Don't Panic!

The Cryptographers' Guide to Robust Authenticated (On-line) Encryption Draft, March 11, 2015

Farzaneh Abed, Christian Forler*, Eik List, Stefan Lucks, and Jakob Wenzel

Bauhaus-Universität Weimar, Germany
`<first name>.<last name>@uni-weimar.de`

Abstract. In [13], Hoang et al. discuss the security definitions for nonce-misuse-resistant authenticated on-line encryption. They argue that (1) all on-line authenticated encryption schemes are vulnerable to the chosen-prefix secret-suffix (CPSS) attack; therefore, none deserves to be called "misuse-resistant" unlike misuse-resistant authenticated encryption (MRAE) schemes, which are off-line and invulnerable to that attack, and (2) if on-line encryption is required, then the OAE1 notion as defined by [10] is too weak for its claimed purpose. Therefore, the authors of [13] propose OAE2 as a putatively stronger alternative.
This work addresses their arguments in three directions. Firstly, we show that all AE schemes – including MRAE schemes – are vulnerable to an attack which we call chosen-plaintext overwrite-secret (CPOS), which is structurally similar to the CPSS attack. Secondly, while there is a clear need for AE schemes that limit the damage in the case of a nonce reuse, the phrase "misuse resistance" has always promised more than it can hold. Thus, we propose to replace it by *"robustness"* or *"damage limitation"*. Thirdly, we discuss the differences between OAE1 and OAE2. While the latter is an interesting security notion of its own, we argue that it solves a different problem than OAE1. We show that the combination of decryption robustness and OAE1 security (called OAE1$^+$) implies OAE2 security, i.e., an OAE2 scheme may be vulnerable to decryption-misuse, but any decryption-robust OAE1 scheme can be transformed into a decryption-robust OAE2 scheme. Moreover, we introduce the notion OAE2$^+$ for a decryption-robust OAE2 scheme. Finally, we argue that POET satisfies the properties of an OAE1$^+$ scheme and show how it can be transformed into an OAE2$^+$ scheme.

Keywords: on-line authenticated encryption, nonce-misuse resistant

1 Introduction

In 2006, Rogaway and Shrimpton [19] defined deterministic authenticated encryption for the key-wrap problem. Additionally, they coined the term misuse-resistant AE (MRAE) for nonce-based schemes that can guarantee to retain a certain level of security when message numbers (that were to be nonces) repeat. More detailed, authenticity must be unaffected, and the only thing an adversary can learn in terms of privacy is repeated encryptions of tuples of the same associated data, nonce, and message under the same key. The authors stressed that nonce-misuse resistance could be achieved only by two-pass schemes since every output bit must depend on every input bit.

Building on Bellare et al. [5], Fleischmann, Forler, and Lucks [10] discussed which security on-line authenticated encryption (OAE) schemes can retain if nonces repeat. They showed that, apart from generic composition (Encrypt-then-Mac [7]), no previously published on-line scheme could provide security against nonce-ignoring adversaries. The authors called an OAE scheme *nonce-misuse-resistant* if authenticity is not weakened from nonce reuse, and in terms of privacy, the only thing an adversary can learn is the longest common prefix of the current associated data, nonce, and message with other queries under the same key.

During the recent years (and particularly in the frame of the ongoing CAESAR competition [8]), many recent OAE design teams adopted the definition of Fleischmann et al; thereby, the original security guarantees became blurred. A considerable shift became apparent, from regarding on-line nonce-misuse resistance as a "second line of defense" for the case of an emergency [3,10] towards a must-have feature, culminating in the misinterpretation and advertisement of some schemes as "nonce-free" and therefore almost encouraging users to repeat nonces.

Rogaway continuously [12,16,17] emphasized that misuse resistance in the sense of [19] could not be a feature of on-line schemes. He was supported by the results of Reyhanitabar and Vizár [15], who proposed a generic nonce-misusing attack, called CPSS (Chosen-Prefix Secret-Suffix), against the privacy of on-line schemes.

In [13], Hoang, Reyhanitabar, Rogaway, and Vizár consolidated their arguments against nonce-misuse-resistant on-line AE: the authors renamed the definition of [10] as OAE1 and introduced a second kind of on-line AE, called

*The research leading to these results received funding from the Silicon Valley Community Foundation, under the Cisco Systems project Misuse Resistant Authenticated Encryption for Complex and Low-End Systems (MIRACLE).

OAE2, which splits messages into segments that are encrypted *and* authenticated each. The authors proposed two instances, called CHAIN and STREAM, the former for turning an MRAE into an OAE2 scheme and the latter for turning a nonce-based two-pass into a nonce-based on-line scheme, respectively. Furthermore, they highlighted the differences between the meanings of the terms used for defining remaining security guarantees under nonce misuse for OAE schemes in the classifications [3,14] and recalled the CPSS attack, underlining their critique that the privacy security of OAE schemes usually depend strongly on a fixed block size of the underlying basic primitive.

