Almost Universal Forgery Attacks on the COPA and Marble Authenticated Encryption Algorithms*

Jiqiang Lu
Institute for Infocomm Research,
Agency for Science, Technology and Research
1 Fusionopolis Way, Singapore 138632
jlu@i2r.a-star.edu.sg, lvjiqiang@hotmail.com

ABSTRACT

The COPA authenticated encryption mode was proved to have a birthday-bound security on integrity, and its instantiation AES-COPA (v1/2) was claimed or conjectured to have a full security on tag guessing. The Marble (v1.0/1.1/1.2) authenticated encryption algorithm was claimed to have a full security on authenticity. Both AES-COPA (v1) and Marble (v1.0) were submitted to the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) in 2014, and Marble was revised twice (v1.1/1.2) in the first round of CAESAR, and AES-COPA (v1) was tweaked (v2) for the second round of CAESAR. In this paper, we cryptanalyse the basic cases of COPA, AES-COPA and Marble, that process messages of a multiple of the block size long; we present collision-based almost universal forgery attacks on the basic cases of COPA, AES-COPA (v1/2) and Marble (v1.0/1.1/1.2), and show that the basic cases of COPA and AES-COPA have roughly at most a birthday-bound security on tag guessing and the basic case of Marble has roughly at most a birthday-bound security on authenticity. The attacks on COPA and AES-COPA do not violate their birthday-bound security proof on integrity, but the attack on AES-COPA violates its full security claim or conjecture on tag guessing. Therefore, the full security claim or conjecture on tag guessing of AES-COPA and the full security claim on authenticity of Marble are incorrectly far overestimated in the sense of a general understanding of full security of these security notions. Designers should pay attention to these attacks when designing authenticated encryption algorithms with similar structures in the future, and should be careful when claiming the security of an advanced form of a security notion without making a corresponding proof after proving the security of the security notion only under its most fundamental form.

Keywords

Cryptology; Authentication, Authenticated encryption algorithm; COPA; Marble; Universal forgery attack

1. INTRODUCTION

In symmetric cryptography, an authenticated encryption algorithm is an algorithm that transforms an arbitrary-length data stream (below an upper bound generally), called a message or plaintext, into another data stream of the same length, called a ciphertext, and generates an (authentication) tag for the message at the same time, under the control of a secret key [19]. It combines the functionalities of a symmetric cipher and a message authentication code (MAC), and achieves data confidentiality and integrity/authenticity at one pass. We refer the reader to Bellare and Namprempre's work [6] for an introduction to authenticated encryption and a few security notions under provable security, such as privacy/confidentiality, integrity/authenticity, and unforgeability, although one may use a different definition for a security notion.

Like existential and universal forgery attacks [20, 25] on a MAC, an existential forgery attack on an authenticated encryption algorithm is to produce a correct ciphertext-tag pair which is not given before (under the secret key and some public nonce if any. Thus, during the decryption and tag verification phase, the message resulted from decrypting the forged ciphertext can result in the forged tag under the same key and nonce if any.), while a universal forgery attack on an authenticated encryption algorithm is to produce the correct ciphertext-tag pair for any specified message whose ciphertext-tag pair is not given before (under the secret key and public nonce if any. Thus, during the decryption and tag verification phase, the specified message will be generated from decrypting the forged ciphertext, and result in the forged tag under the same key and nonce if any). Note that a universal forgery attack implies an existential forgery attack, and thus an algorithm secure against existential forgery attacks is also secure against universal forgery attacks, but not vice versa — a universal forgery attack represents a more serious security threat and usually has a higher complexity level than an existential forgery attack. Besides, Dunkelman, Keller and Shamir [9] recently introduced the notion of almost universal forgery attack on MAC, which works for almost any specified message although not for any.

*Earlier versions of the materials of this paper appeared in 2015 in the forum of the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) [16, 17] and in IACR Cryptology ePrint Archive Report 2015/079 [18].

ASIA CCS '17, April 02-06, 2017, Abu Dhabi, United Arab Emirates

DOI: http://dx.doi.org/10.1145/3052973.3052981

Table 1: Main (almost) universal forgery attacks on COPA and Marble, where n is the block length of the underlying block cipher and e is the base of the natural logarithm.

Algorithm	Associated Data	Data	Memory	Time	Success Rate	Source
COPA	variable	$2^{\sigma}+2^{\varphi}$queries ($1 \le \sigma, \varphi \le \frac{n}{2}$)	$n \cdot 2^{\sigma}$bits	2^{φ}memory accesses	$1-\mathsf{e}^{-2^{\sigma+\varphi-n}}$	Sect. 3.1
	constant	$2^{\theta}+2^{\phi}$queries ($\frac{n}{2} < \theta, \phi < n$)	$3n \cdot 2^{\phi}$bits	2^{ϕ}simple operations	See Sect. 3.2	Sect. 3.2
		2^{64}queries	$2^{69.6}$bytes	2^{64}memory accesses	20%	Sect. 3.3§
AES-COPA v1/2	variable (nonce-respecting)	$\sim 2^{63}$queries ($\sim 2^{64}$blocks)	2^{66}bytes	2^{62}memory accesses	6%	Sect. 3.1.4
	constant (nonce-misuse)	2^{124}queries	$2^{120.6}$bytes	2^{124}simple operations	32%	Sect. 3.2.4
		$\sim 2^{63}$ queries ($\sim 2^{64}$blocks)	$2^{68.6}$bytes	2^{63}memory accesses	6%	Sect. 3.3§
Marble v1.1/1.2	variable	2^{65}queries	2^{68}bytes	2^{65}memory accesses	32%	[10, 11]†
Marble v1.0/1.1/1.2	variable	2^{65}queries ($2^{66.6}$blocks)	2^{68}bytes	2^{65}memory accesses	32%	Sect. 4‡ [18]‡

†: A forgery is based on modifying associated data. ‡: A forgery is based on modifying message or associated data.
§: Suggested by an anonymous reviewer.

Proposed for parallel architectures such as general-purpose graphics processing units (GPGPUs), COPA [3] is a block-cipher-based authenticated encryption mode; and its instantiation with the AES [24] block cipher under 128 key bits is called AES-COPA (v1) [1]. Marble (v1.0) [12] is an AES-based COPA-like authenticated encryption algorithm. For both COPA and Marble, the key length is equal to the tag length. In March 2014, AES-COPA (v1) and Marble (v1.0) were submitted to the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) [7]. Shortly later, a revision (v1.1) [13] to Marble was made in the first round of CAESAR.

The COPA designers [3,4] proved that COPA has (roughly) a birthday-bound security on integrity (which is mainly associated with existential forgery) under the assumption that the underlying block cipher is a strong pseudorandom permutation, put a birthday-bound constraint on the maximum number of data blocks that AES-COPA (v1) can process with a single key, and claimed that AES-COPA (v1) had a full (i.e. 128-bit) security against tag guessing (which is associated with universal forgery) by writing '*security against tag guessing is 128 bits*', in addition to a birthday-bound (i.e. 64-bit) security on integrity. The Marble designer claimed that Marble achieved a full (i.e. 128-bit) security on privacy and authenticity. However, in January 2015, Fuhr et al. [10] presented universal forgery and key recovery attacks on the revised version (i.e. v1.1) of Marble, and the Marble designer made another revision (v1.2) [14] to Marble. In May 2015, Nandi [22] presented an existential forgery attack on the case of COPA that processes fractional messages (that is, messages are not a multiple of the block size long, and thus message padding is required), basing it on his earlier cryptanalysis result [21] on the XLS [26] pseudorandom permutation construction. In September 2015, the AES-COPA designers made a tweak (v2) [2] to the case of fractional messages for the second round of CAESAR, and conjectured that '*security against tag guessing is 128 bits.*'

1.1 Our Contributions

In this paper, we cryptanalyse the basic cases of COPA (as well as AES-COPA v1/2) and Marble (v1.0/1.1/1.2), that process messages of a multiple of the block size long, against almost universal forgery, and obtain the following main cryptanalytic results on COPA, AES-COPA and Marble, with only chosen queries to their message encryption and tag generation oracles:

- We present collision-based almost universal forgery attacks on the basic case of COPA under variable associated data, each of which has a complexity that is very near the birthday bound, by using an idea similar to but much simpler than Fuhr et al.'s attack on Marble. More importantly, when applied to the basic case of AES-COPA (v1/2) in the nonce-respecting scenario, each attack requires slightly less than 2^{63} encryption queries with the total (associated data, message) pairs having a length slightly less than the maximum block number 2^{64} that AES-COPA can process with a single key, and a memory of about 2^{66} bytes, and has a time complexity of about 2^{62} memory accesses and a success probability of about 6%. Though the success probability 6% is not very high, it is not negligible even in reality, and the attack is of semi-practical significance.

