An extended abstract of this article appears in IACR ToSC 2020 (1). It was assigned DOI 10.13154/tosc.v2020.i1.121-143 [PR20]. Our results will further be presented at FSE 2020. This is the article's full version. It is available as entry 2020/232 in the IACR eprint archive.

Combiners for AEAD

Bertram Poettering[1] and Paul Rösler[2]

[1] IBM Research – Zurich, Switzerland
poe@zurich.ibm.com

[2] Chair for Network and Data Security, Ruhr University Bochum, Germany
paul.roesler@rub.de

Abstract. The Authenticated Encryption with Associated Data (AEAD) primitive, which integrates confidentiality and integrity services under a single roof, found wide-spread adoption in industry and became indispensable in practical protocol design. Recognizing this, academic research put forward a large number of candidate constructions, many of which come with provable security guarantees. Nevertheless, the recent past has shaken up with the discovery of vulnerabilities, some of them fatal, in well-regarded schemes, stemming from weak underlying primitives, flawed security arguments, implementation-level vulnerabilities, and so on. Simply reacting to such findings by replacing broken candidates by better(?) ones is in many cases unduly, costly, and sometimes just impossible. On the other hand, as attack techniques and opportunities change over time, it seems venturous to propose any specific scheme if the intended lifetime of its application is, say, twenty years.

In this work we study a workable approach towards increasing the resilience against unforeseen breaks of AEAD primitives. Precisely, we consider the ability to combine two AEAD schemes into one such that the resulting AEAD scheme is secure as long as at least one of its components is (or: as long as at most one component is broken). We propose a series of such combiners, some of which work with fully generic AEAD components while others assume specific internal structures of the latter (like an encrypt-then-MAC design). We complement our results by proving the optimality of our constructions by showing the impossibility of combiners that get along with less invocations of the component algorithms.

Keywords: Secure Combiners · Provable Security · AEAD · Encrypt-then-MAC · Ciphertext Translation · Impossibility Result

1 Introduction

AEAD. Authenticated Encryption with Associated Data (AEAD, [Rog02, McG08]) is a cryptographic primitive that gained more and more importance in the recent years. It consists of two symmetrically keyed algorithms: an encryption algorithm that is invoked by a sender and transforms a message into a ciphertext, and a decryption algorithm that is invoked by a receiver and transforms the ciphertext back to the message. The two algorithms further depend on an additional input referred to as associated data into which the sender and receiver can encode the context in which they perform their operations; equality of these contexts (i.e., associated data strings) is assumed for successful message recovery. The security typically expected of an AEAD scheme covers both confidentiality and authenticity.

Combiners. The idea of cryptographic combiners is to introduce redundancy in security by combining multiple constructions of the same primitive into one, to prepare for the case that one or more, but not all, of these building blocks turn out to be insecure. As long as at least one component stays secure, the combined primitive does as well.

Combiners have been studied for multiple cryptographic primitives, including one-way functions, hash functions, message authentication codes, signatures, symmetric encryption schemes, key encapsulation mechanisms, indistinguishability obfuscation [MH81, Her05, HKN+05, DK05, FL07, FL08, FLP08, FHNS16, GHP18]. Some of these works are more on the theoretical side by considering puristic feasibility

results, while others aim at proposing solutions to a practical problem. For instance, while the KEM combiners of [GHP18] employ hash functions and PRFs as additional building blocks, which is in line with the practical use cases envisioned by the authors, Dodis and Katz [DK05] consider a more pure form of PKE combiner that does not add more hardness assumptions but gets along with adding only information-theoretic ingredients. While the focus of our work is on practical AEAD combiners, we achieve our goals without adding more primitives or assumptions.

1.1 Motivation for AEAD Combiners

Research in symmetric cryptography succeeded with proposing a plethora of AEAD schemes. While many of these come with formal security arguments, there remain a number of reasons why combining two AEAD schemes into one to strengthen their security might still be attractive. The arguments span over various domains and are connected to flawed security arguments, implementation issues, management practices, or simply politics.

Flawed Security Arguments. Even if a cryptographic scheme comes with a "security proof", the unfortunate truth is that the corresponding arguments might be flawed and the scheme weak. This also holds for AEAD designs, where it seems fair to say that the high level of integration that the most efficient blockcipher-based schemes feature also makes the schemes challenging to fully comprehend and surround on a formal level. A prime example of where the security argument of an intricate AEAD scheme turned out to be fatally flawed, and this went unnoticed for a rather long period of time, is given by the recent results on OCB2 that completely broke the scheme [IIMP19]. There are several similar cases to report on, including the flawed EAX' scheme [MLMI14], a flawed argument in the security proof of GCM [IOM12], and flaws in some of the CAESAR submissions [Nan14, BS16, SMAP16].

Untested Assumptions. Schemes might rely on relatively new and untested building blocks. For instance, a majority of the many AEAD submissions to the recent CAESAR[1] and NIST Lightweight Cryptography[2] competitions use ad-hoc constructions that are at best only loosely related to established ciphers like plain AES. While this does not at all serve as an indication that the modes would be insecure, one could still argue that the assumptions are too fresh to be fully relied upon.

Implementation-Based Attacks. Even if a construction comes with a correct formal security argument, securely implementing it might be tricky. For instance, if components of the primitive are conveniently implemented via look-up tables, cache timing side channels are hard to evade. See [Ber05] for classic practical attacks against AES implementations, and the suggestion in [KS09] for side-channel attacks against table-based implementations of GCM's field arithmetic.

Standards. The use of a particular AEAD scheme might be required to achieve standard conformance, or to obtain a positive certification by government bodies. This might be the case even if the AEAD scheme is known to be weak, or at least its security is questionable. Examples may include cases in the payment industry where the use of 3DES still seems to be fashionable even though its blocksize is too small to achieve security in many applications [BL16], or where the use of ISO standardized AEAD schemes like the broken OCB2 are required, etc. An AEAD combiner allows using weak yet mandated schemes without risking security.

Political Issues. The Snowden revelations of 2013 suggested that state agencies might engage in subverting cryptographic primitives. While the best and most clear example remains to be the Dual-EC-DRBG incident[3], some actors actively avoid other NSA approved cryptosystems as well. For instance, instant messenger vendor SilentCircle switched from using the AES blockcipher to TwoFish[4], and the Simon and Speck blockciphers were not adopted by the ISO for standardization[5].[6] Not all of these decisions are purely justifiable on an academic level, but seem partially also political in nature. The use of an AEAD combiner might contribute to satisfying the politically induced requirements of some people.

[1] https://competitions.cr.yp.to/caesar.html
[2] https://csrc.nist.gov/projects/lightweight-cryptography
[3] https://en.wikipedia.org/wiki/Dual_EC_DRBG
[4] https://silentcircle.wordpress.com/2013/09/30/nncs/
[5] https://mobile.reuters.com/article/amp/idUSKCN1BW0GV
[6] That also symmetric schemes can be subverted is illustrated in works like [BPR14, AP19a, AP19b].

To conclude: For many popular encryption schemes, more or less compelling arguments can be found for not using them. It seems to be a matter of personal preference which of these schemes to trust or distrust. Using an AEAD combiner might help finding a common denominator among communication participants. In any case, as we argue next, it is essential that such combiners fulfill two important properties: simplicity and efficiency.

We advocate that AEAD combiners be simple. A complex combiner might lead to incomprehensible security analyses, and might be hard to implement correctly, and thus potentially introduce new security risks. In such a case it is conceivable that the combination of two secure AEAD schemes is actually insecure, with the combiner being responsible for the problems.

We further advocate that AEAD combiners be efficient. Indeed, AEAD schemes are used in almost all modern protocols, protecting billions of communications every day, and top performance is therefore a top priority. Of course a combined AEAD scheme is less efficient than its components; but the overhead should be reduced to the absolute minimum, ideally none.

All our combiners have a very clear design, come with rigorous, yet easy to verify proofs, do not require any exotic hardness assumptions of their components, and are highly efficient (in some cases we even claim optimality).

1.2 Approach and Results

In the course of this article we expose a number of AEAD combiners. While those of Section 3 are generic in the sense that they do not rely on any property of the component AEAD schemes that goes beyond their blackbox properties (i.e., their syntax, correctness, and security), those of Section 4 are non-generic and require a specific internal structure of at least one of their components (e.g., that it is induced by an encrypt-then-MAC design). All our combiners take two AEAD schemes and achieve security if (at least) one of them is secure. The resulting scheme can, of course, be combined in the same way with a third AEAD scheme, a fourth, and so on.[7] Conveniently, our non-generic combiners *preserve* the non-blackbox properties they require of their components. That is, also these combiners are smoothly amenable to a scaling process to more input schemes.

One efficiency metric for combiners is the number of internal invocations of their AEAD components. The intuitive minimum is likely four, suggested by each combined encryption operation requiring one internal invocation of each component's encryption algorithm, and each combined decryption operation requiring one internal invocation of each component's decryption algorithm. In Section 5 we present a result that shows that for generic (blackbox) AEAD schemes this minimum cannot be reached, but that at least five invocations are necessary.[8] Five internal invocations are also sufficient: Our generic combiners are optimal in this sense, requiring two internal encryptions for a combined encryption, and two internal decryptions plus one internal encryption for a combined decryption. Also our non-generic combiners get along with five internal invocations. They outperform our generic combiners nevertheless, but according to different metrics. For instance, observing that the inner building blocks of AEAD schemes have often dedicated functions, like providing 'passive confidentiality' and 'strong authentication' in an encrypt-then-MAC design, we can employ the right building blocks for the right tasks (e.g., save on encryption if confidentiality is not a matter), ultimately making the combiner faster. Further, our non-generic combiners achieve the shortest ciphertext expansions (which is constant for all our combiners anyway).

In Table 1 we compare our various combiners with respect to their performance and the assumptions posed on the underlying schemes. Observe that no combiner outperforms all others in all categories. Thus, which combiner to pick depends on the specific use case.

The intuitive minimum for the 'Processed Data' column is $2A+2M$ $2A+2M$ (as each component should process each associated data and each message at least once). Our impossibility result from Section 5 shows that this minimum is not attainable. However the table clarifies that the combiners we propose are quite close to it. Note that the shortest ciphertext length to be expected is $M + \tau$ (the message itself, plus a tag), and that one of the combiners meets this efficiency goal. (And again, the other combiners get quite close.)

[7] Note that the overall combiner's performance may depend on the order of combinations.

[8] This result can be evaded by requiring additional (non-blackbox) properties of the component schemes. For instance, our discussion at the end of Section 3 appreciates that many practical schemes feature a specific property, *tidiness*, that allows for combiners that require just four invocations.

Table 1: AEAD combiners compared with respect to assumptions on their underlying component schemes and their overall performance. Column 'Structure' describes the required construction principle of underlying scheme AE_i (blackbox, encrypt-then-MAC, ciphertext translated), 'Required Security' lists the underlying schemes' security guarantees (such that either of the two cells in a row must be met for a secure combination), 'Processed Data' adds up necessary processing of data ('A' is a unit for the length of processed associated-data strings, and 'M' is a unit for the length of encrypted messages) per combined encryption and decryption, and 'Ctxt. Length' adds up the combined ciphertexts' length (depending on message length and tag length; we use symbol τ as a placeholder for the common tag length of all AEAD schemes).

	Structure		**Required Security**		**Processed Data**		**Ctxt.**
	AE_0	AE_1	AE_0 has:	or AE_1 has:	enc	dec	**Length**
Blackbox 1	BB	BB	INT, IND	INT, IND	$2A+2M$	$3A+3M$	$M+2\tau$
Enc-then-MAC 1	EtM	BB	SUF, pIND	INT, IND	$2A+3M$	$2A+3M$	$M+2\tau$
Enc-then-MAC 2	EtM	EtM	SUF, pIND	SUF, pIND	$2A+4M$	$2A+4M$	$M+\tau$
Ciphertext transl.	BB	CT	INT, IND	INT, IND	$2A+2M$	$2A+3M$	$M+2\tau$
Blackbox 2	BB	BB	INT, IND	INT, IND	$2A+3M$	$2A+3M$	$\geq M+3\tau$
Blackbox 3	BB	BB	IND$	IND$	$2A+4M$	$2A+4M$	$M+3\tau$
Impossible	BB	BB	INT, IND	INT, IND	$\Sigma \leq 4A+4M$		

1.3 Related Work

The literature on combiners is as broad as the literature on cryptography in general since, theoretically, combiners can be studied for any cryptographic primitive. We concentrate here on work that focuses on combining symmetric encryption or message authentication codes (since both primitives are close to AEAD).