This work discusses the points of [13], We discuss the applicability of OAE2 in practice, in particular, that of the user-defined segmentation. We demonstrate an attack that we call *chosen-position overwrite-secret (CPOS)*, which is similar to CPSS, but can be applied against the privacy of any MRAE scheme. Thus, we argue that under nonce reuse, no AE scheme can provide security comparable to nonce-based AE. We propose the notions OAE1$^+$ (OAE2$^+$) for OAE1 (OAE2) schemes that provide OPERM-CCA- and INT-RUP-security. We show how an OAE1$^+$ scheme can be easily transformed into a secure OAE2$^+$ scheme by adding well-formed redundancy to the message. Thus, we say that schemes such as POET with minor tweaks for the integration of intermediate tags are almost instantaneously well-suited for the use cases of OAE2.

2 Discussion on [13]

Specific Issues with OAE1

"[...] schemes targeting OAE1 conflate the blocksize of the tool being used to construct the scheme and the memory restrictions or latency requirements that motivate OAE in the first place [20]." [p. 2]

We agree with the authors in this point. All OAE1 schemes, we are aware of, use the block size of an underlying primitive as the *on-line permutation* block size, implied by the security definition. For many practical applications, the *on-line permutation* block size could be much larger than, say, 128 bits if the primitive is the AES. We stress that this is not an issue of the OAE1 definition, but rather one of the schemes proposed so far.

The Differences between OAE1 and OAE2

"OAE1 measures privacy against an idealized object that is an on-line cipher followed by a tag. But having such a structure is not only unnecessary for achieving on-line encryption, but also undesirable for achieving good security." [p. 2]

OAE2 proposes to parse long messages and to split them into different variable-sized segments. Segment boundaries are determined by the messages semantics. For example, a video stream might treat each frame as a segment that must be authenticated separately, with the previous frames being treated like the authenticated data of a classical AE scheme.

Obviously, OAE1 and OAE2 address different problems; Hoang et al. argue that OAE2 addresses the right problem unlike OAE1:

"OAE1 fails to capture the real-world characteristics of the problem that practitioners actually need to have solved." [p. 19]

OAE2 is actually a very interesting security notion of its own right that will deserve further attention. Whenever an implementation is able to parse messages, split them into segments and add an authentication tag to each, OAE2 may tackle the right problem. On the other hand, the *whenever* puts severe constraints on the application and its design. Very often, OAE2 would not be an acceptable solution for practitioners. From our point of view, OAE2 implies the existence of a supervising overlying protocol layer that is capable of choosing a well-suited segment size. (Figure 3, p. 8, just writes "$(M_1, \ldots, M_m) \leftarrow M$" to indicate the message M being split up into m segments.) OAE2 may render it highly difficult to transparently implement cryptographic schemes into security protocols.

In fact, OAE2 assumes the receiver to *know* the sizes of segments or their boundaries. (See Figure 3, p. 8: "$(C_1, \ldots, C_m) \leftarrow C$".) It does not consider how this knowledge is transmitted from the sender to the receiver. This is an implementation problem, but also a security issue. Any out-of-band technique to transmit the segment boundaries to the receiver may break the privacy by leaking the sizes of segments. This would not violate the OAE2 security notion, because it is not part of the system that is considered by the OAE2 security notion. But the receiver cannot decrypt without the out-of-band transmission, in whatever form.

OAE2 with Fixed-Size Blocks

If parsing and splitting up the message into semantic segments is difficult, a remedy may be to split the message into fixed-size (n-bit) blocks, for some reasonable n. The last block may be smaller than n bit. But, as we will elaborate below, one can easily tweak OAE1 to support this kind of intermediate tags as well, with the same benefits as OAE2 with fixed-size segments. In that context, there is not much need for the OAE2 formalism.

Actually, there are a couple of CAESAR candidates, such as ELmD [9], which implement this idea by supporting the generation of intermediate authentication tags. After every n message bits, an intermediate authentication tag is inserted into the cipher text. After checking the tag, the receiver can safely emit the decrypted n-bit message block. Nevertheless, this is only relevant if there is additional space/bandwidth available for intermediate authentication tags.

Comparing OAE1 and OAE2

> "OAE1 with blocksize and tagsize n remains weaker than OAE2 with a $(0, n)$-expanding scheme and all segments required to have exactly n bits." [p. 22f.]

This point is well taken and refers to the case of OAE2 without an intermediate authentication tag (indicated by the first parameter, 0), and with a final authentication tag of n bit (indicated by the second parameter). Unlike OAE2, OAE1, as originally defined in [10], does not imply any security under decryption misuse.

This point is also moot which ignores recent work on security under decryption misuse, cf. [11] or [4]. If a scheme is OAE1 and resistant to decryption misuse, such as the CAESAR candidate POET [2], the argument made by Hoang et al. in Appendix D in [13] does not apply any more.

