Reforgeability of Authenticated Encryption Schemes

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

[1] Beuth Hochschule für Technik Berlin, Berlin, Germany
cforler@beuth-hochschule.de
[2] Bauhaus-Universitä Weimar, Weimar, Germany
{eik.list,stefan.lucks,jakob.wenzel}@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. The goal of authenticated encryption (AE) schemes is to simultaneously protect authenticity and privacy of messages. 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 [44], where the user must provide an additional nonce for every message it wants to process – a number used once (nonce). AE schemes that

J. Pieprzyk and S. Suriadi (Eds.): ACISP 2017, Part II, LNCS 10343, pp. 19–37, 2017.
DOI: 10.1007/978-3-319-59870-3_2

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 [45] 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 [35]. They introduced the measure of expected number of forgeries and conducted a thorough analysis of GCM [34], HMAC [6], and CBC-MAC [8]. In 2008, Handschuh and Preneel [22] 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 τ 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 τ 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^τ 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 [37], GCM [34], COLM [3], CWC [29], and OCB [30]), there already exist forgery attacks in the literature (see [19] for details) 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 [4,32,39], 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 [20,27,28,36,42,46] 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 approach to the wide-pipe mode introduced for hash functions [33], 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 [17,45]. 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 approaches to provide resistance in the sense of reforgeability and j-IV-CAs. Moreover, in the full version of this work [19], for AES-OTR, COLM, and OCB, we describe three attacks making multi-forgery attacks more efficient than our generic approach.

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.

Scheme	NI	NR	Scheme	NI	NR
3rd-round CAESAR candidates					
ACORN [47]	$t_1 + j$	$t_1 \cdot j/2$	Ketje [12]	$t_2 + j$	$t_2 \cdot j/2$
AEGIS [50]	$t_2 + j$	$t_2 \cdot j/2$	Keyak [21]	$t_2 + j$	$t_2 \cdot j/2$
AES-OTR [37]	t_1	t_1	MORUS [48]	$t_2 + j$	$t_2 \cdot j/2$
AEZv4 [23]	$t_1 + j$	$t_1 \cdot j/2$	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 [24]	$t_1 + j$	$t_1 \cdot j$	OCB [30]	t_1	t_1
COLM [3]	t_1	$t_1 + j$	SILC [24]	$t_1 + j$	$t_1 \cdot j$
Deoxys [26]	$t_1 + j$	$t_1 \cdot j$	Tiaoxin [40]	$t_2 + j$	$t_2 \cdot j/2$
JAMBU [49]	$t_1 + j$	$t_1 \cdot j/2$			
Classical schemes					
CWC [29]	t_1	t_1	CCM [16]	$t_1 + j$	$t_1 + j$
EAX [9]	$t_1 + j$	$t_1 \cdot j$	GCM [34]	t_1	t_1

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 and a generic security analysis. Section 5 contains possible remedies to j-IV-CAs and Sect. 6 concludes our work.

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 $X \xleftarrow{} \mathcal{X}$ 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 $\mathbf{Adv}_F^X(\mathbf{A})$ for the advantage $\mathbf{A}$ 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 $\mathbf{Adv}_F^X(q, \ell, t) = \max_{\mathbf{A}}\{\mathbf{Adv}_F^X(\mathbf{A})\}$ 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 ℓ 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, ℓ, 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 ℓ blocks in sum, where $\mathbf{A}$ runs in time at most t. We define a scheme Π 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, ℓ, t)-X-adversaries on Π is upper bounded by ϵ. 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) [43] 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 $\mathcal{E}_K^{A,N}(M)$ and $\mathcal{D}_K^{A,N}(C,T)$ 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, $\mathcal{D}_K^{A,N}(C,T)$ 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 $\mathcal{E}_K^{A,N}(M) = (C,T)$, then $|C| = |M|$ and $|T| = \tau$, *correctness*: if $\mathcal{E}_K^{A,N}(M) = (C,T)$, then $\mathcal{D}_K^{A,N}(C,T) = M$, and *tidiness*: if $\mathcal{D}_K^{A,N}(C,T) = M \neq \perp$, then $\mathcal{E}_K^{A,N}(M) = (C,T)$, 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

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

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 Π 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 Π 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 Π that ask at most q_E encryption queries, q_D decryption queries, which sum up to at most ℓ 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 [38], 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:

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

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 $M \in \mathcal{M}$).