- We present a (multi-)collision-based almost universal forgery attack on the basic case of COPA under constant or no associated data, by using a novel idea. When applied to the basic case of AES-COPA (v1/2) in the nonce-misuse scenario, it requires about 2^{124} encryption queries and a memory of $2^{120.6}$ bytes and has a computational complexity of about 2^{124} simple operations and a success probability of about 32%. The attack is mainly of academic interest, due to its large data complexity that is far beyond the birthday bound. Anyway, there is an efficient birthday-bound attack suggested by an anonymous reviewer.

- We present collision-based almost universal forgery attacks on the basic case of Marble (v1.0/1.2) under variable associated data (in our earlier work [17, 18]), following Fuhr et al.'s attack [10] on Marble v1.1. Each attack has a data/time/memory complexity of about 2^{65}. However, since Fuhr et al. recently extended their attack on Marble v1.1 to Marble v1.2 in the final publication version [11] of the earlier work [10], who acknowledged our attacks by writing 'as shown independently by ourselves and Lu', we only focus on a different forgery way for an almost universal forgery on Marble v1.0/1.1 in this final publication version of our work.

Table 1 summarises previously published and our main (almost) universal forgery attacks on COPA and Marble.

Our attacks on COPA and AES-COPA do not violate their birthday-bound security proof on integrity, but the attack on AES-COPA violates its full (i.e. 128-bit) security claim or conjecture on tag guessing. In summary, our attacks suggest that the full security claim and conjecture on tag guessing of AES-COPA and the full security claim on authenticity of Marble are incorrectly far overestimated in the sense of a general understanding of full security of these security notions. More specifically, our attacks have the following meanings:

1. Our attacks suggest that the AES-COPA designers should also claim a birthday-bound security on tag guessing, instead of a full security. Although the AES-COPA designers proved a birthday-bound security on integrity (i.e. existential forgery resistance) by referring to the integrity security proof of COPA, they did not prove its security on tag guessing (i.e. universal forgery), but they claimed a full security for it. Our attacks have a complexity similar to the complexity of the proven birthday-bound security on integrity, showing that AES-COPA (v1/2) has roughly (at most) a birthday-bound security against tag guessing in the nonce-respecting scenario, rather than a full security as the designers claimed or conjectured. (Note that AES-COPA merged recently with another second round candidate of CAESAR and the merger [5] went into the third round of CAESAR in August 2016. The merger uses a completely different nonce process and does not make any security claim or conjection on tag guessing or universal forgery resistance.)

2. The COPA designers proved a birthday-bound security on integrity (i.e. existential forgery resistance), but did not specify its security against universal forgery. As mentioned earlier, existential and universal forgery attacks represent different threat levels and usually have different complexity levels. The security claim and conjecture of AES-COPA (v1/2) indicated that the designers might have thought that COPA had a full security against universal forgery (even under the birthday-bound data constraint), however, our attacks show that COPA has roughly (at most) a birthday-bound security against universal forgery, the same security level as for integrity. Thus, COPA users should not take it for granted that the general belief of a full security on universal forgery holds for COPA, and should not misuse COPA for such a full security in reality.

3. Our attacks show that Marble has roughly (at most) a birthday-bound security on authenticity, rather than a full security that the designer claimed. We would like to mention that as a consequence, our attacks resulted partially in the withdrawal of Marble from the CAESAR competition in January 2015, together with Fuhr et al.'s attack [10].

4. Our attacks are mainly based on the structures of COPA and Marble, and thus designers should pay attention to these attacks when designing authenticated encryption algorithms with similar structures in the future.

5. Lastly, if some security notion of a cryptographic algorithm is proved under its most fundamental form, it should be careful when claiming the security of an advanced form of the security notion without making a corresponding proof, for example, claiming universal forgery security after proving integrity only under existential forgery security, claiming key/plaintext/state recovery security after proving confidentiality/privacy only under distinguishing attack security [27]. Strictly speaking, a corresponding proof or justification is also required for a security claim on such an advanced form.

1.2 Organization

The remainder of the paper is organised as follows. In the next section, we give the notation used throughout this paper and briefly describe the basic cases of the COPA and Marble algorithms that process messages of a multiple of the block size long. We present our almost universal forgery attacks on COPA (as well as AES-COPA) and Marble in Sections 3 and 4, respectively. Section 5 concludes this paper.

2. PRELIMINARIES

In this section, we give the notation used throughout this paper and briefly describe the concerned basic cases of COPA and Marble that process messages of a multiple of the block size long (that is, no message padding is required). We refer the reader to [1–4,12–14] for detailed specifications of COPA and Marble.

2.1 Notation

We use the following notation throughout this paper.

$\oplus$	bitwise logical exclusive OR (XOR) operation
$*$	polynomial multiplication modulo the polynomial $x^{128} \oplus x^7 \oplus x^2 \oplus x \oplus 1$ in GF(2^{128})
$\|\|$	string concatenation
e	the base of the natural logarithm ($\mathsf{e} = 2.71828\cdots$)

2.2 The COPA Authenticated Encryption Algorithm

The COPA [3] authenticated encryption mode was published in 2013. Its internal state, key and tag have the same length as the block size of the underlying block cipher. It has mainly three phases: processing associated data, message encryption, and tag generation. Fig. 1 illustrates the message encryption and tag generation phase of COPA, where

- $\mathbf{E}_K$ is an n-bit block cipher with a k-bit user key K;

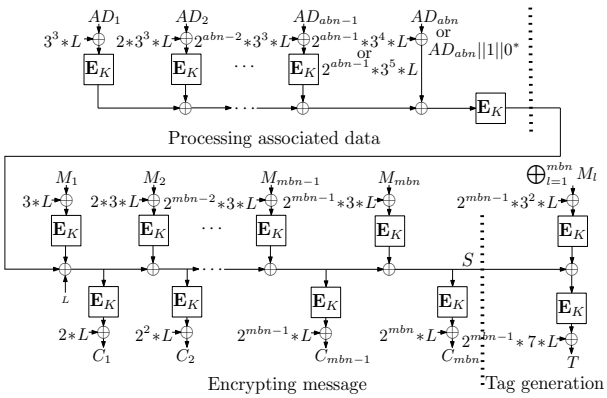

Figure 1: Message encryption and tag generation of COPA

- $L = \mathbf{E}_K(0)$ is an n-bit secret internal parameter, which is called subkey sometimes [1, 2];
- S is an n-bit internal state;
- $(AD_1, AD_2, \cdots, AD_{abn})$ is an associated data of abn n-bit blocks;
- $(M_1, M_2, \cdots, M_{mbn})$ is a message of mbn n-bit blocks;
- $(C_1, C_2, \cdots, C_{mbn})$ is the ciphertext for $(M_1, M_2, \cdots, M_{mbn})$; and
- T is the tag for $(M_1, M_2, \cdots, M_{mbn})$.

COPA first computes the secret parameter L, and then generates a number of dummy masks of the form $2^i * 3^j * 7^l * L$ for specific indices i, j and l. During the processing associated data phase, associated data should be padded if it is not a multiple of n bits long, by appending first a one then as many zeros as required to reach a multiple of n; then the (padded) associated data is divided into a series of n-bit blocks, each block is XORed with its corresponding mask, and the XORed value goes though a block cipher encryption operation $\mathbf{E}_K$; and finally the outputs of the block cipher encryption operations are XORed and the resulting value goes though another block cipher encryption operation $\mathbf{E}_K$. During the message encryption phase, the message is divided into a series of n-bit blocks, each message block is XORed with its corresponding mask, goes though a block cipher encryption operation $\mathbf{E}_K$, is XORed with the most recent state value (and the parameter L only for the first message block), and finally the XORed value goes though another block cipher encryption operation $\mathbf{E}_K$ and is XORed with another corresponding mask to produce a ciphertext block. During the tag generation phase, the XOR sum of the message blocks is XORed with the corresponding mask, goes though a block cipher encryption operation $\mathbf{E}_K$, is XORed with the most recent state value, and finally the XORed value goes though another block cipher encryption operation $\mathbf{E}_K$ and is XORed with another corresponding mask to produce the tag for the message.