Online Decryption

> "[...] while OAE1 aims to ensure that encryption is on-line, it ignores decryption. [...] We question the utility of on-line encryption when one still needs to buffer the entire ciphertext before any portion of the (speculative) plaintext may be disclosed, the implicit assumption behind OAE1." [p. 2]

We are in violent agreement with Hoang et al.; since it is completely unnatural to require encryption to be on-line, while the security definition assumes decryption to be off-line.

The CAESAR submission POET has been designed to address this issue. See also [11] and [4] studying the *release of unverified plaintext.*

Weaker Notions

> "[...] weakened variants of OAE1 have proliferated. [...] In this race to the bottom, it may seem as though the scheme comes first and whatever properties it provides is branded as some form misuse resistance." [p. 16]

There is a big difference between a security notion being developed to capture the maximum security that can be maintained under certain constraints (such as support for an on-line workflow), and a security notion being discovered as *this is what that scheme just happens to provide.* We agree with Hoang et al. that most of the "below-OAE1" security definitions appear of limited relevance for the design of secure applications.

On the other hand, the large number of "me too" claims in the CAESAR competition shows that there is a demand for the combination of on-line encryption and nonce-misuse robustness in the cryptographic community.

*The Phrase "***Misuse Resistance***"*

> "While many real-world settings won't enable a CPSS attack, our own take is that, for a general purpose tool, such a weakness effectively refutes any claim of misuse resistance." [p. 2]

We agree that the phrase *misuse resistance* promises more than it can possibly keep. But unlike Hoang et al., we argue that this is not a specific issue for on-line AE. As the CPOS attack shows, off-line AE is in the same ballgame as on-line AE.

More precisely: As the difference between the CPSS attack, which applies to all on-line AE schemes, and the CPOS attack, which applies to on-line and off-line AE schemes, shows that off-line AE schemes can be more secure than on-line AE schemes. But, as the CPOS attack shows, there is a huge security gap between nonce-misuse security, and the standard security notion where nonces are never reused.

> "The paper defining MRAE [...] never suggested that nonce-reuse was OK [...]" [p. 16]

Agreed, and neither did [10] suggest that nonce-reuse was OK. Nonce-reuse is misuse, and never acceptable. No AE scheme can possibly *resist* to this kind of misuse and therefore *misuse resistance* may not be a suitable name for that property. All that cryptosystem designers can possibly do is to try to limit the harm from nonce misuse and to hinder possible exploits. This is the reason why one should not call it *misuse resistance*, but rather *damage limitation* or, as we would prefer, *robustness*.

> "[...] if reusing a scheme's nonce greatly weakens a security guarantee, it seems best to avoid calling it misuse-resistant, arbitrary-IV, or nonce-free. Our language should try to signal the opposite, that, absent some application-specific analysis, nonces must not be reused." [p. 20]

Again, we agree with Hoang et al.; our conclusion is that one should not call any AE scheme *misuse-resistant*, neither on-line nor off-line.

3 Preliminaries

3.1 AE Definitions

Throughout this work, we use lower case for literals, upper case for strings and calligraphic font for sets and algorithms. We write $X \twoheadleftarrow \mathcal{X}$ to say that we sample X from a set $\mathcal{X}$ uniformly at random.

We denote by $\mathcal{N} \subseteq \{0,1\}^n$ a non-empty nonce space. For some $k \geq 1$, $\tau \geq 0$, we call $\mathcal{K} \subseteq \{0,1\}^k$ and $\mathcal{T} = \{0,1\}^\tau$ key and tag space. Further, we write $\tilde{\mathcal{A}}, \mathcal{M}, \mathcal{C} \subseteq \{0,1\}^*$ for associated-data, message and ciphertext spaces. The header space is defined to contain associated data and nonce $\mathcal{H} = \tilde{\mathcal{A}} \times \mathcal{N}$.

An adversary $\mathcal{A}$ is an efficient algorithm that interacts with a given set of oracles, which appear as black boxes to $\mathcal{A}$. We write $\mathcal{A}^{\mathcal{O}}$ for the output of $\mathcal{A}$ after interacting with some oracle $\mathcal{O}$. $\mathcal{O}_i \hookrightarrow \mathcal{O}_j$ denotes that $\mathcal{A}$ first queries $\mathcal{O}_i$ and later $\mathcal{O}_j$ with the output of $\mathcal{O}_i$. Wlog., we assume that adversaries never ask queries to which they already knows the answer. If the oracles $\mathcal{O}_i$, $\mathcal{O}_j$ represent a family of algorithms indexed by inputs (nonce, header, key), the indices must match. For example when $\mathcal{E}_K(H,M)$ and $\mathcal{D}_K(H,M)$ represent encryption and decryption algorithms with a fixed K and indexed by H, then $\mathcal{E}_K \hookrightarrow \mathcal{D}_K$ says that $\mathcal{A}$ first queries $\mathcal{E}_K(H,M)$ and later $\mathcal{D}_K(H,M)$.