The initial motivation for combining schemes was not to rely on either of the underlying schemes' security but to increase the (assumed) security of one specific scheme by applying it multiple times. Merkle and Hellman [MH81] analyze the security of the Data Encryption Standard (DES) when being iteratively applied on some input, and specifically compare two versus three iterations of DES.

Today's motivation for combining schemes is to maintain security if at least one of them is secure. In this setting, Herzberg [Her05] considers different encryption and MAC combiners. He shows for randomized schemes that neither nested encryption nor certain parallel variants of encryption (where ciphertexts of the underlying schemes are encrypted iteratively or in parallel) can achieve strong confidentiality (IND-CCA security) if only one underlying scheme reaches this property. In contrast, he shows that certain nested and parallel combiners of MACs and signatures inherit unforgeability (EUF-CMA) guarantees of an underlying scheme. We note that [Her05] does not consider *strong* unforgeability (SUF), which is necessary for AEAD.

Zhang et al. [ZHSI04], Dodis and Katz [DK05], Giacon et al. [GHP18], as well as Bindel et al. [BBF+19] consider combiners for public key encryption and key encapsulation mechanisms. Their results include combiners that achieve IND-CCA security from either of the underlying schemes. Harnik et al. [HKN+05] propose combiners for one-way functions, oblivious transfer, public key encryption, as well as key exchange. In a line of work Fischlin and Lehmann [FL07, FL08], also together with Pietrzak [FLP08], analyze and propose combiners for hash functions. Finally, Fischlin et al. [FHNS16] consider combiners for obfuscation.

2 Preliminaries

We introduce general notations and the syntax and security of AEAD and MAC schemes.

NOTATION. We denote deterministic assignments with '$\leftarrow$', probabilistic assignments with '$\leftarrow_{\$}$', definitional assignments with ':='. If variables A, B represent sets we write '$A \xleftarrow{\cup} B$' shorthand for '$A \leftarrow A \cup B$'. We denote with $\oplus$ the bit-wise XORing operation of two bit-strings; if the strings have different lengths, the shorter one is padded with $\mathtt{0}$ bits at the end before the operation is performed.

With '$a \parallel b \parallel \ldots$' we denote the classic concatenation of strings $a, b, \ldots$ (into a single string), whereas with '$a \shortparallel b \shortparallel \ldots$' we denote an *injective* encoding of $a, b, \ldots$ into a string such that $a, b, \ldots$ can be

unambiguously recovered from the result.

We label algorithm specifications with '**Proc**', security games with '**Game**', and oracles that can be queried in games with '**Oracle**'. Security games are terminated via certain instructions: 'Reward x' means that the game terminates with return value 1 if condition x is met (otherwise execution continues), 'Require x' means that the game terminates with value 0 if condition x is not met (otherwise execution continues), and 'Stop with b' means that the game terminates with return value b. These terms convey an intuitive meaning, e.g., for rewarding an adversary if a winning condition is reached, or penalizing an adversary that does not follow the game rules. For a game G that outputs a Boolean value when run with an adversary $\mathcal{A}$, we write $\Pr[\mathrm{G}(\mathcal{A})]$ for the probability that the game outputs 1.

AEAD. A scheme for providing (nonce-based) authenticated encryption with associated data (AEAD) consists of deterministic algorithms $\mathrm{enc}, \mathrm{dec}$ and associated spaces $\mathcal{K}, \mathcal{N}, \mathcal{AD}, \mathcal{M}, \mathcal{C}$. The encryption algorithm enc takes a key $k \in \mathcal{K}$, a nonce $n \in \mathcal{N}$, an associated-data string $ad \in \mathcal{AD}$, and a message $m \in \mathcal{M}$, and returns a ciphertext $c \in \mathcal{C}$. The decryption algorithm dec takes a key $k \in \mathcal{K}$, a nonce $n \in \mathcal{N}$, an associated-data string $ad \in \mathcal{AD}$, and a ciphertext $c \in \mathcal{C}$, and returns either a message $m \in \mathcal{M}$ or the rejection symbol $\perp \notin \mathcal{M}$. A shortcut notation for this syntax is

$$\mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathrm{enc} \to \mathcal{C} \qquad \text{and} \qquad \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \mathrm{dec} \to \mathcal{M} \cup \{\perp\}.$$

For correctness we require of an AEAD scheme that for all $k \in \mathcal{K}$, $n \in \mathcal{N}$, $ad \in \mathcal{AD}$, $m \in \mathcal{M}$, and $c \in \mathcal{C}$, we have that $\mathrm{enc}(k, n, ad, m) = c$ implies that $\mathrm{dec}(k, n, ad, c) = m$.

Practical AEAD schemes have spaces $\mathcal{K}, \mathcal{N}, \mathcal{AD}, \mathcal{M}, \mathcal{C}$ such that $\mathcal{K} = \{0,1\}^\kappa$ for some $\kappa \geq 80$, $\{0,1\}^{96} \subseteq \mathcal{N} \subseteq \{0,1\}^*$, and $\mathcal{AD} = \mathcal{M} = \mathcal{C} = \{0,1\}^*$,[9] and they have *constant expansion*, i.e., all ciphertexts are a constant number of bits longer than the messages they encode. Formally, the latter property is given if there exists a number $\tau \in \mathbb{N}$ such that for any $(k, n, ad, m) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M}$ we have that $|\mathrm{enc}(k, n, ad, m)| = |m| + \tau$. We assume for all AEAD schemes considered in this paper that they are of this type.[10]

We formalize three standard security requirements: That the integrity of ciphertexts be protected (INT-CTXT), that encryptions be indistinguishable in the presence of chosen-ciphertext attacks (IND-CCA), and that encryptions be indistinguishable in the presence of (passive) chosen-plaintext attacks (IND-CPA). These notions are defined via the INT and IND^b games in Figure 1, and the games pIND^b which are like IND^b but with the decryption oracle Dec removed. Intuitively, a scheme provides *integrity* if the maximum advantage $\mathbf{Adv}^{\mathrm{int}}(\mathcal{A}) := \Pr[\mathrm{INT}(\mathcal{A})]$ that can be attained by realistic adversaries $\mathcal{A}$ is negligible, it provides *indistinguishability* if the same holds for the advantage $\mathbf{Adv}^{\mathrm{ind}}(\mathcal{A}) := |\Pr[\mathrm{IND}^1(\mathcal{A})] - \Pr[\mathrm{IND}^0(\mathcal{A})]|$, and it provides *indistinguishability against passive adversaries* if the same holds for the advantage $\mathbf{Adv}^{\mathrm{ind}}(\mathcal{A}) := |\Pr[\mathrm{pIND}^1(\mathcal{A})] - \Pr[\mathrm{pIND}^0(\mathcal{A})]|$.[11]

Message Authentication Codes. A message authentication code (MAC) for message space $\mathcal{M}$ consists of a key space $\mathcal{K}$ and a tag space $\mathcal{C}$, and a deterministic algorithm M that processes a key and a message into a tag:

$$\mathcal{K} \times \mathcal{M} \to \mathrm{M} \to \mathcal{C}.$$

We formalize the standard security requirement that a MAC scheme provide authenticity. This is defined via the SUF game in Figure 1. We say that a scheme provides *strong unforgeability* if the maximum advantage $\mathbf{Adv}^{\mathrm{suf}}(\mathcal{A}) := \Pr[\mathrm{SUF}(\mathcal{A})]$ that can be attained by realistic adversaries $\mathcal{A}$ is negligible.

3 Blackbox Combiners for AEAD

We present an AEAD combiner that is fully black-box, meaning that it works generically for any two component AEADs. Clearly, any such combiner invokes each component scheme at least once: The combined encryption will invoke each component's encryption algorithm, and the combined decryption will invoke each component's decryption algorithm. We refer to such combiners as *two-enc-two-dec* combiners. The combiner we present here requires more invocations, as its decryption algorithm makes

[9] Among other standards, these parameter sets are mandated by RFC 5116 [McG08].

[10] This is a mild assumption: We are not aware of *any* practically relevant AEAD scheme that has non-constant expansion.

[11] Our approach deliberately does not formalize of the intuitive meanings of "realistic" and "negligible". We formally define only the security games and adversary advantages.

Game INT($\mathcal{A}$)	**Oracle** Enc(n, ad, m)	**Oracle** Dec(n, ad, c)
00 $k \leftarrow_{\$} \mathcal{K}$	05 Require $n \notin$ N	10 $m \leftarrow \mathrm{dec}(k, n, ad, c)$
01 Q $\leftarrow \emptyset$	06 $c \leftarrow \mathrm{enc}(k, n, ad, m)$	11 If $m = \perp$: Return $\perp$
02 N $\leftarrow \emptyset$	07 Q $\xleftarrow{\cup} \{(n, ad, c)\}$	12 Reward $(n, ad, c) \notin$ Q
03 Invoke $\mathcal{A}$	08 N $\xleftarrow{\cup} \{n\}$	13 Return m
04 Stop with 0	09 Return c	
Game IND^b($\mathcal{A}$)	**Oracle** Enc(n, ad, m^0, m^1)	**Oracle** Dec(n, ad, c)
14 $k \leftarrow_{\$} \mathcal{K}$	19 Require $n \notin$ N	25 $m \leftarrow \mathrm{dec}(k, n, ad, c)$
15 Q $\leftarrow \emptyset$	20 Require $\lvert m^0 \rvert = \lvert m^1 \rvert$	26 If $m = \perp$: Return $\perp$
16 N $\leftarrow \emptyset$	21 $c \leftarrow \mathrm{enc}(k, n, ad, m^b)$	27 If $(n, ad, c) \in$ Q:
17 $b' \leftarrow \mathcal{A}$	22 Q $\xleftarrow{\cup} \{(n, ad, c)\}$	28 $m \leftarrow \diamond$
18 Stop with b'	23 N $\xleftarrow{\cup} \{n\}$	29 Return m
	24 Return c	
Game SUF($\mathcal{A}$)	**Oracle** Tag(m)	**Oracle** Vfy(m, t)
30 $k \leftarrow_{\$} \mathcal{K}$	34 $t \leftarrow \mathrm{M}(k, m)$	37 $t' \leftarrow \mathrm{M}(k, m)$
31 Q $\leftarrow \emptyset$	35 Q $\xleftarrow{\cup} \{(m, t)\}$	38 If $t' \neq t$: Return 0
32 Invoke $\mathcal{A}$	36 Return t	39 Reward $(m, t) \notin$ Q
33 Stop with 0		40 Return 1

Figure 1: AEAD games INT, IND^0, IND^1 and MAC game SUF. The condition $n \notin$ N of lines 05, 19 checks for nonce freshness. The symbol '$\diamond$' of line 28 is used to suppress sharing the value of m with the adversary.

three invocations of component algorithms. This is only seemingly suboptimal, as in Section 5 we prove that generically secure two-enc-two-dec combiners actually do not exist. Indeed, the three-invocation combiner from the current section can be seen as tightly complementing the impossibility result.

Actually, it may even be surprising that AEAD combiners that leverage on the security of just one component, while requiring effectively nothing of the other, can exist in the first place. Consider that, during encryption, one component will be invoked first and transform some message into some ciphertext, and then the second component will be invoked on input a message that may depend on the first ciphertext, and output its own ciphertext; the combined ciphertext will be a function of the former two and is required to be INT-secure, i.e., none of its parts may be malleable. While this intuitively requires that the two components protect their ciphertexts *mutually* (the one scheme protects the other's ciphertext, and vice versa), in practice one component has to be invoked last, so its output cannot be integrity protected by the other.