Decryption is the inverse of encryption, and tag verification is identical to tag generation. COPA can be used without associated data, by setting the output of the processing associated data phase to zero.

In 2014, AES-COPA (v1) [1] — an instantiation of COPA that uses AES with 128 key bits [24]— was submitted to the CAESAR competition [7], where a nonce of 128 bits long is used and is appended to associated data, and the resulting value is treated as the associated data in the COPA mode. The designers claimed a 128-bit security against tag guessing for AES-COPA (v1) [1] without giving a proof or explanation on the security. In 2015, the designers made a tweaked version (v2) [2], and also changed the previous security claim on tag guessing to a conjecture without explanation. Under the basic cases that process messages of a multiple of the block size long, AES-COPA v2 differs from AES-COPA v1 only in that the last mask parameter of the tag generation phase becomes $2^{mbn} * 7 * L$.

2.3 The Marble Authenticated Encryption Algorithm

The Marble [12] authenticated encryption algorithm is similar to COPA. Marble has four phases: initialization, processing associated data, message encryption, and tag generation. Compared with COPA, Marble has mainly two structural distinctions at a high level: First, it has three layers of block cipher encryption operations to have an internal state that is twice as long as its key or tag in order to achieve a full security; second, the processing associated data phase produces anther secret parameter τ, which is to be used in the tag generation phase. Fig. 2 illustrates the message encryption and tag generation phase of the newest version (i.e. v1.2) of Marble, where

- each of the operations $\mathbf{E_1}, \mathbf{E_2}$ and $\mathbf{E_3}$ is a 4-round reduced version of the AES block cipher, with four fixed round subkeys chosen from the eleven round subkeys of the AES with 128 key bits;
- the TRANS operation is defined as $\text{TRANS}(x, y) = (x \oplus y, 3 * x \oplus y)$, where x and y are 128-bit inputs;
- $Const_0, Const_1$ and $Const_2$ are three 128-bit constants;
- S_1 and S_2 are two 128-bit internal states;
- $(AD_1, AD_2, \cdots, AD_{abn})$ is an associated data of abn 128-bit blocks;
- L and τ are 128-bit secret parameters;
- $(M_1, M_2, \cdots, M_{mbn})$ is a message of mbn 128-bit blocks;
- $(C_1, C_2, \cdots, C_{mbn})$ is the ciphertext for $(M_1, M_2, \cdots, M_{mbn})$; and
- T is the tag for $(M_1, M_2, \cdots, M_{mbn})$.

No nonce is used in Marble. (Note that in the last two versions (v1.1/1.2) [13, 14] the designer mentioned that one can opt to replace $Const_0$ with a nonce, but this option is not recommended by the designer). Decryption is the inverse of encryption, and tag verification is identical to tag generation.

Under the basic cases that process messages of a multiple of the block size long, the distinctions among the three versions of Marble are: (1) associated data with the last block being full should not be padded in Marble v1.0, but should also be padded in Marble v1.1/1.2; (2) the mask parameter before $\mathbf{E_1}$ for the last block of associated data is

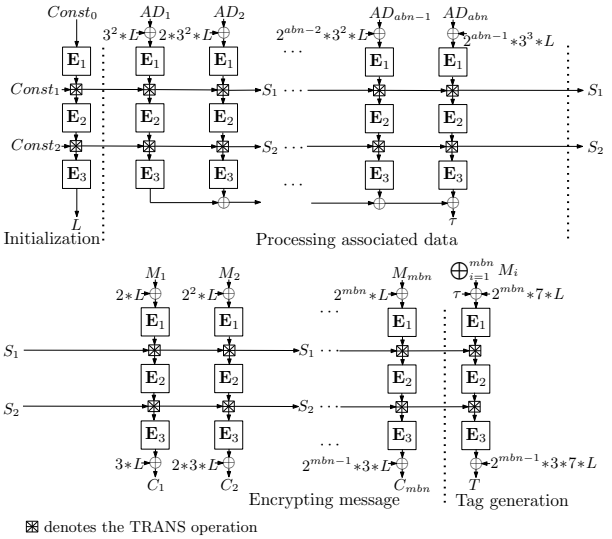

Figure 2: Message encryption and tag generation of Marble

$2^{abn-1} * 3^3 * L$ in Marble v1.0/1.2, and is $2^{abn-1} * 3^2 * L$ in Marble v1.1; and (3) when there is no associated data, Marble v1.0/1.1 simply sets $\tau = 0$ (but an empty message is not allowed), while Marble v1.2 processes a padded block of associated data.

3. (ALMOST) UNIVERSAL FORGERY ATTACKS ON THE BASIC CASES OF COPA AND AES-COPA

In this section, we first present almost universal forgery attacks on the basic case of COPA (that processes messages of a multiple of the block size long) under variable associated data, then present an almost universal forgery attack on the basic case of COPA under constant (or no) associated data, and describe their applications to the basic case of AES-COPA (v1/2); at last, we brief a more efficient attack on COPA and AES-COPA under constant (or no) associated data, suggested by an anonymous reviewer. Note that the distinction between the two versions of AES-COPA does not make much sense in these attacks.

3.1 (Almost) Universal Forgery Attacks on the Basic Case of COPA under Variable Associated Data

We first describe our attack idea at a high level, then show how to recover the secret parameter L in a more advantageous way than exhaustive key search, next describe three ways to make an almost universal forgery once L is recovered, and more importantly we apply them to AES-COPA (v1/2) in the nonce-respecting scenario.

3.1.1 Attack Idea

Each of the attacks consist of two phases: recovering the secret parameter L, followed by a forgery if L is recovered, while the attacks share the same phase of recovering L but use different ways for a forgery.

To recover L we use an idea similar to but much simpler than Fuhr et al.'s universal forgery attack on Marble v1.1, due to the structure of COPA. We fix a one-block message and choose a set of associated data of one block long and the other set of associated data of less than one block long which meet a condition after padding. The two sets of associated data mean that two different mask parameters are used for the two sets of associated data by the padding rule. At last, we recover L by looking for a collision on the ciphertext blocks.

We then use three ways to make a forgery: modifying only message, or modifying only associated data, or modifying both message and associated data.

3.1.2 Recovering the Secret Parameter L

The procedure is as follows, which is illustrated in Fig. 3.

1. Choose 2^σ (associated data of one n-bit block long, fixed message of one n-bit block long) pairs $(AD_1^{(i)}, M_1) = (i, M_1)$, where $0 < \sigma \leq \frac{n}{2}$ and $i = 0, 1, \cdots, 2^\sigma - 1$. Query the COPA encryption and tag generation oracle, and obtain all the ciphertexts and tags for the 2^σ (associated data, message) pairs; we denote by $C_1^{(i)}$ and $T^{(i)}$ the ciphertext and tag under associated data $AD_1^{(i)}$, respectively. Store $C_1^{(i)}$ into a table indexed by $C_1^{(i)}$.

2. Choose $(2^\varphi - 1)$ (associated data of less than n bits long, the same fixed message of one n-bit block long) pairs such that the (padded associated data, message) pairs $(\widehat{AD_1}^{(j)}, M_1) = (j \times 2^{\frac{n}{2}}, M_1)$, where $0 < \varphi \leq \frac{n}{2}$, $j = 1, 2, \cdots, 2^\varphi - 1$. (The padded associated data are possible by the padding rule for associated data of COPA, namely, first a one then as many zeros as required to reach a multiple of the block size n. Note that 0 is an impossible value for the block of padded associated data.) Query the COPA encryption and tag generation oracle, and obtain all the ciphertexts and tags for the $(2^\varphi - 1)$ (associated data, message) pairs; we denote by $\widehat{C}_1^{(j)}$ and $\widehat{T}^{(j)}$ the ciphertext and tag under associated data $\widehat{AD_1}^{(j)}$, respectively.

