Journal of Information Security and Applications 70 (2022) 103342

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Plaintext recovery and tag guessing attacks on authenticated encryption algorithm COLM

Sırrı Erdem Ulusoy [a,b,*], Orhun Kara [b,c], Mehmet Önder Efe [a]

[a] Hacettepe University, Graduate School of Science and Engineering, Department of Computer Engineering, 06532, Beytepe, Ankara, Turkey
[b] Scientific and Technological Research Council of Turkey, National Research Institute of Electronics and Cryptology, 41470, Gebze, Kocaeli, Turkey
[c] İzmir Institute of Technology, Faculty of Science, Department of Mathematics, 35430, Urla, Izmir, Turkey

ARTICLE INFO

Keywords:
COLM
CAESAR competition
Authenticated encryption with associated data
AEAD
Universal forgery
Tag guessing
Plaintext recovery
Key recovery
AES
Biclique
Impossible differential
Meet-in-the-middle
SEBC
SDBC

ABSTRACT

There are three main approaches related to cryptanalysis of Authenticated Encryption with Associated Data (AEAD) algorithms: Simulating the encryption oracle (universal forgery attack), simulating the decryption oracle (plaintext recovery attack) and producing the valid tag of a given ciphertext (tag guessing attack). In this work, we analyze the security of COLM in these approaches. COLM is one of the AEAD algorithms chosen in the final portfolio for defense-in-depth use case of the CAESAR competition. The ciphers in this portfolio are supposed to provide robust security with their multiple layered defense mechanisms. The main motivation of this work is to examine if COLM indeed satisfies defense-in-depth security. We make cryptanalysis of COLM, particularly in the chosen ciphertext attack (CCA) scenario, once its secret whitening parameter $L = E_K(0)$ is recovered. To the best of our knowledge, we give the first example of querying an EME/EMD (Encrypt-linearMix-Encrypt/Decrypt) AEAD scheme in its decryption direction for arbitrary ciphertexts, not produced previously by the oracle, namely either a forgery or tag guessing attack. We construct SEBC/SDBC (Simulation models of the Encryption/Decryption oracles of the underlying Block Cipher) of COLM, thereby forming the first examples of these models of an authenticated EME scheme simultaneously. The combination of our SEBC/SDBC is a powerful tool to mount a universal forgery attack, a tag guessing attack and a plaintext recovery attack. All of these attacks have polynomial time complexities once L is recovered in the offline phase, indicating that the security of COLM against plaintext recovery and tag guessing attacks is limited by the birthday bound. Apart from exploiting SEBC/SDBC, we mount a pair of plaintext recovery attacks and another universal forgery attack. Finally, we make some suggestions to prevent our attacks.

1. Introduction

In information security, confidentiality is protected by an encryption algorithm while authenticity is provided by a digital signature or a Message Authentication Code (MAC). An Authenticated Encryption with Associated Data (AEAD) [1] scheme provides both confidentiality and authenticity of a message with a single key, simultaneously. It produces a ciphertext (CT) and a tag from an input consisting of an associated data (AD) and a plaintext (PT). Note that the PT is not released unless the tag is verified in the decryption. Advanced Encryption Standard - Galois/Counter Mode (AES-GCM) [2] is one of the leading AEAD algorithms. It has a wide range of usage across the Internet applications such as Transport Layer Security (TLS) and Internet Protocol Security (IPsec).

The design of secure AEAD schemes is one of the most popular topics in cryptography nowadays. "The Competition for Authenticated Encryption: Security, Applicability, and Robustness" (CAESAR) was launched by the International Cryptologic Research Community in January 2014 [3]. Also, the algorithms submitted to the ongoing lightweight cryptography competition organized by National Institute of Standards and Technology are supposed to be AEAD schemes [4]. Besides the algorithms submitted to CAESAR or the NIST competition, there have been several AEAD schemes in the literature such as [5–8] for the last five years.

The ciphers selected for the final portfolio of the CAESAR competition are expected to become possible alternatives of AES-GCM, outperforming it in either performance or security. The final portfolio is classified in three categories as lightweight, high-performance and

* Corresponding author at: Hacettepe University, Graduate School of Science and Engineering, Department of Computer Engineering, 06532, Beytepe, Ankara, Turkey.
E-mail addresses: erdem.ulusoy@tubitak.gov.tr (S.E. Ulusoy), orhunkara@iyte.edu.tr (O. Kara), onderefe@hacettepe.edu.tr (M.Ö. Efe).

https://doi.org/10.1016/j.jisa.2022.103342

Available online 13 October 2022

Table 1
Attacks against COPA, ELmD and COLM. EF: Existential Forgery, UF: Universal Forgery, TagG: Tag Guessing, PtR: Plaintext Recovery, SEBC/SDBC: Simulations of Encryption/Decryption Oracles of the Underlying Block Cipher, QDOAC: Querying AEAD Decryption Oracle for Arbitrary Ciphertexts. As can be seen from the table, for the first time TagG, PtR and decryption of AEAD attacks are mounted.

	AEAD	EF	UF	TagG	PtR	SEBC/SDBC	QDOAC
[9]	COPA	No	Yes	No	No	None	No
[10]	ELmD	No	No	No	No	SDBC	No
[11]	COLM	No	No	No	No	None	No
[12]	COLM	Yes	Yes	No	No	SEBC	No
This work	COLM	Yes	Yes	Yes	Yes	SEBC/SDBC[a]	Yes

[a]This SDBC is the only simulation model built by querying an AEAD decryption (itself a forgery attack).

defense-in-depth use cases. The competition has attracted the cryptography community worldwide and 57 candidate algorithms were submitted. The ciphers have been analyzed extensively (e.g.[9–20]). There are also various studies related to efficient implementations of the candidates (e.g. [21–26]).

Two AEAD schemes which qualified for the second round of the CAESAR competition, COPA [27] and ELmD (Encrypt-Linear mix-Decrypt) [28], are merged to one cipher, COLM [29], to inherit their best properties (see Table 2 in Section 2). It is an authenticated Encrypt-linearMix-Encrypt (EME) scheme whose underlying block cipher is AES-128 [30].

COLM and Deoxys [31] are the winners of the CAESAR competition in the defense-in-depth use case [32]. The both ciphers chosen in this use case are supposed to provide highly robust security services by utilizing multiple layers of security mechanisms. So, even if some of the defense layers of such a cipher fail, the cipher is expected to sustain performing its functions without depriving of the authenticity level or the security strength. The most critical mechanism is stated as providing authenticity against nonce misuse attacks where the attacker can control the nonce, [33].

Three main approaches for cryptanalysis of an AEAD algorithm can be listed as universal forgery attack, plaintext recovery attack and tag guessing attack. The encryption oracle of an AEAD scheme is simulated to construct both the ciphertext and the tag of any given plaintext in a universal forgery attack while the simulation model of the decryption oracle of an AEAD scheme is built to compute the plaintext of a given ciphertext in a plaintext recovery attack. As for a tag guessing attack, the goal is to compute the valid tag of a given ciphertext.

In this work, we analyze the security of COLM in these three approaches under the assumption that the whitening mask $L = E_K(0)$ is known, which can be recovered by Lu's method in 2^{65} nonce misuse queries [9,11]. We build Simulation models of the Encryption/Decryption oracles of the underlying Block Cipher (SEBC/SDBC), yielding attacks in all three approaches. These two simulations together provide a powerful tool to an attacker so that she can query the COLM decryption oracle even for ciphertexts that have no tags. Unlike the previous simulation models in [10,12], our simulation model queries the AEAD scheme in decryption direction. Any decryption query of an AEAD is itself either a forgery or a tag guessing attack. To the best of our knowledge, our SDBC is the first example of querying an authenticated EME/EMD scheme in the decryption direction for arbitrary ciphertexts. We compare our results with the previous works in Table 1. COLM can still be considered secure against collision based forgeries. However, we show that the security of COLM against plaintext recovery and tag guessing attacks is limited by the birthday bound. On the other hand, the security level of an AEAD is expected to be the minimum of the key length or the block length against these attacks. In particular, this must be a strict security requirement for a cipher chosen in a defense-in-depth portfolio.

1.1. Related work

As a member of the final portfolio, COLM has drawn significant interest of the cryptography community both in security analyses and implementations. Several cryptanalysis works related to COLM and its ancestors have been published for half a decade. In [9], Lu suggests a nonce misuse method to recover the whitening mask $L = E_K(0)$, the encryption of the zero-vector with the underlying block cipher using the key K, for COPA [27] and Marble [34] and the author mounts almost universal forgery attack against COPA and Marble. In [11], Forler et al. mount a semi-universal forgery attack against COLM once they recover L by Lu's method. In [10], Bay et al. also recover the L parameter of ELmD and then use it for mounting some forgery attacks. In addition to forging ELmD, they adapted Demirci and Selcuk Meet-in-the-Middle (MitM) Attack [35] for the key recovery of 6-round AES-128, one of the underlying block cipher options in ELmD. Their attacks cannot go beyond forgery attacks against the full-round AES option of ELmD.

In [13], Datta et al. introduce three INT-RUP (integrity under release of unverified plaintext) attacks to mount an existential forgery against COLM and they state that the attack is no more applicable if the intermediate tags are used. In [36], the CAESAR finalists along with AES-GCM are assessed in terms of security and performance. Consequently, COLM is listed as one of the most resilient ciphers. In [12], Vaudenay and Vizár mount various forgery attacks against the candidates selected for the 3rd round of the CAESAR competition. Their existential forgery and semi-universal forgery attacks against COLM are based on the birthday bound. Both of the attacks exploit a collision deduced from ADs whose corresponding PTs are chosen by the attacker. Moreover, Vaudenay and Vizár mount a semi-universal forgery attack by permuting two AD blocks as a special case of the attack given in [11]. Finally, they introduce an SEBC by arranging the AD so that the first CT block is produced as the encryption of a given input. We introduce another SEBC which is a chosen plaintext query for COLM instead of arranging the AD.

There are SDBC or SEBC constructions of some AEADs in the literature. The difficulty level and the structures of these attacks may differ completely according to the AEAD scheme. Bay et al. build an SDBC for ELmD [10]. One should note that constructing an SDBC is much easier than an SEBC on ELmD since it is an Encrypt-mix-**Decrypt** scheme and requires only chosen plaintexts for an SDBC. On the other hand, constructing an SEBC of ELmD is still an open problem. However, the situation is converse for COLM. That is, there is an SEBC construction of COLM by Vaudenay and Vizár [12]. It is much more difficult to construct an SDBC for COLM and has been an open problem so far.

