# Reforgeability of Authenticated Encryption Schemes

Christian Forler[1], Eik List[2], Stefan Lucks[2], and Jakob Wenzel[2]

[1] Beuth Hochschule für Technik Berlin, Germany
cforler(at)beuth-hochschule.de
[2] Bauhaus-Universität Weimar, Germany
<first name>.<last name>(at)uni-weimar.de

**Abstract.** This work pursues the idea of multi-forgery attacks as introduced by Ferguson in 2002. We recoin reforgeability for the complexity of obtaining further forgeries once a first forgery has succeeded. First, we introduce a security notion for the integrity (in terms of reforgeability) of authenticated encryption schemes: $j$-INT-CTXT, which is derived from the notion INT-CTXT. Second, we define an attack scenario called $j$-IV-Collision Attack ($j$-IV-CA), wherein an adversary tries to construct $j$ forgeries provided a first forgery. The term *collision* in the name stems from the fact that we assume the first forgery to be the result from an internal collision within the processing of the associated data and/or the nonce. Next, we analyze the resistance to $j$-IV-CAs of classical nonce-based AE schemes (CCM, CWC, EAX, GCM) as well as all 3rd-round candidates of the CAESAR competition. The analysis is done in the nonce-respecting and the nonce-ignoring setting. We find that none of the considered AE schemes provides full built-in resistance to $j$-IV-CAs. Based on this insight, we briefly discuss two alternative design strategies to resist $j$-IV-CAs.

**Keywords:** authenticated encryption, CAESAR, multi-forgery attack, reforgeability.

## 1 Introduction

**(Nonce-Based) Authenticated Encryption.** Simultaneously protecting authenticity and privacy of messages is the goal of authenticated encryption (AE) schemes. AE schemes with support for Associated Data (AEAD) provide additional authentication for associated data. The standard security requirement for AE schemes is to prevent leakage of any information about secured messages except for their respective lengths. However, stateless encryption schemes would enable adversaries to detect whether the same associated data and message has been encrypted before under the current key. Thus, Rogaway proposed nonce-based encryption [43], where the user must provide an additional nonce for every message it wants to process – a number used once (nonce). AE schemes that require a nonce input are called nonce-based authenticated encryption (nAE) schemes.

**Reforgeability.** In the cryptographic sense, *reforgeability* refers to the complexity of finding subsequent forgeries once a first forgery has been found. Thus, it defines the hardness of forging a ciphertext after the first forgery succeeded. The first attack known was introduced in 2002 by Ferguson by showing collision attacks on OCB [44] and a Ctr-CBC-like MAC [17]. He showed that finding a collision within the message processing of OCB "*leads to complete loss of an essential function*" (referring to the loss of authenticity/integrity).
Later on, in 2005, the term *multiple forgery attacks* was formed and defined by McGrew and Fluhrer [34]. They introduced the measure of expected number of forgeries and conducted a thorough analysis of GCM [33], HMAC [6], and CBC-MAC [8]. In 2008, Handschuh and Preneel [21] introduced key recovery and universal forgery attacks against several MAC algorithms. The term

*Reforgeability* was first formally defined by Black and Cochran in 2009, where they examined common MACs regarding their security to this new measurement [13]. Further, they introduced WMAC, which they argue to be the "*best fit for resource-limited devices*".

**Relevance.** For a reforgeability attack to work, an adversary must be provided with a verification oracle in addition to its authentication (and encryption) oracle. In practice, such a setting can, for example, be found when a client tries to authenticate itself to a server and has multiple tries to log in to a system. Thus, the server would be the verification oracle for the client.
Obviously, the same argument holds for the case when the data to be send is of sensitive nature, i.e., the data itself has to be encrypted. Thus, besides the resistance of MACs to reforgeability, also the resistance of AE schemes is of high practical relevance.
Since modern and cryptographically secure AE schemes should provide at least INT-CTXT security in terms of integrity, the first forgery is usually not trivially found and depends on the size of the tag or the internal state. For that reason, reforgeability becomes especially essential when considering resource-constrained devices limited by, e.g., radio power, bandwidth, area, or throughput. This is not uncommon in the area of low-end applications such as sensor networks, VoIP, streaming interfaces, or, for example, devices connected to the Internet of Things (IoT). In these domains, the tag size $\tau$ of MACs and AE schemes is usually quite small, e.g., $\tau = 64$ or $\tau = 32$ bits, or even smaller ($\tau = 8$ bits) as mentioned by Ferguson in regard to voice systems [18]. Therefore, even if the AE scheme is secure in the INT-CTXT setting up to $\tau$ bits, it is not unreasonable for an adversary to find a forgery for such a scheme in general. Nevertheless, even if finding the first forgery requires a large amount of work, a rising question is, whether it can be exploited to find more forgeries with significantly less than $2^\tau$ queries to an authentication oracle per forgery. For our analysis, we derive a new security notion $j$-INT-CTXT, which states that an adversary who finds the first forgery using $t_1$ queries, can generate $j$ additional forgeries in polynomial time depending on $j$. In general, the best case would be to find $j$ additional forgeries using $t_1 + j$ queries. Nevertheless, for five schemes (AES-OTR [36], GCM [33], COLM [3], CWC [28], and OCB [29]), there already exist forgery attacks in the literature (mentioned in Section 4.3 and Appendix C) leading to $j$ forgeries using only $t_1$ queries (thus, the $j$ additional authentication queries are not even required).

Due to the vast number of submissions to the CAESAR competition [10], cryptanalysis proceeds slowly for each individual scheme. For instance, forgery attacks on 3rd-round CAESAR candidates have only been published for AES-COPA [4,31,38], which even might become obsolete since AES-COPA and ELmD [14] have been merged to COLM [3]. Besides looking at 3rd-round CAESAR candidates, we also analyze other existing and partially widely-used AE schemes, e.g., GCM, EAX [9], CCM [16], and CWC. Naturally, due to their longer existence, there exist a lot more cryptanalysis on those schemes in comparison to the CAESAR candidates (see [19,26,27,35,41,45] for some examples. The hope is that an INT-CTXT-secure AE scheme does not lose its security when considering reforgeability, i.e., $j$-INT-CTXT.
We briefly introduce what we mean by *resistant to* $j$-IV-CAs, whereby we assume the first forgery to be the results from an internal collision of the processing of the associated data and/or the nonce.

- **Nonce-Ignoring:** We call an nAE scheme resistant to $j$-IV-CAs if the required number of queries of a *nonce-ignoring* $j$-IV-CA adversary for finding $1 + j$ forgeries (including the first) is greater than $t_1 + j$, where $t_1$ denotes the number of queries for finding the first forgery.
- **Nonce-Respecting:** We call an nAE scheme resistant to $j$-IV-CAs if the required number of queries of a *nonce-respecting* $j$-IV-CA adversary for finding $1 + j$ forgeries (including the first) is greater than $t_1 \cdot j/2$, where $t_1$ denotes the number of queries for finding the first forgery.

Further, we say that an nAE scheme is *semi-resistant* to $j$-IV-CAs if the internal state is of wide size and the scheme itself is not trivially insecure in terms of $j$-IV-CA. Thereby, following a similar

| Scheme | NI | NR | Scheme | NI | NR |
|---|---|---|---|---|---|
| 3rd-Round CAESAR Candidates | | | | | |
| ACORN [48] | $t_1 + j$ | $t_1 \cdot j/2$ | KETJE [12] | $t_2 + j$ | $t_2 \cdot j/2$ |
| AEGIS [51] | $t_2 + j$ | $t_2 \cdot j/2$ | KEYAK [20] | $t_2 + j$ | $t_2 \cdot j/2$ |
| AES-OTR [36] | $t_1$ | $t_1$ | MORUS [49] | $t_2 + j$ | $t_2 \cdot j/2$ |
| AEZv4 [22] | $t_1 + j$ | $t_1 \cdot j/2$ | NORX [5] | $t_2 + j$ | $t_2 \cdot j/2$ |
| ASCON [15] | $t_2 + j$ | $t_2 \cdot j/2$ | NR-NORX [5] | $t_2 + j$ | $t_2 \cdot j$ |
| CLOC [23] | $t_1 + j$ | $t_1 \cdot j$ | OCB [29] | $t_1$ | $t_1$ |
| COLM [3] | $t_1$ | $t_1 + j$ | SILC [23] | $t_1 + j$ | $t_1 \cdot j$ |
| DEOXYS [25] | $t_1 + j$ | $t_1 \cdot j$ | TIAOXIN [39] | $t_2 + j$ | $t_2 \cdot j/2$ |
| JAMBU [50] | $t_1 + j$ | $t_1 \cdot j/2$ | | | |
| Classical Schemes | | | | | |
| CWC [28] | $t_1$ | $t_1$ | CCM [16] | $t_1 + j$ | $t_1 + j$ |
| EAX [9] | $t_1 + j$ | $t_1 \cdot j$ | GCM [33] | $t_1$ | $t_1$ |

**Table 1:** Expected #oracle queries required for $j$ forgeries for IV/nonce-based classical schemes and 3rd-round CAESAR candidates. By $t_1$ and $t_2$, we denote the computational cost for obtaining the first forgery, where $t_2$ relates to wide-state designs. **NR** = nonce-respecting setting; **NI** = nonce-ignoring setting. Since we obtained the same results for DEOXYS-I and DEOXYS-II, we combine them to DEOXYS in this table. NR-NORX (draft) means the nonce-misuse-resistant version of NORX.

approach to the wide-pipe mode introduced for hash functions [32], the internal state of an nAE scheme is at least twice as big as the output, i.e., the tag value. Such a design is, for example, given by the widely used Sponge construction [11]. That would make the search for a generic collision significantly harder than the search for multiple forgeries. We denote the number of queries required for finding a collision within a wide internal state by $t_2$. Finally, we call an nAE scheme *vulnerable* to $j$-IV-CAs if it is neither resistant nor semi-resistant to $j$-IV-CA.