3. Check whether $\widehat{C}_1^{(j)}$ matches one of the set $\{C_1^{(i)} | i = 0, 1, \cdots, 2^\sigma - 1\}$ for $j = 1, 2, \cdots, 2^\varphi - 1$. We denote the match(es) by $(\widehat{C}_1^{(\omega)}, C_1^{(\mu)})$ if any, that is $\widehat{C}_1^{(\omega)} = C_1^{(\mu)}$.

4. For the match $(\widehat{C}_1^{(\omega)}, C_1^{(\mu)})$, we have $AD_1^{(\mu)} \oplus 3^4 * L = \widehat{AD_1}^{(\omega)} \oplus 3^5 * L$ by the structure of COPA. Thus, we can recover L from this equation.

The reason that we use padded associated data in Step 2 is that an input mask (i.e. $3^5 * L$) different from the one (i.e. $3^4 * L$) used in Step 1 will be introduced for the first block of (padded) associated data. This state recovery attack requires approximately $2^\sigma + 2^\varphi$ encryption queries, a memory of approximately $n \cdot 2^\sigma$ bits (as we do not need to store $\widehat{C}_1^{(j)}$), and has a time complexity of about 2^φ memory accesses (from Step 3) and a success probability of approximately $1 - \binom{2^\sigma \cdot (2^\varphi - 1)}{0} \cdot (2^{-n})^0 \cdot (1 - 2^{-n})^{2^\sigma \cdot (2^\varphi - 1)} \approx 1 - \mathsf{e}^{-2^{\sigma+\varphi-n}}$.

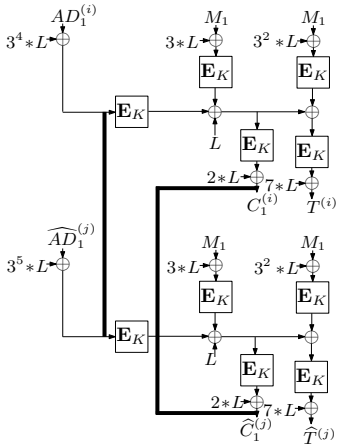

Figure 3: State recovery attack on COPA under variable associated data

3.1.3 Making an (Almost) Universal Forgery

If the secret parameter L is recovered by the above state recovery attack, we have three ways to make a universal forgery attack on COPA with a single query at a 100% success probability. Below we assume a target (associated data of abn n-bit blocks long, message of mbn n-bit blocks long) pair $(AD, M) = (AD_1, AD_2, \cdots, AD_{abn}, M_1, M_2, \cdots, M_{mbn})$, where $abn > 0$ and $mbn \geq 0$.

One way is similar to Fuhr et al.'s universal forgery attack [10] on Marble v1.1, which is based on modifying only associated data and can make a forgery on the same message under different associated data. Its main idea is to insert two additional blocks of associate data and cancel their outputs immediately after the first layer of block cipher encryptions, due to the XOR sum feature of the processing associated data phase. It works as follows.

1. Query the COPA encryption and tag generation oracle with the (associated data of $(abn + 2)$ blocks long, the same message) pair $(\widetilde{AD}, M) = (AD_1, AD_2, \cdots, AD_{abn-1}, \widetilde{AD}_{abn}, \widetilde{AD}_{abn} \oplus 2^{abn} * 3^3 * L \oplus 2^{abn-1} * 3^3 * L, AD_{abn} \oplus 2^{abn-1} * 3^4 * L \oplus 2^{abn+1} * 3^4 * L, M_1, M_2, \cdots, M_{mbn})$, where $\widetilde{AD}_{abn}$ is an arbitrary block. Obtain its ciphertext and tag, denoted respectively by $\widetilde{C} = (\widetilde{C}_1, \widetilde{C}_2, \cdots, \widetilde{C}_{mbn})$ and $\widetilde{T}$.

2. The ciphertext for (AD, M) is $C = (\widetilde{C}_1, \widetilde{C}_2, \cdots, \widetilde{C}_{mbn})$, and the tag for (AD, M) is $\widetilde{T}$.

The second way is based on modifying only message, and can make a forgery on the same associated data under different messages. Its main idea is to append an additional block of message with a particular value and deduce the correct tag from the corresponding ciphertext block, due to the fact that the tag generation phase has the same internal structure as the two block cipher encryptions after a message block. It works as follows.

1. Query the COPA encryption and tag generation oracle with the (the same associated data, message of $(mbn+1)$ n-bit blocks long) pair $(AD, \widetilde{M}) = (AD_1, AD_2, \cdots, AD_{abn}, M_1, M_2, \cdots, M_{mbn}, 2^{mbn} * 3 * L \oplus 2^{mbn-1} * 3^2 * L \oplus \bigoplus_{i=1}^{mbn} M_i)$, and obtain its ciphertext $\widetilde{C} = (\widetilde{C}_1, \widetilde{C}_2, \cdots, \widetilde{C}_{mbn}, \widetilde{C}_{mbn+1})$.

2. The ciphertext for (AD, M) is $C = (\widetilde{C}_1, \widetilde{C}_2, \cdots, \widetilde{C}_{mbn})$, and the tag for (AD, M) is $\widetilde{C}_{mbn+1} \oplus 2^{mbn+1} * L \oplus 2^{mbn-1} * 7 * L$.

The third way is based on modifying both message and associated data, which is a combination of the first two ways, and can make a forgery under different associated data and different messages, as follows.

1. Query the COPA encryption and tag generation oracle with the (associated data of $(abn+2)$ blocks long, message of $(mbn + 1)$ n-bit blocks long) pair $(\widetilde{AD}, \widetilde{M}) = (AD_1, AD_2, \cdots, AD_{abn-1}, \widetilde{AD}_{abn}, \widetilde{AD}_{abn} \oplus 2^{abn} * 3^3 * L \oplus 2^{abn-1} * 3^3 * L, AD_{abn} \oplus 2^{abn-1} * 3^4 * L \oplus 2^{abn+1} * 3^4 * L, M_1, M_2, \cdots, M_{mbn}, 2^{mbn} * 3 * L \oplus 2^{mbn-1} * 3^2 * L \oplus \bigoplus_{i=1}^{mbn} M_i)$, and obtain its ciphertext $\widetilde{C} = (\widetilde{C}_1, \widetilde{C}_2, \cdots, \widetilde{C}_{mbn}, \widetilde{C}_{mbn+1})$.

2. The ciphertext for (AD, M) is $C = (\widetilde{C}_1, \widetilde{C}_2, \cdots, \widetilde{C}_{mbn})$, and the tag for (AD, M) is $\widetilde{C}_{mbn+1} \oplus 2^{mbn+1} * L \oplus 2^{mbn-1} * 7 * L$.

The correctness of the three ways can be easily verified. Particularly, when $n = 128$ and $\sigma = \varphi = 64$, each universal forgery attack that includes the phase of recovering L requires about 2^{65} encryption queries, a memory of about 2^{68} bytes, and has a time complexity of 2^{64} memory accesses and a success probability of about 63%. (Here, typically as suggested in [9, 20], encrypting chosen messages is associated with the data complexity of an attack and is not counted as part of the time complexity of the attack. The same statement applies to subsequent attacks, although we do not make any further explicit statements. However, if one would treat the time complexity for encrypting chosen messages as part of the time complexity of the attack, the resulting time complexity would be about $2^{65} \times 5 \approx 2^{67.4}$ block cipher encryptions.)

3.1.4 An Application to AES-COPA in the Nonce-Respecting Scenario

Different from COPA, AES-COPA (v1/2) has an additional (public) input parameter called nonce, which has a constant length of 128 bits. It is appended to associated data (if any), and then the resulting value is treated as associated data in COPA. As a consequence, when applying the state recovery attack of Section 3.1.2 to AES-COPA, we should obtain associated data satisfying Steps 1 and 2; this can be easily done, for example:

- In Step 1, we choose (associated data of one 128-bit block long, nonce of one 128-bit block long) pairs $(AD, N^{(i)})$, where $N^{(i)} = AD_1^{(i)}$, (and $AD_1^{(i)}$ is from Section 3.1.2);

- In Step 2, we choose the (associated data of less than 128 bits long, nonce of one 128-bit block long) pairs such that the padded (associated data, nonce) pairs are $(AD, X^{(j)})$, where $X^{(j)} = \widehat{AD}_1^{(j)}$, (and $\widehat{AD}_1^{(j)}$ is from Section 3.1.2);

- For instance, a value of AD can be $(1, \cdots, 1, 0)$ in binary form, which can guarantee that the nonces in Step 2 before padding are different (i.e., the rightmost 128 bits after removing the padded one and zero (if any) bits from the right-hand side of $(AD, X^{(j)}) = ((1, \cdots, 1, 0)||(j \times 2^{\frac{n}{2}}))$, and the leftmost remaining bits are chosen associated data).