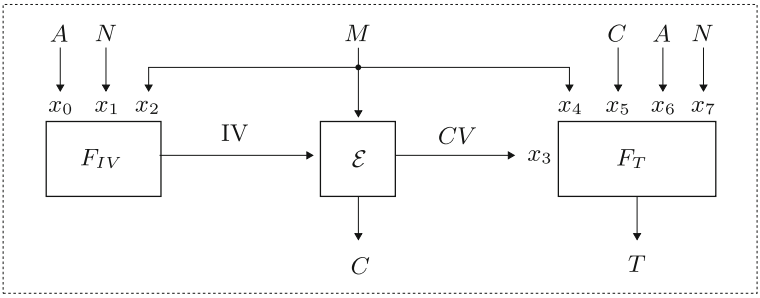

Fig. 1. Generic AE scheme as considered in our analysis.

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 [38] 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 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 Fig. 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 [24]. 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.,

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

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

$\{(01****00), (01****01), (10****00), (10****10), (00****01), (00****10)\}$. 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

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.

4.1 j-IV-Collision Attack

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 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:

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

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_ℓ) and receive (C'_ℓ, T'_ℓ)
13: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(*, *, M_\ell, *, *)\}$
14: Output the forgery $(A_k, N_k, C'_\ell, T'_\ell)$

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 $(A_i, N_i) \neq (A_k, N_k)$)[1] 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_ℓ, 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 4 of Appendix B, the reader can find the security claims of the considered AE schemes provided by their respective designers.

Nonce-Respecting Setting. The second setting prohibits an adversary from repeating any value N_i during its encryption queries. Therefore, we introduce a

[1] 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 Fig. 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

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_ℓ) for a fresh message M_ℓ to the encryption oracle and receives (C'_ℓ, T'_ℓ), where it is trivial to see that the pair (C'_ℓ, T'_ℓ) will also be valid

for (A_k, N_k, M_ℓ). **A** then only has to repeat this process for j pairwise distinct messages M_ℓ.

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 $\mathcal{C}_0$.

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 Fig. 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. Due to space constraints, the discussion of 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, is provided in Appendix C.

5 Countermeasures to j-IV-C Attacks

This section 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[2], 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*

[2] 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.

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 (A_i, N_i) would be processed to two independent n-bit values τ_1 and τ_2. Then, one could use τ_1 as initialization vector to the encryption step and τ_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 [31].

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.

A Classification of NRS'14 Schemes

This section shows the eleven "favored" nAE schemes considered by [38] and how we map them according to our classification. From Table 3, 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 [38] regarding to whether the message/ciphertext can be processed in parallel or if the tag can be truncated. For the scheme N3, it

Table 3. The eleven "favored" nAE schemes considered by the authors of [38] according to our classification.

Name & Class [38]	Class Sect. 3	Name & Class [38]	Class Sect. 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)		

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.

B Security Claims

In Table 4, we state the security as claimed by the authors of the corresponding scheme. We denote by τ, n, c, and r the tag length, block length, capacity, and the rate, respectively.

Table 4. Claimed INT-CTXT bounds. **NR** = nonce-respecting adversary, **NI** = nonce-ignoring adversary, where τ 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	NI	NR	Scheme	NI	NR
3rd-round CAESAR candidates					
ACORN	–	2^{τ}	JAMBU	$2^{2n/2}$	$2^{2n/2}$
AEGIS	–	2^{τ}	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^{τ}	OCB	–	2^{τ}
CLOC	$2^{n/2}$	$2^{n/2}$	SILC	–	$2^{\tau/2}$
COLM	2^{64}	2^{64}	NORX	–	$2^{\lvert\tau\rvert}$
DEOXYS-I	–	2^{τ}	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	–	$2^{n/2}$

C 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 5, we give an overview of the resistance the considered AE schemes to j-IV-CAs and we additionally provide a brief discussion for those cases that are not trivially observable. 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 the full version of this work [19].