**Contribution.** This work classifies nonce-based AE schemes depending on the usage of their inputs to the initialization, encryption, and authentication process, and categorize the considered AE schemes regarding to that classification. To allow for a systematic analysis of the reforgeability of AE schemes, we introduce the $j$-IV-Collision Attack based on the introduced security definition $j$-INT-CTXT, providing us with expected upper bounds on the hardness of further forgeries (a summary of our results can be found in Table 1). For our attack, we pursue the idea of the message-block-collision attacks presented in [44] and [17]. However, in contrast, we focus on an internal collision within the processing of the associated data and/or the nonce. In the last section, we provide two alternative approaches to provide resistance in the sense of reforgeability and $j$-IV-CAs. Moreover, for AES-OTR, COLM, and OCB, we describe three attacks making multi-forgery attacks more efficient than our generic approach.

**Outline.** Section 2 provides necessary preliminaries including our security notions. Section 3 introduces our classification of generic AE schemes. Section 4 presents the $j$-IV-CA, a generic security analysis, and categorization of authenticated encryption schemes regarding to the introduced classes. Section 5 contains possible remedies to $j$-IV-CAs and Section 6 concludes our work.

**Algorithm 1** The $j$-INT-CTXT Experiment.

**Experiment** $j$-INT-CTXT
1: $K \twoheadleftarrow \mathcal{K}$
2: Run $\mathbf{A}^{\mathcal{E}(\cdot),\mathcal{D}(\cdot)}$ such that $\mathbf{A}$ never queries $\mathcal{E} \hookrightarrow \mathcal{D}$
3: **if** $\mathbf{A}$ made $j$ distinct decryption queries $(A_i, N_i, C_i, T_i)$, $1 \leq i \leq j$ such that $\mathcal{D}_K(A_i, N_i, C_i, T_i) \neq \perp$ for all $1 \leq i \leq j$ **then return** 1
4: **return** 0

# 2 Preliminaries

We use lowercase letters $x$ for indices and integers, uppercase letters $X, Y$ for binary strings and functions, and calligraphic uppercase letters $\mathcal{X}, \mathcal{Y}$ for sets and combined functions. We denote the concatenation of binary strings $X$ and $Y$ by $X \,\|\, Y$ and the result of their bitwise XOR by $X \oplus Y$. We indicate the length of $X$ in bits by $|X|$, and write $X_i$ for the $i$-th block (assuming that $X$ can be split into blocks of, e.g., $n$ bits). Furthermore, we denote by $X \twoheadleftarrow \mathcal{X}$ that $X$ is chosen uniformly at random from the set $\mathcal{X}$. For an event $E$, we denote by $\Pr[E]$ the probability of $E$.

**Adversaries and Advantages.** An adversary $\mathbf{A}$ is an efficient Turing machine that interacts with a given set of oracles that appear as black boxes to $\mathbf{A}$. We denote by $\mathbf{A}^{\mathcal{O}}$ the output of $\mathbf{A}$ after interacting with some oracle $\mathcal{O}$. We write $\mathbf{Adv}^{X}_{F}(\mathbf{A})$ for the advantage of an adversary $\mathbf{A}$ against a security notion $X$ on a function/scheme $F$. All probabilities are defined over the random coins of the oracles and those of the adversary, if any. We write $\mathbf{Adv}^{X}_{F}(q, \ell, t) = \max_{\mathbf{A}}\{\mathbf{Adv}^{X}_{F}(\mathbf{A})\}$ to refer to the maximal advantage over all $X$-adversaries $\mathbf{A}$ on a given scheme/function $F$ that run in time at most $t$ and pose at most $q$ queries consisting of at most $\ell$ blocks in total to the available oracles. Wlog., we assume that $\mathbf{A}$ never asks queries to which it already knows the answer, and by $\mathcal{O}_1 \hookrightarrow \mathcal{O}_2$ we denote that $\mathbf{A}$ never queries $\mathcal{O}_2$ with the output of $\mathcal{O}_1$.
We define as $(q_E, q_D, \ell, t)$-adversary $\mathbf{A}$ an adversary that asks at most $q_E$ queries to its first oracle, $q_D$ queries to its second oracle, which consist of at most $\ell$ blocks in sum, where $\mathbf{A}$ runs in time at most $t$. We define a scheme $\Pi$ to be $(q_E, q_D, \ell, t, \epsilon)$-X-secure to a notion X if the maximal advantage of all $(q_E, q_D, \ell, t)$-X-adversaries on $\Pi$ is upper bounded by $\epsilon$.
During the query phase, we say that an adversary $\mathbf{A}$ maintains a query history $\mathcal{Q}$ collecting all requests together with their corresponding answer. We write $\mathcal{Q}_{|X}$, if we refer only to all entries of type $X$ in the query history. For example, $N_i \notin \mathcal{Q}_{|N}$ denotes that the nonce $N_i$ is not contained in the set of nonces already in the query history.

**Nonce-Based AE Schemes.** A nonce-based authenticated encryption (nAE) scheme (with associated data) [42] is a tuple $\Pi = (\mathcal{E}, \mathcal{D})$ of a deterministic encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{A} \times \mathcal{N} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}$, and a deterministic decryption algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{A} \times \mathcal{N} \times \mathcal{C} \times \mathcal{T} \to \mathcal{M} \cup \{\perp\}$, with associated non-empty key space $\mathcal{K}$, associated data space $\mathcal{A} \subseteq \{0,1\}^*$, the non-empty nonce space $\mathcal{N}$, and $\mathcal{M}$, $\mathcal{C} \subseteq \{0,1\}^*$ denote the message and ciphertext space, respectively. We define a tag space $\mathcal{T} = \{0,1\}^\tau$ for a fixed $\tau \geq 0$. We write $\mathcal{E}^{A,N}_K(M)$ and $\mathcal{D}^{A,N}_K(C,T)$ as short forms of $\mathcal{E}(K, A, N, M)$ and $\mathcal{D}(K, A, N, C, T)$. If a given tuple $(A, N, C, T)$ is valid, $\mathcal{D}^{A,N}_K(C,T)$ returns the corresponding plaintext $M$, and $\perp$ otherwise. We assume that for all $K \in \mathcal{K}$, $A \in \mathcal{A}$, $N \in \mathcal{N}$, and $M \in \mathcal{M}$ holds *stretch-preservation*: if $\mathcal{E}^{A,N}_K(M) = (C,T)$, then $|C| = |M|$ and $|T| = \tau$, *correctness*: if $\mathcal{E}^{A,N}_K(M) = (C,T)$, then $\mathcal{D}^{A,N}_K(C,T) = M$, and *tidiness*: if $\mathcal{D}^{A,N}_K(C,T) = M \neq \perp$, then $\mathcal{E}^{A,N}_K(M) = (C,T)$, for all $C \in \mathcal{C}$ and $T \in \mathcal{T}$.

**Security Notions for Reforgeability.** In 2004, Bellare et al. introduced the two security notions INT-PTXT-M and INT-CTXT-M [7]; however, these notions capture the setting that an

adversary can pose multiple verification queries for a *single* forgery. In contrast, we are interested in finding *multiple* (in general $j \geq 1$) forgeries based on multiple verification queries. In the scenario of INT-CTXT, an adversary wins if it can find any valid forgery, that is a tuple $(A, N, C, T)$ for which the decryption returns anything different from the invalid symbol $\perp$ and which has not been previously obtained by $\mathbf{A}$ as response of the encryption oracle. The $j$-INT-CTXT security notion, as shown in Algorithm 1, is derived from INT-CTXT in the sense that $\mathbf{A}$ now has to provide $j$ distinct valid forgeries that all have not been obtained from the encryption oracle. In the following, we define the $j$-INT-CTXT Advantage of an adversary.

**Definition 1 ($j$-INT-CTXT Advantage).** *Let $\Pi = (\mathcal{E}, \mathcal{D})$ be a nonce-based AE scheme, $K \twoheadleftarrow \mathcal{K}$, and $\mathbf{A}$ be a computationally bounded adversary on $\Pi$ with access to two oracles $\mathcal{E}$ and $\mathcal{D}$ such that $\mathbf{A}$ never queries $\mathcal{E} \hookrightarrow \mathcal{D}$. Then, the $j$-INT-CTXT advantage of $\mathbf{A}$ on $\Pi$ defined as*

$$\mathbf{Adv}_{\Pi}^{j\text{-INT-CTXT}}(\mathbf{A}) := \Pr\left[\mathbf{A}^{\mathcal{E},\mathcal{D}} \text{ forges } j \text{ times}\right],$$

*where "forges" means that $\mathcal{D}_K$ returns anything other than $\perp$ for a query of $\mathbf{A}$, and "forges $j$ times" means that $\mathbf{A}$ provides $j$ distinct decryption queries $(A_i, N_i, C_i, T_i)$, $1 \leq i \leq j$ such that $\mathcal{D}_K(A_i, N_i, C_i, T_i) \neq \perp$ for all $1 \leq i \leq j$.*

We define $\mathbf{Adv}_{\Pi}^{j\text{-INT-CTXT}}(q_E, q_D, \ell, t)$ for the maximal advantage over all adversaries $\mathbf{A}$ on $\Pi$ that ask at most $q_E$ encryption queries, $q_D$ decryption queries, which sum up to at most $\ell$ blocks in total, and run in time at most $t$.