Then, the first blocks for all the $(2^\sigma + 2^\varphi - 1)$ (padded) (associated data, nonce) pairs are identical, and the first block cipher encryption operations produce the same output, and we only need to modify the above state recovery attack slightly. As a result, the nonces used are different one another, and the state recovery attack works in the nonce-respecting scenario. Of course, it can also work in the nonce-misuse scenario.

For AES-COPA (v1/2), when we set $\sigma = \varphi$ to be slightly smaller than 62 extremely, the attack requires slightly less than 2^{63} queries with the total (associated data, message) pairs having a length slightly less than 2^{64} blocks (which is the maximum number of data blocks that AES-COPA can process with a single key), and a memory of about $2^{62} \times 16 = 2^{66}$ bytes, and has a time complexity of about 2^{62} memory accesses and a success probability of about 6%. (For a longer (associated data, nonce, message) triple, we need to reduce the values of σ and φ accordingly.)

Because of the constraint on the maximum number of data blocks that can be processed with a single key, the success probability 6% is not very high, but it is not negligible even in reality and it still represents a semi-practical security concern, considering particularly that COPA was proposed for GPGPU-like parallel architectures.

3.2 (Almost) Universal Forgery Attack on the Basic Case of COPA under Constant Associated Data

There are real situations that only allow for constant associated data, for example, sending some files with the same public header, where the header is used as associated data. Thus, the above attacks are not applicable in such situations.

In this subsection, we show how to recover the secret parameter L in the basic case of COPA under constant associated data in a more advantageous way than exhaustive key search, then describe a way to make an almost universal forgery after L is recovered, and finally brief its application to AES-COPA (v1/2) in the nonce-misuse scenario. We start with our attack idea.

3.2.1 Attack Idea

The attack also consists of two phases: recovering the secret parameter L, followed by a forgery if L is recovered. Note that associated data is fixed here.

Different from the idea used in Section 3.1, a novel idea is used here to recover L. First, we choose a number of two-block messages, and then from these messages we select a certain small number of messages whose second ciphertext blocks are identical, like finding a multi-collision [15, 23] in hash function cryptanalysis. Next, we choose a number of three-block messages with the first two blocks fixed to one of the messages with the second ciphertext blocks being identical, which means an identical internal state S immediately after the second block. At last, we recover L by looking for a general collision between the process of the third blocks of the three-block messages and the tag generation process of the two-block messages with the second ciphertext blocks being identical; this general collision is different in nature from the general one used in Section 3.1.

To make a forgery on a message, we query with the message obtained by modifying the target message so that the pair of messages make a general collision similar to the one in the phase of recovering L. Note that here we cannot use the forgery ways based on modifying associated data and modifying associated data and message, since associated data is constant.

3.2.2 Recovering the Secret Parameter L

The procedure for recovering the secret parameter L is as follows, which is illustrated in Fig. 4. Since the same associated data is used, we will omit it in the attack description.

1. Choose uniformly at random 2^θ messages $M^{(i)} = (M_1^{(i)}, M_2^{(i)})$ of two n-bit blocks long (a specific value of θ will be given below, and $i = 1, 2, \cdots, 2^\theta$). Query the COPA encryption and tag generation oracle, and obtain all the ciphertexts and tags for the 2^θ messages; we denote by $C^{(i)} = (C_1^{(i)}, C_2^{(i)})$ and $T^{(i)}$ the ciphertext and tag for message $M^{(i)}$, respectively.

2. Select a tuple of δ messages $(M^{(i_1)}, M^{(i_2)}, \cdots, M^{(i_\delta)})$ such that
$$C_2^{(i_1)} = C_2^{(i_2)} = \cdots = C_2^{(i_\delta)}. \quad (1)$$
(A specific value of δ will be given below.) This can be done efficiently by storing $(M^{(i)}, C^{(i)}, T^{(i)})$ into a table indexed by $C_2^{(i)}$. Go to Step 1 if there does not exist such a δ-tuple.

3. Choose two n-bit constants α and β such that
$$\alpha * (2 * 3^2 * L \oplus 2^2 * 3 * L) = \beta * (2^3 * L \oplus 2 * 7 * L). \quad (2)$$
Observe that the secret parameter L is not required when solving Eq. (2) for α and β, because it cancels out.

4. Choose uniformly at random 2^ϕ messages $\widehat{M}^{(j)} = (\widehat{M}_1^{(j)}, \widehat{M}_2^{(j)}, \widehat{M}_3^{(j)})$ of three n-bit blocks long (a specific value of ϕ will be given below, and $j = 1, 2, \cdots, 2^\phi$), such that $\widehat{M}_l^{(j)} = M_l^{(i_1)}$ for $1 \le l \le 2$; that is, $\widehat{M}^{(j)} = (M_1^{(i_1)}, M_2^{(i_1)}, \widehat{M}_3^{(j)})$. Query the COPA encryption and tag generation oracle, and obtain all the ciphertexts and tags for the 2^ϕ messages; we denote by $\widehat{C}^{(j)} = (\widehat{C}_1^{(j)}, \widehat{C}_2^{(j)}, \widehat{C}_3^{(j)})$ and $\widehat{T}^{(j)}$ the ciphertext and tag for message $\widehat{M}^{(j)}$, respectively.[1] Since the same user key and constant associated data are used, clearly $\widehat{C}_l^{(j)} = C_l^{(i_1)}$ for $1 \le l \le 2$; i.e., $\widehat{C}^{(j)} = (C_1^{(i_1)}, C_2^{(i_1)}, \widehat{C}_3^{(j)})$.

5. Select the message-ciphertext pair $(\widehat{M}^{(j)}, \widehat{C}^{(j)})$ such that the following two equations hold for some t, here $1 \le t \le \delta$:
$$\widehat{M}_3^{(j)} \oplus 2^2 * 3 * L = \bigoplus_{l=1}^{2} M_l^{(i_t)} \oplus 2 * 3^2 * L; \quad (3)$$
$$\widehat{C}_3^{(j)} \oplus 2^3 * L = T^{(i_t)} \oplus 2 * 7 * L. \quad (4)$$

[1] The tags for the 2^ϕ chosen messages are not required in this attack.

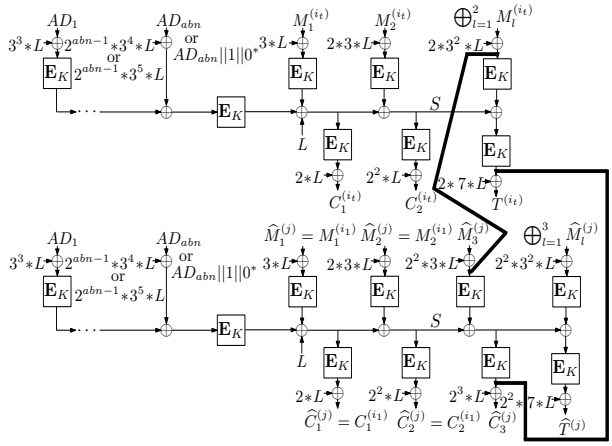

Figure 4: State recovery attack on COPA under constant associated data

This can be partially done efficiently by checking whether

$$\alpha * \widehat{M}_3^{(j)} \oplus \beta * \widehat{C}_3^{(j)} = \alpha * \bigoplus_{l=1}^{2} M_l^{(i_t)} \oplus \beta * T^{(i_t)}; \quad (5)$$

we denote the qualified message-ciphertext pair(s) by $(\widehat{M}^{(\omega)}, \widehat{C}^{(\omega)})$ (if any), where $1 \le \omega \le 2^{\phi}$.