Our combiner uses an intriguing technique to side-step this seeming contradiction. Its code is specified in Figure 2. (We silently assume sufficiently compatible nonce spaces, associated data spaces, message spaces, and ciphertext spaces.) The combined encryption is simply a nested encryption of the two component algorithms, using the same nonce and associated data for both (lines 01,02). Note that this is already sufficient to achieve confidentiality (against passive adversaries), if at least one of the components provides confidentiality. The combined decryption first recovers the message by reversing the nesting (lines 05–08). The integrity protection of the outer ciphertext (i.e., the component ciphertext not protected by the other component) is accomplished by re-computing and confirming it (lines 09–10). Note that this approach only works because AEAD encryption is deterministic. (We generalize this re-encryption technique in Section 4.3, where we show that it can be used to turn every INT-PTXT secure scheme into an INT-CTXT secure one.)

We show that if either of the two component schemes is a secure AEAD scheme, then so is the combined scheme. For the following formal statement we assume $\mathcal{N} \subseteq \mathcal{N}_0 \cap \mathcal{N}_1$, $\mathcal{AD} \subseteq \mathcal{AD}_0 \cap \mathcal{AD}_1$, $\mathcal{M} \subseteq \mathcal{M}_0$, and $\mathcal{C}_0 \subseteq \mathcal{M}_1$, where $\mathcal{N}_i$, $\mathcal{AD}_i$, $\mathcal{M}_i$, $\mathcal{C}_i$ are the spaces of nonces, associated data, messages, and ciphertexts of scheme AE_i, and, without indices, of combiner CB.

Theorem 1. *For all pairs of correct AEAD schemes* AE_0 *with constant expansion and* AE_1*, all indices* $i \in \{0, 1\}$*, and all adversaries* $\mathcal{A}$ *against the combined scheme* $\mathsf{CB}(\mathsf{AE}_0, \mathsf{AE}_1)$ *as defined in Fig-*

Proc $\mathrm{enc}(k, n, ad, m)$	**Proc** $\mathrm{dec}(k, n, ad, c)$
00 $(k_0, k_1) \leftarrow k$	04 $(k_0, k_1) \leftarrow k$
01 $c_0 \leftarrow \mathrm{enc}_0(k_0, n, ad, m)$	05 $c_0 \leftarrow \mathrm{dec}_1(k_1, n, ad, c)$
02 $c_1 \leftarrow \mathrm{enc}_1(k_1, n, ad, c_0)$	06 If $c_0 = \perp$: Return $\perp$
03 Return c_1	07 $m \leftarrow \mathrm{dec}_0(k_0, n, ad, c_0)$
	08 If $m = \perp$: Return $\perp$
	09 $c_1' \leftarrow \mathrm{enc}_1(k_1, n, ad, c_0)$
	10 If $c_1' \neq c$: Return $\perp$
	11 Return m

Figure 2: Blackbox AEAD combiner $\mathsf{CB}(\mathsf{AE}_0, \mathsf{AE}_1) = (\mathrm{enc}, \mathrm{dec})$ from two AEAD schemes $\mathsf{AE}_i = (\mathrm{enc}_i, \mathrm{dec}_i), i \in \{0, 1\}$. The combiner invokes each scheme's algorithms once per operation plus one additional encryption under scheme AE_1 during combined decryption. **Associated cost:** enc: $(A+M)+(A+M) = 2A+2M$; dec: $(A+M)+(A+M)+(A+M) = 3A+3M$. **Ciphertext length:** $M+\tau+\tau$.

ure 2, there exists an adversary $\mathcal{B}_i^{\mathrm{int}}$ *such that* $\mathbf{Adv}^{\mathrm{int}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathsf{AE}_i}(\mathcal{B}_i^{\mathrm{int}})$*, an adversary* $\mathcal{B}_0^{\mathrm{ind}}$ *such that* $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}_0}(\mathcal{B}_0^{\mathrm{ind}})$*, and an adversary* $\mathcal{B}_1^{\mathrm{ind}}$ *such that* $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\mathrm{int}}) + \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\mathrm{ind}})$*. The running time of the adversaries* $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

Proof. From an adversary $\mathcal{A}$ against INT security of the combined scheme $\mathsf{CB}(\mathsf{AE}_0, \mathsf{AE}_1)$ we construct an adversary $\mathcal{B}_i^{\mathrm{int}}$ against the AEAD security of scheme AE_i as follows: $\mathcal{B}_i$ entirely simulates scheme $\mathsf{AE}_j, j \in \{0,1\} \setminus \{i\}$ itself and derives the encryptions and decryptions under scheme AE_i by forwarding the inputs to the encryption and decryption oracles respectively of game INT against scheme AE_i.

Adversary $\mathcal{B}_1^{\mathrm{int}}$, obtaining a forgery for a combined ciphertext c in the decryption oracle, can immediately forward this forgery c to the decryption oracle of the INT game against scheme AE_1; note that set Q in the combined INT game (against which $\mathcal{A}$ plays) directly complies with set Q in the INT game against scheme AE_1 (against which $\mathcal{B}_1^{\mathrm{int}}$ plays). Observe that the re-encryption in combined decryption does not need to be simulated in a reduction to AE_1's INT security as the adversary either provides a forgery (which is directly reduced with the simulation of line 05), or a ciphertext that has been obtained from the combined encryption oracle (for which no integrity verification is necessary), or a ciphertext that decrypts to '$\perp$' in the first place.

Adversary $\mathcal{B}_0^{\mathrm{int}}$ can compute all algorithms of AE_1 directly and use the oracles from the INT game against scheme AE_0 to simulate AE_0. For a ciphertext c, queried to the combined decryption oracle, that has not been an output under the same nonce and associated data of the combined encryption oracle, $\mathcal{B}_0^{\mathrm{int}}$ first decrypts c to obtain c_0. If c_0 was neither derived as part of a combined encryption query under the same nonce and associated data, its decryption either fails or it represents a forgery against INT security of scheme AE_0 (such that the reduction is successful in this case). If c_0 was, however, derived as part of such a previous combined encryption query, then $c_1' \neq c$ would have been the resulting ciphertext under scheme AE_1 (as computed in lines 09–10). Consequently, the combined decryption rejects this ciphertext (as line 10 would output $\perp$). Thereby, the only way, adversary $\mathcal{A}$ produces a valid forgery, can be reduced to INT security of scheme AE_0 such that $\mathbf{Adv}^{\mathrm{int}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathsf{AE}_i}(\mathcal{B}_i^{\mathrm{int}})$ for both $i \in \{0,1\}$.

Reducing successful adversaries against confidentiality to AE_0's IND security is trivial: $\mathcal{B}_0^{\mathrm{ind}}$ derives c_0 from the encryption oracle in the IND game against scheme AE_0 by querying both messages m^0 and m^1 under (n, ad), and encrypts c_0 under AE_1 itself with (k_1, n, ad). The decryption oracle in the IND game against the combiner can accordingly be simulated by actually computing decryptions under scheme AE_1 and deriving decryptions under scheme AE_0 from the IND game against AE_0. A successful guess b' for the challenge bit b by adversary $\mathcal{A}$ is thereby immediately a successful guess for $\mathcal{B}_0^{\mathrm{ind}}$ as well. Hence, we have $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}_0}(\mathcal{B}_0^{\mathrm{ind}})$.

For the reduction to scheme AE_1, we proceed in one game hop: in game G_1 we abort on decryptions (in the IND game's decryption oracle against the combiner) that output a message (i.e., not $\perp$). An adversary, detecting this game hop, breaks INT security (see above for the reduction). As a result, we have $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\mathrm{int}}) + \mathbf{Adv}^{G_1}$. In game G_1 we can assume that none of the ciphertexts, given to the combiner's decryption oracle, result in a valid message (hence the decryption

oracle does not need to be simulated anymore).[12] From game G_1, we can directly reduce to the IND security of scheme AE_1. $\mathcal{B}_1^{\text{ind}}$ encrypts both messages m^0 and m^1 under (k_0, n, ad) itself and then queries the encryption oracle of the IND game against scheme AE_1 for both resulting ciphertexts c_0^0 and c_0^1 as input messages (where $|c_0^0| = |c_0^1|$). The guessed bit b' is therefore valid in case of a successful adversary $\mathcal{A}$, resulting in $\mathbf{Adv}^{\text{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\text{int}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\text{int}}) + \mathbf{Adv}^{\text{ind}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\text{ind}})$. □

Note that our combiner inherits the security guarantees of its underlying secure scheme. As a result, it can be applied recursively to combine more than two AEAD schemes (such that only one of them needs to remain secure). Also it is notable that for the combiner's INT and IND security it suffices to have scheme AE_0 only IND secure and scheme AE_1 only INT secure.[13]

In Appendix A we present two further blackbox combiners that have different performance properties (i.e., slightly better in some dimensions while worse in other dimensions). We provide these combiners for didactic reasons since we believe the underlying concepts support the understanding of AEAD combiners in general.

Remarks on Tidy Encryption Schemes The tidiness property [NRS14] of an AEAD scheme requires that different ciphertexts decrypt to different messages. Equivalently, it is required that any ciphertext that validly decrypts to a message is also the result of an encryption of that message (under the same key, nonce, and associated data). Given a candidate AEAD scheme it is typically straight-forward to decide from its specification whether it is tidy or not, and in fact most modern AEAD schemes turn out to have this property. (This includes EtM constructions as well as highly integrated modes like OCB. We note however that tidiness is neither a necessary nor a sufficient condition for attaining INT or IND security.)

For tidy schemes, the decrypt-then-encrypt-then-compare component of our blackbox combiner collapses to a simple decryption. As a consequence, securely combining two AEAD schemes, one of which is guaranteed to be tidy, can be done in a particularly efficient way. (The combiner would be just like the one of Figure 2, but with lines 09,10 removed.) We caution however that blindly relying on this simpler and more efficient combiner *in practice*, even if all component schemes are evidently tidy according to their specification, might lead to fatal results. This is because 'tidiness by specification' means little if 'tidiness of implementation' is what ultimately counts. We note that it is fairly easy to (mis)implement a theoretically tidy scheme in a practically untidy way: The decryption process of most current AEAD schemes concludes with a step where some authentication value is computed from the results of the plaintext recovery and compared with a tag positioned at the end of the ciphertext. It seems conceivable that a flawed implementation would compare these two values in an incomplete way, e.g., by comparing the values except for their last bits. Such an implementation would still be formally correct, and could also provide a very fair level of authenticity and confidentiality, yet it would be untidy. As we believe that effectively assessing the practical tidiness of schemes implemented as closed-source software or in silicon is unrealistically costly, and even in the case of open-source software may represent a considerable challenge, we ultimately only recommend using our generic combiner (or one of the combiners from Section 4).

We finally note that a combination based on a tidy scheme allows for a circumvention of our performance bound from Section 5 since then only one algorithm invocation per scheme and operation is necessary. Our impossibility result is based on combining a secure AEAD scheme with an insecure untidy scheme, and crucially exploits the untidiness of the latter.

4 Non-Blackbox Combiners for AEAD

In the previous section we proposed an AEAD combiner that used its component AEAD schemes fully generically, i.e., we assumed no property of the latter other than correctness and security. In this section

[12] The simulation of the re-encryption in the combiner's decryption is the problematic part because of which we first reduce to INT security. We cannot use the encryption oracle in the IND game against AE_1 to simulate this re-encryption because either the respective nonce was used in a combined encryption query before (not allowing us to query encryption under the same nonce again), or the adversary wants to query a combined encryption under the same nonce in the future (which it then cannot do as our simulation exhausted this nonce already). For nonce misuse-resilient schemes (for which the security experiment allows for multiple encryptions under the same nonce, as long as e.g., associated data changes), this issue is resolved.

[13] The inverse combination of security properties between the underlying schemes (i.e., AE_0 is only INT secure and AE_1 is only IND secure) is, however, not implied to be sufficient by our proof (also reflected in the advantage bounds) of Theorem 1. We give an intuition why our proof uses INT and IND security to reduce the combiner's IND security to scheme AE_1 below.

we study more efficient AEAD combiners, at the expense of having to require specific structures of at least one component. Concretely, we propose two classes of improved combiners. The first class applies in cases where at least one of the AEAD components is induced by the encrypt-then-MAC paradigm, and the second class can be used if the AEAD component is constructed from an AE scheme and a PRF using ciphertext translation.[14]

4.1 EtM-Structured AEAD

In practice, authenticated encryption is often achieved from an encryption scheme and a message authentication code (MAC) using the encrypt-then-MAC (EtM) paradigm, either explicitly or implicitly. The idea is that first the message is encrypted using some passively secure encryption scheme, and then the ciphertext is integrity-protected using the MAC, providing INT security and lifting IND-CPA to IND-CCA security. Many real-world protocols such as TLS [Gut14] and SSH [SSH, ADHP16] can be used with such an AEAD construction. Further, many direct AEAD constructions make use of this design principle as well, though rather implicitly. Examples for the latter include the GCM [MV04] and CWC [KVW04] modes of operation.