# 3 Classification of AE Schemes

In our work, we consider AE schemes from a general point of view. Therefore, in comparison to the classification of Namprempre, Rogaway, and Shrimpton [37], we introduce one additional optional input to the tag-generation step (a key-dependent chaining value) and further, we distinguish between the message and the ciphertext being input to the tag generation.
We classify AE schemes according to their inputs to an initialization function $F_{IV}$ and a tag-generation function $F_T$. Let $\mathcal{K}, \mathcal{A}, \mathcal{N}, \mathcal{IV}, \mathcal{T}, \mathcal{M}, \mathcal{CV}$, and $\mathcal{C}$ define the key, associated data, nonce, IV, tag, message, chaining-value, and ciphertext space, respectively. We define three functions $F_{IV}$, $\mathcal{E}$, and $F_T$ as follows:

$$\begin{aligned} F_{IV} &: \mathcal{K}[\times\mathcal{A}][\times\mathcal{N}][\times\mathcal{M}] && \to \mathcal{IV}, \\ \mathcal{E} &: \mathcal{K} \times \mathcal{IV} \times \mathcal{M} && \to \mathcal{C}[\times\mathcal{CV}], \\ F_T &: \mathcal{K}[\times\mathcal{CV}][\times\mathcal{M}][\times\mathcal{C}][\times\mathcal{A}][\times\mathcal{N}] && \to \mathcal{T}, \end{aligned}$$

where $\mathcal{A}, \mathcal{N}, \mathcal{M}, \mathcal{CV}, \mathcal{C} \subseteq \{0,1\}^*$, $\mathcal{T} \subseteq \{0,1\}^\tau$, and $\mathcal{IV} \subseteq \{0,1\}^*$. The expressions (sets) given in brackets are *optional* inputs to the corresponding function, e.g., the function $F_{IV}$ must be provided with at least one input (the key $K \in \mathcal{K}$), but is able to process up to four inputs (including associated data $A \in \mathcal{A}$, nonce $N \in \mathcal{N}$, and message $M \in \mathcal{M}$).
From this, we introduce a generic classification based on which input is used in $F_{IV}$ and $F_T$. Note that the encryption algorithm $\mathcal{E}$ is equal for all classes described, i.e., it encrypts a message $M$ under a key $K$ and an $IV \in \mathcal{IV}$, and outputs a ciphertext $C \in \mathcal{C}$. However, the authors of [37] distinguished between IV-based (ivE) and nonce-based (nE) encryption schemes. Such a distinction is covered by our generalized approach since one can simply assume the only input to $F_{IV}$ to be the nonce (and the key) and making $F_{IV}$ itself the identity function, i.e., it forwards the nonce $N$ to the encryption function $\mathcal{E}$. Moreover, AE schemes built from generic composition can be modelled by setting $x_3 = 0$ and assuming $F_T$ to be a PRF-secure MAC (see below for the meaning of $x_3$).
In the following, we encode the combination of inputs as a sequence of eight bits $x_0, \ldots, x_7$, where each bit denotes whether an input is used (1) or not (0), resulting in a total of $2^8 = 256$ possible

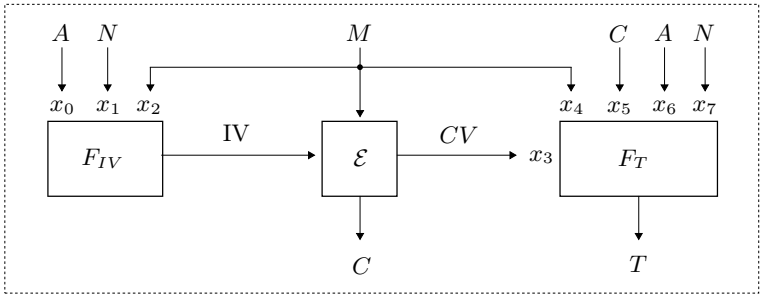


**Fig. 1:** Generic AE scheme as considered in our analysis.

| **Set of Classes** | **Input to $F_{IV}$** | **Input to $F_T$** |
|---|---|---|
| $(01****10)$ | $\mathcal{K} \times \mathcal{N}[\times\mathcal{M}]$ | $\mathcal{K}[\times\mathcal{CV}][\times\mathcal{M}][\times\mathcal{C}] \times \mathcal{A}$ |
| $(01****11)$ | $\mathcal{K} \times \mathcal{N}[\times\mathcal{M}]$ | $\mathcal{K}[\times\mathcal{CV}][\times\mathcal{M}][\times\mathcal{C}] \times \mathcal{A} \times \mathcal{N}$ |
| $(11****00)$ | $\mathcal{K} \times \mathcal{A} \times \mathcal{N}[\times\mathcal{M}]$ | $\mathcal{K}[\times\mathcal{CV}][\times\mathcal{M}][\times\mathcal{C}]$ |
| $(11****01)$ | $\mathcal{K} \times \mathcal{A} \times \mathcal{N}[\times\mathcal{M}]$ | $\mathcal{K}[\times\mathcal{CV}][\times\mathcal{M}][\times\mathcal{C}] \times \mathcal{N}$ |
| $(11****10)$ | $\mathcal{K} \times \mathcal{A} \times \mathcal{N}[\times\mathcal{M}]$ | $\mathcal{K}[\times\mathcal{CV}][\times\mathcal{M}][\times\mathcal{C}] \times \mathcal{A}$ |
| $(11****11)$ | $\mathcal{K} \times \mathcal{A} \times \mathcal{N}[\times\mathcal{M}]$ | $\mathcal{K}[\times\mathcal{CV}][\times\mathcal{M}][\times\mathcal{C}] \times \mathcal{A} \times \mathcal{N}$ |

**Table 2:** Overview of accepted classes. All excluded classes are trivially insecure.

classes. More detailed, the first three bits $x_0, x_1, x_2$ denote whether the associated data $A$, the nonce $N$, or the message $M$ is used as input to $F_{IV}$, respectively. The bits $x_3, \ldots, x_7$ denote whether a key-dependent chaining value $CV$, $M$, $C$, $A$, or $N$ is used as input to $F_T$, respectively (see Figure 1 for a depiction of our generic AE scheme). For example, the string (11010011) represents $F_{IV} : \mathcal{K} \times \mathcal{A} \times \mathcal{N} \to \mathcal{IV}$ and $F_T : \mathcal{K} \times \mathcal{CV} \times \mathcal{A} \times \mathcal{N} \to \mathcal{T}$ as it would be the case for, e.g., POET [2], CLOC, and SILC [23]. Further, we mark a bit position by '*' if we do not care about whether the specific input is available or not.

Our next step is to significantly reduce the number of possible classes by disregarding those that are trivially insecure. First, we can simply discard $2^4 = 16$ classes of the form $(00 * * * *00)$, where neither the nonce $N$ nor the associated data $A$ is considered as input. Similarly, we can exclude $6 \cdot 2^4 = 96$ classes which lack the use of either the nonce *or* the associated data, i.e., $\{(01****00), (01****01), (10****00), (10****10), (00****01), (00****10)\}$. Finally, since a secure nonce-based AE scheme requires the nonce to influence at least the encryption step, we can further disregard the $3 \cdot 2^4 = 48$ classes $\{(00****11), (10****01), (10****11)\}$ which omit the nonce in the initialization function $F_{IV}$. As a result, we reduced the number of relevant classes to 96. An overview can be found in Table 2.

# 4 $j$-Int-CTXT-Analysis of nAE schemes

## 4.1 $j$-IV-Collision Attack

In this section, we introduce a new attack type called *$j$-IV-Collision Attack ($j$-IV-CA)* as one possible way to analyze the security of a nonce-based AE scheme regarding to reforgeability. We provide two variants (1) for the nonce-ignoring (NI; also known as nonce misuse) and (2) the nonce-respecting (NR) setting. The core idea of a $j$-IV-CA is to (1) assume a first forgery can be found caused by an internal collision within the processing of the associated data $A$ and/or the nonce $N$