There are some other studies related to implementations of COLM. In [37], Zhang et al. endorse the designers of COLM about its performance and potential for parallel implementation. In [21], Jahanbani et al. implement COLM in a differential power analysis (DPA) protected manner. In [22], Bossuet et al. present pipelined hardware implementation and in [14], Gruber et al. perform a persistent fault analysis. In [15], Khairallah et al. mount adapting differential fault attacks against COLM along with some other CAESAR finalists.

1.2. Our contributions

Our main contribution can be summarized as reducing the security of COLM to the confidentiality of its L-parameter in all aspects, particularly in constructing SEBC/SDBC, plaintext recovery and tag guessing attacks. We show that both the encryption and the decryption oracles of the underlying block cipher of COLM can be simulated for any given input. That is, it is possible to use the COLM oracle to query $\alpha = E_K(\beta)$ (SEBC) and $\beta = D_K(\alpha)$ (SDBC) for any given β and α, respectively, where $D_K = E_K^{-1}$ and is run only in the decryption of COLM since it is an Encrypt-mix-**Encrypt** scheme.

There have been no known examples of constructing SDBC to an authenticated EME scheme or SEBC to an authenticated EMD scheme

in the literature yet. To the best of our knowledge, our SDBC of COLM is the first such simulator in an arbitrarily chosen ciphertext scenario against an authenticated EME/EMD scheme. In general, it is much more difficult to mount an attack against an AEAD scheme in this scenario since the attacker must generate the valid tag for her query.

As related to all of the main security approaches of an AEAD algorithm, we mount generic universal forgery, plaintext recovery and tag guessing attacks against COLM, based on SEBC and SDBC. The security levels of exploiting the decryption oracle for arbitrary ciphertexts or producing a valid tag are supposed to be 128 bits for COLM. Once L is recovered in 2^{65} offline queries (see [11]), we can decrypt any given CT, and produce its tag in a polynomial time complexity. If the proposed attacks are mounted within the birthday bound complexity (2^{64} queries), the success rate would be as high as one quarter. Also, we show how to collect chosen plaintexts and ciphertexts through SEBC and SDBC, respectively, to mount some well known key recovery attacks against (the round reduced) AES.

We introduce some other attacks not based on SEBC/SDBC. We present another plaintext recovery attack for any given ciphertext even though its tag is unknown by querying both encryption and the decryption oracles of COLM once. As our third plaintext recovery attack, we introduce how to decrypt any given ciphertext of at least five blocks where only the COLM decryption oracle is queried twice. Unlike our other plaintext recovery attacks, we need the tag in this attack. Moreover, as another universal forgery attack, we produce the ciphertext and the tag of a given plaintext of length at least three blocks by querying the COLM encryption oracle twice and the COLM decryption oracle once.

In general, authenticated EME schemes have the whitening process with the whitening mask L, usually defined by $L = E_K(0)$. It is possible to mount a stand alone attack to recover L. Therefore, the attacks on such schemes can be separated as recovering L and attacking the EME structure. Recall that L was recovered in COLM by Forler et al. [11], but the security levels of the EME structure against plaintext recovery and tag guessing have been open questions. An SEBC or an SDBC itself does not threaten the 128-bit security of the EME structure against these attacks. Notice that one can mount these attacks if there exist both an SEBC and an SDBC. By introducing them together, we show that COLM does not satisfy 128-bit security against plaintext recovery and tag guessing attacks. Moreover, we make some suggestions in order to prevent our attacks.

1.3. Organizations

The rest of this paper is structured as follows. In Section 2, COLM is explained. We introduce how to recover $S = D_K(0)$ parameter in COLM in Section 3. Then, in Section 4, the simulation models of the encryption/decryption oracles of the underlying block cipher are introduced. In Section 5, a universal forgery attack, a plaintext recovery attack and a tag guessing attack based on SEBC and SDBC are presented. In Section 6, we explain how to decrypt a given ciphertext without its tag, and an existential forgery attack. We introduce another universal forgery attack and the third plaintext recovery attack in Section 7. In Section 8, we list some examples for the complexity of the data collection to mount CPA/CCA against (reduced-round) AES-128. Finally, in Section 9, we give our concluding remarks along with some suggestions to prevent our attacks against COLM.

2. Brief description of COLM

COLM is an Encrypt-Mix-Encrypt (EME) AEAD algorithm. COPA and ELmD are merged as COLM in the second round of CAESAR to enhance the performance without sacrificing the security, see Table 2.

In the encryption, COLM takes a 128-bit key, an AD and a PT, both of arbitrary length, as the inputs then outputs a CT, a tag and optionally intermediate tags (we ignore this option for simplicity; our attacks still work in this case). The encryption scheme of COLM is depicted in Fig. 1. The last block is shown separately to emphasize the verification process. For details see [29].

Table 2
Comparison of COPA, ELmD and COLM, [29].

	COPA	ELmD	COLM
Simplified masking	No	No	Yes
Fully parallelizable authentication	No	Yes	Yes
XOR mixing for authentication	Yes	No	Yes
ρ mixing for encryption	No	Yes	Yes
Bottom layer encryption	Yes	No	Yes
Intermediate tags	No	Yes	Yes

COLM uses two sorts of operations: linear mixing defined in $GF(2^{128})$ and a symmetric encryption with the cipher AES-128. We do not exploit any property of AES-128 in our attacks. Therefore, throughout the paper, we use E_K and D_K to represent the encryption and the decryption of the underlying cipher (AES-128), respectively.

Three whitening masks, L, L_1 and L_2, are produced as whitening masks of PT, AD, and CT, respectively. They are defined as:

$$L = E_K(0), \quad L_1 = 3 \cdot L, \quad L_2 = 3^2 \cdot L.$$

In the encryption, a CT and a tag are generated whereas, in the decryption, a PT is generated but released only if the tag is verified. The encryption and the decryption are denoted as $(CT, \text{tag}) = \mathcal{E}_K(AD, PT)$ and $(PT \text{ or } \perp) = \mathcal{D}_K(AD, CT, \text{tag})$, respectively. The data is divided into 128-bit blocks and if the last block is shorter than 128 bits, it is padded by concatenating the pattern 10^* with one "1" and enough number of zeros. AD, PT, CT and tag are denoted as:

$$\begin{aligned} AD &= (A[1], \ldots, A[\alpha-1], A^*[\alpha]),\\ PT &= (M[1], \ldots, M[l-1], M^*[l]),\\ CT &= (C[1], \ldots, C[l-1], C[l]),\\ tag &= C[l+1]. \end{aligned}$$

and $M[l] = \bigoplus_{i=1}^{l-1} M[i] \bigoplus (M^*[l]10^*)$.

The whitening masks are given in the equations below:

$$\Delta_{A[i]} = \begin{cases} 7 \cdot 2^{i-1} \cdot L_1 & \text{if } i = \alpha \& |A^*[\alpha]| < 128,\\ 2^i \cdot L_1 & \text{otherwise.} \end{cases}$$

$$\Delta_{M[i]} = \begin{cases} 7 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l, l+1\} \& |M^*[l]| = 128,\\ 7^2 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l, l+1\} \& |M^*[l]| < 128,\\ 2^i \cdot L & \text{otherwise.} \end{cases}$$

$$\Delta_{C[i]} = \begin{cases} 7 \cdot 2^{i-1} \cdot L_2 & \text{if } i \in \{l, l+1\} \& |M^*[l]| = 128,\\ 7^2 \cdot 2^{i-1} \cdot L_2 & \text{if } i \in \{l, l+1\} \& |M^*[l]| < 128,\\ 2^i \cdot L_2 & \text{otherwise.} \end{cases}$$

After whitening, the input blocks, denoted as AA for AD and MM for PT, are ready for E_K. The W' blocks (the encryptions of the AA-blocks) are XORed (denoted $\oplus$) to generate IV. The AES-128 encryption is called twice for each block of PT during encryption. The linear operation, ρ, is executed between these two AES-128 encryptions. For details, see [29] and Fig. 1 where MM, X, W, Y and CC are the intermediate values and C is the output. The ρ-operation and its inverse (ρ^{-1}) are defined as:

$$\begin{aligned} (y, st') &= \rho(x, st),\\ y &= x \oplus 3 \cdot st,\\ st' &= x \oplus 2 \cdot st.\\ (x, st') &= \rho^{-1}(y, st), \end{aligned}$$

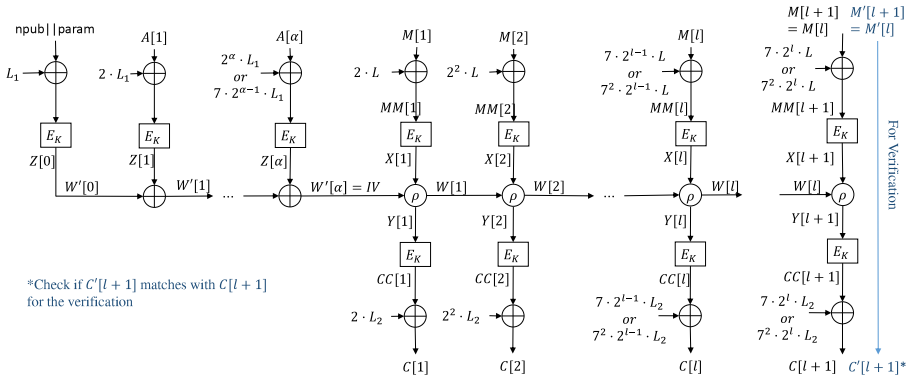

Fig. 1. Encryption and tag verification of COLM.

$$x = y \oplus 3 \cdot st,$$
$$st' = y \oplus st.$$

Here, st and st' denote the previous and the next state values for the ith block $W[i-1]$ and $W[i]$, respectively.

The IV generation is given as:

$$\begin{aligned}
A[0] &= npub \| param, \\
AA[i] &= A[i] \oplus \Delta_A[i] && \text{for } i = 1, \ldots, \alpha, \\
Z[i] &= E_K(AA[i]) && \text{for } i = 1, \ldots, \alpha, \\
W'[0] &= E_K(AA[0]), \\
W'[i] &= W'[i-1] \oplus Z[i] && \text{for } i = 1, \ldots, \alpha, \\
IV &= W'[\alpha].
\end{aligned}$$

Here *npub* is the nonce and *param* is a constant value to indicate configuration of the cipher.