We propose two EtM based combiners: One that combines a generic AEAD scheme with an EtM-structured one, and a second that combines two EtM-structured AEAD schemes. By explicitly considering the design principle of the component schemes, the combiners can distribute the processing overhead between encryption and decryption more evenly compared to the blackbox combiner of Section 3. The second combiner has also particularly short ciphertexts.

We say that an AEAD scheme is *EtM-structured* if its key space and its ciphertext space have the form $\mathcal{K} = \mathcal{K}' \times \mathcal{K}''$ and $\mathcal{C} = \mathcal{C}' \times \mathcal{C}''$ for some sets $\mathcal{K}', \mathcal{K}'', \mathcal{C}', \mathcal{C}''$, and there exist three algorithms

$$\mathcal{K}' \times \mathcal{N} \times \mathcal{M} \to \mathrm{E} \to \mathcal{C}' \qquad \mathcal{K}' \times \mathcal{N} \times \mathcal{C}' \to \mathrm{D} \to \mathcal{M} \cup \{\perp\} \qquad \mathcal{K}'' \times \mathcal{M}' \to \mathrm{M} \to \mathcal{C}'' \ ,$$

where we write $\mathcal{M}' = \mathcal{N} \times \mathcal{AD} \times \mathcal{C}'$, such that the code in Figure 3 implements algorithms enc and dec. Note how algorithms E and D stand for the encryption and decryption algorithms, respectively, while M implements the MAC. An EtM-structured AEAD scheme is secure if algorithms (E, D) provide indistinguishability against passive adversaries (according to game pIND)[15] and algorithm M is a strongly unforgeable MAC (according to game SUF).

Proc $\mathrm{enc}(k, n, ad, m)$	**Proc** $\mathrm{dec}(k, n, ad, c)$
00 $(k_e, k_m) \leftarrow k$	05 $(k_e, k_m) \leftarrow k$
01 $c' \leftarrow \mathrm{E}(k_e, n, m)$	06 $c' \,\|\|\, t \leftarrow c$
02 $t \leftarrow \mathrm{M}(k_m, n \,\|\|\, ad \,\|\|\, c')$	07 $t' \leftarrow \mathrm{M}(k_m, n \,\|\|\, ad \,\|\|\, c')$
03 $c \leftarrow c' \,\|\|\, t$	08 If $t \neq t'$: Return $\perp$
04 Return c	09 $m \leftarrow \mathrm{D}(k_e, n, c')$
	10 Return m

Figure 3: EtM construction.

EtM with Blackbox Combiner We specify our first EtM combiner in Figure 4. It intertwines an EtM-structured AEAD scheme AE_0 implemented by algorithms $(\mathrm{E}_0, \mathrm{D}_0, \mathrm{M}_0)$ with a blackbox AEAD scheme $\mathsf{AE}_1 = (\mathrm{enc}_1, \mathrm{dec}_1)$. For combined encryption, the ciphertext of the message encryption under the EtM-scheme (line 01) is fed into the blackbox encryption algorithm (line 02). The resulting ciphertext is then MACed together with the associated data (line 03). The combined decryption first verifies the MAC (lines 08–09) and then, if verification succeeds, reverses the nested encryption by decrypting under both schemes (lines 10–12).

Confidentiality is protected by the combiner due to the nested encryption, and integrity is maintained since either the MAC is secure and c thus protected directly, or c_1 is integrity protected by enc_1 such that the MAC reduces to being just a deterministic computation on it (which cannot be manipulated).

[14] With 'ciphertext translation' [Rog02] we refer to a technique that transforms an AE scheme into an AEAD scheme by XOR-ing the PRF evaluation of the associated data into the ciphertext. This is, for instance, used in EAX and OCB.

[15] Note that (E, D) have no associated-data input. Oracle Enc thus ignores the ad input in game pIND against these algorithms.

Proc $\mathrm{enc}(k, n, ad, m)$	**Proc** $\mathrm{dec}(k, n, ad, c)$
00 $((k_e, k_m), k_1) \leftarrow k$	06 $((k_e, k_m), k_1) \leftarrow k$
01 $c_0 \leftarrow \mathrm{E}_0(k_e, n, m)$	07 $c_1 \mathbin{\|} t \leftarrow c$
02 $c_1 \leftarrow \mathrm{enc}_1(k_1, n, ad, c_0)$	08 $t' \leftarrow \mathrm{M}_0(k_m, n \mathbin{\|} ad \mathbin{\|} c_1)$
03 $t \leftarrow \mathrm{M}_0(k_m, n \mathbin{\|} ad \mathbin{\|} c_1)$	09 If $t \neq t'$: Return $\perp$
04 $c \leftarrow c_1 \mathbin{\|} t$	10 $c_0 \leftarrow \mathrm{dec}_1(k_1, n, ad, c_1)$
05 Return c	11 If $c_0 = \perp$: Return $\perp$
	12 $m \leftarrow \mathrm{D}_0(k_e, n, c_0)$
	13 Return m

Figure 4: EtM combiner. **Associated cost:** enc: $M + (A + M) + (A + M) = 2A + 3M$; dec: $(A + M) + (A + M) + M = 2A + 3M$. **Ciphertext length:** $M + \tau + \tau$.

Our EtM combiner is INT and IND secure for an EtM-structured scheme AE_0 and a blackbox AEAD scheme AE_1 if AE_0's algorithms provide SUF and pIND security respectively, or AE_1 is a secure AEAD scheme. For the following formal statement we assume $\mathcal{N} \subseteq \mathcal{N}_0 \cap \mathcal{N}_1$, $\mathcal{AD} \subseteq \mathcal{AD}_0 \cap \mathcal{AD}_1$, $\mathcal{M} \subseteq \mathcal{M}_0$, $\mathcal{C}_0 \subseteq \mathcal{M}_1$, and $\mathcal{N} \times \mathcal{AD} \times \mathcal{C}_1 \subseteq \mathcal{M}_0'$, where $\mathcal{N}_i$, $\mathcal{AD}_i$, $\mathcal{M}_i$, $\mathcal{M}_i$', $\mathcal{C}_i$ are the spaces of nonces, associated data, messages, and ciphertexts of scheme AE_i and, without indices, of combiner CB.

Theorem 2. *For all pairs of correct AEAD schemes* AE_0*, implemented by* $(\mathrm{E}_0, \mathrm{D}_0, \mathrm{M}_0)$ *with constant expansion, and* AE_1*, all indices* $i \in \{0, 1\}$*, and all adversaries* $\mathcal{A}$ *against the combined scheme* $\mathsf{CB}(\mathsf{AE}_0, \mathsf{AE}_1)$ *from Figure 4, there exists an adversary* $\mathcal{B}_0^{\mathrm{int}}$ *such that* $\mathbf{Adv}^{\mathrm{int}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{suf}}_{\mathrm{M}_0}(\mathcal{B}_0^{\mathrm{int}})$*, an adversary* $\mathcal{B}_1^{\mathrm{int}}$ *such that* $\mathbf{Adv}^{\mathrm{int}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\mathrm{int}})$*, an adversary* $\mathcal{B}_0^{\mathrm{ind}}$ *such that* $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{suf}}_{\mathrm{M}_0}(\mathcal{B}_0^{\mathrm{int}}) + \mathbf{Adv}^{\mathrm{pind}}_{\mathsf{AE}_0}(\mathcal{B}_0^{\mathrm{ind}})$*, and an adversary* $\mathcal{B}_1^{\mathrm{ind}}$ *such that* $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\mathrm{ind}})$*. The running time of the adversaries* $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

Proof. We split our proof into reductions to integrity of each scheme and reductions to confidentiality of each scheme.

The reduction from INT security of the combiner to SUF security of scheme AE_0's MAC is straight forward. All encryption and decryption queries of an adversary $\mathcal{A}$ against the combiner are simulated by reduction $\mathcal{B}_0^{\mathrm{int}}$ by directly computing all algorithms except for the MAC M_0. MAC computations are instead derived from the SUF game such that lines 08–09 are obtained from oracle Vfy (cf. Figure 1). In case adversary $\mathcal{A}$ queries the decryption oracle on a ciphertext that it did not obtain from the encryption oracle under the same pair (n, ad), then the MAC verification either fails (such that it is neither a forgery in game INT nor in game SUF) or succeeds such that the forgery in the INT game is directly forwarded as a forgery in the SUF game. Thus, we have $\mathbf{Adv}^{\mathrm{int}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{suf}}_{\mathrm{M}_0}(\mathcal{B}_0^{\mathrm{int}})$.

Adversary $\mathcal{B}_1^{\mathrm{int}}$, reducing INT security of the combiner to INT security of scheme AE_1, computes all algorithms of scheme AE_0 directly and derives all computations under AE_0 from the INT game, against which it plays. A ciphertext $c = c_1 \mathbin{\|} t$, provided to the combined decryption oracle, that differs from all ciphertexts under the same pair (n, ad), output by the combined encryption oracle, can differ in the ciphertext part c_1 and in the tag part t. If it only differs in the tag part, then the tag verification will fail as there only exists one valid tag per tuple (n, ad, c_1) (as MAC computation is deterministic). If it differs in the ciphertext part, then either decryption under dec_1 outputs $\perp$ or $\mathcal{A}$ broke INT security of scheme AE_1 such that $\mathbf{Adv}^{\mathrm{int}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\mathrm{int}})$.

For the reduction of confidentiality from the combiner to the EtM-structured scheme, we proceed in a (short) sequence of games. In the first game, we output $\perp$ on any combined decryption query. An adversary, distinguishing this game from the original IND game against the combiner, breaks the SUF security of MAC M_0 as shown above. Due to this game hop, adversary $\mathcal{B}_0^{\mathrm{ind}}$, reducing successful IND adversaries $\mathcal{A}$ from this game to the pIND security of scheme AE_0's algorithms, does not need to simulate the combined decryption oracle anymore. The simulation of the combined encryption oracle is conducted by querying the encryption oracle of game pIND against algorithm E_0 of scheme AE_0 for both messages m^0 and m^1 that are input to the combined encryption oracle. The remaining computations are conducted directly by reduction $\mathcal{B}_0^{\mathrm{ind}}$. As distinguishing between combined encryptions of m^0 and m^1 distinguishes encryptions of m^0 and m^1 under the EtM-structured scheme, we have $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{suf}}_{\mathrm{M}_0}(\mathcal{B}_0^{\mathrm{int}}) + \mathbf{Adv}^{\mathrm{pind}}_{\mathsf{AE}_0}(\mathcal{B}_0^{\mathrm{ind}})$.

The reduction $\mathcal{B}_1^{\text{ind}}$, reducing IND security of the combiner to IND security of scheme AE_1, computes all algorithm invocations except for enc_1 and dec_1 directly. Computations of enc_1 and dec_1 are obtained from the IND game against scheme AE_1. In order to embed the challenge, reduction $\mathcal{B}_1^{\text{ind}}$ computes algorithm E_0 twice during combined encryption: once under each message m^0 and m^1. Then it queries the two resulting equally long ciphertexts as challenge messages to oracle Enc in game IND against scheme AE_1. The resulting ciphertext c_1 is MACed by $\mathcal{B}_1^{\text{ind}}$ to obtain $c = c_1 \mathbin{\|} t$. An adversary $\mathcal{A}$, successfully guessing bit b in the IND game against the combiner, thereby successfully guesses bit b in the IND game against scheme AE_1. Consequently we have $\mathbf{Adv}^{\text{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ind}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\text{ind}})$ which completes the proof. □

EtM with EtM Combiner We now consider a combiner that takes two EtM-structured schemes AE_0 and AE_1 implemented by algorithms $(\text{E}_0, \text{D}_0, \text{M}_0)$ and $(\text{E}_1, \text{D}_1, \text{M}_1)$, respectively, and produces an AEAD scheme that is EtM-structured as well. As many real-world protocols are based on EtM-structured AEAD schemes, when aiming to combine these schemes with others there is a wide range of options to choose from. This combiner features the shortest ciphertext length among the ones considered in this article. Last not least, since the combined scheme is again EtM-structured, it can straight-forwardly be combined with further EtM-structured schemes.

