \oplus 2^{i-1} \cdot 3^2 L$. In other words, we can query (DD, MM) and obtain CC values. For the simplicity, we usually use (DD, MM, CC) triples. It is important to note that the last message block cannot be controlled since $MM_{\ell+1} = MM_\ell \oplus 2^{\ell-1}L \oplus 2^\ell L$.

4.1 2-multiplicative Pairs: (R_1, R_2) with $2 \cdot E(R_1) = E(R_2)$

Initially, for any given/fixed $D_0 = \mathsf{pub} \| \mathsf{param}$, we make a query for one block message $MM_1^1 = DD_0$ without an additional associated data and obtain the corresponding ciphertext and tag pair (C^1, T^1). As seen in Figure 3, $IV^1 = E_K(DD_0)$. Because of our message choice, X_1^1 is also equal to IV^1 and therefore $Y_1^1 = 2 \cdot IV^1$. Even without knowing IV^1 value, we obtain CC_1^1 such that $E_K(CC_1^1) = 2 \cdot IV^1 = 2 \cdot E_K(DD_0)$. Here, it is important to note that D_0 has a special structure and cannot take any 128-bit value. For any R_1, using the same D_0, query $DD_1^2 = CC_1^1, MM_1^2 = MM_2^2 = R_1$ and obtain the corresponding ciphertext and tag pair (C^2, T^2). It can be seen that $IV^2 = \rho(IV^1, 2 \cdot IV^1) = 0$ and therefore $X_1^2 = W_1^2 = E_K(MM_1^2)$. $W_1^2 = E_K(MM_1^2) = X_2^2$ implies $Y_2^2 = 2 \cdot X_2^2$ and $E_K(CC_2^2) = 2 \cdot E_K(MM_1^2)$. As can be seen in Figure 3, by setting $R_2 = CC_2^2$, we obtain (R_1, R_2) pair such that $2 \cdot E(R_1) = E(R_2)$. The complexity to obtain N such 2-multiplicative pairs is only $N + 1$

queries if the same $D_0 = \mathsf{pub}\|\mathsf{param}$ is used. Therefore, the complexity of getting a 2-multiplicative pair is approximately one block query.