6. Recover L from Eq. (3) with respect to $\widehat{M}^{(\omega)}$, that is $\widehat{M}_3^{(\omega)} \oplus 2^2 * 3 * L = \bigoplus_{l=1}^{2} M_l^{(i_t)} \oplus 2 * 3^2 * L$, and output the recovered L.

Step 1 requires a memory of about $5n \cdot 2^{\theta}$ bits, which can be reduced to $3n \cdot 2^{\theta}$ bits by storing only $(\bigoplus_{l=1}^{2} M_l^{(i)}, C_2^{(i)}, T^{(i)})$. By a mathematical analysis (namely, Eq. 7.5) on the coincidence theory from [8], the probability p that given 2^{θ} randomly chosen messages there is at least one δ-tuple satisfying Eq. (1) is approximately given by the equation

$$\begin{aligned} & 2^{\theta} \times \mathsf{e}^{-\frac{2^{\theta}}{\delta \cdot 2^{n}}} \times (1 - \frac{2^{\theta}}{(\delta+1) \cdot 2^{n}})^{-\frac{1}{\delta}} \\ = \; & [2^{(\delta-1)\cdot n} \times \delta! \times \log_{\mathsf{e}}^{\frac{1}{1-p}}]^{\frac{1}{\delta}}. \end{aligned}$$

Thus, we have $p = 1 - \mathsf{e}^{\frac{-2^{\theta} \cdot \delta \times \mathsf{e} - 2^{\theta} - n}{(1 - \frac{2^{\theta} - n}{\delta+1}) \times 2^{(\delta-1)\cdot n} \times \delta!}}$, which is approximately equal to $1 - \mathsf{e}^{-\binom{2^{\theta}}{\delta} \cdot 2^{-n(\delta-1)}}$ for $\theta \ll n$ and a small δ. Eq. (1) guarantees that messages $M^{(i_1)}, M^{(i_2)}, \cdots, M^{(i_\delta)}$ have the same internal state S immediately before the tag generation phase.

Observe that for the correct value of L, Eq. (4) holds once Eq. (3) holds, and vice versa. If both Eqs. (3) and (4) hold, then Eq. (5) always holds, because from Eqs. (3) and (4) we have

$$\begin{aligned} & \widehat{M}_3^{(j)} \oplus \bigoplus_{l=1}^{2} M_l^{(i_t)} = 2 * 3^2 * L \oplus 2^2 * 3 * L; \\ & \widehat{C}_3^{(j)} \oplus T^{(i_t)} = 2^3 * L \oplus 2 * 7 * L. \end{aligned}$$

Then, we can obtain Eq. (5) after applying α and β to the above two equations and XORing the resulting two equations.

Note that once we obtain the ciphertext-tag pair for a message in Step 4, we can discard it if it does not meet Eq. (5), and thus we only need to store the qualified message-ciphertext-tag tuples in Step 4. Particularly, if we choose $\alpha = 1$ or $\beta = 1$, then Eq. (5) can be checked with one $*$ operation and one $\oplus$ operation (which is generally negligible compared with one $*$ operation) for a message-ciphertext pair, since the right-hand side of Eq. (5) is one-off.

For a random message-ciphertext pair $(\widehat{M}^{(j)}, \widehat{C}^{(j)})$, it is expected that Eq. (5) holds for a given i_t with a probability of $2^{-n} \times 1 + (1 - 2^{-n}) \times 2^{-n} \approx 2^{1-n}$, assuming that Eq. (5) holds uniformly at random when at least one of Eqs. (3) and (4) does not hold. On the other hand, for a given i_t the (conditional) probability that both Eqs. (3) and (4) hold when Eq. (5) holds is

$$\begin{aligned} & \text{Pr.(Both Eqs. (3) and (4) hold when Eq. (5) holds)} \\ = \; & \frac{\text{Pr.(Eq. (5) holds when Eqs. (3) and (4) hold)}}{\text{Pr.(Eq. (5) holds)}} \\ & \times \text{Pr.(Eqs. (3) and (4) hold)} \\ = \; & \frac{1}{2^{1-n}} \times 2^{-n} \\ = \; & \frac{1}{2}. \end{aligned}$$

Since there are 2^{ϕ} message-ciphertext pairs $(\widehat{M}^{(j)}, \widehat{C}^{(j)})$, the expected number of qualified message-ciphertext pairs satisfying Eq. (5) for an i_t is approximately $2^{\phi} \times 2^{1-n} \times \delta = \delta \cdot 2^{\phi-n+1}$. The probability that there is at least one message-ciphertext pair satisfying Eq. (5) for an i_t is approximately $1 - (1 - \delta \cdot 2^{1-n})^{2^{\phi}} \approx 1 - \mathsf{e}^{-\delta \cdot 2^{\phi-n+1}}$, and the probability that the recovered L is correct is $\frac{1}{2} \cdot (1 - \mathsf{e}^{-\delta \cdot 2^{\phi-n+1}})$.

Therefore, the state recovery attack requires $2^{\theta} + 2^{\phi}$ encryption queries (the tags for the 2^{ϕ} chosen messages are not required) and a memory of approximately $3n \cdot 2^{\theta}$ bits, and has a computational complexity of about 2^{ϕ} simple $*$ operations, with a success probability of approximately $\frac{1}{2} \cdot (1 - \mathsf{e}^{-\binom{2^{\theta}}{\delta} \cdot 2^{-n(\delta-1)}}) \cdot (1 - \mathsf{e}^{-\delta \cdot 2^{\phi-n+1}})$. (If one would treat the time complexity for encrypting chosen messages as part of the time complexity of the attack, the resulting time complexity would be about $\lambda \cdot (2^{\theta} + 2^{\phi}) + 2^{\phi+1}$ block cipher encryptions, (2^{ϕ} simple $*$ operations are negligible compared with the block cipher encryptions), where λ is the number of block cipher encryptions for one of the 2^{θ} messages.)

3.2.3 Making an (Almost) Universal Forgery

Once the correct n-bit secret parameter L is recovered by the above state recovery attack, we can make a universal forgery attack on the COPA with a single query at a 100% success probability, by modifying message as in Section 3.1.3. Its illustration is similar to Fig. 4.

In summary, the universal forgery attack that includes the phase of recovering L requires approximately $2^{\theta} + 2^{\phi}$ encryption queries (the tags for 2^{ϕ} chosen messages are not required actually) and a memory of approximately $3n \cdot 2^{\theta}$ bits, and has a computational complexity of about 2^{ϕ} simple $*$ operations, with a success probability of approximately $\frac{1}{2} \cdot (1 - \mathsf{e}^{-\binom{2^{\theta}}{\delta} \cdot 2^{-n(\delta-1)}}) \cdot (1 - \mathsf{e}^{-\delta \cdot 2^{\phi-n+1}})$. (Note that if one would treat the time complexity for encrypting chosen messages as part of the time complexity of the attack, the resulting time complexity would be about $\lambda \cdot (2^{\theta} + 2^{\phi}) + 2^{\phi+1}$ block cipher encryptions, where λ is the number of

block cipher encryptions for one of the 2^θ messages.) The success probability is a bit complex, anyway, we can make an attack faster than exhaustive key search if we choose the parameters θ, δ and ϕ appropriately, as applied to AES-COPA next, which holds for COPA.

3.2.4 An Application to AES-COPA in the Nonce-Misuse Scenario

We have $n(=k) = 128$ for AES-COPA (v1/2). By setting $\theta = 115, \delta = 8$ and $\phi = 124$, the above attack requires about 2^{124} encryption queries in the nonce-misuse scenario and a memory of about $2^{120.6}$ bytes, and has a time complexity of about 2^{124} simple $*$ operations, with a success probability of about 32%. This attack is mainly of academic interest, since its data complexity is far beyond the birthday bound constraint.

3.3 More Efficient (Almost) Universal Forgery Attack on COPA and AES-COPA under Constant Associated Data

An anonymous reviewer mentioned a more efficient almost universal forgery attack on COPA and AES-COPA under constant associated data, which works as follows: (1) Choose uniformly at random 2^{64} messages $M^{(i)} = (M_1^{(i)}, M_2^{(i)} = M_1^{(i)})$ of two 128-bit blocks long ($i = 1, 2, \cdots, 2^{64}$); (2) Filter out message pairs such that $C_2^{(i_1)} = C_2^{(i_2)}$, where $1 \le i_1 \ne i_2 \le 2^{64}$; and (3) For a qualified message pair, $M_1^{(i_1)} \oplus 3*L = M_2^{(i_2)} \oplus 2*3*L$ holds with probability 50% similarly. Next, L can be recovered, and a forgery can be made.