We specify our combiner in Figure 5, where we make the algorithms $(\text{E}, \text{D}, \text{M})$ of the EtM-structured result explicit. To obtain the full AEAD scheme from this, apply the procedures from Figure 3. The following description refers to the latter.

The combined encryption encrypts the message in a nested form under both schemes (see lines 01–02) and then computes MACs over the final ciphertext under both MAC algorithms (see lines 10–11). The resulting MAC tags are then XORed and appended to the final encryption ciphertext. For combined decryption, first the MAC computations are repeated and compared with the provided XOR of MAC tags in the input ciphertext (compare with Figure 3 lines 07–08). Finally, the nested encryption is reversed (lines 05–07 in Figure 5). Note that in most practical cases the ciphertext output by E has the same length as the input message. If we further assume that both component schemes use the same tag length, the overall combined ciphertext has the length of the message plus this tag length.

Confidentiality is again achieved by the nested encryption, and integrity is protected under each MAC. If one MAC is secure, the other MAC can be seen as a function, deterministically computed on integrity protected input, whose output is added only for transportation.

Proc $\text{E}(k_e, n, m)$	**Proc** $\text{D}(k_e, n, c)$	**Proc** $\text{M}(k_m, m)$
00 $(k_{e,0}, k_{e,1}) \leftarrow k_e$	04 $(k_{e,0}, k_{e,1}) \leftarrow k_e$	09 $(k_0, k_1) \leftarrow k$
01 $c_0 \leftarrow \text{E}_0(k_{e,0}, n, m)$	05 $c_0 \leftarrow \text{D}_1(k_{e,1}, n, c)$	10 $t_0 \leftarrow \text{M}_0(k_0, m)$
02 $c \leftarrow \text{E}_1(k_{e,1}, n, c_0)$	06 If $c_0 = \perp$: Return $\perp$	11 $t_1 \leftarrow \text{M}_1(k_1, m)$
03 Return c	07 $m \leftarrow \text{D}_0(k_{e,0}, n, c_0)$	12 $tag \leftarrow t_0 \oplus t_1$
	08 Return m	13 Return tag

Figure 5: EtM-EtM combiner. **Associated cost:** enc: $M + M + (A + M) + (A + M) = 2A + 4M$; dec: $(A + M) + (A + M) + M + M = 2A + 4M$. **Ciphertext length:** $M + \tau$.

If either of the underlying MtE-structured schemes provides SUF and pIND security for its respective algorithms, then the combiner from Figure 5 achieves the same security properties. Moreover, it suffices if one scheme has a secure MAC and the other scheme's encryption provides confidentiality against passive adversaries to obtain a secure combination. For the formal statement we assume $\mathcal{N} \subseteq \mathcal{N}_0 \cap \mathcal{N}_1$, $\mathcal{M} \subseteq \mathcal{M}_0$, $\mathcal{C}_0 \subseteq \mathcal{M}_1$, and $\mathcal{N} \times \mathcal{AD} \times \mathcal{C}_1 \subseteq \mathcal{M}'_i$ where $\mathcal{N}_i$, $\mathcal{AD}_i$, $\mathcal{M}_i$, $\mathcal{M}'_i$, $\mathcal{C}_i$ are the spaces of nonces, associated data, messages, and ciphertexts of scheme AE_i and, without indices, of combiner CB, and

Theorem 3. *For all pairs of correct AEAD schemes* AE_0 *with constant expansion and* AE_1 *implemented by* $(\text{E}_i, \text{D}_i, \text{M}_i)$*, all indices* $i \in \{0, 1\}$*, and all adversaries* $\mathcal{A}$ *against the combined scheme* $\mathsf{CB}(\mathsf{AE}_0, \mathsf{AE}_1)$ *as depicted in Figure 5, there exists an adversary* $\mathcal{B}_i^{\text{suf}}$ *such that* $\mathbf{Adv}^{\text{suf}}_{\mathsf{CB}(\text{M}_0,\text{M}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\text{suf}}_{\text{M}_i}(\mathcal{B}_i^{\text{suf}})$*, and an adversary* $\mathcal{B}_i^{\text{pind}}$ *such that* $\mathbf{Adv}^{\text{pind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\text{pind}}_{\mathsf{AE}_i}(\mathcal{B}_i^{\text{pind}})$*. The running time of the adversaries* $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

Parts of this statement have been proven before when pure symmetric encryption was considered

(e.g., by Herzberg [Her05]) but we are not aware of SUF secure MAC combiners in the literature. For completeness we give a full proof.

Proof. Reduction $\mathcal{B}_i^{\text{suf}}$, playing the SUF game against the MAC of scheme AE_i while simulating the SUF game for adversary $\mathcal{A}$ against the combined MAC M, computes algorithm $\mathrm{M}_j, j \in \{0,1\} \setminus \{i\}$ directly. In order to obtain MAC tags t_i for $\mathcal{A}$'s queries to oracle Tag, it queries the Tag oracle in the SUF game against MAC M_i. For the simulation of oracle Vfy towards $\mathcal{A}$ on input (m,t), $\mathcal{B}_i^{\text{suf}}$ first computes t_j under M_j directly and then queries oracle Vfy for input $(m, t \oplus t_j)$. The result of this query is forwarded to adversary $\mathcal{A}$. For a forgery against the combined MAC, input (m,t) to the combined Vfy oracle must not have been an input-output pair of a combined Tag oracle query before. If only the tag input t to a combined Vfy oracle query differs from previous queries to the combined Tag oracle, then, by the determinism of both algorithms M_i and M_j, this query is answered with $\perp$. If, on a new message input (i.e., not yet queried to combined Tag) to the combined Vfy oracle, the tag verification succeeds (such that this is a valid forgery against the combined MAC), then $(m, t \oplus t_j)$ is a valid forgery against algorithm M_i (since 1. m was not queried before and 2. $t \oplus t_j$ must be the MAC under M_i for m). Consequently, we have $\mathbf{Adv}^{\text{suf}}_{\mathsf{CB}(\mathrm{M}_0,\mathrm{M}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\text{suf}}_{\mathrm{M}_i}(\mathcal{B}_i^{\text{suf}})$.

Reducing pIND security from the combiner to each scheme AE_i works by simulating the combined encryption as follows (remember that the combined decryption does not need to be simulated in game pIND): Reduction $\mathcal{B}_0^{\text{pind}}$ derives ciphertext c_0 for a combined encryption query under input (m^0, m^1) by querying the encryption oracle in game pIND against scheme AE_0 under the same input. Encryptions under scheme AE_1 are thereby directly computed. For reducing to pIND security of scheme AE_1, reduction $\mathcal{B}_1^{\text{pind}}$ computes two equally long ciphertexts c_0^0, c_0^1 by encrypting each combined encryption oracle input m^0 and m^1 with algorithm E_0. These ciphertexts are then forwarded to the encryption oracle in game pIND against scheme AE_1, which outputs the combined ciphertext c. In either of both reductions, the challenge bit b in the combined pIND game equals the challenge bit in the pIND game against which $\mathcal{B}_i^{\text{pind}}$ plays, such that a successful adversary $\mathcal{A}$ breaks both underlying schemes' pIND security with $\mathbf{Adv}^{\text{pind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\text{pind}}_{\mathsf{AE}_i}(\mathcal{B}_i^{\text{pind}})$. This completes the proof. □

4.2 Ciphertext Translation based AEAD

We consider the combination of AEAD schemes where at least one was obtained via *ciphertext translation* [Rog02]. As in Section 4.1 this results in particularly efficient combiners.

The term 'ciphertext translation' refers to a technique that transforms an AE scheme into an AEAD scheme by XORing a PRF value derived from the associated data into the ciphertext. That is, encryption and integrity protection of the message are conducted independently of the associated data. Examples where this technique is used include EAX [BRW04] and the OCB family (OCB3 [KR11] and the *insecure* OCB2 [Rog04, IIMP19]).

We say that an AEAD scheme is *ciphertext translated* if there exist three algorithms

$$\mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathrm{E} \to \mathcal{C} \qquad \mathcal{K} \times \mathcal{N} \times \mathcal{C} \to \mathrm{D} \to \mathcal{M} \cup \{\perp\} \qquad \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \to \mathrm{F} \to \mathcal{C} \ ,$$

such that the code in Figure 6 implements algorithms enc and dec. (Note that function F sees the nonce, but this is actually optional.)

Security-wise, our results assume of a ciphertext translated AEAD scheme that it offers INT and IND security for the composed algorithms $(\mathrm{enc}, \mathrm{dec})$. That is, our consideration of ciphertext translated schemes is only restrictive regarding their syntax but not regarding security.

Proc $\mathrm{enc}(k,n,ad,m)$	**Proc** $\mathrm{dec}(k,n,ad,c)$
00 $c' \leftarrow \mathrm{E}(k,n,m)$	03 $c' \leftarrow c \ominus \mathrm{F}(k,n,ad)$
01 $c \leftarrow c' \oplus \mathrm{F}(k,n,ad)$	04 $m \leftarrow \mathrm{D}(k,n,c')$
02 Return c	05 Return m

Figure 6: Ciphertext translated AEAD construction.

The idea behind our combiner of a ciphertext translated AEAD scheme AE_1, implemented by algorithms $(\mathrm{E}_1, \mathrm{D}_1, \mathrm{F}_1)$, with a blackbox AEAD scheme $\mathsf{AE}_0 = (\mathrm{enc}_0, \mathrm{dec}_0)$ is very close to our blackbox

combiner from Section 3. However, by treating ciphertext translation explicitly, the combiner can save one operation in decryption and hence reach better efficiency.

We specify our ciphertext translation based combiner in Figure 7. It nests the blackbox AEAD encryption into the ciphertext translated AEAD encryption for combined encryption (lines 01–03) and reverses this operation for combined decryption (lines 06–10). To protect the integrity of the ciphertext, the re-encryption technique from Section 3 is applied: The outer encryption (of the ciphertext translated scheme) is re-computed to verify that, if the ciphertext translated scheme provides no ciphertext integrity, the input ciphertext c_1 would have been the result of an honest encryption under (k_1, n, c_0) (lines 11–12). Since the computation of function F is independent of the input ciphertext c, it does not need to be re-computed and compared in order to verify integrity.

Proc $\mathrm{enc}(k, n, ad, m)$	**Proc** $\mathrm{dec}(k, n, ad, c)$
00 $(k_0, k_1) \leftarrow k$	05 $(k_0, k_1) \leftarrow k$
01 $c_0 \leftarrow \mathrm{enc}_0(k_0, n, ad, m)$	06 $c_1 \leftarrow c \ominus \mathrm{F}_1(k_1, n, ad)$
02 $c_1 \leftarrow \mathrm{E}_1(k_1, n, c_0)$	07 $c_0 \leftarrow \mathrm{D}_1(k_1, n, c_1)$
03 $c \leftarrow c_1 \oplus \mathrm{F}_1(k_1, n, ad)$	08 If $c_0 = \bot$: Return $\bot$
04 Return c	09 $m \leftarrow \mathrm{dec}_1(k_1, n, ad, c_0)$
	10 If $m = \bot$: Return $\bot$
	11 $c_1' \leftarrow \mathrm{E}_1(k_1, n, c_0)$
	12 If $c_1' \neq c_1$: Return $\bot$
	13 Return m

Figure 7: Ciphertext translation based combiner from blackbox AEAD scheme AE_0 and ciphertext translated AEAD scheme AE_1. **Associated cost:** enc: $(A+M)+(A+M) = 2A+2M$; dec: $(A+M)+(A+M)+M = 2A+3M$. **Ciphertext length:** $M+\tau+\tau$.