The encryption has three phases:

1. The encryption of the blocks with the underlying block cipher ($X[i] = E_K(MM[i])$)
2. Linear mixing of the intermediate blocks ($(Y[i], W[i]) = \rho(X[i], W[i-1])$)
3. The encryption of the mixed blocks with the underlying block cipher ($CC[i] = E_K(Y[i])$)

The pseudocode for the whole encryption process is as follows:

$$\begin{aligned}
W[0] &= IV, \\
MM[i] &= M[i] \oplus \Delta_M[i] && \text{for } i = 1, \ldots, l+1, \\
X[i] &= E_K(MM[i]) && \text{for } i = 1, \ldots, l+1 \\
(Y[i], W[i]) &= \rho(X[i], W[i-1]) && \text{for } i = 1, \ldots, l+1, \\
CC[i] &= E_K(Y[i]) && \text{for } i = 1, \ldots, l+1, \\
C[i] &= CC[i] \oplus \Delta_C[i] && \text{for } i = 1, \ldots, l+1.
\end{aligned}$$

A new block, $M[l+1] = M[l]$, is concatenated to PT as the $l+1$st block. For a padding of p bits in a PT, the last p bits of the CT are removed from its last block, $C[l]$. Notice that, for an l-block PT an $(l+1)$-block CT is produced. We consider the $(l+1)$st block of CT as the tag and call the resulting CT an l-block CT.

The pseudocode of the decryption process after the IV generation is as follows:

$$\begin{aligned}
W[0] &= IV, \\
CC[i] &= C[i] \oplus \Delta_C[i] && \text{for } i = 1, \ldots, l, \\
Y[i] &= D_K(CC[i]) && \text{for } i = 1, \ldots, l, \\
(X[i], W[i]) &= \rho^{-1}(Y[i], W[i-1]) && \text{for } i = 1, \ldots, l, \\
MM[i] &= D_K(X[i]) && \text{for } i = 1, \ldots, l, \\
M[i] &= MM[i] \oplus \Delta_M[i] && \text{for } i = 1, \ldots, l, \\
M^*[l] &= M[1] \oplus \cdots \oplus M[l].
\end{aligned}$$

Pseudocode of the tag verification process apart from the padding check is as follows

$$\begin{aligned}
M'[l+1] &= M[l], \\
MM'[l+1] &= M'[l+1] \oplus \Delta_M[l+1], \\
X'[l+1] &= E_K(MM'[l+1]), \\
Y'[l+1] &= X'[l+1] \oplus 3 \cdot W'[l], \\
CC'[l+1] &= E_K(Y'[l+1]), \\
C'[l+1] &= CC'[l] \oplus \Delta_C[l+1], \\
C[l+1] &\stackrel{?}{=} C'[l+1].
\end{aligned}$$

2.1. Our assumption on L

When L is known, it is straightforward to calculate AAs, MMs and CCs from As, Ms and Cs, respectively or vice versa. Forler et al. show how to recover the whitening mask L in [11]. We assume that L is known and, for the simplicity, consider AAs, MMs and CCs as AD, PT and CT, respectively throughout the paper. Also, the condition $M[l] = M[l+1]$ in the tag verification yields the condition $MM[l+1] = MM[l] \oplus 7 \cdot 3 \cdot 2^{l-1} \cdot L$ for the full last block. We use the notation $\widetilde{CT}$ to represent the forged ciphertexts produced by the attacker. Similarly, we use the notation $\widetilde{T}$ to represent any data T produced during the decryption of forged ciphertexts, $\widetilde{CT}$.

3. How to recover $S = D_K(0)$ through an existential forgery

Our principal goal is to recover $S = D_K(0)$, the decryption of the zero-vector for the underlying block cipher of COLM, to mount both the tag guessing and plaintext recovery attacks. Furthermore, it is a milestone as being the first step of the simulation models of both the encryption (see Section 4.2) and the decryption (see Section 4.1) oracles of the underlying block cipher. To recover S, we need to make decryption query of the AEAD-scheme. We further extend the decryption query to a universal decryption query in Sections 5–7. Recall that it is much more difficult to query an AEAD structure in decryption direction since one needs the valid tag for the query. To the best of our knowledge, this is the first work making an existential forgery that discloses plaintext–ciphertext pair from the underlying block cipher of an authenticated EME/EMD scheme. We remind that the simulation of underlying block cipher attacks in [10,12] make encryption queries of the AEAD-schemes, ELmD and COLM, respectively.

Theorem 1 states how to recover $S = D_K(0)$ with an existential forgery attack. Figs. 2 and 3 depict how to recover $S = D_K(0)$. For simplicity, we take $\widetilde{MM}[1] = 0$ in Fig. 2 and the tag verification is illustrated by using $\widetilde{MM'}[3]$ in Fig. 3. The proof of Theorem 1 uses Lemma 1, Proposition 1 and Theorem 2. Therefore, we give the proof as a separate section.

Theorem 1. *Let $CC[1]\|CC[2]$ be the CT of $AA[0]\|AA[1]\|MM[1]\|MM[2]$ where $AA[0] = AA[1]$ and $MM[1] \oplus MM[2] = 7 \cdot 3 \cdot 2 \cdot L$. Then $\widetilde{CT} = L\|CC[1]$ is a valid ciphertext with $AA[0]\|AA[1]$ as $\widetilde{AD}$ and $CC[2]$ as the tag. Moreover, the first block of its plaintext is $\widetilde{MM}[1] = S = D_K(0)$.*

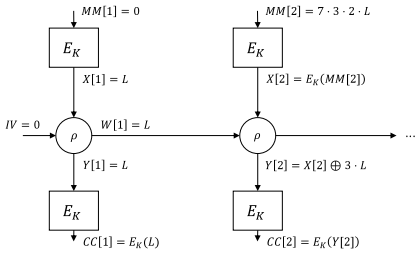

Fig. 2. Encryption phase of obtaining $S = D_K(0)$.

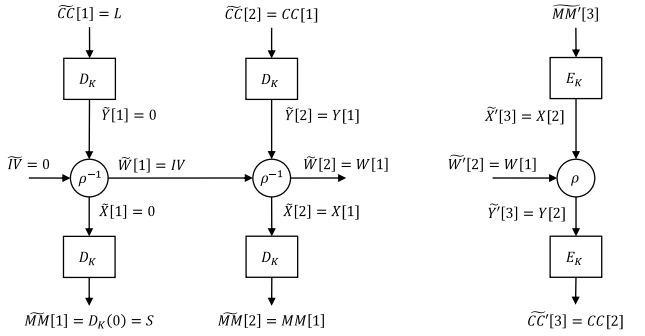

Fig. 3. Decryption phase of obtaining $S = D_K(0)$. Note that the tag is verified since $\widetilde{MM}'[3] = MM[1] \oplus 7 \cdot 3 \cdot 2 \cdot L = MM[2]$.

3.1. Proof of Theorem 1

Theorem 1 has two claims. The former is that $\widetilde{CT}$ is a valid ciphertext and the latter is $S = D_K(0) = \widetilde{MM}[1]$.

We introduce three statements to prove these claims. The first statement is Lemma 1 where we show a method to satisfy $\widetilde{W}[l-1] = IV$. In Proposition 1, we introduce how to produce the tag of a set of $\widetilde{CT}$s under the condition $\widetilde{W}[l-1] = IV$. We combine Lemma 1 and Proposition 1 to produce a forged $\widetilde{CT}$ with a valid tag in Theorem 2.

Lemma 1 exploits a property of the inverse of the linear operator, ρ^{-1}, to satisfy $\widetilde{W}[l-1] = IV$.

Lemma 1. *For any l-block CT whose initial value is IV and whose $CC[i]$ blocks take any value, except $L = E_K(0)$, even number of times for $1 \le i < l$ then $W[l-1] = IV$. (An example is illustrated in Fig. 4)*

Proof. Let CT be a ciphertext of l blocks such that its $CC[i]$ blocks take any value, except $L = E_K(0)$, even number of times for $1 \le i < l$. That is, for any given $i < l$ if $CC[i] \neq L$ then $\exists j \neq i$ such that $CC[i] = CC[j]$ and the number of $j < l$ satisfying this equality is odd. This simply implies that the $Y[i]$ blocks take a value except $D_K(L) = 0$ even number of times for $1 \le i < l$. On the other hand, $W[l-1] = \bigoplus_{i=1}^{l-1} Y[i] \oplus IV$. Hence one can deduce $W[l-1] = IV$ since nonzero $Y[i]$s will vanish pairwise. □

The following statement provides us a method to produce the tag of a set of $\widetilde{CT}$s where $\widetilde{W}[l-1] = IV$ by using the first two CT blocks of a chosen PT of at least two blocks.

Proposition 1. *Let a given plaintext PT be encrypted with an initial value IV and the first two blocks of the produced CT be $CC[1]$ and $CC[2]$. Assume $MM[1] \oplus MM[2] = 7 \cdot 3 \cdot 2^{l-1} \cdot L$. Then any forged l-block $\widetilde{CT}$ whose $\widetilde{W}[l-1] = IV$ and $\widetilde{CC}[l] = CC[1]$ has the valid tag $\widetilde{CC}[l+1] = CC[2]$. (See Fig. 5.)*

Proof. Let a plaintext PT be given such that $MM[1] \oplus MM[2] = 7 \cdot 3 \cdot 2^{l-1} \cdot L$. Assume $CC[1]$ and $CC[2]$ are the first two blocks of the produced CT by encrypting the PT with IV. Let a forged l-block $\widetilde{CT}$ have $\widetilde{CC}[l] = CC[1]$ and $\widetilde{CC}[l+1] = CC[2]$. Observe that

$$\begin{aligned} \widetilde{MM}[l] &= D_K(D_K(\widetilde{CC}[l]) \oplus 3 \cdot \widetilde{W}[l-1]) \\ &= D_K(D_K(CC[1]) \oplus 3 \cdot IV) \\ &= MM[1] \end{aligned}$$

if $\widetilde{W}[l-1] = IV$. Similarly $\widetilde{MM}[l+1] = MM[2]$. Therefore the tag is valid since $MM[1] \oplus MM[2] = 7 \cdot 3 \cdot 2^{l-1} \cdot L$. □

We can combine Lemma 1 and Proposition 1 to produce a forged $\widetilde{CT}$ with the valid tag which is stated by Theorem 2.

Theorem 2. *Let a given plaintext PT be encrypted with the initial value IV and the first two blocks of the produced CT be $CC[1]$ and $CC[2]$. Assume $MM[1] \oplus MM[2] = 7 \cdot 3 \cdot 2^{l-1} \cdot L$. For any forged l-block $\widetilde{CT}$ whose $\widetilde{CC}[i]$ blocks take any value, except $L = E_K(0)$, even number of times for $1 \le i < l$, if $\widetilde{CC}[l] = CC[1]$ and $\widetilde{CC}[l+1] = CC[2]$, then $\widetilde{CT}$ with IV is a valid ciphertext.*

Proof. For an encrypted plaintext PT, assume $MM[1] \oplus MM[2] = 7 \cdot 3 \cdot 2^{l-1} \cdot L$. Let the first two blocks of its ciphertext be $CC[1]$ and $CC[2]$. Let $\widetilde{CT}$ be a l-block forged ciphertext such that its $\widetilde{CC}[i]$ blocks take any value, except $L = E_K(0)$, even number of times for $1 \le i < l$. Then, we have $\widetilde{W}[l-1] = IV$ by Lemma 1. Therefore, $\widetilde{CT}$ is a valid ciphertext by Proposition 1. □

Now, we can give the proof of Theorem 1.