Definition 1 (OAE1 Scheme [13]). *An OAE1 scheme is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with a non-empty key space $\mathcal{K}$, a deterministic encryption algorithm*

$$\mathcal{E} : \mathcal{K} \times \mathcal{H} \times \mathcal{M} \to \mathcal{M} \times \mathcal{T},$$

and a deterministic decryption algorithm

$$\mathcal{D} : \mathcal{K} \times \mathcal{H} \times \mathcal{M} \times \mathcal{T} \to \mathcal{M} \cup \{\perp\}.$$

The validity condition is the following: $\forall K \in \mathcal{K}$, $\forall H \in \mathcal{H}$, $\forall M \in \mathcal{M}$: it holds that $\mathcal{D}_K(H, \mathcal{E}_K(H,M)) = M$.

Definition 2 (OAE2 Scheme [13]). *Fix $\sigma, \tau \geq 0$. An OAE2 scheme is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with a non-empty key space $\mathcal{K}$, a triple $\mathcal{E} = (\mathcal{E}_{init}, \mathcal{E}_{next}, \mathcal{E}_{final})$, and a triple $\mathcal{D} = (\mathcal{D}_{init}, \mathcal{D}_{next}, \mathcal{D}_{final})$ of deterministic algorithms with signatures:*

$$\begin{array}{ll} \mathcal{E}_{init} : \ \mathcal{K} \times \mathcal{H} \to \mathcal{S} & \mathcal{D}_{init} : \ \mathcal{K} \times \mathcal{H} \to \mathcal{S} \\ \mathcal{E}_{next} : \ \mathcal{K} \times \mathcal{S} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}_1 \times \mathcal{S} & \mathcal{D}_{next} : \ \mathcal{K} \times \mathcal{S} \times \mathcal{C} \times \mathcal{T}_1 \to (\mathcal{M} \times \mathcal{S}) \cup \{\perp\} \\ \mathcal{E}_{final} : \mathcal{K} \times \mathcal{S} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}_2 & \mathcal{D}_{final} : \mathcal{K} \times \mathcal{S} \times \mathcal{C} \times \mathcal{T}_2 \to \mathcal{M} \cup \{\perp\} \end{array}$$

with a header space $\mathcal{H} = \mathcal{N} \times \mathcal{A}$, nonce space $\mathcal{N}$, AD space $\tilde{\mathcal{A}}$, a state space $\mathcal{S}$, for $\mathcal{N}, \tilde{\mathcal{A}}, \mathcal{S} \subseteq \{0,1\}^*$, and tag spaces $\mathcal{T}_1 \subseteq \{0,1\}^\sigma$, $\mathcal{T}_2 \subseteq \{0,1\}^\sigma$.

Validity condition: Consider a sequence $(M_1, \ldots, M_m)$ of message segments, a key $K \in \mathcal{K}$ and header $H \in \mathcal{H}$. Encrypt $(M_1, \ldots, M_m)$ to $(C_1, T_1, C_2, T_2, \ldots, C_{m-1}, T_{m-1}, C_m, T)$ by calling $\mathcal{E}_{init}$ once, $\mathcal{E}_{next}$ $(m-1)$-times, and $\mathcal{E}_{final}$ once. Decrypt $(C_1, T_1, C_2, T_2, \ldots, C_{m-1}, T_{m-1}, C_m, T)$ by calling $\mathcal{D}_{init}$ once, $\mathcal{D}_{next}$ $(m-1)$-times, and $\mathcal{D}_{final}$ once. Then the result must be the same sequence $(M_1, \ldots, M_m)$ as at the beginning.

Separated AE Schemes. Andreeva et al [4] proposed to separate the decryption and verification algorithms for easier analysis of release of unverified plaintext. A separated AE scheme is defined as a tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{V})$ with a non-empty key space $\mathcal{K}$ and deterministic encryption, decryption, and verification algorithms with signatures

$$\begin{aligned} \mathcal{E} &: \mathcal{K} \times \mathcal{H} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}, \\ \mathcal{D} &: \mathcal{K} \times \mathcal{H} \times \mathcal{C} \times \mathcal{T} \to \mathcal{M}, \\ \mathcal{V} &: \mathcal{K} \times \mathcal{H} \times \mathcal{C} \times \mathcal{T} \to \{\texttt{true}, \perp\}. \end{aligned}$$

$\mathcal{D}_K(H, C, T)$ always returns some message $M \in \mathcal{M}$; the verification algorithm $\mathcal{V}_K(H, C, T)$ returns `true` if (H, C, T) is valid and $\perp$ otherwise. For all $K \in \mathcal{K}$, $M \in \mathcal{M}$, $H \in \mathcal{H}$ holds that $\mathcal{D}_K(H, \mathcal{E}_K(H, M)) = M$.

3.2 Security Definitions

For $n \geq 1$, let $\{0,1\}^n$ denote the set of n-bit blocks. Similar to Bellare et al. [5], we define OPerm_n as the set of all length-preserving permutations π on $(\{0,1\}^n)^*$ where the i-th output block of $\pi(\cdot)$ depends only on the input blocks 1 to i.