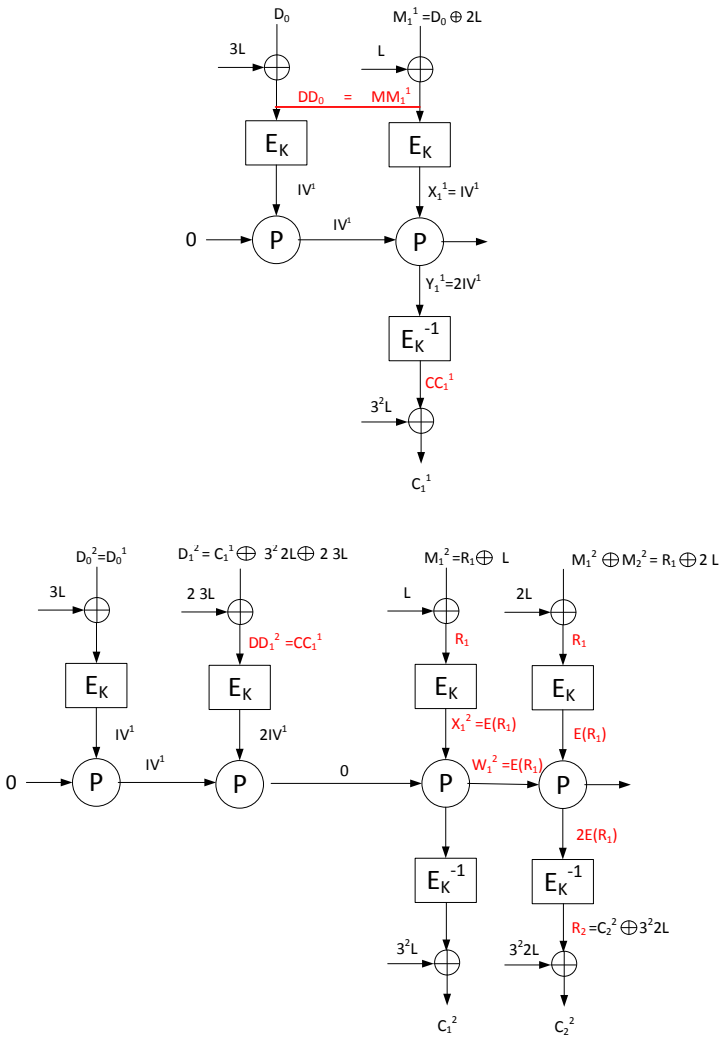

Fig. 3. 2-multiplicative Pairs

4.2 μ-multiplicative Pairs: (P_1, P_2) with $\mu \cdot E(P_1) = E(P_2)$

Here, we present a method to generate (P_1, P_2) pair satisfying $\mu \cdot E(P_1) = E(P_2)$ for any P_1 and μ values with the help of observations in the previous part. First, for a given P_1, we obtain the plaintext R_2 such that $2 \cdot E(P_1) = E(R_2)$. Also, we arrange associated data to make $IV = 0$.

Let $\mu' = 3^{-1}\mu \oplus 1$ where 3^{-1} represents the multiplicative inverse of 3 in the given field. It can be seen that any $\mu' \in \mathbb{F}_{2^{128}}$ can be represented as $2^{127} \cdot m_1 \oplus 2^{126} \cdot m_2 \oplus \cdots \oplus 2 \cdot m_{127} \oplus m_{128}$ where $m_i \in \{1, 2\}$.

As shown in Figure 4, by querying 129-block message with $MM_i = R_{m_i}$ for $i = 1, \ldots, 128$ and $MM_{129} = R_1$, we can obtain the plaintext $P_2 = CC_{129}$ satisfying $E(P_2) = \mu \cdot E(P_1)$. The complexity to obtain any multiplicative pair of a given P_i is about 2^7 block encryptions. In other words, obtaining the plaintext of a given multiple of a given ciphertext costs 2^7 block ELmD encrytions which is approximately 2^8 block cipher calls.

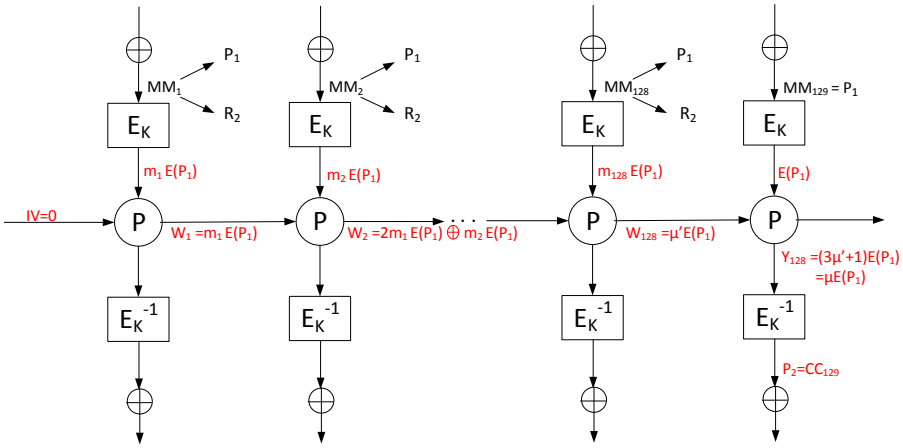

Fig. 4. μ-multiplicative Pairs

Note that using μ-multiplicative pairs, we can obtain the plaintext P_0 satisfying $E(P_0) = 0 \cdot E(\cdot) = 0$.