**Algorithm 2** $j$-IV-Collision Attack for nonce-ignoring adversaries.

1: Choose an arbitrary fixed message $M$
2: $\mathcal{Q} \leftarrow \emptyset$
3: **for** $i \leftarrow 1$ to $t_1$ **do**
4: Choose $(A_i, N_i)$ with $(A_i, N_i) \notin \mathcal{Q}_{|A,N}$
5: Query $(A_i, N_i, M)$ and receive $(C_i, T_i)$.
6: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(A_i, N_i, M, C_i, T_i)\}$
7: **if** $T_i \in \mathcal{Q}_{|T}$ **then**
8: Store the tuples $(A_i, N_i, M, C_i, T_i)$ and $(A_k, N_k, M, C_k, T_k)$ for which $T_i = T_k$
9: **break**
10: **for** $\ell \leftarrow 1$ to $j$ **do**
11: Choose $M_\ell \notin \mathcal{Q}_{|M}$
12: Query $(A_i, N_i, M_\ell)$ and receive $(C'_\ell, T'_\ell)$
13: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(*, *, M_\ell, *, *)\}$
14: Output the forgery $(A_k, N_k, C'_\ell, T'_\ell)$

and (2) to exploit this collision for efficiently constructing $j$ further forgeries. Depending on the class of an AE scheme, such a collision can occur during the invocation of $F_{IV}$, $F_T$, or both.
Due to the character of the attacks presented in this section, we can derive a set of classes $\mathcal{C}_0$ of nAE schemes for which those attacks are trivially applicable. For all schemes belonging to that class, it holds that neither the message $M$, a message/ciphertext-depending chaining $CV$, nor the ciphertext $C$ influence the first collision found by our adversary, e.g., if an adversary tries to construct a collision for the outputs of $F_{IV}$, the only possible inputs to $F_{IV}$ are either the nonce $N$, the associated data $A$, or both. Therefore, the set $\mathcal{C}_0$ contains the following 22 classes of AE schemes:

$$\mathcal{C}_0 = \{(110***0*), (01*0001*), (11000011), (11000010)\}.$$

**Nonce-Ignoring Setting.** The attack for the nonce-ignoring setting is described in Algorithm 2. An adversary **A** starts by choosing a fixed arbitrary message $M$ and pairs $(A_i, N_i)$ not queried before ($(A_i, N_i) \notin \mathcal{Q}_{|A,N}$, see Line 4). That builds up a query $(A_i, N_i, M)$ resulting in an oracle answer $(C_i, T_i)$ which is stored by **A** in the query history $\mathcal{Q}$. Once a collision of two tag values $T_i$ and $T_k$ (implying a collision of two pairs $(A_i, N_i) \neq (A_k, N_k)$)[3] was found (Line 7 of Algorithm 2), **A** starts to generate $j$ additionally queries with an effort of $\mathcal{O}(j)$ (Lines 10-14). In Lines 6 and 13, the adversary is collecting all tuples queried so far, where in Line 13 we are only interested in the values of $M_\ell$, since these are not allowed to repeat (see Line 11) by the definition of **A**.
It is easy to observe that **A** has to use the same nonce twice, i.e., $N_i$ is chosen in Line 4 and reused in Line 12 of Algorithm 2. Independent from the number of queries of finding the $j$ additional forgeries, **A** always (in the nonce-ignoring as well as in the nonce-respecting setting) has to find a collision for two pairs $(A_i, N_i) \neq (A_k, N_k)$. That number of queries (denoted by $t_1$ in general, or by $t_2$ if the scheme employs a wide state of $\geq 2n$ bits (or $\geq 2\tau$ bits, when referring to the size of the tag value), see Table 1) always depends on the concrete instantiation of our generic AE scheme and is usually bounded by at least $\mathcal{O}(q^2/2^n)$ (birthday bound), where $q$ denotes the number of queries and $n$ the state size in bit. In Table 5 of Appendix B, the reader can find the security claims of the considered AE schemes provided by their respective designers.

[3] Based on our assumption, the case $T_i = T_k$ can be caused by an internal collision of the processing of two pairs $(A_i, N_i) \neq (A_k, N_k)$. Moreover, since we are considering the nonce-ignoring setting allowing an adversary for repeating the values $N_i$, we can say wlog. that we must have found two associated data values $A_i \neq A_k$ leading to an equal output of the processing of the associated data, e.g., the initialization vector $IV$ (see Figure 1).

**Algorithm 3** $j$-IV-Collision Attack for nonce-respecting adversaries.

1: Choose an arbitrary fixed message block $M$
2: $\mathcal{Q} \leftarrow \emptyset$
3: **for** 1 to $j$ **do**
4: **for** $i \leftarrow 1$ to $t_1$ **do**
5: Choose $(A_i, N_i)$ with $(A_i, N_i) \notin \mathcal{Q}_{|A,N}$
6: Choose $P_i$ with $P_i \notin \mathcal{Q}_{|P}$
7: Query $(A_i, N_i, M \,\|\, P_i)$ and receive $(C_i^1 \,\|\, C_i^{P_i}, T_i)$.
8: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(A_i, N_i, C_i^1 \,\|\, C_i^{P_i}, T_i)\}$
9: **if** $C_i^1 \in \mathcal{Q}_{|C^1}$ **then**
10: **A** outputs the tuples $(A_i, N_i, C_i^1 \,\|\, C_k^{P_k}, T_k)$ and $(A_k, N_k, C_k^1 \,\|\, C_i^{P_i}, T_i)$
11: for which $C_i^1 = C_k^1$ holds
12: **goto** Step 4

**Nonce-Respecting Setting.** The second setting prohibits an adversary from repeating any value $N_i$ during its encryption queries. Therefore, we introduce a modified version of the $j$-IV-CA as proposed above. Such an attack works for all schemes that allow to observe a collision of the outputs of the IV-generation step by just looking at the ciphertext blocks. Thus, during the first step, we do not care about finding the first forgery but only about the collision during $F_{IV}$ as shown in Algorithm 3. This attacks works also for nAE schemes that consider the associated data $A_i$ only as input to $F_T$. In such a situation, **A** would leave $A_i$ constant (or empty when considering $F_{IV}$) and would vary only $N_i$ to find a collision within $F_{IV}$.
If the number of queries for finding a collision during the processing of the associated data is given by $t_1$, an adversary requires $j \cdot t_1$ queries in average to obtain $2 \cdot j$ forgeries. Clearly, this attack is weaker than that in the nonce-misuse setting above, but still reduces the number of queries for finding $j$ forgeries from $j \cdot t_1$ to $1/2 \cdot (j \cdot t_1)$.

## 4.2 Security Analysis

For all nAE schemes which belong to $\mathcal{C}_0$, there exist a straight-forward argument that they are insecure in the nonce-ignoring setting. A $j$-IV-CA, as defined in Algorithm 2, requires an adversary **A** to choose $j$ pair-wise distinct messages $M_1, \ldots, M_j$. Beforehand, we assume **A** to be successful in finding the first forgery for two distinct pairs $(A_i, N_i)$ and $(A_k, N_k)$ (Lines 3-9 of Algorithm 2) using $t_1$ queries.
Therefore, the $j$-IV-CA adversary **A** queries $t_1$ distinct pairs $(A_i, N_i) \neq (A_k, N_k)$, together with a fixed message $M$, until an internal collision leads to the case $T_i = T_k$. Since the event of that very first collision does not depend on the message, a chaining value, and/or the ciphertext (requirement for an nAE scheme to be placed in $\mathcal{C}_0$), we can always choose a new message and still can ensure the internal collision for the pairs $(A_i, N_i)$ and $(A_k, N_k)$. Then, **A** only has to query $(A_i, N_i, M_\ell)$ for a fresh message $M_\ell$ to the encryption oracle and receives $(C'_\ell, T'_\ell)$, where it is trivial to see that the pair $(C'_\ell, T'_\ell)$ will also be valid for $(A_k, N_k, M_\ell)$. **A** then only has to repeat this process for $j$ pairwise distinct messages $M_\ell$.

In the case of a nonce-respecting adversary (see Algorithm 3), an internal collision of the processing of ($(A_i)$ and) $N_i$ is detected by observing colliding ciphertext blocks (see Line 9). Since the attack requires an internal collision within the IV-generation step and the nonce $N_i$ must not directly influence the tag-generation step $F_T$, the nonce $N_i$ must be given as input to $F_{IV}$, but not to $F_T$. The associated data $A_i$ can be given as input to $F_{IV}$, $F_T$, or both. Therefore, the attack described in Algorithm 3 is applicable to all schemes belonging to the subset $\{(11{*}{*}{*}{*}00), (11{*}{*}{*}{*}00), (01{*}{*}{*}{*}10)\}$ of $\mathcal{C}_0$.

| Scheme | Class | NI | NR | Scheme | Class | NI | NR |
|---|---|---|---|---|---|---|---|
| 3rd-Round CAESAR Candidates ($\mathcal{C}_0$) | | | | 3rd-Round CAESAR Candidates ($\mathcal{C}_1$) | | | |
| ACORN | (11011000) | – | – | AEGIS | (11011010) | ∘ | ∘ |
| AES-OTR (ser.) | (11001100) | – | – | AES-OTR (par.) | (01001110) | – | – |
| Ascon | (11010100) | ∘ | ∘ | AEZv4 | (11011011) | – | – |
| COLM | (11011000) | – | – | CLOC | (11010101) | – | • |
| JAMBU | (11011000) | – | – | Deoxys-I | (01011001) | – | • |
| Ketje | (11010000) | ∘ | ∘ | Deoxys-II | (01011001) | – | • |
| NORX | (11010100) | ∘ | ∘ | Keyak | (01011010) | ∘ | ∘ |
| Classical AE Schemes ($\mathcal{C}_1$) | | | | MORUS | (11011010) | ∘ | ∘ |
| CCM | (01011011) | – | • | NR-NORX | (11110100) | ∘ | • |
| CWC | (01010110) | – | – | OCB | (01001010) | – | – |
| EAX | (01000111) | – | • | SILC | (11010101) | – | • |
| GCM | (01000111) | – | – | Tiaoxin | (11011010) | ∘ | ∘ |

**Table 3:** $j$-IV-CA-Resistance of the third-round CAESAR candidates and considered classical AE schemes, in the nonce-ignoring (**NI**) and the nonce-respecting (**NR**) setting. '•' indicates resistance, '∘' vulnerability under certain requirements (e.g., the scheme employs a wide state), and '–' vulnerability. AES-OTR (ser.) means the serial and (par.) the parallel mode.