3.4 Notes

The attack of Section 3.2 aims for the basic case of COPA that processes messages of a multiple of the block size long under constant associated data. If the case of COPA that processes messages with the last block being not full is considered, or if associated data is not constant, there is a more efficient attack with an idea similar to that described in Section 3.1.

The attacks of Section 3.1.3 does not work for an associated data with the number of blocks being equal to or one smaller than the preset maximum number, or for a message with the preset maximum number of blocks; and the attack of Section 3.2.3 does not work for a message with the preset maximum number of blocks. Thus, the attacks are almost universal forgery attacks [9].

4. (ALMOST) UNIVERSAL FORGERY ATTACKS ON THE BASIC CASE OF MARBLE UNDER VARIABLE ASSOCIATED DATA

In January 2015, Fuhr et al. [10] released an (almost) universal forgery attack on Marble v1.1, then the Marble designer made another revision, namely Marble v1.2, and shortly later we showed that Marble v1.2 still suffered from (almost) universal forgery attacks based on Fuhr et al.'s (almost) universal forgery attack on Marble v1.1. Finally, the Marble designer withdrew Marble from the CAESAR competition in January 2015, due to Fuhr et al.'s and our attacks.

Fuhr et al. extended their attack on Marble v1.1 described in [10] to Marble v1.2 in the final publication version [11] of their work, and they acknowledged our attacks by writing 'as shown independently by ourselves and Lu'. Our attack and Fuhr et al.'s attack on Marble v1.2 consist of two phases: recovering the secret parameter L, followed by a forgery if L is recovered. Since our attack and Fuhr et al.'s attack on Marble v1.2 are similar and Fuhr et al.'s attack has been published, here we do not focus on our attack on Marble v1.2, but nevertheless we give how to recover L of Marble v1.2, for the reader to have an understanding on it, and then focus on a different forgery way on Marble v1.0/1.1.

4.1 A State Recovery Attack for the Secret Parameter L in Marble v1.2

The idea of the attack is as follows, which is illustrated in Fig. 5.

1. Choose $(2^{64} - 1)$ ((padded) associated data of two blocks long, message of one block long) pairs $(AD_1^{(i)}, AD_2^{(i)}, M_1^{(i)}) = ((3^2 \oplus 3^3) * i, (2 * 3^3 \oplus 2) * i, (2 \oplus 2^2) * i)$, and obtain their ciphertexts (and tags), where $i = 1, 2, \cdots, 2^{64} - 1$; we denote by $C_1^{(i)}$ the ciphertext for message $M_1^{(i)}$. Store $C_1^{(i)}$ into a table indexed by $C_1^{(i)} \oplus (3 \oplus 2 * 3) * i$ (i.e. $C_1^{(i)} \oplus 5 * i$).

2. Choose $(2^{64}-1)$ ((padded) associated data of one block long, message of two blocks long) pairs $(\widehat{AD}_1^{(j)}, \widehat{M}_1^{(j)}, \widehat{M}_2^{(j)}) = ((3^2 \oplus 3^3) * (j \times 2^{64}), (2 * 3^3 \oplus 2) * (j \times 2^{64}), (2 \oplus 2^2) * (j \times 2^{64}))$, and obtain their ciphertexts (and tags), where $j = 1, 2, \cdots, 2^{64} - 1$; we denote by $(\widehat{C}_1^{(j)}, \widehat{C}_2^{(j)})$ the ciphertext for message $(\widehat{M}_1^{(j)}, \widehat{M}_2^{(j)})$.

3. Check whether $\widehat{C}_2^{(j)} \oplus (3 \oplus 2 * 3) * (j \times 2^{64})$ (i.e. $\widehat{C}_2^{(j)} \oplus 5 * (j \times 2^{64})$) matches one of the set $\{C_1^{(i)} \oplus 5 * i | i = 1, 2, \cdots, 2^{64} - 1\}$ for $j = 1, 2, \cdots, 2^{64} - 1$. We denote the match(es) by $(\widehat{C}_2^{(\omega)} \oplus 5 * (\omega \times 2^{64}), C_1^{(\mu)} \oplus 5 * \mu)$ if any, that is, $\widehat{C}_2^{(\omega)} \oplus 5 * (\omega \times 2^{64}) = C_1^{(\mu)} \oplus 5 * \mu$.

4. Recover L from $\widehat{C}_2^{(\omega)} \oplus C_1^{(\mu)} = 5 * (\omega \times 2^{64}) \oplus 5 * \mu = 5 * L$.

For $(AD_1^{(i)}, AD_2^{(i)}, M_1^{(i)})$, the immediate inputs to the three $\mathbf{E_1}$ operations are $(3^2 \oplus 3^3) * i \oplus 3^2 * L, (2 * 3^3 \oplus 2) * i \oplus 2 * 3^3 * L, (2 \oplus 2^2) * i \oplus 2 * L$, respectively; for $(\widehat{AD}_1^{(j)}, \widehat{M}_1^{(j)}, \widehat{M}_2^{(j)})$, the immediate inputs to the three $\mathbf{E_1}$ operations are $(3^2 \oplus 3^3) * (j \times 2^{64}) \oplus 3^3 * L, (2 * 3^3 \oplus 2) * (j \times 2^{64}) \oplus 2 * L, (2 \oplus 2^2) * (j \times 2^{64}) \oplus 2^2 * L$, respectively. Thus, the input difference to the three $\mathbf{E_1}$ operations under $(AD_1^{(i)}, AD_2^{(i)}, M_1^{(i)})$ and $(\widehat{AD}_1^{(j)}, \widehat{M}_1^{(j)}, \widehat{M}_2^{(j)})$ are $(3^2 \oplus 3^3) * [i \oplus (j \times 2^{64}) \oplus L], (2 * 3^3 \oplus 2) * [i \oplus (j \times 2^{64}) \oplus L], (2 \oplus 2^2) * [i \oplus (j \times 2^{64}) \oplus L]$, respectively. Now, if $i \oplus (j \times 2^{64}) = L$, then the input difference to the corresponding three $\mathbf{E_1}$ operations will be zero, and $\widehat{C}_2^{(j)} \oplus 2 * 3 * L = C_1^{(i)} \oplus 3 * L$, which is equivalent to $\widehat{C}_2^{(j)} \oplus (3 \oplus 2 * 3) * (j \times 2^{64}) = C_1^{(i)} \oplus (3 \oplus 2 * 3) * i$.

The state recovery attack requires about 2^{65} encryption queries and a memory of about $2^{64} \times 16 = 2^{68}$ bytes, and has a time complexity of about 2^{65} memory accesses and a success probability of about $\frac{1}{2} \times [1 - \binom{2^{128}}{0} \cdot (2^{-128})^0 \cdot (1 - 2^{-128})^{2^{128}}] \approx 32\%$, where $\frac{1}{2}$ has a similar meaning to that explained in Section 3.2.2.