4.3 1-difference Pairs: (R_1, R_2) with $E(R_1) = E(R_2) \oplus 1$

In this part, we show how to construct (R_1, R_2) pairs such that $E(R_1) = E(R_2) \oplus 0^{127}1$ by using 2-multiplicative pairs (see Figure 5). For any D_0 (resp. M_1), we can obtain D_1 (resp. M_2) such that $E(DD_1) = 2 \cdot E(DD_0)$ (resp. $E(MM_2) = 2 \cdot E(MM_1)$). By querying the corresponding associated data and message pair, we can obtain $R_1 = MM_3$ and $R_2 = CC_3$ satisfying $E(R_1) = E(R_2) \oplus 1$. The complexity to obtain a 1-difference pair is simply a query of 1 associated data block and 2 message blocks where associated data and message blocks are 2-multiplicative pairs.

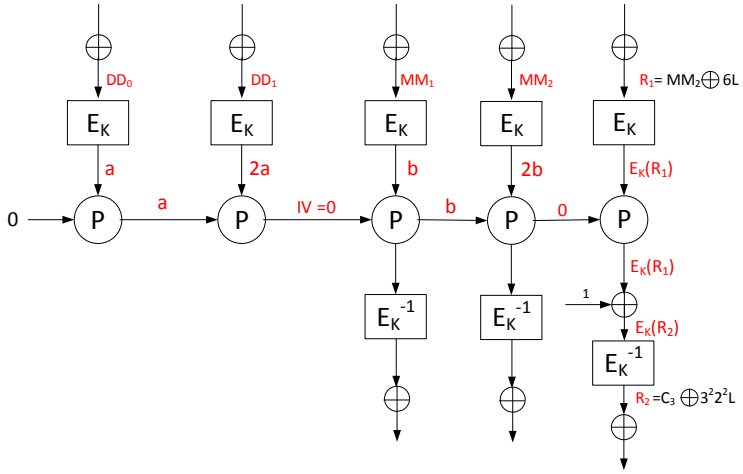

Fig. 5. 1-difference Pairs

4.4 Δ-difference Pairs: (Q_1, Q_2) with $E(Q_1) = E(Q_2) \oplus \Delta$

First, we generate a 1-difference pair: $\{R_1, R_2\}$ where $E(R_1) = E(R_2) \oplus 0^{127}1$. Then, for any Δ, compute $\delta = \delta_1 \| \delta_2 \| \cdots \| \delta_{128}$ such that $3 \cdot \delta = \Delta$ over the defined field.

We construct two messages M, M' each containing 129 blocks with the same associated data D such that

$$MM_i = R_1 \quad \text{and} \quad MM'_i = R_{\delta_i + 1} \quad \text{for} \quad i = 1, 2, \ldots, 129$$

where $\delta_{129} = 0$.

As illustrated in Figure 6, 129^{th} ciphertext blocks of (D, M) and (D, M') differ by Δ. Here, we briefly, explain the differential path of two messages (D, M) and (D, M'). As their associated data are equal, they will provide the same IV, that is $IV \oplus IV' = 0$. After processing of the first blocks of two messages R_1 and R_{δ_1+1} in the upper layer of encryption, we will get difference in X_1's as $\Delta X_1 = X_1 \oplus X'_1 = \delta_1$. Since $\Delta IV = 0$, $\Delta W_1 = W_1 \oplus W'_1 = \delta_1$. For the second message blocks R_1 and R_{δ_2+1}, we get $\Delta X_2 = X_2 \oplus X'_2 = \delta_2$. Then, we have $\Delta W_2 = W_2 \oplus W'_2 = 2\delta_1 + \delta_2$. Similarly, after the encryption of 128th blocks, we have $\Delta W_{128} = W_{128} \oplus W'_{128} = 2^{127}\delta_1 + 2^{126}\delta_2 + \cdots + \delta_{128} = \delta$. Finally, as we choose the last message blocks equal, we have $\Delta X_{129} = X_{129} \oplus X'_{129} = 0$. Since no difference is coming from upper encryption layer $\Delta Y_{129} = Y_{129} \oplus Y'_{129} = 3 \cdot \Delta W_{128} = 3 \cdot \delta = \Delta$. Hence, we obtain plaintexts $Q_1 = CC_{129}$ and $Q_2 = CC'_{129}$ having required ciphertext difference: $E(Q_1) = E(Q_2) \oplus \Delta$. Note that by changing the last message block, we can get several message pairs having desired ciphertext difference.