All remaining 74 classes in the set $\mathcal{C}_1$ provide resistance to $j$-IV-CAs from a theoretical point of view, i.e., with regard to our generalized AE scheme as shown in Figure 1.

$$\begin{aligned}\mathcal{C}_1 = \{&(01*0011*), (01*0101*), (01*0111*), (01*1001*), (01*1011*),\\ &(01*1101*), (01*1111*), (1100011*), (1100101*).(1100111*),\\ &(1101001*), (1101011*), (1101101*), (1101111*), (111*****)\}\end{aligned}$$

However, in practice, their security highly depends on the specific instantiation of $F_{IV}$ and/or $F_T$. In the next section, we look at concrete instantiations from the class $\mathcal{C}_1$ as well as from $\mathcal{C}_0$ when considering classical nAE schemes and 3rd-round CAESAR candidates.

## 4.3 Concrete Instantiations of $\mathcal{C}_1$ and $\mathcal{C}_0$

The resistance of the classes in $\mathcal{C}_1$ to $j$-IV-CA regarding to our generalized AE scheme stems from the fact that the message, and/or a chaining value, and/or the ciphertext affect the generation of the IV or the tag, i.e., is input to $F_{IV}$ and/or $F_T$. However, if we move from our generalized approach to concrete instantiations of these classes, i.e., to existing AE schemes whose structure is defined by a class in $\mathcal{C}_1$, we will see that some of those classes do not provide resistance to $j$-IV-CAs. However, AE schemes whose classes belong to $\mathcal{C}_0$ are vulnerable to $j$-IV-CAs in both the NI and the NR setting. In Table 3, we give an overview of the resistance the considered AE schemes to $j$-IV-CAs. We also provide a brief discussion for those cases that are not trivially observable in the following. In addition to the generic $j$-IV-CAs in this section, we recall stronger multi-forgery attacks on OCB, AES-OTR, and COLM from the literature in Appendix C.

**AEGIS, MORUS, and Tiaoxin.** These schemes are semi-resist to $j$-IV-CAs in the nonce-respecting and the nonce-ignoring setting. This stems from the fact that they employ very wide states, which are initialized by nonce and associated data, and which are more than twice as large as the final ciphertext stretch; therefore, the search for state collisions is at best a task of sophisticated cryptanalysis, and at worst by magnitudes less efficient than the trivial search by querying many forgery attempts. As a side effect, the search for state collisions is restricted to associated data and messages of equal lengths since their lengths are used in $F_T$ (for that reason, we set the bit $x_6$).

**CWC and GCM.** In the nonce-ignoring setting, forgeries for CWC and GCM can be obtained with a few queries. The tag-generation procedures of both modes employ a Carter-Wegman MAC consisting of XORing the encrypted nonce with an encrypted hash of associated data and ciphertext. The employed hash are polynomial hashes in both cases, which is well-known to lead to a variety of forgeries after a few queries when nonces are repeated.
In the nonce-respecting setting, both CWC and GCM possess security proofs that show that they provide forgery resistance up to the birthday bound (Iwata et al. [24] invalidated those for GCM and presented revised bounds which still are bound by the birthday paradox). However, a series of works from the past five years [45,40,1] illustrated that the algebraic structure of polynomial hashing may allow to retrieve the hashing key from forgery polynomials with many roots. The most recent work by Abdelraheem et al. [1] proposes universal forgery attacks that work on a weak key set. Thus, a nonce-respecting adversary could find the hash key and possess the power to derive universal forgeries for those schemes, even with significantly less time than our nonce-respecting attack.

**AES-OTR and OCB.** In the nonce-ignoring setting, these schemes are trivially insecure, as has been clearly stated by their respective authors. We consider OCB as an example, a similar attack can be performed on AES-OTR if nonces are reused. A nonce-ignoring adversary simply performs the following steps:

1. Choose $(A, N, M)$ such that $M$ consists of at least three blocks: $M = (M_1, M_2, \ldots)$, and ask for their authenticated ciphertext $(C_1, C_2, \ldots, T)$.
2. Choose $\Delta \neq 0^n$, and derive $M'_1 = M_1 \oplus \Delta$ and $M'_2 = M_2 \oplus \Delta$. For $M' = M'_1, M'_2$ and $M'_i = M_i$, for $i \geq 3$, ask for the authenticated ciphertext $(C'_1, C'_2, \ldots, T)$ that corresponds to $(A, N, M')$.
3. Given the authenticated ciphertext $(C'', T'')$ for any further message $(A,\ N,\ M'')$ with $M'' = (M_1, M_2, \ldots)$, the adversary can forge the ciphertext by replacing $(C''_1, C''_2) = (C_1, C_2)$ with $(C'_1, C'_2)$.

Therefore, the complexities for $j$ forgeries under nonce-ignoring adversaries are only $t_1$ (and not $t_1 + j$, see Table 1). Because of their structure, there exist nonce-respecting forgery attacks on AES-OTR and OCB that are stronger than our generic $j$-IV-CA. Those can be found in Appendix C.

**AEZv4.** Since AEZv4 does not separate the domains of $(A_i, N_i)$ for $IV$ and tag generation, our $j$-IV-CAs work out-of-the box here. More detailed, nonce and associated data are parsed into a string $T_1, \ldots, T_t$ of $n$-bit strings $T_i$, and simply hashed in a PHASH-like manner inside AEZ-hash: $\Delta \leftarrow \bigoplus_{i=1}^{t} E_K^{i+2,1}(T_i)$, where $E$ denotes a variant of four-round AES. The adversary can simply ask for the encryption of approximately $2^{64}$ tuples $(A_i, N_i, M)$ for fixed $M$. Obtaining a collision for this hash (requiring birthday-bound complexity) can be easily detected when the message is kept constant over all queries. Given such a hash collision for $(A_i, N_i)$ and $(A_k, N_k)$, the adversary can directly construct subsequent forgeries by asking for the encryption of $(A_i, N_i, M')$ and the same ciphertext will be valid for $(A_k, N_k, M')$ for arbitrary $M'$.

**Deoxys.** Deoxys-I, i.e., the nonce-requiring variant, possesses a similar structure as OCB. Hence, there are trivial multi-forgery attacks with few queries if nonces repeat:

1. Choose $(A, N, M)$ arbitrarily and ask for $(C, T)$.
2. Choose $A' \neq A$, leave $N$ and $M$ constant and ask for $(C' = C, T')$. Since the tag is computed by the XOR of $\mathrm{Hash}(A)$ with the encrypted checksum under the nonce as tweak, the adversary sees the difference in the hash outputs in the tags: $\mathrm{Hash}(A) \oplus \mathrm{Hash}(A') = T \oplus T'$.
3. Choose $(A, N', M')$ and ask for $(C'', T'')$. It instantly follows that for $(A', N', M')$, $(C'', T''' = T \oplus T' \oplus T'')$ will be valid.

However, in the nonce-respecting setting, the use of a real tweaked block cipher that employs the nonce in tweak (instead of the XEX construction as in AES-OTR and OCB) prevents the attacks in Appendix C; the tag generation seems surprisingly strong in the sense that an adversary can not detect collisions between two associated data since the hash is XORed with an output of a fresh block cipher (because of the nonce is used as tweak) for every query. Therefore, we indicate that DEOXYS-I provides resistance in the nonce-respecting setting.
DEOXYS-II is a two-pass mode, i.e., the message is processed twice (1) once for the encryption process and (2) for the authentication process. In the nonce-ignoring setting, an adversary can simply fix $N_i$ and vary $A_i$ for finding a collision for `Auth`, which renders the scheme vulnerable to $j$-IV-CAs. Therefore, that kind of two-pass scheme (in comparison to SIV, where the message is used as input to $F_{IV}$), does not implicitly provide resistance to $j$-IV-CAs.

**NORX.** The authors of NORX presented a nonce-misuse resistant version of their scheme in Appendix D of [5]. NR-NORX follows the MAC-then-Encrypt paradigm, which yields a two-pass scheme similar to SIV. Therefore, NR-NORX provides at the least resistance to $j$-IV-CAs in the NR setting, which renders it stronger than NORX. However, this security comes at the cost of being off-line and two-pass.

**CCM, EAX, CLOC, and SILC.** The resistance to $j$-IV-CAs in the nonce-respecting setting provided by CCM, EAX, CLOC, and SILC stems from similar reasons as for DEOXYS-II; the tag is generated by the XOR of the MAC of the nonce with the MAC of the ciphertext and the MAC of the associated data. Hence, collisions in ciphertext or header can not be easily detected since the MAC of a fresh nonce is XORed to it.

# 5 Countermeasures to $j$-IV-C Attacks

This section briefly describes two possible approaches for providing resistance to $j$-IV-CAs in the nonce-respecting (NR) as well as in the nonce-ignoring (NI) setting.