Proof of Theorem 1. Let $CC[1] \| CC[2]$ be the CT of

$$AA[0] \| AA[1] \| MM[1] \| MM[2]$$

where $AA[0] = AA[1]$ and $MM[1] \oplus MM[2] = 7 \cdot 3 \cdot 2 \cdot L$. Then, the forged ciphertext $\widetilde{CT} = L \| CC[1]$ is valid with $AD = AA[0] \| AA[1]$ and the tag $CC[2]$ by Theorem 2. Note that $IV = E_K(AA[0]) \oplus E_K(AA[1]) = 0$, $\widetilde{Y}[1] = D_K(L) = 0$ and $\widetilde{X}[1] = \widetilde{Y}[1] \oplus 3 \cdot IV = 0$. Hence, we obtain the first block $\widetilde{MM}[1] = D_K(0) = S$ by decrypting $\widetilde{CT}$. □

3.2. A method to produce ADs having the same IV

Similar to Lemma 1, we can manipulate any AD by inserting certain values by utilizing $\widetilde{MM}[1] = D_K(0) = S$ outcome of Theorem 1. Lemma 2 states this manipulation precisely. Note that instead of the whitening mask L, the S value is used as in Lemma 1.

Lemma 2. *For any given AD, inserting any number of the S value or any even number of other values into the AD will not change IV.*

Proof. Let an $AD = AA[0] \| \dots \| AA[a]$ be given with the blocks, its $IV = \bigoplus_{i=0}^{a} X[i]$ where $X[i] = E_K(AA[i])$. Inserting the S value into any block position of AD will result $IV = \bigoplus_{i=0}^{a} X[i] \oplus E_K(S) = IV$. Similarly, inserting a different value, V twice into any two block positions of AD will result

$$IV = \bigoplus_{i=0}^{a} X[i] \oplus E_K(V) \oplus E_K(V) = IV.$$

Therefore, by induction, inserting any number of S or any even number of other values into the AD will not change its IV. □

Remark 1. Lemma 2 gives us a method to produce several other ADs having the same IV as that of a given AD. Therefore, in all our attacks, we can choose different associated data without changing IV while querying the encryption/decryption oracles of COLM. For the sake of simplicity, we query plaintexts or ciphertexts without changing the AD throughout the paper just to mean that we use the same IV. It is straightforward that we can easily produce different associated data, when necessary, for each query by Lemma 2.

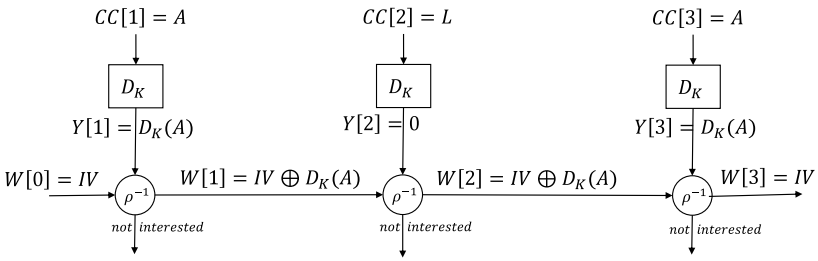

Fig. 4. An example of Lemma 1, $CC[2] = L$ does not have any effect on state value and $CC[3] = A$ compensates effect of $CC[1] = A$.

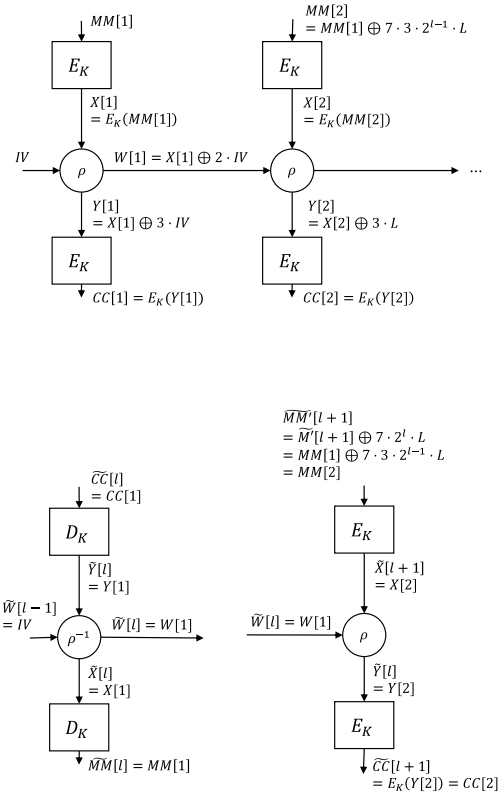

Fig. 5. Illustration of Proposition 1.

4. Simulation models of encryption/decryption oracles of the underlying block cipher

In this section, we show how to simulate both the encryption and the decryption oracles of the underlying block cipher (SEBC/SDBC) for any given input. We show that the COLM oracle can be used to query $\alpha = E_K(\beta)$ and $\beta = D_K(\alpha)$ for any given β and α, respectively. For both these tasks, We use the $S = D_K(0)$ value besides the whitening mask L. Recall that $S = D_K(0)$ can be recovered by Theorem 1.

4.1. Simulation model of the decryption oracle of the underlying block cipher: Computing $\beta = D_K(\alpha)$

In this section, we introduce a simulation model of the decryption oracle of the underlying block cipher (SDBC) for any given input, i.e. how to compute $\beta = D_K(\alpha)$ for any given α.

Both simulation models of Bay et al. in [10] and of Vaudenay and Vizár in [12] are standard encryption queries of an AEAD scheme, ELmD and COLM, respectively. We query COLM, the AEAD, in decryption direction, in our attack. An AEAD should not release the plaintext without verifying the tag. Therefore, we mount a forgery against COLM as part of this attack. We begin our attack with the query of the COLM encryption twice as a precomputation to forge a valid tag for a chosen set of ciphertexts.

In the first query, the set of $\{E_K(\gamma), E_K(2 \cdot \gamma), \ldots, E_K(2^{127} \cdot \gamma)\}$ is constructed for an arbitrary non-zero γ where both γ and $D_K(3^{-1} \cdot \gamma)$ are known. We choose $\gamma = 3 \cdot L$ and set $IV_1 = L$. Then, encrypting $MM_1[i] = S$ for $i = 1, \ldots, 128$, we get $CC_1[i] = E_K(2^{i-1} \cdot 3 \cdot L)$, by Lemma 3 (see Fig. 6). Taking $AA_1[1] = AA_1[0]$ and $AA_1[2] = 0$ yield to $IV_1 = L$ (see Fig. 7). The process of constructing the set $\{E_K(3 \cdot L), E_K(2 \cdot 3 \cdot L), \ldots, E_K(2^{127} \cdot 3 \cdot L)\}$ is given in Algorithm 1.

Lemma 3. *Let $AA[0] = AA[1]$ and $AA[2] = D_K(3^{-1}\gamma)$. Then the corresponding CT of the PT where $MM[i] = S$ for $i = 1, \ldots, 128$ will result in $CC[i] = E_K(2^{i-1} \cdot \gamma)$ for $i = 1, \ldots, 128$.*

Proof. Let $AA[0] = AA[1]$ and $AA[2] = D_K(3^{-1}\gamma)$. Then $IV = W[0] = 3^{-1} \cdot \gamma$. For the PT if $MM[i] = S$ then $X[i] = 0$, for $i = 1, \ldots, 128$. Therefore, the inputs of the linear operator ρ will be $W[i-1] = 3^{-1} \cdot 2^{i-1} \cdot \gamma$ and $X[i] = 0$. So the output $Y[i]$ will be $2^{i-1} \cdot \gamma$, for $i = 1, \ldots, 128$. That is, $CC[i] = E_K(Y[i]) = E_K(2^{i-1} \cdot \gamma)$. □

Algorithm 1 Constructing the Set of $E_K(3 \cdot L)$, $E_K(2 \cdot 3 \cdot L)$, …, $E_K(2^{127} \cdot 3 \cdot L)$

```
procedure CONSTRUCT THE SET
    IV ← L
    for i ← 1, 128 do
        MM[i] ← S
    end for
    (CT, tag) ← E_K(AD, PT)
    return CC                          ▷ CC = CC[1]||CC[2]||| ··· ||CC[128]
end procedure
```

We query another PT of at least 2 blocks during the offline phase to construct a tag which is valid for any 258-block CTs whose CC blocks appear even number of times. This query is depicted in Fig. 8. The first two blocks of the resulting CT, which we denote $(CC_2[1], CC_2[2])$, will be used for the verification according to Theorem 2 during the online phase.

In the decryption query, the COLM oracle is forged and $\beta = D_K(\alpha)$ for the given α is obtained. The explicit statement and the pseudocode of this process are given in Theorem 3 and in Algorithm 2, respectively.

Theorem 3. *Let $CC[1]$ and $CC[2]$ be the first two CT blocks of a PT with $MM[1] = MM[2] \oplus 7 \cdot 3 \cdot 2^{257} \cdot L$ encrypted with the 2-block AD where $AA[0] = AA[1]$. Assign $\alpha_d = (3 \cdot \gamma)^{-1} \cdot \alpha$ for a given value α. Let*

$$\widetilde{CC}[i] = \widetilde{CC}[i + 129] = \begin{cases} E_K(2^{128-i} \cdot \gamma) & \text{if the } i\text{th MSB} \\ & \text{(Most Significant Bit) of} \\ & \alpha_d = 1, \\ L & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, 128$ and $\widetilde{CC}[129] = L$. Let $\widetilde{CC}[258] = CC[1]$ and $\widetilde{CC}[259] = CC[2]$. Then, the $\widetilde{CT} = \widetilde{CC}[1]\| \ldots \|\widetilde{CC}[259]$ will be a valid ciphertext

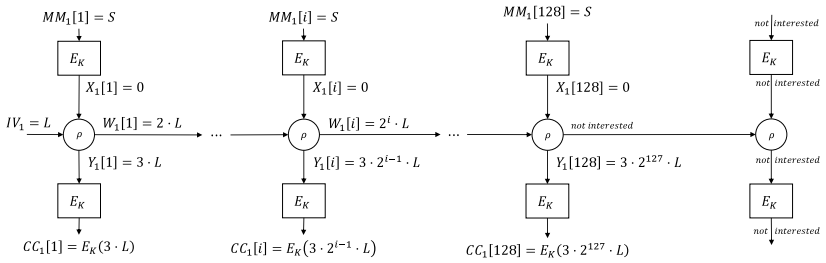

Fig. 6. Constructing $E_K(2^i \cdot \gamma)$ for $\gamma = 3 \cdot L$ (see Fig. 7 for setting $IV_1 = L$).