Definition 3 (OPERM-CCA Security [1]). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an on-line AE scheme, and $\mathcal{A}$ a computationally bounded adversary with access to two oracles $\mathcal{O}_1, \mathcal{O}_2$, such that $\mathcal{A}$ never queries $\mathcal{O}_1 \hookrightarrow \mathcal{O}_2$. Let π be a family of independent random on-line permutations. This means that for each header $H \in \mathcal{H}$, $\pi(H, \cdot)$ is a random on-line permutation with the inverse $\pi^{-1}(H, \cdot)$. Then, we define the OPERM-CCA advantage of an adversary $\mathcal{A}$ by*

$$\mathbf{Adv}_{\Pi}^{\textsf{OPERM-CCA}}(\mathcal{A}) = \left| \Pr\left[\mathcal{A}^{\mathcal{E}_K, \mathcal{D}_K} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\pi, \pi^{-1}} \Rightarrow 1\right] \right|,$$

where the probabilities are taken over $K \twoheadleftarrow \mathcal{K}$ and $\forall H \in \mathcal{H} : \pi(H, \cdot) \twoheadleftarrow \mathsf{OPerm}_n$.

Clearly, an upper bound for the maximal OPERM-CCA advantage of some adversary $\mathcal{A}$ on Π is always also an upper bound for the maximal OPERM-CPA advantage of $\mathcal{A}$ on Π.

Definition 4 (INT-RUP Advantage [4]). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{V})$ be a separated AE scheme, and $\mathcal{A}$ a computationally bounded adversary with access to three oracles $\mathcal{O}_1, \mathcal{O}_2$, and $\mathcal{O}_3$ such that $\mathcal{A}$ never queries $\mathcal{O}_1 \hookrightarrow \mathcal{O}_3$. Then, the INT-RUP advantage of $\mathcal{A}$ with respect to Π is defined as*

$$\mathbf{Adv}_{\Pi}^{\textsf{INT-RUP}}(\mathcal{A}) := \Pr\left[\mathcal{A}^{\mathcal{E}_K, \mathcal{D}_K, \mathcal{V}_K} \text{ forges}\right].$$

where the probability is taken over $K \twoheadleftarrow K$, and random coins of $\mathcal{A}$. "Forges" means that $\mathcal{V}_K$ returns `true` for a query of $\mathcal{A}$.

Note that security under release of unverified plaintext (INT-RUP) is contained in similar form also in the notion of robust AE by Hoang, Krovetz, and Rogaway [12].

Definition 5 (OAE1$^+$). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an OAE1 scheme as defined in Definition 1. We call Π an OAE1$^+$ scheme iff it provides OPERM-CCA and INT-RUP security.*

Definition 6 (OAE2$^+$). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an OAE2 scheme as defined in Definition 2. We call Π an OAE2$^+$ scheme iff it provides OPERM-CCA and INT-RUP security.*

We call an on-line AE scheme providing OPERM-CCA security and INT-RUP security *decryption-robust.* Table 1 provides an overview of OAE1, OAE2, OAE1$^+$, and OAE2$^+$.

Properties		**OAE1** [10]	**OAE1$^+$**	**OAE2** [13]	**OAE2$^+$**
Functional	Definitional Idea	On-line cipher + tag		Segments auth. + encr.	
	Segmentation	Fixed-size blocks		Variable-size segments	
	Ciphertext Expansion	Constant		Linear in m=(#segments)	
	in bit	τ		$(m-1) \cdot \tau + \sigma$	
	Message Space	$(\{0,1\}^n)^*$		$\{0,1\}^*$	
Security	Implies Online Decryption	–	–	✓	✓
	Can AEncrypt Streams	–	–	✓	✓
	Resistant to CPSS	–	–	–	–
	Implies Decryption-Robustness	–	✓	–	✓

Table 1. Comparison of the OAE security definitions.

3.3 Decryption Robustness

Note that an OAE2 scheme that avoids decryption misuse has better chances to be implemented properly than an arbitrary AE scheme. Of course, all segments have to be reasonably small and the size τ of intermediate tags must be large enough to neglect a $2^{-\tau}$ probability. But decryption robustness is about unverified plaintexts being compromised, *in spite* of invalid authentication tags. In such cases, the security assurance provided by OAE2 is actually somewhat paradoxical: it depends on the size τ of intermediate tags:

– If τ is small then OAE2 security implies decryption robustness, i.e., $\mathsf{OAE2} = \mathsf{OAE2}^+$.
– If τ is large, then OAE2 security does not imply decryption robustness, i.e., $\mathsf{OAE2} \neq \mathsf{OAE2}^+$.