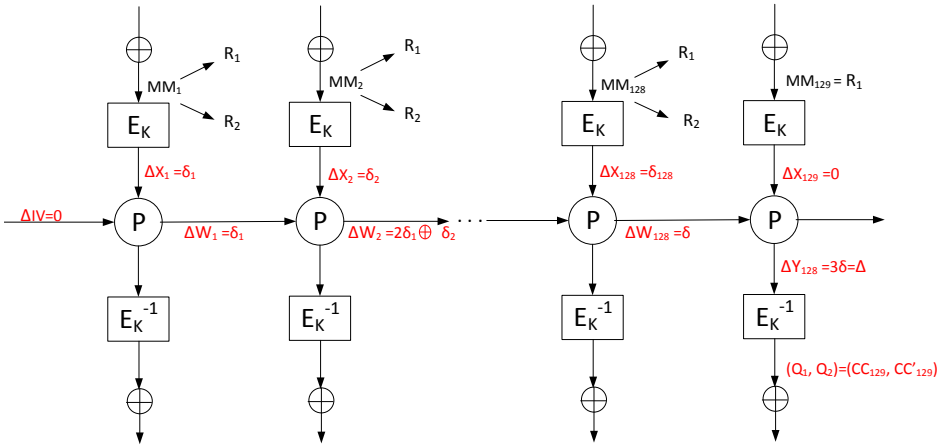

Fig. 6. Δ-difference Pairs

4.5 Querying Decryption Oracle of the Block Cipher

Here, we describe how to query inner block cipher of ELmD, AES^{rd}. Since, we can obtain any multiple of a given ciphertext in μ-multiplicative pairs, it is obvious that any ciphertext can be queried, i.e., plaintext of a given ciphertext can be obtained, if the decryption of $0^{127}1$ is known.

First, using 1-difference pairs, we obtain a pair (R_1, R_2) with $E(R_1) = E(R_2) \oplus 1$. Then, using μ-multiplicative pairs, we acquire R_3 such that $3^{-1}E(R_1) = E(R_3)$. By querying associated data satisfying $IV = 0$ and message with $MM_1 = R_3$, $MM_2 = R_2$, we obtain CC_2 which is equal to decryption of 1, i.e., $E(CC_2) = 0^{127}1$. After obtaining decryption of 1, we can query any ciphertext with the help of μ-multiplicative pairs.

This property enables us to mount a chosen ciphertext attack.

5 Key Recovery

The encryption function E_K used in ElmD is either 6-round AES (AES^6) or 10-round AES (AES^{10}) depending on the application. For both versions of ELmD, the designers claim that ELmD provides 128 bits of security against plaintext and key recovery attacks. In this section, we show that this claim is not valid if the function E_K is AES^6.

In Section 4, after recovering L parameter, it is shown how to obtain corresponding plaintext for any given ciphertext in a time complexity of about 2^8 encryption operations. As a result, we can mount attacks on 6-round AES with chosen ciphertexts. In [7], by using partial sums an attack on 6-round AES was given with a time and data complexities of 2^{44} and $2^{34.6}$, respectively in chosen plaintext scenario. This attack can be easily adapted to chosen ciphertext case because of the AES structure. MixColumns and AddRoundKey operations can be swapped with applying the inverse of MixColumns to the round key. As known, the inverse of AES without the MixColumns operation in the last round has the same structure with AES, the similar attack can be applied. Note that the MixColumn operations at the end of the cipher is not important because ciphertexts can be easliy manipulated. The total time complexity of key recovery is $2^{65} + 2^8 \times 2^{34.6} + 2^{44} \approx 2^{65}$ which is dominated by the cost of recovery of L.