The ciphertext translation based combiner reaches INT and IND security if either of the underlying schemes provides these properties. For the formal statement we assume $\mathcal{N} \subseteq \mathcal{N}_0 \cap \mathcal{N}_1$, $\mathcal{AD} \subseteq \mathcal{AD}_0 \cap \mathcal{AD}_1$, $\mathcal{M} \subseteq \mathcal{M}_0$, and $\mathcal{C}_0 \subseteq \mathcal{M}_1$, where $\mathcal{N}_i$, $\mathcal{AD}_i$, $\mathcal{M}_i$, $\mathcal{C}_i$ are the spaces of nonces, associated data, messages, and ciphertexts of scheme AE_i and, without indices, of combiner CB.

Theorem 4. *For all pairs of correct AEAD schemes AE_0 with constant expansion and AE_1, implemented by $(\mathrm{E}_1, \mathrm{D}_1, \mathrm{F}_1)$, all indices $i \in \{0,1\}$, and all adversaries $\mathcal{A}$ against the combined scheme $\mathsf{CB}(\mathsf{AE}_0, \mathsf{AE}_1)$ as depicted in Figure 7, there exists an adversary $\mathcal{B}_i^{\mathrm{int}}$ such that $\mathbf{Adv}^{\mathrm{int}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathsf{AE}_i}(\mathcal{B}_i^{\mathrm{int}})$, an adversary $\mathcal{B}_0^{\mathrm{ind}}$ such that $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}_0}(\mathcal{B}_0^{\mathrm{ind}})$, and an adversary $\mathcal{B}_1^{\mathrm{ind}}$ such that $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\mathrm{int}}) + \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\mathrm{ind}})$. The running time of the adversaries $\mathcal{B}$ is about that of $\mathcal{A}$.*

Proof. The proof proceeds in two steps: first we slightly change the combiner by adding a redundant computation. Then we apply Theorem 1 since adding the redundant computation makes our combiner here equivalent with the blackbox combiner from Section 3.

In our first game, we add the computation $c' \leftarrow c_1' \oplus \mathrm{F}_1(k_1, n, ad)$ to the code from Figure 7 between lines 11 and 12, and call this addition 'line 11.5' hereafter. Additionally, we change the first part of line 12 from "If $c_1' \neq c_1$: ..." to "If $c' \neq c$: ...". These two changes are fully oblivious to the adversary since, by line 06, c_1 and c are in the exact same relation as c_1' and c' (according to our changes). Consequently, no adversary can detect this game hop.

Now observe that, due to our changes, all computations of encryptions and decryptions under scheme AE_1 can be merged (disregarding the ciphertext translation idea) to blackbox algorithms $(\mathrm{enc}_1, \mathrm{dec}_1)$. By reversing the explicit consideration from Figure 6, lines 02–03 and lines 11-11.5 are replaced by blackbox enc_1 respectively and lines 06–07 are replaced by blackbox dec_1. The resulting scheme is exactly the same as the blackbox combiner from Figure 2. As a result, INT and IND security in this game follow from the proof of Theorem 1.

Consequently, we obtain $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}_0}(\mathcal{B}_0^{\mathrm{ind}})$, $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\mathrm{int}}) + \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}_1}(\mathcal{B}_1^{\mathrm{ind}})$, and $\mathbf{Adv}^{\mathrm{int}}_{\mathsf{CB}(\mathsf{AE}_0,\mathsf{AE}_1)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{int}}_{\mathsf{AE}_i}(\mathcal{B}_i^{\mathrm{int}})$ for both $i \in \{0,1\}$ which concludes our proof. □

4.3 Deriving INT-CTXT from INT-PTXT

So far, all of our combiners use the idea of verifying the integrity of ciphertexts by partially re-computing them. This technique can be generalized by considering INT-PTXT and INT-CTXT security, where the latter is defined by our INT game and a formal definition of INT-PTXT security is obtained by changing lines 07 and 12 of game INT in Figure 1 such that tuples (n, ad, m) are considered in the comparison instead. Abstractly, every nonce based scheme that is INT-PTXT secure can efficiently be turned into a scheme that is INT-CTXT secure by verifying the respective ciphertext via re-computation. This additional computation induces tidiness of the AEAD scheme (see Section 3) such that manipulations of ciphertexts must result in manipulations of plaintexts.

Jost et al. [JBB18] prove that INT-PTXT and INT-CTXT security are equivalent for schemes that are also IND secure. Our technique is actually independent of any confidentiality requirements. We are not aware that this result has been stated in the literature before and thus provide a formal proof.

Lemma 1. *For all adversaries $\mathcal{A}$ and all nonce-based AEAD schemes $\mathsf{AE} = (\mathrm{enc}, \mathrm{dec})$ it holds for nonce-based AEAD scheme $\mathsf{AE}' = (\mathrm{enc}, \mathrm{dec}')$ with*

$$\mathrm{dec}'(k, n, ad, c)\text{: } m \leftarrow \mathrm{dec}(k, n, ad, c)\text{; If } c = \mathrm{enc}(k, n, ad, m)\text{: Return } m\text{; Else: Return } \perp$$

that there exists an adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{AE}'}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{int\text{-}ptxt}}_{\mathsf{AE}}(\mathcal{B})$.

Proof. Obviously all queries of an adversary $\mathcal{A}$ in the INT-CTXT game can be answered by an adversary $\mathcal{B}$ against the INT-PTXT game by simply relaying the algorithm invocations. If an adversary $\mathcal{A}$ provides a ciphertext to the decryption oracle of the INT-CTXT game, simulated by adversary $\mathcal{B}$, that was not output by an earlier query to the encryption oracle of the INT-CTXT game, then this ciphertext 1. either results in a decryption failure under algorithm dec (such that it is no forgery in either game), 2. decrypts via dec to a message which was queried under the same pair of nonce and associated data to the encryption oracle before (which is not a forgery in the INT-PTXT game but probably in the INT-CTXT game), or 3. decrypts to a message that has not been queried to the encryption oracle before under the same pair of nonce and associated data (resulting in a valid forgery against INT-PTXT security). Case 3. can obviously be reduced to INT-PTXT security. In case 2., the adversary (by definition of the cases) produced a new ciphertext c' for a message m that was queried to the encryption oracle, outputting ciphertext c, such that $c \neq c'$. Since the consistency check via re-computation will again produce ciphertext c from message m, the output of dec' will be $\perp$ such that ciphertext c' cannot be a forgery in either security game. Consequently, $\mathcal{A}$ can only win the INT-CTXT game against scheme AE' if it produces an INT-PTXT forgery for scheme AE. This completes the proof. □

As a consequence of this result, it actually suffices for our re-computation based combiners that the outer AEAD scheme (i.e., the one invoked secondly during combined encryption) reaches only INT-PTXT security in order to achieve INT security for the combined scheme.

Note finally that our result is independent of the associated data. It thus also applies to pure authenticated encryption.

5 On the Impossibility of Two-Invocation Combiners

We proposed explicit AEAD combiners in Sections 3 and 4. The constructions, though efficient, require more than the intuitively minimal number of invocations of the components' encryption and decryption algorithms. Concretely, while combiners *in general* will not do without invoking for each encryption operation at least once the encryption algorithm of each component, and for each decryption operation at least once the decryption algorithm of each component, it is a priori unclear why a combiner would necessarily need more than these. In this section we give a negative result, showing that no generic AEAD combiner that reaches the intuitive minimum can exist.

We note that our impossibility result holds for the combination of *generic* AEAD schemes, i.e., that fulfill the set of properties formalized in Section 2. If one demands further properties (that is, if one makes the primitive stronger) the result may not hold any more. For instance, in Section 4 we demonstrate that zero-overhead combiners exist for EtM schemes, and in Section 3 we do the same for schemes the tidiness of which can be effectively confirmed.

We say that an AEAD combiner CB is *two-enc-two-dec* if for any AEAD schemes $\mathsf{AE}_0 = (\mathrm{enc}_0, \mathrm{dec}_0)$ and $\mathsf{AE}_1 = (\mathrm{enc}_1, \mathrm{dec}_1)$ the algorithms enc and dec of $\mathsf{CB}(\mathsf{AE}_0, \mathsf{AE}_1)$ both internally make precisely two algorithm invocations: enc invokes enc_0 and enc_1 one time each, and dec invokes dec_0 and dec_1 one time each. The order in which enc and dec invoke their algorithms determines two classes of two-enc-two-dec combiners: If the first invocations made by enc and dec are both of algorithms of the same component (i.e., both enc and dec invoke AE_0 first, or both enc and dec invoke AE_1 first), we say the combiner is *synchronized*. Otherwise, if the first invocations are of algorithms of different components (i.e., enc invokes AE_0 first and dec invokes AE_1 first, or the other way round), we say the combiner is *reversed*. The algorithms specified in Figure 8 characterize synchronized and reversed two-enc-two-dec combiners: A combiner CB is of the synchronized type if there exist algorithms $\alpha, \beta, \gamma, \varphi, \pi, \rho$ such that CB's algorithms coincide with enc^S and dec^S, and it is of the reversed type if algorithms $\alpha, \beta, \gamma, \varphi, \pi, \rho$ exist such that the combiner's algorithms coincide with enc^R and dec^R.

Proc $\mathrm{enc}^S(k, n, ad, m)$	**Proc** $\mathrm{dec}^S(k, n, ad, c)$
00 $(k_0, k_1) \leftarrow k$	07 $(k_0, k_1) \leftarrow k$
01 $(st_0, n_0, ad_0, m_0) \leftarrow \alpha(n, ad, m)$	08 $(st'_0, n_0, ad_0, c_0) \leftarrow \varphi(n, ad, c)$
02 $c_0 \leftarrow \mathrm{enc}_0(k_0, n_0, ad_0, m_0)$	09 $m_0 \leftarrow \mathrm{dec}_0(k_0, n_0, ad_0, c_0)$
03 $(st_1, n_1, ad_1, m_1) \leftarrow \beta(st_0, c_0)$	10 If $m_0 = \perp$: Return $\perp$
04 $c_1 \leftarrow \mathrm{enc}_1(k_1, n_1, ad_1, m_1)$	11 $(st'_1, n_1, ad_1, c_1) \leftarrow \pi(st'_0, m_0)$
05 $c \leftarrow \gamma(st_1, c_1)$	12 $m_1 \leftarrow \mathrm{dec}_1(k_1, n_1, ad_1, c_1)$
06 Return c	13 If $m_1 = \perp$: Return $\perp$
	14 $m \leftarrow \rho(st'_1, m_1)$
	15 Return m
Proc $\mathrm{enc}^R(k, n, ad, m)$	**Proc** $\mathrm{dec}^R(k, n, ad, c)$
16 $(k_0, k_1) \leftarrow k$	23 $(k_0, k_1) \leftarrow k$
17 $(st_0, n_0, ad_0, m_0) \leftarrow \alpha(n, ad, m)$	24 $(st'_0, n_1, ad_1, c_1) \leftarrow \varphi(n, ad, c)$
18 $c_0 \leftarrow \mathrm{enc}_0(k_0, n_0, ad_0, m_0)$	25 $m_1 \leftarrow \mathrm{dec}_1(k_1, n_1, ad_1, c_1)$
19 $(st_1, n_1, ad_1, m_1) \leftarrow \beta(st_0, c_0)$	26 If $m_1 = \perp$: Return $\perp$
20 $c_1 \leftarrow \mathrm{enc}_1(k_1, n_1, ad_1, m_1)$	27 $(st'_1, n_0, ad_0, c_0) \leftarrow \pi(st'_0, m_1)$
21 $c \leftarrow \gamma(st_1, c_1)$	28 $m_0 \leftarrow \mathrm{dec}_0(k_0, n_0, ad_0, c_0)$
22 Return c	29 If $m_0 = \perp$: Return $\perp$
	30 $m \leftarrow \rho(st'_1, m_0)$
	31 Return m

Figure 8: Structure of two-enc-two-dec combiners: synchronized (top) and reversed (bottom). Note that algorithms enc^S and enc^R are identical.

We now formulate our result, stating that secure two-enc-two-dec combiners do not exist:

Theorem 5. *No secure two-enc-two-dec AEAD combiner exists (neither synchronized nor reversed). More precisely, for any $x \in \{S, R\}$ and any set of algorithms $\alpha, \beta, \gamma, \varphi, \pi, \rho$ for Figure 8 there exists a pair $(\mathsf{AE}_0, \mathsf{AE}_1)$ of AEAD schemes, one of which secure, and an adversary $\mathcal{A}$ that breaks the combined scheme $\mathsf{CB}^x(\mathsf{AE}_0, \mathsf{AE}_1) = (\mathrm{enc}^x, \mathrm{dec}^x)$ with high probability.*

The theorem follows as a corollary from Lemmas 2 and 3, which handle the synchronized and reversed case separately.