Table 5. 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.

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)	○	○

AEGIS, MORUS, and Tiaoxin. These schemes provide semi-resistance 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. [25] 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 [1,41,46] 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 the full version of this work [19].

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. The nonce-requiring variant of Deoxys, i.e., Deoxys-I, 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 Hash(A) with the encrypted checksum under the nonce as tweak, the adversary sees the difference in the hash outputs in the tags: Hash$(A) \oplus$ 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 shown in [19]; 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.

References

1. Abdelraheem, M.A., Beelen, P., Bogdanov, A., Tischhauser, E.: Twisted polynomials and forgery attacks on GCM. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 762–786. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46800-5_29
2. Abed, F., Fluhrer, S., Foley, J., Forler, C., List, E., Lucks, S., McGrew, D., Wenzel, J.: The POET Family of On-Line Authenticated Encryption Schemes (2014). http://competitions.cr.yp.to/caesar-submissions.html
3. Andreeva, E., Bogdanov, A., Datta, N., Luykx, A., Mennink, B., Nandi, M., Tischhauser, E., Yasuda, K.: COLM v1 (2016). http://competitions.cr.yp.to/caesar-submissions.html
4. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: AES-COPA (2014). http://competitions.cr.yp.to/caesar-submissions.html
5. Aumasson, J.-P., Jovanovic, P., Neves, S.: NORX (2016). http://competitions.cr.yp.to/caesar-submissions.html
6. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996). doi:10.1007/3-540-68697-5_1
7. Bellare, M., Goldreich, O., Mityagin, A.: The power of verification queries in message authentication and authenticated encryption. IACR Cryptology ePrint Arch. **2004**, 309 (2004)
8. Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. **61**(3), 362–399 (2000)
9. Bellare, M., Rogaway, P., Wagner, D.: The EAX mode of operation. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 389–407. Springer, Heidelberg (2004). doi:10.1007/978-3-540-25937-4_25
10. Bernstein, D.J.: CAESAR Call for Submissions, Final, 27 January 2014. http://competitions.cr.yp.to/caesar-call.html
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. ECRYPT Hash Function Workshop (2007)
12. Bertoni, G., Daemen, J., Peeters, M., Van Keer, R., Van Assche, G.: CAESAR submission, Ketje v2 (2016). http://competitions.cr.yp.to/caesar-submissions.html
13. Black, J., Cochran, M.: MAC reforgeability. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 345–362. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03317-9_21
14. Datta, N., Nandi, M.: ELmD (2014). http://competitions.cr.yp.to/caesar-submissions.html
15. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2 (2016). http://competitions.cr.yp.to/caesar-submissions.html
16. Dworkin, M.J.: 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. Ferguson, N.: Collision Attacks on OCB. Unpublished manuscript (2002). http://www.cs.ucdavis.edu/rogaway/ocb/links.htm
18. Ferguson, N.: Authentication weaknesses in GCM (2005). http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf
19. Forler, C., List, E., Lucks, S., Wenzel, J.: Reforgeability of Authenticated Encryption Schemes. Cryptology ePrint Archive, Report 2017/332 (2017). http://eprint.iacr.org/2017/332