In addition, we propose a Demirci-Selçuk meet-in-the-middle attack [6] using the distinguisher on 3-round AES [9]. This attack also uses chosen ciphertexts. The time and data complexities of this attack is 2^{66} and 2^{33}, respectively. With this attack the time complexity of key recovery attack on ELmD is $2^{65} + 2^8 \times 2^{33} + 2^{66} \approx 2^{66.6}$ encryptions. Even though the time complexity is relatively higher than the previous attack, this attack uses relatively less data and illustrates Demirci-Selçuk MITM in a splice-and-cut [3] perspective.

While presenting the attack we use the following notation. AES^6 consists of 6 full rounds of AES with initial key whitening and supports a key

size of 128 bits. One full round of AES is composed of SubBytes (SB), ShiftRows (SR), MixColumns (MC) and AddRoundKey (AK) operations [4]. The whitening key and i-th round key ($i \in \{1, 2, 3, 4, 5, 6\}$) are denoted by k_0 and k_i respectively. We use x_i, y_i, z_i and w_i to represent the blocks in i-th round before the SubBytes, ShiftRows, MixColumns and AddRoundKey operations respectively where the input of the first round is $x_1 = P \oplus k_0$ and P is the plaintext. In the case of swapping MixColumns and AddRoundKey operations we denote the round key as $u_i = \mathrm{MC}^{-1}(k_i)$ and the state after round key addition as $\bar{w}_i$. Also x_i^j, y_i^j, z_i^j, w_i^j and $\bar{w}_i^j$ denotes the blocks for j-th plaintext and $a(m, n, ..., l)$ are used for $m, n, ..., l$-th bytes of a block a. The orders of 128-bit blocks' bytes in 4×4 matrix of bytes is as conventional, that is the first row is composed of 0, 4, 8 and 12-th bytes of 128-bit block where 0-th byte is the left-most byte.

The attack is given in Algorithm 3 and depicted in Figure 7. The number of bits guessed in the attack is 144 and the probability that a wrong guess passes the condition in Step 10 is 2^{-144}. Thus, with the correct guess, a wrong one can be returned by the algorithm. In Step 3 in Algorithm 3, $2^{80} \times 19 \times \frac{10}{16 \times 6} \approx 2^{81}$ encryptions are performed by guessing 10 bytes. Note that this step can be done offline. For a ciphertext the time complexity of getting the corresponding plaintext is approximately 2^8 encryptions as mentioned in Section 4. Thus the number of operations performed in Step 6 is $2^{32} \times 2^8 = 2^{40}$ encryptions. In Step 9, 144-bit differences are computed performing $2^{64} \times 19 \times \frac{10}{16 \times 6} \approx 2^{65}$ encryption operations. As a result the time complexity of Algorithm 3 is 2^{81} offline and 2^{65} online encryptions. To store the 144-bit difference for possible 2^{80} values $2^{80} \times 144$-bit memory is required. Note that in the attack 12 bytes of $\bar{w}_6^j$ are fixed to a constant 0. Thus the attack needs 2^{32} chosen ciphertexts and corresponding plaintexts.

Notice that with this attack we obtain 4 bytes of k_0 so far. With slight modifications in the attack it can be seen easily that other 4 bytes of k_0 can be found. Remaining 64 bits of the key can be recovered by brute force. The total complexity of recovering 128-bit key will be $2 \times 2^{81} = 2^{82}$ offline and $2 \times 2^{65} + 2^{64} \approx 2^{66}$ online encryptions.

The memory and data complexities will be $2 \times 2^{80} \times 144$-bit memory and $2 \times 2^{32} = 2^{33}$ data respectively. Note that the offline time complexity can be reduced to 2^{74} by removing the guess of $y_5(0)$ from the offline step and adding 8-bit guess for $u_5(0)$ to online step. In that case the time complexity of online step will be 2^{74} encryption operations.