Lemma 2 (Synchronized case)**.** *For any algorithms $\alpha, \beta, \gamma, \varphi, \pi, \rho$ there exist a secure AEAD scheme AE_0, an (insecure) AEAD scheme AE_1, and an adversary $\mathcal{A}$, such that $\mathcal{A}$ breaks the integrity of the combined scheme $\mathsf{CB}^S(\mathsf{AE}_0, \mathsf{AE}_1)$ with high probability.*

Proof. Let $\mathsf{AE}, \mathsf{AE}'$ be secure AEAD schemes, and derive the schemes $\mathsf{AE}^{\shuffle 00}, \mathsf{AE}^{\shuffle 11}, \mathsf{AE}^{\shuffle 0?}, \mathsf{AE}^{\shuffle 1?}$ from AE' as specified in Figure 9. Note that $\mathsf{AE}^{\shuffle 00}$ and $\mathsf{AE}^{\shuffle 11}$ provide authenticity, while ciphertexts of $\mathsf{AE}^{\shuffle 0?}$ and $\mathsf{AE}^{\shuffle 1?}$ are forgeable by flipping a single bit.

For arbitrarily picked (n, ad, m), consider an encryption $c \leftarrow \mathsf{CB}.\mathrm{enc}(n, ad, m)$ followed by a decryption $m \leftarrow \mathsf{CB}.\mathrm{dec}(n, ad, c)$, where the combiner $\mathsf{CB} := \mathsf{CB}^S(\mathsf{AE}_0, \mathsf{AE}_1)$ is operated with components $\mathsf{AE}_0 = \mathsf{AE}$ and either (a) $\mathsf{AE}_1 = \mathsf{AE}^{\shuffle 00}$ or (b) $\mathsf{AE}_1 = \mathsf{AE}^{\shuffle 11}$. As the difference between the two cases becomes visible

Proc $\mathrm{enc}^{\shortparallel bb}(k, n, ad, m)$	**Proc** $\mathrm{dec}^{\shortparallel bb}(k, n, ad, c)$
00 $c' \leftarrow \mathrm{enc}'(k, n, ad, m)$	03 $c' \shortparallel b' \leftarrow c$
01 $c \leftarrow c' \shortparallel b$	04 If $b' \neq b$: Return $\perp$
02 Return c	05 $m \leftarrow \mathrm{dec}'(k, n, ad, c')$
	06 Return m
Proc $\mathrm{enc}^{\shortparallel b?}(k, n, ad, m)$	**Proc** $\mathrm{dec}^{\shortparallel b?}(k, n, ad, c)$
07 $c' \leftarrow \mathrm{enc}'(k, n, ad, m)$	10 $c' \shortparallel b' \leftarrow c$
08 $c \leftarrow c' \shortparallel b$	11 $m \leftarrow \mathrm{dec}'(k, n, ad, c')$
09 Return c	12 Return m

Figure 9: Schemes $\mathsf{AE}^{\shortparallel 00}, \mathsf{AE}^{\shortparallel 11}, \mathsf{AE}^{\shortparallel 0?}, \mathsf{AE}^{\shortparallel 1?}$, all parameterized with a secure AEAD scheme $\mathrm{enc}', \mathrm{dec}'$.

to CB.enc not before line 04 of Figure 8, the values of n_0, ad_0, m_0, c_0 from line 02 are independent of the case. Further, the values n_0, ad_0, c_0, m_0 from line 09 will coincide with those of line 02, as otherwise the combiner would, with noticeable probability, either abort in line 10 or present a forgery for AE_0[16], contradicting either the correctness of CB or the security of AE_0. Observe that the values n_0, ad_0, c_0, m_0 can be recovered by exploiting knowledge of values n, ad, m, c and algorithms α, φ (see lines 01 and 08). This further allows to recover the values n_1, ad_1, c_1, m_1 (see lines 03 and 11), which again will coincide in the $\mathrm{enc}^S, \mathrm{dec}^S$ executions, for the same reason as above. Note that n_1, ad_1, m_1 will be independent of the case we are in, i.e., (a) vs. (b), while c_1 will depend on it.

Consider the adversary that poses an encryption query $\mathrm{Enc}(n, ad, m)$, recovers c_1 as just described, derives c_1' from it by flipping its last bit, then computes $c' \leftarrow \gamma(st_1, c_1')$ as in line 05 (note that state st_1 can be recovered by knowledge of $n, ad, m, c_0, \alpha, \beta$), and finally poses a decryption query $\mathrm{Dec}(n, ad, c')$. That is, the adversary transforms a case-(a) ciphertext to a case-(b) ciphertext, and vice versa. The internals of the CB.dec algorithm are oblivious of the case we are in until at least line 12, so values n_1, ad_1 in that line will be independent of the case, and coincide with the original ones from line 04. We expect, of course, the decryption operation of line 12 to reject (as the last bit of c_1 is wrong).

Consider finally the two additional cases (a') $\mathsf{AE}_1 = \mathsf{AE}^{\shortparallel 0?}$ and (b') $\mathsf{AE}_1 = \mathsf{AE}^{\shortparallel 1?}$, i.e., the instantiation of the AE_1 component almost as above, the difference being that the decryption algorithms tolerate bit flips. As the view of CB.enc does not change at all, and the view of CB.dec does not change until line 13 is reached, all arguments made above continue to apply, except that the decryption operation will not reject but continue operation with the unmodified m_1 value. This means that line 14 will recover the original message m, meaning that ciphertext c' is a valid forgery. □

Lemma 3 (Reversed case)**.** *For any algorithms $\alpha, \beta, \gamma, \varphi, \pi, \rho$ there exist a secure AEAD scheme AE_0, an (insecure) AEAD scheme AE_1, and an adversary $\mathcal{A}$, such that $\mathcal{A}$ breaks the integrity of the combined scheme $\mathsf{CB}^R(\mathsf{AE}_0, \mathsf{AE}_1)$ with high probability.*

Proof. Let AE be a secure AEAD scheme and M a strongly unforgeable MAC, and derive the schemes $\mathsf{AE}^{mt00}, \mathsf{AE}^{mt11}, \mathsf{AE}^{mt0?}, \mathsf{AE}^{mt1?}$ from M as specified in Figure 10. Note that AE^{mt00} and AE^{mt11} provide authenticity, while ciphertexts of $\mathsf{AE}^{mt0?}$ and $\mathsf{AE}^{mt1?}$ are forgeable by flipping the last bit of any authentic ciphertext.

For arbitrarily picked (n, ad, m), consider an encryption $c \leftarrow \mathsf{CB}.\mathrm{enc}(n, ad, m)$ followed by a decryption $m \leftarrow \mathsf{CB}.\mathrm{dec}(n, ad, c)$, where the combiner $\mathsf{CB} := \mathsf{CB}^R(\mathsf{AE}_0, \mathsf{AE}_1)$ is operated with components $\mathsf{AE}_0 = \mathsf{AE}$ and either (a) $\mathsf{AE}_1 = \mathsf{AE}^{mt00}$ or (b) $\mathsf{AE}_1 = \mathsf{AE}^{mt11}$. The values n_0, ad_0, c_0, m_0 from line 28 in Figure 8 will coincide with those of line 18, as otherwise the combiner would, with noticeable probability, either abort in line 29 or present a forgery for AE_0, contradicting either the correctness of CB or the unforgeability of AE_0. Similarly the values n_1, ad_1, c_1, m_1 from line 25 will coincide with those of line 20, as otherwise the combiner would, with noticeable probability, either abort in line 26 or present a forgery for AE_1, contradicting either the correctness of CB or the unforgeability of AE_1. We show that values st_1 and c_1 from line 21 are publicly recoverable: Recover st_0', c_1 from φ, n, ad, c as in line 24, recover m_1 from c_1 by

[16] We note that neither of the combiner's algorithms has access to either of the schemes' symmetric key. Furthermore, the combiner has no state that stores information (e.g., ciphertexts) beyond one execution of combined encryption or decryption respectively. Consequently, crafting a different valid ciphertext is for the combiner as hard as for any outside adversary.

Proc $\mathrm{enc}^{mtbb}(k, n, ad, m)$	**Proc** $\mathrm{dec}^{mtbb}(k, n, ad, c)$
00 $t \leftarrow \mathrm{M}(k, n \mid\mid ad \mid\mid m)$	03 $m \mid\mid t' \mid\mid b' \leftarrow c$
01 $c \leftarrow m \mid\mid t \mid\mid b$	04 $t \leftarrow \mathrm{M}(k, n \mid\mid ad \mid\mid m)$
02 Return c	05 If $(t', b') \neq (t, b)$: Return $\perp$
	06 Return m
Proc $\mathrm{enc}^{mtb?}(k, n, ad, m)$	**Proc** $\mathrm{dec}^{mtb?}(k, n, ad, c)$
07 $t \leftarrow \mathrm{M}(k, n \mid\mid ad \mid\mid m)$	10 $m \mid\mid t' \mid\mid b' \leftarrow c$
08 $c \leftarrow m \mid\mid t \mid\mid b$	11 $t \leftarrow \mathrm{M}(k, n \mid\mid ad \mid\mid m)$
09 Return c	12 If $t' \neq t$: Return $\perp$
	13 Return m

Figure 10: Schemes $\mathsf{AE}^{mt00}, \mathsf{AE}^{mt11}, \mathsf{AE}^{mt0?}, \mathsf{AE}^{mt1?}$, all parameterized with a strongly unforgeable MAC scheme M. Note that algorithms $\mathrm{dec}^{mt0?}$ and $\mathrm{dec}^{mt1?}$ are identical.

exploiting that AE_1-ciphertexts encode the message in the clear, recover c_0 from π, st'_0, m_1 as in line 27, recover st_0 from α, n, ad, m as in line 17, finally recover st_1 from β, st_0, c_0 as in line 19.

Consider the adversary that poses an encryption query $\mathrm{Enc}(n, ad, m)$, recovers st_1, c_1 as just described, derives c'_1 from c_1 by flipping its last bit, then computes $c' \leftarrow \gamma(st_1, c'_1)$ as in line 21, and finally poses a decryption query $\mathrm{Dec}(n, ad, c')$.

Consider finally the two additional cases (a') $\mathsf{AE}_1 = \mathsf{AE}^{mt0?}$ and (b') $\mathsf{AE}_1 = \mathsf{AE}^{mt1?}$, i.e., the instantiation of the AE_1 component almost as above, the difference being that the decryption algorithms tolerate bit flips. As the invocation of β in line 19 is independent of the case we are in (hence st_1 is independent of the cases), the above observation that the values n_1, ad_1 from lines 20 and 25 are identical continues to hold in all of these cases. If c_1 was generated in case (a'), c'_1 looks for γ as in cases (b) or (b') and thus c' looks for algorithms φ, π, and ρ as in cases (b) or (b') as well (and vice versa if c_1 was generated in case (b')). Observing that the decryption algorithms $\mathrm{dec}^{mt0?}$ and $\mathrm{dec}^{mt1?}$ coincide, the value m_1 recovered in line 25 in the Dec query of adversary $\mathcal{A}$ is independent of the case, in particular will be processed as in (a) and (b) and thus accepted. That is, adversary $\mathcal{A}$ delivered a successful forgery c'. □

Acknowledgments

We thank Jeroen Pijnenburg for early discussions on the topic and the reviewers of ToSC for their helpful suggestions for improving this article. The research of Poettering was supported by the European Union's Horizon 2020 project FutureTPM (779391) and the research of Rösler was supported by the research training group "Human Centered Systems Security" sponsored by the state of North Rhine-Westphalia, Germany.

References

[ADHP16] Martin R. Albrecht, Jean Paul Degabriele, Torben Brandt Hansen, and Kenneth G. Paterson. A surfeit of SSH cipher suites. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1480–1491. ACM Press, October 2016.