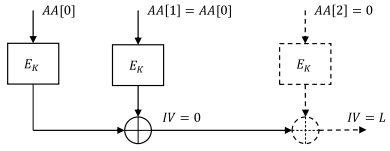

Fig. 7. Setting $IV = 0$ or $IV = L$ ($IV = 0$ when $AD = AA[0]||AA[1]$, and $IV = L$ when $AD = AA[0]||AA[1]||AA[2]$).

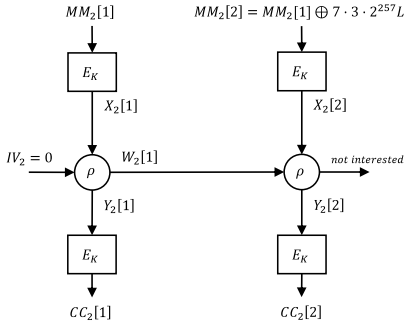

Fig. 8. Generating valid tag.

with the 2-block AD where $AA[0] = AA[1]$ and $MM[129]$ will be equal to $D_K(\alpha)$.

Proof. Let a given plaintext PT be encrypted with the initial value $IV = 0$ and the first two blocks of the produced CT be $CC[1]$ and $CC[2]$. Assume $MM[1] \oplus MM[2] = 7 \cdot 3 \cdot 2^{257} \cdot L$. For any forged $\widetilde{CT}$ of 258 blocks whose $\widetilde{CC}[i]$ blocks take any value, except $L = E_K(0)$, even number of times for $1 \le i < 258$, we have

$$\widetilde{W}[257] = \bigoplus_{i=1}^{128} \left(\widetilde{Y}[i] \oplus \widetilde{Y}[i+129]\right) \bigoplus D_K(L) \bigoplus IV$$
$$= \bigoplus_{i=1}^{128} 0 \bigoplus 0 \bigoplus 0 = 0$$

by Lemma 1. By setting the last two blocks as $\widetilde{CC}[258] = CC[1]$ and $\widetilde{CC}[259] = CC[2]$, the tag is verified by Proposition 1.

The decision parameter $\alpha_d = (3 \cdot \gamma)^{-1} \cdot \alpha$ is calculated. Both $\widetilde{CC}[i]$ and $\widetilde{CC}[i+129]$ are set to $E_K(2^{128-i} \cdot \gamma)$, if ith MSB of $\alpha_d = 1$, or set to L, otherwise and $\widetilde{CC}[129]$ is set to L.

$$\widetilde{W}[128] = \bigoplus_{i=1}^{128} \widetilde{Y}[i] \bigoplus IV = \bigoplus_{i=1}^{128} \alpha_d[i] \cdot 2^{128-i} \cdot \gamma = \alpha_d \cdot \gamma$$

and hence

$$\widetilde{X}[129] = 3 \cdot \widetilde{W}[128] \bigoplus \widetilde{Y}[129] = \alpha_d \cdot 3\gamma \bigoplus D_K(L) = \alpha$$

which yields $\widetilde{MM}[129] = \beta = D_K(\alpha)$ (see Fig. 9). □

Algorithm 2 Forging COLM to Decrypt $D_K(\alpha)$

procedure $D_K(\alpha)$
  $\alpha_d \leftarrow (3^2 \cdot L)^{-1} \cdot \alpha$
  $\widetilde{AA}[1] \leftarrow \widetilde{AA}[0]$
  for $i \leftarrow 1, 128$ **do** ▷ 1 is the MSB, 128 is the LSB (Least Significant Bit)
    if $\alpha_d[i] = 1$ **then**
      $\widetilde{CC}[i] \leftarrow E_K(2^{128-i} \cdot 3 \cdot L)$
    else
      $\widetilde{CC}[i] \leftarrow L$
    end if
    $\widetilde{CC}[i+129] \leftarrow \widetilde{CC}[i]$
  end for
  $\widetilde{CC}[129] \leftarrow L$
  $\widetilde{CC}[258] \leftarrow CC[1]$
  $\widetilde{CC}[259] \leftarrow CC[2]$
  $\widetilde{CC} = \widetilde{CC}[1]||| \cdots ||\widetilde{CC}[259]$
  $\widetilde{PT} \leftarrow D_K(\widetilde{AD}, \widetilde{CT})$
  return $\widetilde{MM}[129]$ ▷ $\widetilde{MM} = \widetilde{MM}[1]|\widetilde{MM}[2]]|| \cdots ||\widetilde{MM}[258]$
end procedure

The online cost of this attack is a single COLM decryption query for an input consisting of a 2-block AD and a 259-block CT including the tag.

4.2. Simulation model of the encryption oracle of the underlying block cipher: Computing $\alpha = E_K(\beta)$

For a given β, we construct a convenient (AD, PT) such that the last CT block will be the required α value. The explicit statement is given in Theorem 4. We always choose AD so that $IV = 0$ independent of the given β. For example AD can be taken as two blocks such that $AA[0] = AA[1]$, then it is clear that $IV = 0$ (see Fig. 7).

Theorem 4. *Let a PT be encrypted with an AD of two blocks such that $AA[1] = AA[0]$. Let $\beta_d = (3 \cdot L)^{-1} \cdot \beta$ for a given β. Take*

$$MM[i] = \begin{cases} S & \text{if the } i\text{th MSB of } \beta_d = 0, \\ 0 & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, 128$. Let $MM[129] = S$. Then the last block of the corresponding CT, $CC[129]$, equals $\alpha = E_K(\beta)$.

Proof. For any given β, we define the decision parameter $\beta_d = (3 \cdot L)^{-1} \cdot \beta$. Then, take $MM[i] = S$ if the ith MSB of β_d is zero and $MM[i] = 0$, otherwise. This leads to

$$W[i] = 2 \cdot W[i-1] \oplus \beta_d[i] \cdot L.$$

Let $\beta_d = \beta_d[1]\beta_d[2] \cdots \beta_d[128]$ where $\beta_d[1]$ is the MSB and $\beta_d[128]$ is the LSB of β_d. Then,

$$W[128] = \beta_d \cdot L = \oplus_{i=1}^{128} \beta_d[i] \cdot 2^{128-i} \cdot L$$

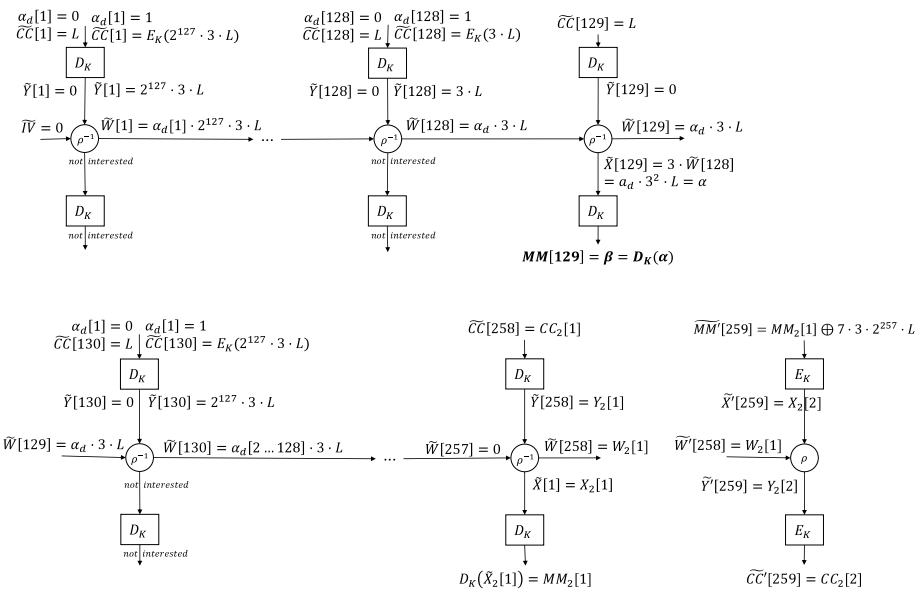

Fig. 9. Forging COLM algorithm for simulating the decryption oracle of underlying block cipher. The upper part depicts producing $\beta = D_K(\alpha)$. The lower part depicts verifying the tag.

as depicted in Fig. 10. On the other hand, $X[129] = 0$ since $MM[129] = S$. Hence

$$Y[129] = 3 \cdot W[128] \oplus X[129] = 3 \cdot \beta_d \cdot L.$$

As a result,

$$CC[129] = E_K[\beta] = \alpha. \quad \square$$

Algorithm 3 gives the pseudocode of how to construct the convenient PT for given β. Then querying $AD\|PT$, the last block of the corresponding CT will be α by Theorem 4 (see Fig. 10). Let us remark that this is a different method for SEBC of COLM from Vaudenay's and Vizár's method in [12]. In [12], the authors precompute the series $E_K((2^n-1)L)$, where $0 \le n \le 134$ and arrange an arbitrary length AD to obtain $Y[1] = \beta$. In this attack, we precompute only $D_K(0)$ and arrange a 129-block PT to obtain $Y[129] = \beta$.

Algorithm 3 Computing $\alpha = E_K(\beta)$ for a given β

```
procedure E_K(β)
    AA[1] ← AA[0]
    β_d ← (3 · L)^{-1} · β
    for i from 1 to 128 do
        if β_d[i] = 0 then                ▷ 1 is the MSB, 128 is the LSB
            MM[i] ← S
        else
            MM[i] ← 0
        end if
    end for
    MM[129] ← S
    (CT, tag) ← E_K(AD, PT)
    return CC[129]                        ▷ α = E_K(β) is CC[129]
end procedure
```

The cost of this attack is a single COLM encryption query for an input consisting of a 2-block AD and a 259-block PT.

5. Universal forgery, tag guessing and plaintext recovery attacks

In a universal forgery attack, the attacker is given any (AD, PT) pair and required to compute the corresponding CT and the tag. The attacker can query the encryption and decryption oracles of COLM for any data except for the given PT with any AD or the given AD with any PT.

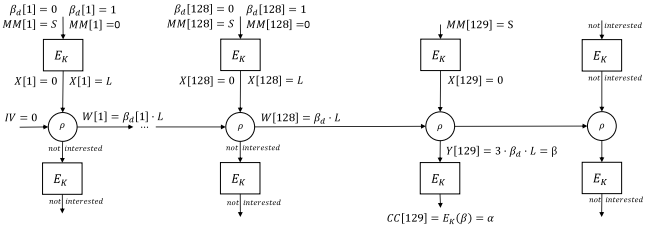

Fig. 10. Simulation of the encryption oracle of the underlying block cipher via COLM oracle.