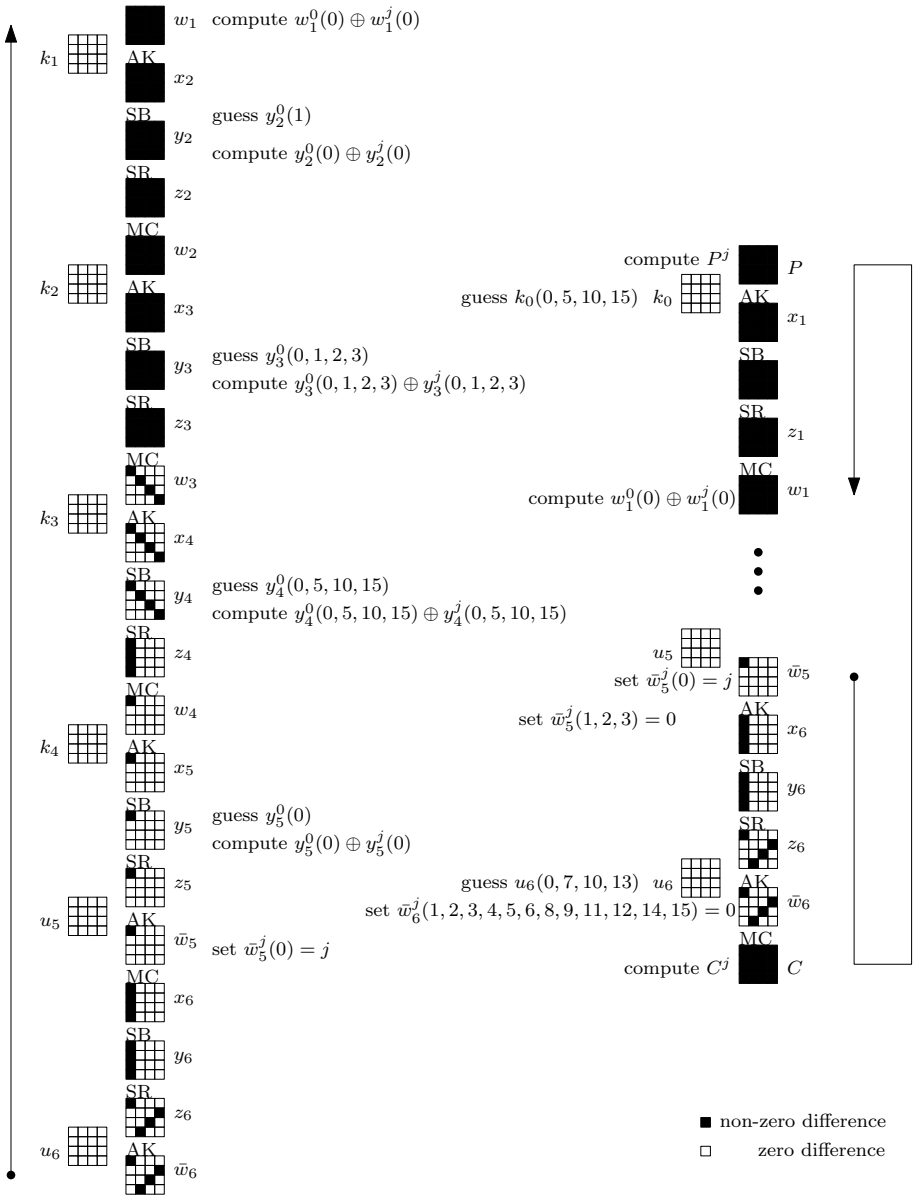

Fig. 7. Demirci-Selçuk MITM attack on 6-round AES. The offline and online steps are on the left-hand and right-hand sides, respectively.

Algorithm 3 Demirci-Selçuk MITM Attack on 6-round AES.