[AP19a] Marcel Armour and Bertram Poettering. Substitution attacks against message authentication. *IACR Transactions on Symmetric Cryptology*, 2019(3):152–168, 2019.

[AP19b] Marcel Armour and Bertram Poettering. Subverting decryption in AEAD. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *Lecture Notes in Computer Science*, pages 22–41. Springer, Heidelberg, December 2019.

[BBF+19] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 206–226. Springer, Heidelberg, 2019.

[Ber05] Daniel J. Bernstein. Cache-timing attacks on AES, 2005. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[BL16] Karthikeyan Bhargavan and Gaëtan Leurent. On the practical (in-)security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 456–467. ACM Press, October 2016.

[BPR14] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 1–19. Springer, Heidelberg, August 2014.

[BRW04] Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, Heidelberg, February 2004.

[BS16] Raphael Bost and Olivier Sanders. Trick or tweak: On the (in)security of OTR's tweaks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 333–353. Springer, Heidelberg, December 2016.

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 188–209. Springer, Heidelberg, February 2005.

[FHNS16] Marc Fischlin, Amir Herzberg, Hod Bin Noon, and Haya Shulman. Obfuscation combiners. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 521–550. Springer, Heidelberg, August 2016.

[FL07] Marc Fischlin and Anja Lehmann. Security-amplifying combiners for collision-resistant hash functions. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 224–243. Springer, Heidelberg, August 2007.

[FL08] Marc Fischlin and Anja Lehmann. Multi-property preserving combiners for hash functions. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 375–392. Springer, Heidelberg, March 2008.

[FLP08] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. Robust multi-property combiners for hash functions revisited. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 655–666. Springer, Heidelberg, July 2008.

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 190–218. Springer, Heidelberg, March 2018.

[Gut14] Peter Gutmann. Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 7366, RFC Editor, September 2014.

[Her05] Amir Herzberg. On tolerant cryptographic constructions. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 172–190. Springer, Heidelberg, February 2005.

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 96–113. Springer, Heidelberg, May 2005.

[IIMP19] Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, and Bertram Poettering. Cryptanalysis of OCB2: Attacks on authenticity and confidentiality. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 3–31. Springer, Heidelberg, August 2019.

[IOM12] Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and repairing GCM security proofs. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 31–49. Springer, Heidelberg, August 2012.

[JBB18] Daniel Jost, Christian Badertscher, and Fabio Banfi. A note on the equivalence of IND-CCA & INT-PTXT and IND-CCA & INT-CTXT. Cryptology ePrint Archive, Report 2018/135, 2018. https://eprint.iacr.org/2018/135.

[KR11] Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In Antoine Joux, editor, *Fast Software Encryption – FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327. Springer, Heidelberg, February 2011.

[KS09] Emilia Käsper and Peter Schwabe. Faster and timing-attack resistant AES-GCM. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 1–17. Springer, Heidelberg, September 2009.

[KVW04] Tadayoshi Kohno, John Viega, and Doug Whiting. CWC: A high-performance conventional authenticated encryption mode. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 408–426. Springer, Heidelberg, February 2004.

[McG08] David McGrew. An Interface and Algorithms for Authenticated Encryption. RFC 5116, January 2008.

[MH81] Ralph C. Merkle and Martin E. Hellman. On the security of multiple encryption. *Commun. ACM*, 24(7):465–467, 1981.

[MLMI14] Kazuhiko Minematsu, Stefan Lucks, Hiraku Morita, and Tetsu Iwata. Attacks and security proofs of EAX-prime. In Shiho Moriai, editor, *Fast Software Encryption – FSE 2013*, volume 8424 of *Lecture Notes in Computer Science*, pages 327–347. Springer, Heidelberg, March 2014.

[MV04] David A. McGrew and John Viega. The security and performance of the Galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004: 5th International Conference in Cryptology in India*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, Heidelberg, December 2004.

[Nan14] Mridul Nandi. Forging attacks on two authenticated encryption schemes COBRA and POET. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 126–140. Springer, Heidelberg, December 2014.

[NRS14] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 257–274. Springer, Heidelberg, May 2014.

[PR20] Bertram Poettering and Paul Rösler. Combiners for AEAD. *IACR Trans. Symmetric Cryptol.*, 2020(1):121–143, 2020.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 2002: 9th Conference on Computer and Communications Security*, pages 98–107. ACM Press, November 2002.

[Rog04] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, Heidelberg, December 2004.

[SMAP16] Willem Schroé, Bart Mennink, Elena Andreeva, and Bart Preneel. Forgery and subkey recovery on CAESAR candidate iFeed. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015: 22nd Annual International Workshop on Selected Areas in Cryptography*, volume 9566 of *Lecture Notes in Computer Science*, pages 197–204. Springer, Heidelberg, August 2016.

[SSH] OpenSSH's deviations and extensions to the published SSH protocol. https://github.com/openssh/libopenssh/blob/master/ssh/PROTOCOL. From the Source Code Repository.

[ZHSI04] Rui Zhang, Goichiro Hanaoka, Junji Shikata, and Hideki Imai. On the security of multiple encryption or CCA-security+CCA-security=CCA-security? In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 360–374. Springer, Heidelberg, March 2004.

A Additional Combiners

We discuss a couple of combiners that are potentially slightly outperformed regarding efficiency by our combiners from the main body of the paper. Nevertheless, we believe that the description of these combiners and their efficiency trade-offs support the understanding of AEAD combiners in general.

A.1 Concatenation Combiner

We specify the encryption and decryption routines of our concatenation combiner in Figure 11. Unlike the combiners proposed in the main body, the encryption routine performs three invocations of the component schemes' routines: two of enc_0 and one of enc_1. The task of the first enc_0 invocation is to encrypt the message but not to integrity-protect the assocated data or ciphertext, while the second enc_0 invocation is used exclusively to establish integrity. (Note that the second invocation does not provide any confidentiality as its message input is empty.) We provide the two invocations with the required different nonces by appending a bit to the input nonce.

Proc $\mathrm{enc}(k, n, ad, m)$	**Proc** $\mathrm{dec}(k, n, ad, c)$
00 $(k_0, k_1) \leftarrow k$	05 $(k_0, k_1) \leftarrow k$
01 $c_0 \leftarrow \mathrm{enc}_0(k_0, n \mathbin{\shortparallel} 0, \epsilon, m)$	06 $c_1 \mathbin{\shortparallel} c_2 \leftarrow c$
02 $c_1 \leftarrow \mathrm{enc}_1(k_1, n, ad, c_0)$	07 $c_0 \leftarrow \mathrm{dec}_1(k_1, n, ad, c_1)$
03 $c_2 \leftarrow \mathrm{enc}_0(k_0, n \mathbin{\shortparallel} 1, c_1 \| ad, \epsilon)$	08 Require $c_0 \neq \perp$
04 Return $c_1 \mathbin{\shortparallel} c_2$	09 $m \leftarrow \mathrm{dec}_0(k_0, n \mathbin{\shortparallel} 0, \epsilon, c_0)$
	10 Require $m \neq \perp$
	11 $c_2' \leftarrow \mathrm{enc}_0(k_0, n \mathbin{\shortparallel} 1, c_1 \| ad, \epsilon)$
	12 Require $c_2 = c_2'$
	13 Return m

Figure 11: Blackbox AEAD combiner $\mathsf{CB}(\mathsf{AE}_0, \mathsf{AE}_1) = (\mathrm{enc}, \mathrm{dec})$ from AEAD schemes $\mathsf{AE}_i = (\mathrm{enc}_i, \mathrm{dec}_i)$, $i \in \{0, 1\}$, that uses nested encryption to protect confidentiality and re-computation to verify integrity. **Associated cost:** enc: $M + (A + M) + (A + M) = 2A + 3M$; dec: $(A + M) + M + (A + M) = 2A + 3M$. **Ciphertext length:** $M + \tau + \tau + \tau$ (where ciphertext c_2 might be larger than τ).

The combined decryption recovers the payload by reversing the nested encryption and re-computing and checking the second ciphertext. The combiner provides IND and INT security as long as one of $\mathsf{AE}_0, \mathsf{AE}_1$ is a secure AEAD scheme: Confidentiality is reached due to the nested encryption, and integrity is reached since either enc_1 protects c_1, and c_2 is derived and verified deterministically, or enc_0 protects c_1 and its own output c_2.

The combined ciphertext is longer and the total number of operations is higher than for our blackbox combiner from Section 3. In the combined decryption, however, associated data and message inputs are differently distributed among the algorithm invocations. Firstly, one processing of associated-data input is saved with this concatenation combiner (the combined decryption here only processes ad twice whereas the combiner in Figure 2 processes it three times). Secondly, if an underlying scheme AE_0 performs better when either associated data or message input is empty, the combiner from Figure 11 is superior.

A.2 XOR Combiner

This combiner assumes that the secure component scheme provides IND$ security (indistinguishability from random strings). This notion is stronger than plain IND, and an AEAD scheme that reaches it can be (mis-)used as a PRF. We specify the routines of our combiner in Figure 12. The confidentiality of the message is, once more, guaranteed by nesting encryption invocations of the components (lines 01–02), while integrity protection is achieved by appending a tag to the resulting ciphertext (line 06). This tag is computed by XORing a τ-long stretch of the ciphertexts obtained by encrypting the empty message (lines 03–05). The latter step can be seen as considering functions $F_i(k_i, x) := \mathrm{enc}_i(k_i, n, x, \epsilon)$ as PRFs, with $F((k_0, k_1), x) := F_0(k_0, x) \oplus F_1(k_1, x)$ being their secure combination. The decryption routine reverses the nested encryption, and recomputes and compares the tag. The combiner is INT and IND secure as long as either of the schemes is INT and IND$ secure.

Proc $\mathrm{enc}(k, n, ad, m)$	**Proc** $\mathrm{dec}(k, n, ad, c)$
00 $(k_0, k_1) \leftarrow k$	07 $(k_0, k_1) \leftarrow k$
01 $c_0 \leftarrow \mathrm{enc}_0(k_0, n \mid\mid 0, \epsilon, m)$	08 Require $\lvert c\rvert > \tau$
02 $c_1 \leftarrow \mathrm{enc}_1(k_1, n \mid\mid 0, \epsilon, c_0)$	09 $c_1 \leftarrow c[1 \ldots \lvert c\rvert - \tau - 1]$
03 $c_2 \leftarrow \mathrm{enc}_0(k_0, n \mid\mid 1, c_1 \Vert ad, \epsilon)$	10 $tag \leftarrow c[\lvert c\rvert - \tau \ldots \lvert c\rvert]$
04 $c_3 \leftarrow \mathrm{enc}_1(k_1, n \mid\mid 1, c_1 \Vert ad, \epsilon)$	11 $c_0 \leftarrow \mathrm{dec}_1(k_1, n \mid\mid 0, \epsilon, c_1)$
05 $tag \leftarrow (c_2 \oplus c_3)[1 \ldots \tau]$	12 Require $c_0 \neq \perp$
06 Return $c_1 \Vert tag$	13 $m \leftarrow \mathrm{dec}_0(k_0, n \mid\mid 0, \epsilon, c_0)$
	14 Require $m \neq \perp$
	15 $c_2 \leftarrow \mathrm{enc}_0(k_0, n \mid\mid 1, c_1 \Vert ad, \epsilon)$
	16 $c_3 \leftarrow \mathrm{enc}_1(k_1, n \mid\mid 1, c_1 \Vert ad, \epsilon)$
	17 If $(c_2 \oplus c_3)[1 \ldots \tau] \neq tag$:
	18 Return $\perp$
	19 Return m

Figure 12: Blackbox AEAD combiner $\mathsf{CB}(\mathsf{AE}_0, \mathsf{AE}_1) = (\mathrm{enc}, \mathrm{dec})$ from AEAD schemes $\mathsf{AE}_i = (\mathrm{enc}_i, \mathrm{dec}_i)$, $i \in \{0, 1\}$, that uses nested encryption to protect confidentiality and XOR of ciphertexts to achieve pseudo-randomness. We denote with '$c[x \ldots y]$' the value of c from position x to position y. **Associated cost:** enc: $M + M + (A + M) + (A + M) = 2A + 4M$; dec: $M + M + (A + M) + (A + M) = 2A + 4M$. **Ciphertext length:** $M + \tau + \tau + \tau$.