Each ciphertext has two different tags in COLM, one for the full last block case and one for the incomplete last block case. In a tag guessing attack, the attacker is given any (AD, CT) pair and required to compute one of these two tags. The attacker can query any data except for the given CT with any AD or the given AD with any CT in the encryption and decryption oracles of COLM.

In a plaintext recovery attack, the attacker is given any (AD, CT) pair and required to compute the corresponding PT. The attacker can query the encryption and the decryption oracles of COLM for anything except for the given AD or CT even with different ciphertexts or associated data, respectively. Notice that the challenging pair (AD, CT) can be given with the valid tag in another version of plaintext recovery attacks. But we adopt the former definition in this section and in Section 6. We introduce also a plaintext recovery attack for the latter definition in Section 7.

All of the attacks above can be deduced easily from the SEBC and SDBC introduced in Section 4. An attacker who can run both SEBC and SDBC has the same capability as a legitimate user. Therefore, in such a scenario the attacker can produce the CT and its tag for any given PT and also decrypt any given CT even if its tag is missing. In the universal forgery attack, the state values are obtained by querying the MM blocks as in Section 4.2; on the other hand, in the tag guessing and plaintext recovery attacks the state values are obtained by querying the CC blocks as in Section 4.1.

In the universal forgery attack, first of all, the $MM[l]$ and the $MM[l+1]$ blocks are prepared as explained in Section 3. All Z, X

and CC blocks in an encryption process of COLM are the outputs of the underlying block cipher encryption. Therefore, we use the SEBC to recover all Z, X and CC blocks from the AA, the MM, and the Y blocks respectively as in Section 4.2. The AA and the MM blocks are given while the Y blocks are computed via the linear combinations of the Z and the X blocks. Hence, we can produce the CT of any given PT by the SEBC. We query COLM $a + 2 \cdot l + 2$ times, where a is the number of the AD-blocks, and l is the number of the PT-blocks.

The tag guessing attack and the plaintext recovery attack are similar to our universal forgery attack. In both attacks, the SDBC introduced in Section 4.1 is also used and the AD-blocks are encrypted through the SEBC to obtain the Z blocks as explained in Section 4.2 and the IV is calculated via XORing the Z blocks. Similarly, the CT-blocks are decrypted through the SDBC to recover the Y-blocks. The W and the X-blocks are calculated via the inverse linear mix. All X blocks are decrypted through the SDBC to recover the PT-blocks. Furthermore, in the tag guessing attack, $MM'[l+1]$ is computed. It is $MM[l] \oplus 7 \cdot 3 \cdot 2^{l-1} \cdot L$ for the PTs whose last block is full. $CC'[l+1]$ is recovered by using $MM'[l+1]$ and $W[l]$ through the SEBC. Let us remark that we do not need the whole PT to obtain the length of its last block. We query COLM $a + l + 3$ times for the tag guessing and $a + 2 \cdot l + 2$ times for plaintext recovery, where a is the number of the AD blocks, and l is the number of the PT blocks.

6. Another plaintext recovery without tag

In this section, we introduce another plaintext recovery attack where we can decrypt any ciphertext without its tag. In this attack, we use Lemmas 2, 1, Proposition 1 and Theorem 2.

Let an (AD, CT) pair be a given ciphertext with its associated data. It is possible to mount an attack to decrypt this CT even if the tag is not known. Let

$$CT = CC[1] \| \dots \| CC[l].$$

Then, query a message of two blocks satisfying Eq. (1) with the associated data AD:

$$\widehat{MM}[1] \oplus \widehat{MM}[2] = 7 \cdot 3 \cdot 2^{2l} \cdot L. \tag{1}$$

Let the first two blocks of the output be $\widehat{CC}[1]$ and $\widehat{CC}[2]$. To forge a $\widetilde{CT}$, set $\widetilde{CC}[2l+1] = \widehat{CC}[1]$, $\widetilde{CC}[2l+2] = \widehat{CC}[2]$ and choose the other $\widetilde{CC}[i]$'s such that

$$\widetilde{CC}[i] = \widetilde{CC}[i+l] = CC[i] \text{ for } 1 \le i \le l.$$

Observe that

$$\widetilde{W}[2l] = \bigoplus_{i=1}^{l} \left(D_K(\widetilde{CC}[i]) \bigoplus D_K(\widetilde{CC}[i+l]) \right) \bigoplus IV = IV. \tag{2}$$

Then the $(\widetilde{AD}, \widetilde{CC})$ pair will be an authenticated ciphertext with the tag $\widetilde{CC}[2l+2]$ (see Fig. 11). Indeed, Eqs. (1) and (2) together yield $\widehat{CC}[2] = \widetilde{CC}'[2l+2]$ by Lemma 1 and Proposition 1. Therefore, $\widetilde{CC}[2l+2] = \widehat{CC}[2]$ is a valid tag by Theorem 2.

It is clear that, the first l blocks of the plaintext of the forged ciphertext $(\widetilde{AD}, \widetilde{CT})$ will be the plaintext of the given ciphertext (AD, CT).

Note that $\widetilde{CC}[2l+2] = \widehat{CC}[2]$ is the valid tag for any forged $(\widetilde{AD}, \widetilde{CT})$ pair where $\widetilde{IV} = IV$ (see Lemma 2), $\widetilde{CC}[2l+1] = \widehat{CC}[1]$ and $\widetilde{CC}[i]$ blocks take any value, except $L = E_K(0)$, even number of times for $1 \le i \le 2l$. Therefore, the plaintext recovery attack can be converted to an existential forgery of $2l+1$-block such $\widetilde{CT}$s (see Lemma 1).

7. Permuting CT-blocks

In this section, we introduce a new property for COLM. The CT blocks can be permuted without any effect on the tag value. This results from a property of the linear operator of COLM explained in Proposition 2. This property can be exploited to mount a plaintext recovery attack and a universal forgery attack. During the encryption, a state value $W[i]$ depends on all $X[j]$-blocks, $1 \le j \le i$, through the linear operation ρ. Hence $W[i]$ can be written as a linear combination of all $X[j]$s along with the IV, namely $W[i] = \bigoplus_{j=1}^{i} 2^{i-j} \cdot X[j] \bigoplus 2^i \cdot IV$. Observe that each $X[j]$ has a different coefficient in this combination. Therefore, reordering the $MM[j]$ blocks changes the state value $W[i]$. However, this is not true for the decryption process. Lemma 4 states this property of COLM.

Lemma 4. *For a given l-block ciphertext CT produced by COLM, the $\widetilde{CT}$ obtained by swapping any two blocks $CC[i]$ and $CC[j]$ for $i, j < l$ is still valid with the same tag $CC[l+1]$.*

Proof. Let a ciphertext $CT = CC[1] \| \cdots \| CC[l]$ with its tag $CC[l+1]$ be given with the initialization value IV. During the decryption, the state value $W[l]$ can be written as the accumulation of all the $Y[j]$ values where $1 \le j \le l$ as $W[l] = \bigoplus_{j=1}^{l} Y[j] \bigoplus IV$ since $W[k] = W[k-1] \oplus Y[k]$ for any $1 \le k \le l$. By swapping $CC[i]$ and $CC[j]$, we swap $Y[i]$ and $Y[j]$ for $i, j < l$. The new CT has the following Y values $\widetilde{Y}[k] = Y[k]$ for any $k \ne i, j$; $\widetilde{Y}[i] = Y[j]$ and $\widetilde{Y}[j] = Y[i]$. Hence,

$$\widetilde{W}[l-1] = \bigoplus_{j=1}^{l-1} \widetilde{Y}[j] \bigoplus IV = \bigoplus_{j=1}^{l-1} Y[j] \bigoplus IV = W[l-1]$$

and

$$\widetilde{W}[l] = \bigoplus_{j=1}^{l} \widetilde{Y}[j] \bigoplus IV = \bigoplus_{j=1}^{l} Y[j] \bigoplus IV = W[l].$$

Therefore, the checksum of the PT blocks $MM[l]$ remains same and hence the new ciphertext $\widetilde{CC}$ is valid with the tag $CC[l+1]$. □

We can generalize the statement in Lemma 4 by applying arbitrary permutation instead of a swap operation.

Proposition 2. *For a given l-block ciphertext CT, the $\widetilde{CT}$ obtained by permuting the $CC[i]$ blocks for $i < l$ is still valid with the same tag $\widetilde{CC}[l+1] = CC[l+1]$.*

Proof. Let CT be a given ciphertext consists of $CC[1] \| \cdots \| CC[l]$ with its tag $CC[l+1]$ and the initialization value IV. Let the new ciphertext $\widetilde{CT}$ be obtained by permuting the $CC[i]$ blocks for $i < l$. On the other hand, any permutation can be written as the combination of two-cycles (swap operations). Hence, this new ciphertext $\widetilde{CT}$ can be obtained by swapping certain pairs of the CC blocks. However, after each swap the tag value remains same by Lemma 4. Therefore, the same tag $CC[l+1]$ is valid for $\widetilde{CT}$. □

Proposition 2 can be used to mount a plaintext recovery attack of any given (AD, CT) pair where $CT = CC[1] \| CC[2] \| CC[3] \| CC[4] \| \dots \| tag$ of at least 5 blocks. One can recover the plaintext by querying

$$D_K(AD, CC[1] \| CC[2] \| CC[4] \| CC[3] \| \dots \| tag)$$
(See Fig. 12)
$$D_K(AD, CC[2] \| CC[1] \| CC[3] \| CC[4] \| \cdots \| tag)$$
(See Fig. 13)

with an associated data AD and obtaining

$$MM[1] \| MM[2] \| \widetilde{MM}[3] \| \widetilde{MM}[4] \| \dots$$
$$\widetilde{MM}[1] \| \widetilde{MM}[2] \| MM[3] \| MM[4] \| \dots$$

respectively and then combining the first two blocks of the former plaintext with the rest of the latter plaintext. (see Figs. 12 and 13).

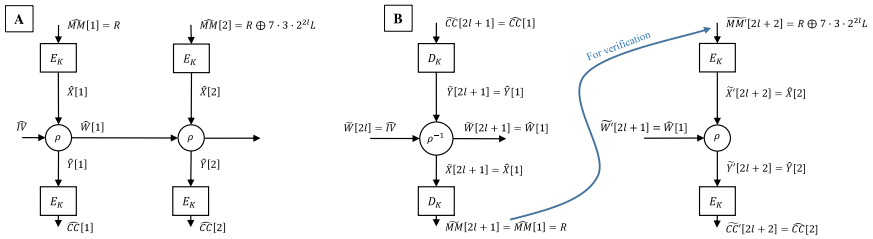

Fig. 11. Plaintext recovery attack in Section 6. In Part A: Two PT blocks with a special difference are queried to obtain $\widetilde{CC}[1]$ and $\widetilde{CC}[2]$. In Part B: $\widetilde{CC}[2l+1] = \widetilde{CC}[1]$ and $\widetilde{CC}[2l+2] = \widetilde{CC}[2]$ are verified through the special difference of $\widetilde{MM}[1]$ and $\widetilde{MM}[2]$ depicted in Part A.