Take 19 different values for $\bar{w}_5^j(0,1,2,3)$ and $\bar{w}_6^j(1,2,3,4,5,6,8,9,11,12,14,15)$ such that $\bar{w}_5^j(0) = j$, and the other bytes are 0 for $0 \leq j \leq 18$.
for each possible values of $y_5^0(0)$, $y_4^0(0,5,10,15)$, $y_3^0(0,1,2,3)$ and $y_2^0(0)$ **do**
    Compute the difference $(w_1^0(0) \oplus w_1^1(0), w_1^0(0) \oplus w_1^2(0), ..., w_1^0(0) \oplus w_1^{18}(0))$ and store it in Table T.
end for
for each possible values of $u_6(0,7,10,13)$ **do**
    Compute C^j's
    Find P^j's by using the method in Section 4.
    for each possible values of $k_0(0,5,10,15)$ **do**
        Compute the difference $(w_1^0(0) \oplus w_1^1(0), w_1^0(0) \oplus w_1^2(0), ..., w_1^0(0) \oplus w_1^{18}(0))$ and find the difference in Table T.
        if a match found **then**
            Return $k_0(0,5,10,15)$ as the correct key
        end if
    end for
end for

6 Conclusion

ELmD is an a block cipher based Encrypt-Linear-mix-Decrypt authentication mode submitted to CAESAR Competition. It is claimed to be strong against misuse forgery attacks, block-wise adaptive adversaries and key recovery attacks with 128-bit security. This work provides universal forgery attacks against ELmD. Furthermore, we disprove the 128-bit security claim of ELmD by applying two key recovery attacks, namely partial-sum and Demirci Selçuk meet-in-the-middle attacks with 2^{65} and $2^{66.6}$ time complexities, respectively.

References

1. CAESAR – Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html
2. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: AES-COPA v.1, Submission to the CAESAR competition. http://competitions.cr.yp.to/round1/aescopav1.pdf (March 2014)
3. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique cryptanalysis of the full AES. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7073, pp. 344–371. Springer (2011), http://dx.doi.org/10.1007/978-3-642-25385-0_19
4. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002), http://dx.doi.org/10.1007/978-3-662-04722-4

5. Datta, N., Nandi, M.: ELmD v2.0, Submission to the CAESAR competition. https://competitions.cr.yp.to/round2/elmdv20.pdf (August 2015)
6. Demirci, H., Selçuk, A.A.: A meet-in-the-middle attack on 8-round AES. In: Nyberg, K. (ed.) Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5086, pp. 116–126. Springer (2008), http://dx.doi.org/10.1007/978-3-540-71039-4_7
7. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved cryptanalysis of rijndael. In: Schneier, B. (ed.) Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1978, pp. 213–230. Springer (2000), http://dx.doi.org/10.1007/3-540-44706-7_15
8. Fuhr, T., Leurent, G., Suder, V.: Collision attacks against CAESAR candidates - forgery and key-recovery against AEZ and marble. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9453, pp. 510–532. Springer (2015), http://dx.doi.org/10.1007/978-3-662-48800-3_21
9. Gilbert, H., Minier, M.: A collision attack on 7 rounds of rijndael. In: AES Candidate Conference. pp. 230–241 (2000)
10. Guo, J.: Marble Specification Version 1.0, Submission to the CAESAR competition. http://competitions.cr.yp.to/round1/marblev10.pdf (March 2014)
11. Lu, J.: On the security of the COPA and marble authenticated encryption algorithms against (almost) universal forgery attack. IACR Cryptology ePrint Archive 2015, 79 (2015), http://eprint.iacr.org/2015/079
12. Wang, L.: SHELL v2.0, Submission to the CAESAR competition. https://competitions.cr.yp.to/round2/shellv20.pdf (August 2015)
13. Zhang, J., Wu, W.: Security analysis of caesar second-round candidate: Elmd. www.escience.cn/system/download/77967 (2016)