20. Fouque, P.-A., Martinet, G., Valette, F., Zimmer, S.: On the security of the CCM encryption mode and of a slight variant. In: Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 411–428. Springer, Heidelberg (2008). doi:10.1007/978-3-540-68914-0_25
21. Peeters, M., Bertoni, G., Daemen, J., Van Assche, G., Van Keer, R.: CAESAR submission, Keyak v2 (2016). http://competitions.cr.yp.to/caesar-submissions.html
22. Handschuh, H., Preneel, B.: Key-recovery attacks on universal hash function based MAC algorithms. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 144–161. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85174-5_9
23. Hoang, V.T., Krovetz, T., Rogaway, P.: AEZ v4.2: Authenticated Encryption by Enciphering (2016). http://competitions.cr.yp.to/caesar-submissions.html
24. Iwata, T., Minematsu, K., Guo, J., Morioka, S.: CLOC and SILC v3 (2016). http://competitions.cr.yp.to/caesar-submissions.html
25. Iwata, T., Ohashi, K., Minematsu, K.: Breaking and repairing GCM security proofs. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 31–49. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_3
26. Jean, J., Nikolić, I., Peyrin, T., Seurin, Y.: Deoxys v1.41 (2016). http://competitions.cr.yp.to/caesar-submissions.html
27. Westerlund, M., Mattsson, J.: Authentication Key Recovery on Galois Counter Mode (GCM). Cryptology ePrint Archive, Report 2015/477 (2015). http://eprint.iacr.org/2015/477
28. Joux, A.: Authentication Failures in NIST version of GCM. NIST Comment (2006)
29. Kohno, T., Viega, J., Whiting, D.: CWC: a high-performance conventional authenticated encryption mode. In: FSE, pp. 408–426, 2004
30. Krovetz, T., Rogaway, P.: OCB (2016). http://competitions.cr.yp.to/caesar-submissions.html
31. List, E., Nandi, M.: Revisiting full-PRF-secure PMAC and using it for beyond-birthday authenticated encryption. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 258–274. Springer, Cham (2017). doi:10.1007/978-3-319-52153-4_15
32. Jiqiang, L.: On the security of the COPA and marble authenticated encryption algorithms against (almost) universal forgery attack. IACR Cryptology ePrint Arch. **2015**, 79 (2015)
33. Lucks, S.: A failure-friendly design principle for hash functions. In: Proceedings of the Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4–8, 2005, pp. 474–494 (2005)
34. McGrew, D., Viega, J.: The Galois/Counter Mode of Operation (GCM). Submission to NIST (2004). http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf
35. McGrew, D.A., Fluhrer, S.R.: Multiple forgery attacks against message authentication codes. IACR Cryptology ePrint Arch. **2005**, 161 (2005)
36. McGrew, D.A., Viega, J.: The security and performance of the Galois/Counter Mode (GCM) of operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30556-9_27
37. Minematsu, K.: AES-OTR v3.1 (2016). http://competitions.cr.yp.to/caesar-submissions.html
38. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering generic composition. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 257–274. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_15

39. Nandi, M.: Revisiting security claims of XLS and COPA. Cryptology ePrint Archive, Report 2015/444 (2015). http://eprint.iacr.org/2015/444
40. Nikolić, I.: Tiaoxin-346 (2016). http://competitions.cr.yp.to/caesar-submissions.html
41. Procter, G., Cid, C.: On weak keys and forgery attacks against polynomial-based MAC schemes. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 287–304. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43933-3_15
42. Rogaway, P., Wagner, D.: A Critique of CCM. Cryptology ePrint Archive, Report 2003/070 (2003). http://eprint.iacr.org/2003/070
43. Rogaway, P.: Authenticated-encryption with associated-data. In: ACM Conference on Computer and Communications Security, pp. 98–107 (2002)
44. Rogaway, P.: Nonce-based symmetric encryption. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 348–358. Springer, Heidelberg (2004). doi:10.1007/978-3-540-25937-4_22
45. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: ACM Conference on Computer and Communications Security, pp. 196–205 (2001)
46. Saarinen, M.-J.O.: Cycling attacks on GCM, GHASH and other polynomial MACs and hashes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 216–225. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34047-5_13
47. Hongjun, W.: A Lightweight Authenticated Cipher (v3) (2016). http://competitions.cr.yp.to/caesar-submissions.html
48. Wu, H., Huang, T.: The Authenticated Cipher MORUS (2016). http://competitions.cr.yp.to/caesar-submissions.html
49. Wu, H., Huang, T.: The JAMBU Lightweight Authentication Encryption Mode (v2.1) (2016). http://competitions.cr.yp.to/caesar-submissions.html
50. Wu, H., Preneel, B.: AEGIS: A Fast Authenticated Encryption Algorithm (v1,1) (2016). http://competitions.cr.yp.to/caesar-submissions.html