**Independence of $F_{IV}$ and $F_T$.** For realizing that approach, the pair $(A_i, N_i)$ has to be processed twice. Let $F_{IV}(A_i, N_i, *)$ be the IV-generation step of an nAE scheme processing the tuple $(A_i, N_i, *)$, where '$*$' denotes that $F_{IV}$ can optionally process the message $M$. Usually, it is proven that $F_{IV}$ behaves like a PRF. Further, let $F_T(*, *, *, A_i, N_i)$ be the tag-generation step of an AE scheme processing the tuple $(*, *, *, A_i, N_i)$, where the first three inputs can be the chaining value $CV$, the message $M$, and or the ciphertext $C$[4], and there exists a proof showing that $F_T$ also behaves like a PRF. Hence, the corresponding scheme would have the class $(11{*}{*}{*}11)$ which belongs to $\mathcal{C}_1$. If one can guarantee independence between $F_{IV}$ and $F_T$, we can say that the outputs of $F_{IV}(A_i, N_i, *)$ and $F_T(*, *, *, A_i, N_i)$ are independent random values. Based on that assumption, a simple collision of the form $F_{IV}(A_i, N_i, *) = F_{IV}(A_k, N_k, *)$ (as required by the $j$-IV-CA) does not suffice to produce a forgery since it is highly likely that $F_T(A_i, N_i, *) \neq F_T(*, *, *, A_k, N_k)$ and vice versa. Therefore, this *two-pass processing* realizes a *domain separation* between the IV-generation and the tag-generation step, providing resistance to $j$-IV-CAs. One way to achieve that goal can be to invoke the same PRF twice (for $F_{IV}$ and $F_T$) but always guarantee distinct inputs, e.g., $F_{IV}(A_i, N_i, *, 1)$ and $F_T(*, *, *, A_i, N_i, 2)$. Another approach would be to just use two independent functions.

**Wide-State IV.** A second approach requires a PRF-processing of the associated data $F_{IV}$ which produces a wide-state output $\tau \leftarrow F_{IV}(A_i, N_i)$ with $|\tau| > n$ bit. For example, for $|\tau| = 2n$, a pair

[4] Note that at least one of the three inputs must be given since else, the tag would be independent from the message, which would make the scheme trivially insecure.

$(A_i, N_i)$ would be processed to two independent $n$-bit values $\tau_1$ and $\tau_2$. Then, one could use $\tau_1$ as initialization vector to the encryption step and $\tau_2$ as initialization vector to the tag-generation step. Therefore, one can always guarantee domain separation between encryption and tag generation, while remaining a one-pass AE scheme. One possible instantiation for such a MAC (which can be utilized for the processing of the associated data) is PMAC2x [30].

# 6 Conclusion

In this work, we followed on the idea of multi-forgery attacks first described by Ferguson in 2002 and went on with introducing the $j$-Int-CTXT notion. Further on, we introduced a classification of nonce-based AE schemes depending of the usage of their inputs to the initialization, encryption, and authentication process, and categorize them regarding to that classification. To allow a systematic analysis of the reforgeability of nonce-based AE schemes, we introduced the $j$-IV-Collision Attack, providing us with expected upper bounds on the hardness of further forgeries. During our analysis, we found that (1) no considered nAE schemes provides full resistance to $j$-IV-CA, (2) ACORN, AES-OTR (serial), Ascon, COLM, JAMBU, Ketje, and NORX belong to the class $\mathcal{C}_0$, rendering them implicitly vulnerable to $j$-IV-CAs, and (3) Ascon, Ketje, Keyak, MORUS, NORX, NR-NORX, and Tiaoxin are *semi-resistant* to $j$-IV-CAs since all of them employ a wide state. This has no impact on the applicability of a $j$-IV-CA itself, but a wide state hardens the computation of the internal collision, e.g., if the internal state is of size $2n$ (wide state) instead of $n$, a generic collision can be found in $2^n$ instead of $2^{n/2}$. Finally, we briefly proposed two alternative approaches which would render an nAE scheme resistant to $j$-IV-CAs in the nonce-respecting as well as the nonce-ignoring setting.

# References


1. Mohamed Ahmed Abdelraheem, Peter Beelen, Andrey Bogdanov, and Elmar Tischhauser. Twisted Polynomials and Forgery Attacks on GCM. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT I*, volume 9056 of *Lecture Notes in Computer Science*, pages 762–786, 2015.
2. Farzaneh Abed, Scott Fluhrer, John Foley, Christian Forler, Eik List, Stefan Lucks, David McGrew, and Jakob Wenzel. The POET Family of On-Line Authenticated Encryption Schemes. `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
3. Elena Andreeva, Andrey Bogdanov, Nilanjan Datta, Atul Luykx, Bart Mennink, Mridul Nandi, Elmar Tischhauser, and Kan Yasuda. COLM v1. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
4. Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar Tischhauser, and Kan Yasuda. AES-COPA. `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
5. Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
6. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In *CRYPTO*, pages 1–15, 1996.
7. Mihir Bellare, Oded Goldreich, and Anton Mityagin. The Power of Verification Queries in Message Authentication and Authenticated Encryption. *IACR Cryptology ePrint Archive*, 2004:309, 2004.
8. Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.
9. Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX Mode of Operation. In *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 2004.
10. Dan J. Bernstein. CAESAR Call for Submissions, Final, January 27 2014. `http://competitions.cr.yp.to/caesar-call.html`.
11. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. ECRYPT Hash Function Workshop, 2007.
12. Guido Bertoni, Joan Daemen, Michaël Peeters, and Ronny Van Keer Gilles Van Assche. CAESAR submission; Ketje v2. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.

13. John Black and Martin Cochran. MAC Reforgeability. In Orr Dunkelman, editor, *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers*, volume 5665 of *Lecture Notes in Computer Science*, pages 345–362. Springer, 2009.
14. Nilanjan Datta and Mridul Nandi. ELmD. `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
15. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
16. Morris J. Dworkin. SP 800-38C. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. Technical report, Gaithersburg, MD, United States, 2004.
17. Niels Ferguson. Collision Attacks on OCB. unpublished manuscript. Available online, 2002. `http://www.cs.ucdavis.edu/rogaway/ocb/links.htm`.
18. Niels Ferguson. Authentication weaknesses in GCM, 2005.
19. Pierre-Alain Fouque, Gwenaëlle Martinet, Frédéric Valette, and Sébastien Zimmer. On the Security of the CCM Encryption Mode and of a Slight Variant. In *ACNS*, pages 411–428, 2008.
20. Michaël Peeters Guido Bertoni, Joan Daemen, Gilles Van Assche, and Ronny Van Keer. CAESAR submission; Keyak v2. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
21. Helena Handschuh and Bart Preneel. Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In *CRYPTO*, pages 144–161, 2008.
22. Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. AEZ v4.2: Authenticated Encryption by Enciphering. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
23. Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, and Sumio Morioka. CLOC and SILC v3. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
24. Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and Repairing GCM Security Proofs. In *CRYPTO*, volume 7417 of *LNCS*, pages 31–49. Springer, 2012.
25. Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannix Seurin. Deoxys v1.41. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
26. Magnus Westerlund John Mattsson. Authentication Key Recovery on Galois Counter Mode (GCM). Cryptology ePrint Archive, Report 2015/477, 2015. `http://eprint.iacr.org/2015/477`.
27. Antoine Joux. Authentication Failures in NIST version of GCM. *NIST Comment*, 2006.
28. Tadayoshi Kohno, John Viega, and Doug Whiting. CWC: A High-Performance Conventional Authenticated Encryption Mode. In *FSE*, pages 408–426, 2004.
29. Ted Krovetz and Phillip Rogaway. OCB. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
30. Eik List and Mridul Nandi. Revisiting Full-PRF-Secure PMAC and Using It for Beyond-Birthday Authenticated Encryption. In Helena Handschuh, editor, *CT-RSA*, volume 10147 of *LNCS*, pages 31–49. Springer, 2017.
31. Jiqiang Lu. On the Security of the COPA and Marble Authenticated Encryption Algorithms against (Almost) Universal Forgery Attack. *IACR Cryptology ePrint Archive*, 2015:79, 2015.
32. Stefan Lucks. A Failure-Friendly Design Principle for Hash Functions. In *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, pages 474–494, 2005.
33. David McGrew and John Viega. The Galois/Counter Mode of Operation (GCM). *Submission to NIST. http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf*, 2004.
34. David A. McGrew and Scott R. Fluhrer. Multiple forgery attacks against Message Authentication Codes. *IACR Cryptology ePrint Archive*, 2005:161, 2005.
35. David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In *INDOCRYPT*, pages 343–355. Springer, 2004.
36. Kazuhiko Minematsu. AES-OTR v3.1. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
37. Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering Generic Composition. In *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 257–274. Springer, 2014.
38. Mridul Nandi. Revisiting Security Claims of XLS and COPA. Cryptology ePrint Archive, Report 2015/444, 2015. `http://eprint.iacr.org/2015/444`.
39. Ivica Nikolić. Tiaoxin-346. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.

| Name & Class [37] | Class Sec. 3 | Name & Class [37] | Class Sec. 3 |
|---|---|---|---|
| A1, A1.100111 | (01001011) | A7, A3.100111 | (01001011) |
| A2, A1.110111 | (11001011) | A8, A3.110111 | (11001011) |
| A3, A1.101111 | (01101011) | N1, N1.111 | (11100000) |
| A4, A1.111111 | (11101011) | N2, N2.111 | (01000111) |
| A5, A2.100111 | (01000111) | N3, N3.111 | (01001011) |
| A6, A2.110111 | (11000111) | | |

**Table 4:** The eleven "favored" nAE schemes considered by the authors of [37] according to our classification.


40. Gordon Procter and Carlos Cid. On Weak Keys and Forgery Attacks against Polynomial-based MAC Schemes. In Shiho Moriai, editor, *FSE*, volume 8424 of *Lecture Notes in Computer Science - Lecture Notes in Computer Science*, pages 287–304. Springer, 2013.
41. P. Rogaway and D. Wagner. A Critique of CCM. Cryptology ePrint Archive, Report 2003/070, 2003. `http://eprint.iacr.org/2003/070`.
42. Phillip Rogaway. Authenticated-Encryption with Associated-Data. In *ACM Conference on Computer and Communications Security*, pages 98–107, 2002.
43. Phillip Rogaway. Nonce-Based Symmetric Encryption. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *LNCS*, pages 348–359. Springer, 2004.
44. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In *ACM Conference on Computer and Communications Security*, pages 196–205, 2001.
45. Markku-Juhani Olavi Saarinen. Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 216–225. Springer, 2012.
46. Raphael Bost; Olivier Sanders. Trick or Tweak: On the (In)security of OTR's Tweaks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT*, volume to appear of *LNCS*. Springer, 2016.
47. Zhelei Sun, Peng Wang, and Liting Zhang. Collision Attacks on Variant of OCB Mode and Its Series. In Miroslaw Kutylowski and Moti Yung, editors, *Inscrypt*, volume 7763 of *LNCS*, pages 216–224. Springer, 2012.
48. Hongjun Wu. ACORN: A Lightweight Authenticated Cipher (v3). `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
49. Hongjun Wu and Tao Huang. The Authenticated Cipher MORUS. `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
50. Hongjun Wu and Tao Huang. The JAMBU Lightweight Authentication Encryption Mode (v2.1). `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.
51. Hongjun Wu and Bart Preneel. AEGIS: A Fast Authenticated Encryption Algorithm (v1,1). `http://competitions.cr.yp.to/caesar-submissions.html`, 2016.