If τ is small, the probability $2^{-\tau}$ to accept a forged ciphertext segment is significant, and the decryption of the forged ciphertext segment can be released. If the adversary could recover any useful information from the released plaintext segment, or even distinguish it from a random plaintext segment of the same size, this would break OAE2 security. As a matter of fact, this is precisely the reason why OAE1 with blocksize and tagsize n is weaker than OAE2 with 0-bit intermadiate tags, see "Comparing OAE1 and OAE2" in Section 2 above, and [13, Appendix D].

On the other hand, a large τ implies a negligible probability $2^{-\tau}$ to accept a forged ciphertext segment. Now, the OAE2 security definition assumes that no information about the corresponding plaintext segment leaks to the adversary. However, if there is such a leak, *in spite of the OAE2 security definition*, the entire OAE2 security assurance is void.

We stress that this is more than just theoretical nitpicking. If the size of intermediate tags is large, one can, for example construct an OAE2 scheme by applying the SIV mode to each segment (cf. [13, Section 6]). SIV is an uses counter mode for the underlying encryption operation. Under decryption misuse, this provides about as much security as the one-time-pad used twice: almost none! With each decryption-misuse query, the adversary can recover one secret message segment. In fact, decryption-misuse would turn the security of an OAE2 scheme into the security of a OAE1d scheme without decryption misuse (cf. [13, Section 8]).

This is not a general issue for OAE2. It is a specific issue for OAE2 schemes which fail to provide decryption robustness for their segment-wise encryption. A counter-example would be another instantiation from [13, Section 6], based on the CAESAR candidate AEZ. It retains decryption robustness, regardless of the intermediate-tag size (assuming the security claims for AEZ hold, of course).

4 Two Attacks: CPSS and CPOS

4.1 The CPSS Attack

In a setting where nonces and headers repeat (i.e. under nonce-misuse), an authenticated on-line cipher can be attacked *if the first plaintext block with non-zero entropy has low entropy.* In many practical settings, this may be an unlikely event. But in some models and practical settings, the adversary may actually make such an event happen.

The Chosen-Prefix Secret-Suffix (CPSS) attack from Reyhanitabar and Vizár [15] demonstrates this (see Figure 1). Consider an on-line AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ operating on a fixed block size n. There is a secret S of ℓ_S bit length. The adversary $\mathcal{A}$ will ask for the encryption of a plaintext $M = P \mid\mid S$, where the length of the known prefix P is $|P| = n - b$ bit, with $b \geq 1$ being as small as possible (e.g., $b = 8$). Thus, the first block of M has at most b bit of entropy, and the adversary can recover the first b bit of S with 2^b further queries. Then, $\mathcal{A}$ will use a new prefix P of length $|P| = n - 2b$ to recover the next b bit of S etc.

We stress that the attack works if and only if the following three assumptions hold:

1. All queries by $\mathcal{A}$ use the same nonce and the same header.
2. $\mathcal{A}$ can choose messages M with a known prefix P, followed by the secret S of interest to $\mathcal{A}$.
3. $\mathcal{A}$ can choose the length of P.

Under these assumptions, the CPSS attack applies to all OAE schemes.

4.2 CPOS Attack

The CPSS attack is not applicable to (off-line) AE (MRAE) schemes. The reason is that there is no low-entropy "first block"; any message $M = P \mid\mid S$ with known P has the full entropy of S. Nevertheless, a slight variation of the CPSS attack applies to all AE schemes, including off-line ones, and even including the so-called misuse-resistant AE (MRAE) schemes.

```
10: function Initialize($\mathit{\Pi} = (\mathcal{K}, \mathcal{E}, \mathcal{D}), \ell_S$)
11:     $K \twoheadleftarrow \mathcal{K}$
12:     $S \twoheadleftarrow \{0,1\}^{\ell_S}$
13: end function
20: function Query($N, A, P$)
21:     $M \leftarrow P \,||\, S$
22:     $C \leftarrow \mathcal{E}_K(N, A, M)$
23:     return $C$
24: end function
30: function Finalize($S'$)
31:     return $S \overset{?}{=} S'$
32: end function
```

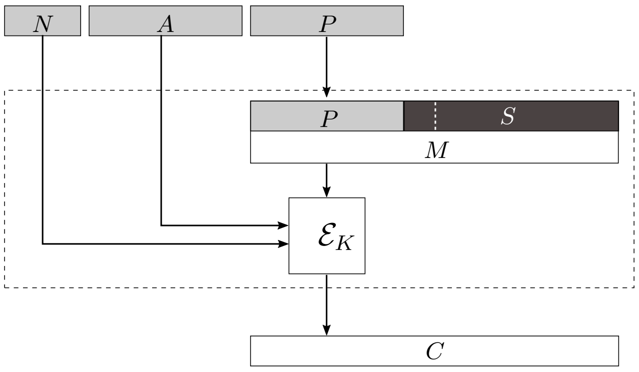

Fig. 1. The setting for a CPSS attack. Elements in light gray are controlled by the adversary $\mathcal{A}$. Dark Elements represent secrets beyond $\mathcal{A}$'s control. The dashed line within the secret S indicates a block boundary after the first byte, or the first few bytes, of S. $\mathcal{A}$ wins by eventually recovering the full S.