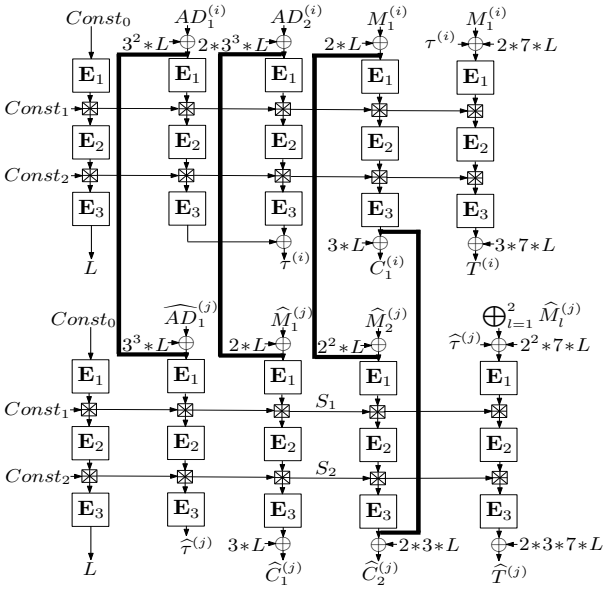

Figure 5: State recovery attack on Marble v1.2 under variable associated data

4.2 Another (Almost) Universal Forgery Attack on Marble

Below we only focus on a different way to make an (almost) universal forgery on Marble v1.0/1.1 after L is recovered by a state recovery attack similar to that described above or in [10, 11]. Fuhr et al. made an (almost) universal forgery by modifying associated data [10, 11], however, we find that there is another way to make an (almost) universal forgery on Marble v1.0/1.1, which is based on modifying message. Different from COPA, Marble uses the additional secret parameter τ in the tag generation phase. As a consequence, this different forgery way targets Marble v1.0/1.1 without associated data, because $\tau = 0$ when there is no associated data in Marble v1.0/1.1. For a message $M = (M_1, M_2, \cdots, M_{mbn})$ of mbn 128-bit message blocks long ($mbn \geq 1$), below is the different forgery way on Marble v1.0/1.1 without associated data.

1. Query the Marble encryption and tag generation oracle with the $(mbn+1)$-block message $\widehat{M} = (M_1, M_2, \cdots, M_{mbn}, 2^{mbn+1}*L \oplus 2^{mbn}*7*L \oplus \bigoplus_{i=1}^{mbn} M_i)$, and obtain its ciphertext $\widehat{C} = (\widehat{C}_1, \widehat{C}_2, \cdots, \widehat{C}_{mbn}, \widehat{C}_{mbn+1})$.

2. The ciphertext for M is $C = (\widehat{C}_1, \widehat{C}_2, \cdots, \widehat{C}_{mbn})$, and the tag for M is $\widehat{C}_{mbn+1} \oplus 2^{mbn}*3*L \oplus 2^{mbn-1}*3*7*L$.

This universal forgery attack including the phase of recovering L requires about 2^{65} encryption queries and a memory of about 2^{68} bytes, and has a time complexity of about 2^{65} memory accesses and a success probability of about 32%. (Note that if one would treat the time complexity of encrypting chosen messages as part of the time complexity of the attack, the resulting time complexity would be about $2^{65} \times 5 \approx 2^{67.4}$ AES encryptions.)

Note that the attack does not work for a message with the preset maximum number of blocks, and is an almost universal attack. This forgery way does not apply to Marble v1.2, since Marble v1.2 will process a padded block of associated data even there is no associated data, which produces an unknown τ. Anyway, the forgery way based on modifying associated data works for Marble v1.2.

5. CONCLUSIONS

In this paper, we have presented almost universal forgery attacks on the basic cases of COPA, AES-COPA and Marble that process messages of a multiple of the block size long, and have shown that the basic cases of COPA, AES-COPA and Marble only have roughly at most a birthday-bound security against universal forgery, particularly for AES-COPA in the nonce-respecting scenario, which may be an undesirable property for AES-COPA, considering that it is proposed for GPGPU-like parallel architectures. Therefore, the full security claim and conjecture on tag guessing of AES-COPA and the full security claim on authenticity of Marble are incorrectly far overestimated in the sense of a general understanding of full security of these security notions.

6. ACKNOWLEDGMENTS

The author is grateful to Hongjun Wu for his conversations on forgery-resistance, to Jian Guo and Kan Yasuda for their discussions on some attacks, to several anonymous referees for their comments on earlier versions of this paper, and to Prof. Yongzhuang Wei and the Natural Science Foundation of China (No. 61572148) for their support.

7. REFERENCES

[1] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, and K. Yasuda. AES-COPA v1. Submission to the CAESAR competition, March 2014. `http://competitions.cr.yp.to/round1/aescopav1.pdf`

[2] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, and K. Yasuda. AES-COPA v2. Submission to the CAESAR competition, September 2015. `http://competitions.cr.yp.to/round2/aescopav2.pdf`

[3] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, and K. Yasuda. Parallelizable and authenticated online ciphers. In K. Sako and P. Sarkar, editors, ASIACRYPT 2013, pages 424–443. Springer, 2013.

[4] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, and K. Yasuda. Parallelizable and authenticated online ciphers. *IACR Cryptology ePrint Archive*, Report 2013/790, 2013. `http://eprint.iacr.org/2013/790`

[5] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, K. Yasuda, N. Datta, and M. Nandi. COLM v1. Submission to the CAESAR competition, 2016. `http://competitions.cr.yp.to/round2/colm.pdf`

[6] M. Bellare and C. Namprempre. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, 2008.

[7] CAESAR — Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html

[8] P. Diaconls and F. Mosteller. Methods for studying coincidences. *Journal of the American Statistical Association*, 84(408):853–861, 1989.

[9] O. Dunkelman, N. Keller, and A. Shamir. Almost universal forgery attacks on AES-based MAC's. *Designs, Codes and Cryptography*, 76(3):431–449, 2015.

[10] T. Fuhr, G. Leurent, and V. Suder. Forgery and key-recovery attacks on CAESAR candidate marble. HAL archive hal-01102031, 13 January 2015. http://hal.inria.fr/hal-01102031v2.

[11] T. Fuhr, G. Leurent, and V. Suder. Collision attacks against CAESAR candidates: forgery and key-recovery against AEZ and Marble. In T. Iwata and J.H. Cheon, editors, ASIACRYPT 2015, pages 510–532. Springer, 2015.

[12] J. Guo. Marble Specification Version 1.0. Submission to the CAESAR competition, 15 March 2014. http://competitions.cr.yp.to/round1/marblev10.pdf

[13] J. Guo. Marble Specification Version 1.1. Submission to the CAESAR competition, 26 March 2014. http://competitions.cr.yp.to/round1/marblev11.pdf

[14] J. Guo. Marble Specification Version 1.2. Submission to the CAESAR competition, 16 January 2015. https://groups.google.com/forum/#!topic/crypto-competitions/FoJITsVbBdM

[15] A. Joux. Multicollisions in iterated hash functions. Application to cascaded constructions. In M. Franklin, editor, CRYPTO 2004, pages 306–316. Springer, 2004.

[16] J. Lu. Attacking the Marble authenticated encryption algorithm. CAESAR forum, 23 January 2015. https://groups.google.com/forum/#!topic/crypto-competitions/dBOAt64POqI

[17] J. Lu. On the security claim of tag guessing of the AES-COPA authenticated encryption algorithm. CAESAR forum, 30 January 2015. https://groups.google.com/forum/#!topic/crypto-competitions/yUGgP-VIS_s

[18] J. Lu. On the security of the COPA and Marble authenticated encryption algorithms against (almost) universal forgery attack, *IACR Cryptology ePrint Archive*, Report 2015/079, 2015. http://eprint.iacr.org/2015/079

[19] J. Lu. On the security of the LAC authenticated encryption algorithm. In J.K. Liu and R. Steinfeld, editors, ACISP 2016, pages 395–408. Springer, 2016.

[20] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[21] M. Nandi. XLS is not a strong pseudorandom permutation. In P. Sarkar and T. Iwata, editors, ASIACRYPT 2014, pages 478–490. Springer, 2014.

[22] M. Nandi. Revisiting security claims of XLS and COPA. *IACR Cryptology ePrint Archive*, Report 2015/444, 2015. http://eprint.iacr.org/2015/444

[23] M. Nandi and D.R. Stinson. Multicollision attacks on some generalized sequential hash functions. *IEEE Transcations on Information Theory*, 53(2):759–767, 2007.

[24] National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES), FIPS-197, 2001.

[25] B. Preneel and P.C. van Oorschot. On the security of iterated message authentication codes. *IEEE Transcations on Information Theory*, 45(1):188–199, 1999.

[26] T. Ristenpart and P. Rogaway. How to enrich the message space of a cipher. In A. Biryukov, editor, FSE 2007, pp. 101–118. Springer, 2007.

[27] J. Zhang, W. Wu, and Y. Zheng. Collision attacks on CAESAR second-round candidate: ELmD. In F. Bao, L. Chen, R.H. Deng, and G. Wang, editors, ISPEC 2016, pp. 122–136. Springer, 2016.