# A Classification of NRS'14 Schemes

This section shows the eleven "favored" nAE schemes considered by [37] and how we map them according to our classification. From Table 4, one can observe that the classes (A1, A7) and (A2, A8) have pairwise the same class according to our generic nAE scheme. That stems from the fact that we do not follow the distinction of nAE schemes from [37] regarding to whether the message/ciphertext can be processed in parallel or if the tag can be truncated. For the scheme N3, it holds that $\mathcal{E}$ gets the two separate inputs $F_L(A, N, M)$ and the nonce $N$. Since there is no segregated tag generation for N3 (the tag is part of the ciphertext), we interpreted $F_L$ as $F_{IV}$ and consider $F_{IV}$ to additionally hand over the nonce $N$ to the encryption $\mathcal{E}$ internally in plain.

| Scheme | NI | NR | Scheme | NI | NR |
|---|---|---|---|---|---|
| 3rd-Round CAESAR Candidates | | | | | |
| ACORN | – | $2^{\tau}$ | JAMBU | $2^{2n/2}$ | $2^{2n/2}$ |
| AEGIS | – | $2^{\tau}$ | KETJE | – | $2^{\min\{\tau,s\}}$ |
| AES-OTR | – | $2^{\tau/2}$ | KEYAK | $2^{\min\{c/2,\tau\}}$ | $2^{\min\{c/2,\tau\}}$ |
| AEZv4 | $2^{55}$ | $2^{55}$ | MORUS | – | $2^{128}$ |
| ASCON | – | $2^{\tau}$ | OCB | – | $2^{\tau}$ |
| CLOC | $2^{n/2}$ | $2^{n/2}$ | SILC | – | $2^{\tau/2}$ |
| COLM | $2^{64}$ | $2^{64}$ | NORX | – | $2^{\lvert\tau\rvert}$ |
| DEOXYS-I | – | $2^{\tau}$ | TIAOXIN | – | $2^{128}$ |
| DEOXYS-II | $2^{\tau/2}$ | $2^{\tau-1}$ | | | |
| Classical AE Schemes | | | | | |
| CCM | – | $2^{n/2}$ | CWC | – | $2^{n/2}$ |
| EAX | – | $2^{n/2}$ | GCM | – | $2^{n/2}$ |

**Table 5:** Claimed INT-CTXT bounds. **NR** = nonce-respecting adversary, **NI** = nonce-ignoring adversary, where $\tau$ denotes the length of the tag, $n$ the size of the internal state (usually the block size of the internally used block cipher), and $c$ the capacity for sponge-based designs.

| Scheme | Type | Attack | Time | Data | Memory | Succ. |
|---|---|---|---|---|---|---|
| AES-OTRv3.1 par. ADP | NR | AUF | $2^{n/2}$ | $2^{n/2}$ | $2^{n/2}$ | 1.0 |
| AES-OTRv3.1 ser. ADP | NR | AUF | $2^{n/2}$ | $2^{n/2}$ | $2^{n/2}$ | 1.0 |
| AES-OTRv3.1 ser. ADP | NI | AUF | $2^{n/2}$ | $2^{n/2}$ | $2^{n/2}$ | 1.0 |
| COLMv1 | NI | AUF | $2^{n/2}$ | $2^{n/2}$ | $2^{n/2}$ | 1.0 |
| OCBv3 | NR | AUF | $2^{n/2}$ | $2^{n/2}$ | $2^{n/2}$ | 1.0 |

**Table 6:** Stronger forgery attacks on AE schemes. **NR/NI** = nonce-respecting/nonce-ignoring; **AUF** = almost universal forgery; **Succ.** = success probability.

# B Security Claims

In Table 5, we state the security as claimed by the authors of the corresponding scheme. We denote by $\tau, n, c$, and $r$ the tag length, block length, capacity, and the rate, respectively.

# C Stronger Forgery Attacks

This section summarizes existing and new attacks on third-round CAESAR candidates and classical AE schemes that yield multiple forgeries. This can be induced by the recovery of e. g., a masking key or authentication key after a collision. While the complexity of the attacks are beyond the proved security bounds and therefore do not invalidate the according proofs, they can be considered as undesirable properties that should be avoided in recommendations by the community or even in future standards.

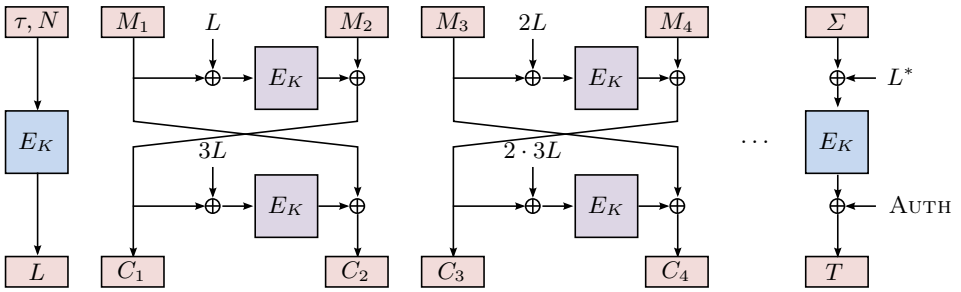


**Fig. 2:** Simplified schematic illustration of the encryption process in AES-OTRv3.1 (parallel). The final two message blocks $M_{m-1}$ and $M_m$ are treated differently; $\Sigma = \bigoplus_{i=1}^{m/2} M_{2i}$, $L^* = 2^{m-1} \cdot L$, and AUTH denotes the result of processing the associated data.

## C.1 OCB

Ferguson [17] showed collision attacks on OCBv1, which allowed to recover the masking key $L$ from a collision of the sums of input and output of two blocks $M_i \oplus C_i = M_j \oplus C_j$. Thereupon, the recovered $L$ allows the construction of many selective forgeries out of a single long message. To address the length restriction of messages in OCBv1, Ferguson also derived attacks from collisions among different messages, which also resulted in selective forgeries. The attacks by Ferguson still hold in similar form also for OCBv3, as pointed out by Sun, Wang, and Zhang [47]. In the following, we recall the details briefly.

For versions v1 and v3 of OCB, the designers used Gray codes for masking the block-cipher inputs. The mask for the $i$-th message/ciphertext block is given by $Z_i := \gamma_i \cdot L \oplus R$, where $L \leftarrow E_K(0^n)$, $R \leftarrow E_K(N \oplus L)$, and $\gamma_i$ represents the integer $i$ in the canonical Gray code. The multiplications are in $\mathbb{GF}(2^n)$ with primitive polynomial $\mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$.

1. Choose $A$ and $N$ arbitrarily, and choose a long query $M = (M_1, \ldots, M_m)$, such that for all $k \in \{1, \lfloor m/4 \rfloor\}$, it holds that $M_{4k} \oplus M_{4k+1} \oplus M_{4k+2} \oplus M_{4k+3} = 0^n$. Ask for its encryption $C = (C_1, \ldots, C_m)$ and $T$.
2. If it holds, for any pair $i, j \in \{1, \ldots, m\}$, $i \neq j$, that $M_i \oplus C_i = M_j \oplus C_j$, then it holds with probability 0.5 that this collision is the result of colliding cipher inputs. Then, we can recover $L$ by $L = (M_i \oplus M_j) \cdot (\gamma_i \oplus \gamma_j)^{-1}$.
3. For any index $d$, change $C'_k \leftarrow C_d \oplus (\gamma_i \oplus \gamma_k) \cdot L$ for $k = 4, \ldots, 7$. Leave other ciphertext blocks, $T$, $A$, and $N$ unchanged. The so-modified ciphertext $C'$ is still valid and will yield $M'_4 \oplus M'_5 \oplus M'_6 \oplus M'_7 = 0^n$, which also held for the original message. Hence, the tag $T$ remains valid also for the modified ciphertext.

## C.2 AES-OTR

Similar collision attacks as for OCB can be applied to AES-OTR. As a reaction to Bost and Sanders' [46] polynomial attacks on the v2 version of AES-OTR, Minematsu updated the tweak usage in v3 of AES-OTR to use the masking key $L$ from encrypting $N$. We show two attacks on AES-OTR v3.1 with birthday-bound complexity of $2^{n/2}$ that recover $L$: an attack with a single, long message, and an attack with multiple messages.