The point is, if *the entire message has low entropy* and if nonces and headers repeat, then all encryption schemes are vulnerable to chosen-plaintext attacks. This holds for authenticated as well as for plain encryption, and for on-line as well as for off-line schemes.

Instead of prepending P to S, we use P to *overwrite* parts of S as shown in Figure 2, which is why we call this the Chosen-Position Overwrite-Secret (CPOS) attack. As above, $b > 1$ is as small as possible, e.g., eight bits. Consider an arbitrary AE scheme $\mathit{\Pi} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and a secret S of length $|S| = \ell_S$. For simplicity, we assume b divides ℓ_S, and we write $S = S[0..\ell_S] = (S_0, \ldots, S_{\ell_S} - 1)$. The adversary $\mathcal{A}$ will choose some P of length $|P| = \ell_S - b$ bit and ask for the encryption of $M = P \,||\, S_{\ell_S - 1}$. Since the entire message M has at most b bit of entropy, $\mathcal{A}$ can find S_{ℓ_S-1} with 2^b queries. Next, $\mathcal{A}$ chooses some P of length $|P| = \ell_S - 2b$ bit and asks for the encryption of $M = P \,||\, S_{\ell_S-2} \,||\, S_{\ell_S-1}$ to recover S_{ℓ_S-2} etc. It is easy to see that $\mathcal{A}$ can recover S in this way after at most $2^b \ell_S / b$ queries.

```
10: function Initialize($\mathit{\Pi} = (\mathcal{K}, \mathcal{E}, \mathcal{D}), \ell_S$)
11:     $K \twoheadleftarrow \mathcal{K}$
12:     $S \twoheadleftarrow \{0,1\}^{\ell_S}$
13: end function
20: function Query($N, A, P$)
21:     $\ell_P \leftarrow |P|$
22:     $M \leftarrow S$
23:     $M[0..\ell_P - 1] \leftarrow P$
24:     $C \leftarrow \mathcal{E}_K(N, A, M)$
25:     return $C$
26: end function
30: function Finalize($S'$)
31:     return $S \overset{?}{=} S'$
32: end function
```

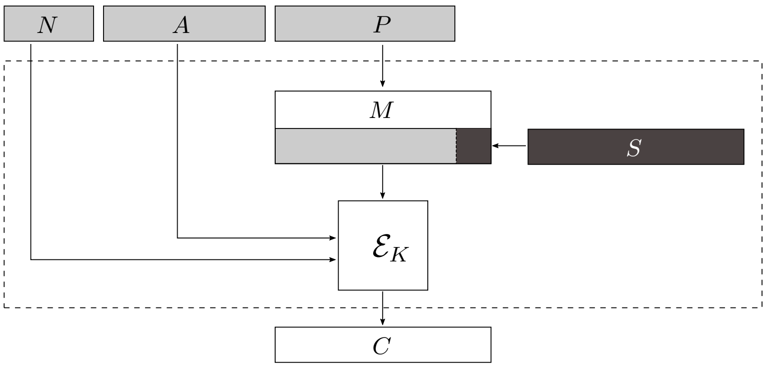

Fig. 2. The setting for a CPOS attack. Elements in light gray are controlled by $\mathcal{A}$, dark elements are secret. The arrow from S to the left indicates M being set to S, the arrow from P down to M indicates M being partially overwritten, with a part of S remaining intact. Lines 22 and 23 are equivalent to $M \leftarrow P \,||\, S[\ell_P..\ell_S - 1]$. $\mathcal{A}$ wins by recovering S.

4.3 Practicality

We are not aware of any practical instance of either the CPSS or the CPOS attack. But note that most applications of AE schemes use schemes that are not robust, anyway. If one of these applications reuses a nonce, then usually other attacks apply, less sophisticated than either the CPSS or the CPOS attack. On the other hand, the setting that the adversary can control a part of the message, while another part holds a secret of interest to the adversary, is a common pattern for many practical attacks. Nice examples are attacks against the TLS record protocol.

Hoang et al. point out that the CPSS attack has been "inspired by the well-known BEAST (Browser Exploit Against SSL/TLS) attack". To the best of our understanding, the BEAST attack is based on *predictable* IVs for CBC encryption, i.e., it does not actually exploit nonce-reuse for a nonce-based AE scheme at all. Moreover, it is not clear to us where the header (or "associated data") would come into play. Nevertheless, a vital part of the BEAST attack is based on controlling the length of a prefix, such that the first message block with non-zero entropy has low entropy. This is essentially the same as setting for the CPSS attack.