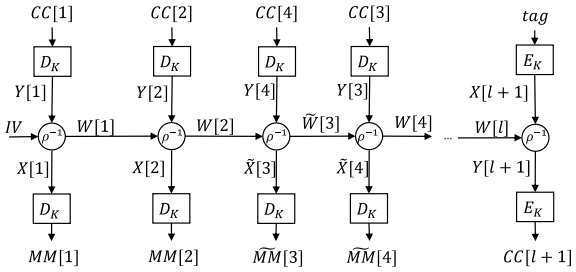

Fig. 12. First query to recover PT.

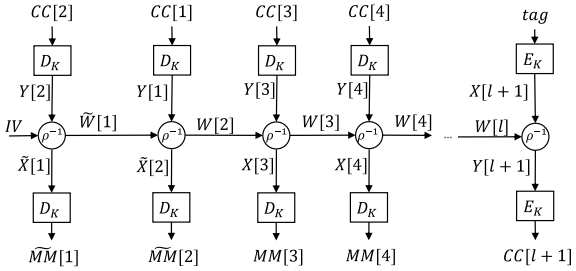

Fig. 13. Second query to recover PT.

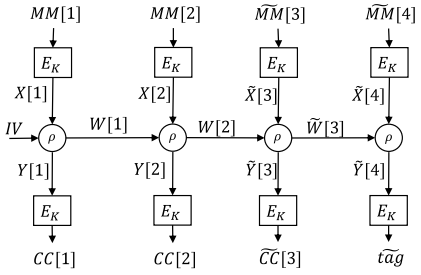

Fig. 14. First query to produce CT.

Proposition 2 can be used also to produce the CT and the tag of a given $PT = MM[1]\|MM[2]\|MM[3]\|\dots$ by querying the oracle as follows:

$CC[1]\|CC[2]\|\widetilde{CC}[3]\|\widetilde{tag}, \leftarrow \mathcal{E}_K(AD, MM[1]\|MM[2]\|\widetilde{MM}[3])$

(See Fig. 14)

$\widetilde{MM}[1]\|\widetilde{MM}[2]\|\widetilde{MM}[3], \leftarrow D_K(AD, CC[2]\|CC[1]\|\widetilde{CC}[3]\|\widetilde{tag})$

(See Fig. 15)

$CC[2]\|CC[1]\|CC[3]\|\dots\|tag. \leftarrow \mathcal{E}_K(AD, \widetilde{MM}[1]\|\widetilde{MM}[2]\|MM[3]\|\dots)$

(See Fig. 16)

with an associated data AD (see Figs. 14–16).

The third block of the target message is modified to be an arbitrary $\widetilde{MM}[3]$ value and the new message is encrypted to $CC[1]\|CC[2]\|$

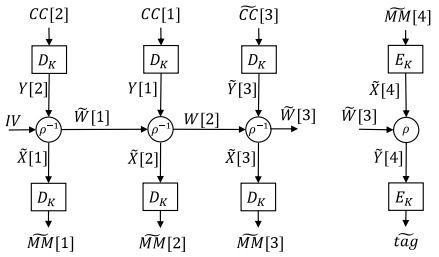

Fig. 15. Second query to produce CT.

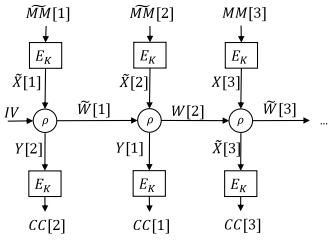

Fig. 16. Third query to produce CT.

Table 3
Some key recovery attacks against AES-128. MitM: Meet-in-the-Middle, ID: Impossible Differential.

Attack	Rounds	Data	Query	Time	Reference
MitM	7	2^{97} CPA	2^{105}	2^{99}	[38]
ID	7	2^{105} CPA	2^{113}	$2^{106.88}$	[39]
Biclique	10	2^{88} CCA	2^{97}	$2^{126.18}$	[40]

$\widetilde{CC}[3]\|\widetilde{tag}$, whose first two blocks will be the original CT blocks. The oracle decrypts this CT when we swap the first two blocks since the tag is still valid by Lemma 4. The first two $\widetilde{MM}$ blocks will result in random values ($\widetilde{MM}[1]$ and $\widetilde{MM}[2]$) but observe that $\widetilde{W}[2]$ will be equal to $W[2]$. Therefore, querying this $\widetilde{MM}[1]\|\widetilde{MM}[2]\|MM[3]\|\dots$ leads to $CC[2]\|CC[1]\|CC[3]\|\dots\|tag$ from which the original CT can be obtained by just swapping the first two blocks.

$MM[l]$, $MM[l+1]$ and $W[l-1]$ parameters are used for the tag verification. Permuting the first $(l-1)$ CT-blocks will not affect these parameters by Proposition 2, resulting in $(l-1)!$ existential forgeries.

8. Key recovery

In this Section, we introduce how to use COLM oracle to collect necessary data for mounting attacks against (reduced-round) AES-128.

There are several attacks against reduced-round AES-128. In general, these attacks require large amount of data, as either CPA or CCA. The question is how to collect data for an AES key used in the COLM encryption. In principle, any weakness of AES is not supposed to be exploited in COLM oracle even in nonce misuse scenario since COLM is chosen to the defense-in-depth final portfolio of the CAESAR

competition. However, as explained in Section 4, we can simulate AES encryption and decryption oracles for any given input, enabling us to collect data required for an attack against (reduced-round) AES-128.

We give three examples of combining these oracles (see Section 4) with published CPA/CCA attacks against reduced-round AES-128, a meet-in-the-middle (MitM) attack [38], an impossible differential (ID) attack [39] and a biclique attack [40]. The baseline complexities of these published attacks are listed in Table 3. In Section 4.2, we show that a single COLM query of a 132-block $AD-PT$ is enough to get the AES encryption of the any given input. Therefore, to collect the data for MitM and ID attacks, we need to make $262\times 2^{97}\approx 2^{105}$ and $262\times 2^{105}\approx 2^{113}$ AES-128 queries in COLM, respectively. In Section 4.1, we show that a single COLM query of a 261-block $AD-CT$ is enough to get the AES decryption of any given input. Therefore, to collect the data for biclique attack, we make $520\times 2^{88}\approx 2^{97}$ AES-128 queries in COLM. Recall that we need 2^{130} AES-128 encryptions ($L=E_K(0)$, $Z[0]=E_K(AA[0])$, $X[1]=E_K(MM[1])$, $CC[1]=E_K(Y[1])$ for each key candidate) to mount a brute force attack against COLM. Therefore, these attacks compared to brute force attacks are 4 times more time efficient in COLM oracle scenario than in AES oracle scenario.

9. Conclusions and suggestions

In this work, we analyze COLM. We mount plaintext recovery attacks, a tag guessing attack, several forgery attacks along with building an SEBC and an SDBC. To the best of our knowledge, a forgery is used in a simulation model for the first time in this work and hence we are able to build both an SEBC and an SDBC together. Note that, an attacker possessing both an SEBC and an SDBC can produce any plaintext or ciphertext like a legitimate user, who has access to the key. It is an open problem how to generalize our methods for mounting plaintext recovery and tag guessing attacks to an arbitrarily formed authenticated EME/EMD scheme.

EME algorithms have generally a whitening mask L defined by designers. In authenticated encryption, algorithms are supposed to have the security level of $\min(|key|,|block|)$ against some attacks such as plaintext recovery and tag guessing. Both the key and the block sizes are 128 bits in COLM. So, the security level of recovering L or the EME structure must be at least 128-bit. Forler et al. illustrate that L can be recovered in 2^{65} queries [11]. In this work, we show that the EME structure in COLM does not satisfy 128-bit security against plaintext recovery and tag guessing attacks. That is, neither the confidentiality of L nor EME of COLM satisfies 128-bit security.

We recommend that both recovering L and the EME structure should have the security level of $\min(|key|,|block|)$ for an authenticated EME scheme designed in defense-in-depth approach. Accordingly, if one of them fails to provide this security level, the authenticated EME scheme still remains secure.

We suggest some modifications for COLM to prevent our attacks. We propose a simple but unconventional suggestion to fulfill the security requirement for recovering L, by producing it with another key deduced from the main key by a slight modification. Even if L is recovered (note that Lu's method is still valid), it cannot be exploited in our attacks in Algorithm 3, Algorithm 1 and Algorithm 2 since L is produced by a different key and it does not give a plaintext–ciphertext pair for the underlying block cipher. Our next suggestion is to modify the ρ-operation. Note that $W[i]=W[i-1]\oplus Y[i]$ in the ρ^{-1} operation and we exploit this property in Lemmas 1, 4, Proposition 2, Theorems 1, 2, 4 and Algorithm 2. Hence, in the computation of $W[i]$ the coefficients of $W[i-1]$ inputs should not be 1 in both ρ and ρ^{-1}. This criterion should also be adopted for $W'[i]$. As an example; $W'[i]=3\cdot W'[i-1]\oplus A[i]$ for processing AD, and $W[i]=4\cdot W[i-1]\oplus X[i]$, $Y[i]=6\cdot W[i-1]\oplus X[i]$ for processing PT (resulting in $W[i]=2\cdot W[i-1]\oplus Y[i]$, $X[i]=6\cdot W[i-1]\oplus Y[i]$) can be used. Our last suggestion is to use different coefficients for the M blocks in the summation to compute the $M[l]$ and the $M[l+1]$ blocks so that one cannot isolate the last two blocks when verifying the tag. This suggestion will render Propositions 1, 2, Theorems 1, 2 and 3 invalid. Lastly, these suggestions are to prevent our attacks and COLM is still required to be analyzed with these amendments.

CRediT authorship contribution statement

Sırrı Erdem Ulusoy: Conceptualization, Methodology, Writing – original draft, Visualization. **Orhun Kara:** Validation, Formal analysis, Writing – review & editing, Supervision. **Mehmet Önder Efe:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Rogaway P. Authenticated-encryption with associated-data. In: Atluri V, editor. Proceedings of the 9th ACM conference on computer and communications security, CCS 2002, Washington, DC, USA, November 18-22, 2002. ACM; 2002, p. 98–107. http://dx.doi.org/10.1145/586110.586125.

[2] Dworkin MJ. SP 800-38D. Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. Tech. rep., Gaithersburg, MD, United States: National Institute of Standards & Technology; 2007.