**Single-Message Attack.** The first attack works as follows:

1. Choose $A$ and $N$ arbitrarily, and choose a long query $M = (M_1, \ldots, M_m)$ for even $m$ such that all even blocks are equal, i.e. $M_2 = M_4 = \ldots = M_{2i}$ for all $i \in \{1, \ldots, m/2 - 1\}$, and random pair-wise distinct odd blocks $M_{2i-1}$. For simplicity, choose arbitrary full blocks $M_{m-1}$ and $M_m$. Ask for its encryption $C = (C_1, \ldots, C_m)$ and $T$.

2. If for any pair $i, j \in \{1, \ldots, m/2-1\}$, it holds that $C_{2i-1} = C_{2j-1}$, it follows from $M_{2i} = M_{2j}$ and from the fact that $E_K(\cdot)$ is a permutation that

$$M_{2i-1} \oplus 2^{i-1}L = M_{2j-1} \oplus 2^{j-1}L, \text{ and hence,}$$
$$L = (M_{2i-1} \oplus M_{2j-1}) \cdot (2^{i-1} \oplus 2^{j-1})^{-1}.$$

3. For any pair of indices $x, y \in \{1, \ldots, m/2-1\}$, derive $\Delta = 3 \cdot (2^{x-1} \oplus 2^{y-1}) \cdot L$ and $\nabla = (2^{x-1} \oplus 2^{y-1}) \cdot L$, and compute

$$C'_{2x-1} = C_{2y-1} \oplus \Delta, \qquad C'_{2x} = C_{2y} \oplus \nabla,$$
$$C'_{2y-1} = C_{2x-1} \oplus \Delta, \quad \text{and} \quad C'_{2y} = C_{2x} \oplus \nabla.$$

   Leave other ciphertext blocks, $T$, $A$, and $N$ unchanged. The so-modified ciphertext $C'$ is still valid and will yield $M'_{2x-1} = M_{2y-1} \oplus \nabla$, $M'_{2x} = M_{2y} \oplus \Delta$, $M'_{2y-1} = M_{2x-1} \oplus \nabla$, and $M'_{2y} = M_{2x} \oplus \Delta$.

Since the tag generation uses the sum of the even-indexed message blocks,

$$\Sigma = \bigoplus_{i=1}^{m/2} M_{2i},$$

it holds that $\Sigma' = \Sigma$ since $M'_{2x} \oplus M'_{2y} = M_{2y} \oplus M_{2x}$.

**Multi-Message Nonce-Respecting Attack.** A variant of Ferguson's collision attack with multiple messages on OCB may also be possible for AES-OTR; however, it would allow to recover the relation of $\Delta = 2^{i-1}L \oplus 2^{j-1}L$. For OCB, Ferguson could inject differences of $(4 \oplus 5 \oplus 6 \oplus 7)\Delta$ which cancels out in $\mathbb{GF}(2^n)$. Since AES-OTR employs doublings instead, this would mean, one would obtain $(2^i \oplus 2^j \oplus 2^k \ldots)\Delta$ when swapping double-blocks from between the colliding messages. Thus, Ferguson's collision attack seems not directly applicable to AES-OTR; at least, we could not find a straight-forward way to cancel values. However, we found two attacks with collision among different messages for the serial-ADP version of AES-OTR. Associated-data attack by Lu on serial ADP works, can recover $Q$ when $N$ is constant, which is allowed by Minematsu.
For the serial version, we can derive a multi-message nonce-respecting collision attack. For this version, the masking key is computed from $L \leftarrow (E_K(\tau, N) \oplus \textsc{Auth}) \cdot 2$. Leaving the (assumed constant) parameter $\tau$ aside, it is easy to see that two values of $L$ can collide at birthday bound.

1. Choose an integer $m \geq 4$, and fix arbitrary values $M, M_{m-1} \in \{0,1\}^n$.
2. For $i = 1..q$, choose pair-wise distinct random pairs of associated data and nonce $(A^i, N^i)$ such that the nonces $N^i$ are all distinct. Choose $M_{2j-1}$, for $j = 1, \ldots, m/2-1$ randomly. Ask for the authenticated encryption of $(A^i, N^i, M^i)$, with

$$M^i = (M_1^i, M, M_3^i, M, \ldots, M_{m-1}, M)$$

   and store $(A^i, N^i, C^i, T^i)$ as well as the odd-indexed blocks of $M^i$ in a table $\mathcal{L}$.
3. If, for any indices $i \neq j$, it holds that $C_m^i = C_m^j$, then it must follow from $M_{m-1}^i = M_{m-1}^j$ and $M_m^i = M_m^j$ that the masking keys for both messages $L^i$ and $L^j$ must be identical. It follows furthermore from the tag generation of AES-OTR that the tags of both messages $M^i$ and $M^j$ are identical.
4. Denote $t = (m/2) - 1$. Leaving the final double-block $(M_{m-1}^i, M_m^i)$ aside that served for detecting the collision of the masking keys, we have $t$ double blocks $(C_{2k-i}^i, C_{2k}^i)$ and $(C_{2k-i}^j, C_{2k}^j)$, for $k = 1, \ldots, m/2-1$, in our two involved ciphertexts that can be swapped across messages. Since both yield $M_{2k}^i = M_{2k}^j = M$ after decryption, the swaps do not change the tag. Assuming $M_{2k-1}^i \neq M_{2k-1}^j$ for all $k = 1, \ldots, m/2-1$, which holds with high probability, we can create in total $2 \cdot 2^t$ different authenticated ciphertexts by exhaustive combination of double blocks with either $(A^i, N^i)$ or $(A^j, N^j)$. Note that we already used two of those combinations for finding the collision.

**Multi-Message Nonce-Ignoring Attack.** Lu [31] published almost-universal forgery attacks on AES-COPA, and Marble. They can be also translated into nonce-ignoring attacks on AES-OTRv3.1. We consider the version with serial ADP since the specification claims that its "*security [...] holds as far as a pair of AD and nonce* $(A, N)$ *is unique for all encryption queries, for privacy and authenticity notions*". Lu proposed attacks with constant associated data, requiring at best about $2^{124}$ queries and time, at about $2^{120.6}$ bytes and a success probability of approximately 0.32. Moreover, he proposed an attack with $2^{65}$ queries and time. Both recover the masking key $L$. We transform the latter with birthday-bound complexity to an attack on AES-OTR that also recovers $L$.
The attack works as follows:

1. Fix any message $M$ and nonce $N$.
2. For $i = 1, \ldots, 2^{n/2}$, choose a single-block associated data $A^i$ of length $< n$ bit, s. t. all $A^i$ are pair-wise distinct. Ask for the encryption of $(A^i, N, M)$ to $C^i, T$ and store them into a table.
3. For $j = 1, \ldots 2^{n/2}$, choose a single-block associated data data $A'^j$ of length $n$ bit. If there exist $i, j$ with $A^i = A'^j$, then, we can recover the associated-data masking key $Q = E_K(0^n)$ from

$$A^i \oplus 2Q = A'^j \oplus 4Q \quad \text{and thus} \quad Q = (A^i \oplus A'^j) \cdot (2 \oplus 4)^{-1}.$$

   Though, it suffices to compute $A^i \oplus A'^j = 2Q \oplus 4Q$.
4. In the following, for each of the $2^{n/2}$ stored tuples $(A^i, N, C^i, T^i)$ with partial $A^i$, derive the padded $n$-bit value $A'^i = (A^i \,\|\, 10^*) \oplus (2Q \oplus 4Q)$. All ciphertexts $(A'^i, N, C^i, T^i)$ are valid forgeries.

## C.3 COLM

The multi-message nonce-ignoring attack by Lu can also be applied in similar form to COLMv1. Here, we can recover first the masking key $L$, which is also used for encryption and tag generation. Using the notation from the attack description on AES-OTR, it holds for COLMv1 that we obtain

$$A^i \oplus 3 \cdot 2 \cdot 7 \cdot L = A'^i \oplus 3 \cdot 2 \cdot L \quad \text{and thus} \quad L = (A^i \oplus A'^j) \cdot 7^{-1}.$$

For each of the $2^{n/2}$ stored tuples $(A^i, N, C^i, T^i)$ with partial $A^i$, derive the padded $n$-bit value $A'^i = (A^i \,\|\, 10^*) \oplus (3 \cdot 2 \cdot 7 \cdot L \oplus 3 \cdot 2 \cdot L)$. Again, all ciphertexts $(A'^i, N, C^i, T^i)$ are valid forgeries. However, the knowledge of $L$ allows almost universal forgeries.
There are various ways to obtain a vast amount of more forgeries. For instance, choose some $N'$, $M'$ and a long associated data $A' = (A_1, \ldots, A_a)$, with $A_1 = A_2 = \ldots = A_a$, such that $(N', A', M')$ has not been queried before. Ask for its corresponding encryption $(C', T')$. From $A_1 = A_2 = \ldots = A_a$ follows that the masked inputs to the block cipher, $AA_i \leftarrow A_i \oplus 2^i \cdot 3 \cdot L$, are pair-wise distinct. Since we know $L$, we can modify the blocks $A'_i$ to obtain a permutation of the $a$ values $AA_i$. Since there exist $a!$ such permutations, we obtain $a! - 1$ forgeries from a single encryption query.