[3] Cryptographic competitions, URL https://competitions.cr.yp.to/caesar-call.html.

[4] Bassham L, Çalık Ç, McKay K, Turan MS. Submission requirements and evaluation criteria for the lightweight cryptography standardization process. US National Institute of Standards and Technology; 2018, URL https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf.

[5] Tran SDP, Seok B, Lee C. HANMRE - an authenticated encryption secure against side-channel attacks for nonce-misuse and lightweight approaches. Appl Soft Comput 2020;97:106663. http://dx.doi.org/10.1016/j.asoc.2020.106663, URL https://www.sciencedirect.com/science/article/pii/S1568494620306013.

[6] Sakamoto K, Liu F, Nakano Y, Kiyomoto S, Isobe T. Rocca: An efficient AES-based encryption scheme for beyond 5G. IACR Trans Symmetric Cryptol 2021;2021(2):1–30. http://dx.doi.org/10.46586/tosc.v2021.i2.1-30, URL https://tosc.iacr.org/index.php/ToSC/article/view/8904.

[7] Naito Y, Sasaki Y, Sugawara T. AES-LBBB: AES mode for lightweight and BBB-secure authenticated encryption. IACR Trans Cryptogr Hardw Embedd Syst 2021;2021(3):298–333. http://dx.doi.org/10.46586/tches.v2021.i3.298-333, URL https://tches.iacr.org/index.php/TCHES/article/view/8976.

[8] Berti F, Guo C, Pereira O, Peters T, Standaert F-X. TEDT, a leakage-resist AEAD mode for high physical security applications. IACR Trans Cryptogr Hardw Embedd Syst 2019;2020(1):256–320. http://dx.doi.org/10.13154/tches.v2020.i1.256-320, URL https://tches.iacr.org/index.php/TCHES/article/view/8400.

[9] Lu J. Almost universal forgery attacks on the COPA and marble authenticated encryption algorithms. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security. ASIA CCS '17, New York, NY, USA: Association for Computing Machinery; 2017, p. 789–99. http://dx.doi.org/10.1145/3052973.3052981, URL https://doi.org/10.1145/3052973.3052981.

[10] Bay A, Ersoy O, Karakoç F. Universal forgery and key recovery attacks on elmd authenticated encryption algorithm. In: Cheon JH, Takagi T, editors. Advances in cryptology - ASIACRYPT 2016 - 22nd international conference on the theory and application of cryptology and information security, Hanoi, Vietnam, December 4-8, 2016, proceedings, Part I. Lecture notes in computer science, Vol. 10031, 2016, p. 354–68. http://dx.doi.org/10.1007/978-3-662-53887-6_13.

[11] Forler C, List E, Lucks S, Wenzel J. Reforgeability of authenticated encryption schemes. In: Pieprzyk J, Suriadi S, editors. Information security and privacy - 22nd Australasian conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, proceedings, Part II. Lecture notes in computer science, Vol. 10343, Springer; 2017, p. 19–37. http://dx.doi.org/10.1007/978-3-319-59870-3_2.

[12] Vaudenay S, Vizár D. Under Pressure: Security of CAESAR Candidates beyond their Guarantees. Cryptology ePrint archive, report 2017/1147, 2017, https://eprint.iacr.org/2017/1147.

[13] Datta N, Luykx A, Mennink B, Nandi M. Understanding RUP integrity of COLM. IACR Cryptol ePrint Arch 2017;2017:431.

[14] Gruber M, Probst M, Tempelmeier M. Persistent fault analysis of OCB, DEOXYS and COLM. In: 2019 workshop on fault diagnosis and tolerance in cryptography (FDTC). 2019, p. 17–24.

[15] Khairallah M, Bhasin S, Chattopadhyay A. On misuse of nonce-misuse resistance : Adapting differential fault attacks on (few) CAESAR winners. In: 2019 IEEE 8th international workshop on advances in sensors and interfaces (IWASI). 2019, p. 189–93.

[16] Sasaki Y. Improved related-tweakey boomerang attacks on deoxys-BC. In: Joux A, Nitaj A, Rachidi T, editors. Progress in cryptology - AFRICACRYPT 2018 - 10th international conference on cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, proceedings. Lecture notes in computer science, Vol. 10831, Springer; 2018, p. 87–106. http://dx.doi.org/10.1007/978-3-319-89339-6_6.

[17] Moazami F, Mehrdad A, Soleimany H. Impossible differential cryptanalysis on deoxys-BC-256. ISC Int J Inf Secur 2018;10(2):93–105. http://dx.doi.org/10.22042/isecure.2018.114245.405, arXiv:http://www.isecure-journal.com/article_66740_e63905b11e7f11a063d97296b15b96ed.pdf URL http://www.isecure-journal.com/article_66740.html.

[18] Eichlseder M, Nageler M, Primas R. Analyzing the linear keystream biases in AEGIS. IACR Trans Symmetric Cryptol 2020;2019(4):348–68. http://dx.doi.org/10.13154/tosc.v2019.i4.348-368, URL https://tosc.iacr.org/index.php/ToSC/article/view/8468.

[19] Ashur T, Eichlseder M, Lauridsen MM, Leurent G, Minaud B, Rotella Y, Sasaki Y, Viguier B. Cryptanalysis of MORUS. In: Peyrin T, Galbraith SD, editors. Advances in cryptology - ASIACRYPT 2018 - 24th international conference on the theory and application of cryptology and information security, Brisbane, QLD, Australia, December 2-6, 2018, proceedings, Part II. Lecture notes in computer science, Vol. 11273, Springer; 2018, p. 35–64. http://dx.doi.org/10.1007/978-3-030-03329-3_2.

[20] Dey P, Rohit RS, Adhikari A. Full key recovery of ACORN with a single fault. J Inf Secur Appl 2016;29:57–64. http://dx.doi.org/10.1016/j.jisa.2016.03.003, URL https://www.sciencedirect.com/science/article/pii/S2214212616300138.

[21] Jahanbani M, Norozi Z, Bagheri N. DPA protected implementation of OCB and COLM authenticated ciphers. IEEE Access 2019;7:139815–26.

[22] Bossuet L, Mancillas-Lopez C, Ovilla-Martinez B. Pipelined hardware implementation of COPA, ELmD, and COLM. IEEE Trans Comput 2020;1.

[23] Tempelmeier M, De Santis F, Sigl G, Kaps J. The CAESAR-API in the real world — Towards a fair evaluation of hardware CAESAR candidates. In: 2018 IEEE international symposium on hardware oriented security and trust (HOST). 2018, p. 73–80.

[24] Katsaiti M, Sklavos N. Implementation efficiency and alternations, on CAESAR finalists: AEGIS approach. In: 2018 IEEE 16th intl conf on dependable, autonomic and secure computing, 16th intl conf on pervasive intelligence and computing, 4th intl conf on big data intelligence and computing and cyber science and technology congress(DASC/PiCom/DataCom/CyberSciTech). 2018, p. 661–5.

[25] Abbas A, Mostafa H, Mohieldin AN. Low area and low power implementation for CAESAR authenticated ciphers. In: 2018 new generation of CAS (NGCAS). 2018, p. 49–52.

[26] Farahmand F, Diehl W, Abdulgadir A, Kaps J, Gaj K. Improved lightweight implementations of CAESAR authenticated ciphers. In: 2018 IEEE 26th annual international symposium on field-programmable custom computing machines (FCCM). 2018, p. 29–36.

[27] Andreeva E, Bogdanov A, Luykx A, Mennink B, Tischhauser E, Yasuda K. Parallelizable and Authenticated Online Ciphers. Cryptology ePrint archive, report 2013/790, 2013, https://eprint.iacr.org/2013/790.

[28] Bossuet L, Datta N, Mancillas-López C, Nandi M. Elmd: A pipelineable authenticated encryption and its hardware implementation. IEEE Trans Comput 2016;65(11):3318–31. http://dx.doi.org/10.1109/TC.2016.2529618.

[29] Andreeva E, Bogdanov A, Datta N, Luykx A, Mennink B, Nandi M, Tischhauser E, Yasuda K. COLM v1. 2016, https://competitions.cr.yp.to/round3/colmv1.pdf.

[30] N I of Standards. Advanced encryption standard, Vol. 197. NIST FIPS PUB; 2001.

[31] Jean J, Nikolic I, Peyrin T, Seurin Y. Deoxys v1. 41, Vol. 124. CAESAR; 2016, submitted for publication.

[32] Cryptographic competitions, URL https://competitions.cr.yp.to/caesar-submissions.html.

[33] Announcement of the CAESAR finalists. 2020, https://cr.yp.to/talks/2018.03.05/slides-djb-20180305-caesar-4x3.pdf, Accessed: 2020-05-25.

[34] Guo J. Marble specification version 1.1. 2014, https://competitions.cr.yp.to/round1/marblev11.pdf.

[35] Demirci H, Selçuk AA. A meet-in-the-middle attack on 8-round AES. In: Nyberg K, editor. Fast software encryption, 15th international workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, revised selected papers. Lecture notes in computer science, Vol. 5086, Springer; 2008, p. 116–26. http://dx.doi.org/10.1007/978-3-540-71039-4_7.

[36] Vaudenay S, Vizár D. Can caesar beat galois? - robustness of CAESAR candidates against nonce reusing and high data complexity attacks. In: Preneel B, Vercauteren F, editors. Applied cryptography and network security - 16th international conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, proceedings. Lecture notes in computer science, Vol. 10892, Springer; 2018, p. 476–94. http://dx.doi.org/10.1007/978-3-319-93387-0_25.

[37] Zhang F, Liang Z, Yang B, Zhao X, Guo S, Ren K. Survey of design and security evaluation of authenticated encryption algorithms in the CAESAR competition. Front Inf Technol Electron Eng 2018;19(12):1475–99. http://dx.doi.org/10.1631/FITEE.1800576.

[38] Derbez P, Fouque P, Jean J. Improved key recovery attacks on reduced-round AES in the single-key setting. In: Johansson T, Nguyen PQ, editors. Advances in cryptology - EUROCRYPT 2013, 32nd annual international conference on the theory and applications of cryptographic techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture notes in computer science, Vol. 7881, Springer; 2013, p. 371–87. http://dx.doi.org/10.1007/978-3-642-38348-9_23.

[39] Boura C, Lallemand V, Naya-Plasencia M, Suder V. Making the impossible possible. J Cryptol 2018;31:101–33. http://dx.doi.org/10.1007/s00145-016-9251-7.

[40] Bogdanov A, Khovratovich D, Rechberger C. Biclique cryptanalysis of the full AES. In: Lee DH, Wang X, editors. 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, Springer; 2011, p. 344–71. http://dx.doi.org/10.1007/978-3-642-25385-0_19.