Another example is the well-known heartbleed attack. To find out if the other party is still active, a user chooses a challenge P and receives the encryption of P as a so-called heartbeat package. Due to an implementation error in the OpenSSL library, the attacker could receive the encryption of $(P \mid\mid S')$, with S' being a part of a secret $S = (S' \mid\mid S'')$. Both the length $|P|$ of P and the length $|S'|$ of S' are under full control of the adversary (up to a certain maximum for $|P| + |S'|$). In the specific case of heartbleed, the adversary has the secret key to decrypt $(P \mid\mid S')$, so applying either the CPSS or CPOS attack would be overly sophisticated.

But without the secret key, and assuming nonces and headers repeat and the secret S doesn't change between queries, the adversary could use the CPSS attack if the encryption scheme in use was on-line, and it could use the CPOS attack, regardless of the encryption scheme in use. In either case, the adversary would eventually recover S with roughly the same amount of work for either CPSS or CPOS.

4.4 Robustness to Nonce-Reuse

The *difference* between the CPSS and the CPOS attacks shows that robust on-line AE is necessarily weaker than robust off-line AE. The *similarity* between the two attacks indicates that robust on-line AE is not much weaker than robust off-line AE. In [13] Hoang et al. make the following argument, that we find very convincing.

> "[...] we find it is reasonable to term an AE scheme nonce-reuse misuse-resistant if what it achieves when nonces are reused is comparable to what an nAE scheme achieves with a proper nonce." [p. 19]

As the CPOS attack shows, there cannot be any AE scheme, which provides under nonce-reuse comparable security to an ordinary nonce-based AE scheme without nonce-reuse. Even robust off-line AE ("MR"AE) fails this criterion. For this reason, we prefer the word "robust" over "misuse-resistant", In any case, we stress that our view of robustness is that of a safety net, or of a second line of defense. Sometimes, nonces are reused due to an occasional glitch. In that case, both robust off-line AE and robust on-line AE should still provide reasonable security, while the security of ordinary non-robust schemes is likely to break apart.

5 Transforming a Decryption-Robust OAE1 Scheme into an OAE2 Scheme

Before we show how one can transform an OAE1 scheme into an OAE2 scheme, we clarify why this transformation does only work if the OAE1 scheme provides decryption robustness.

Necessity of Decryption Robustness. Assume an OAE1 scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ which only provides OPERM-CPA and INT-CTXT security (usually denoted by CCA3 security) but not OPERM-CCA and INT-RUP (which we call decryption robustness). We try to transform $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ into an OAE2 scheme by adding well-formed redundancy r to the message. More detailed, every μ bits we add a new "message block" r of σ bits to an input message M. For simplicity, we consider only one-block messages and we say that $\sigma = n$, where n denotes the block size of the underlying primitive.

Then, assume an OPERM-CCA adversary $\mathcal{A}$ which is capable of choosing a pair (S_1, T_1), where S_1 denotes the first segment of a ciphertext and T_1 its corresponding intermediate tag. If $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is not OPERM-CCA-secure, $\mathcal{A}$ can find a pair (S_1, T_1) so that $E_K^{-1}(T_1, X_1) = r$, where X_1 can be seen as tweaks which may be derived from former message/ciphertext blocks (in S_1), and E_K^{-1} is the inverse of the fixed-input/output length primitive. It is easy to see that $\mathcal{A}$ can efficiently produce segments and intermediate tags which will be accepted by the verify procedure of $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$.

The Transformation. In the next part we show that it is possible to transform an OPERM-CCA and INT-RUP-secure OAE1 scheme into an OAE2 scheme by adding well-formed redundancy to the message before encryption. Let $\Pi_{\mathsf{OAE1}} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ denote an OAE1 scheme as defined in Definition 1, where $\mathcal{K}$ denotes the key-generation step and $\mathcal{E}$ and $\mathcal{D}$ the procedures **Encrypt** and **DecryptAndVerify** as defined in Algorithm 1, respectively. Furthermore, let $E : \{0,1\}^k \times \{0,1\}^+ \times \{0,1\}^* \to \{0,1\}^n$ denote a block cipher used by Π encrypting at least the current message block $m_i \in \{0,1\}^n$ to its corresponding ciphertext block $c_i \in \{0,1\}^n$ under a secret key $K \in \{0,1\}^k$ with $1 \leq i \leq \ell$ and $\ell = |M|/n$. Additionally, let E_K^{-1} be the inverse of E_K where $E_K^{-1}(E_K(m, \cdot), \cdot)$ always holds.

To transform $\Pi_{\mathsf{OAE1}} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ into an OAE2 scheme $\Pi_{\mathsf{OAE2}} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, we redefine the procedures **Encrypt** and **DecryptAndVerify** as shown in Algorithm 2, where the procedure **ExpandMessage** is used to add redundancy to the input message M as shown in Algorithm 2 (Lines 201-208), the procedure **PadMessage**

Algorithm 1 The procedures **Encrypt** and **DecryptAndVerify** of a generic scheme $\Pi_{\mathsf{OAE1}} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$.

```
Encrypt($M$)
101: $\ell \leftarrow |M|/n$
102: for $i \leftarrow 1, \ldots, \ell$ do
103:     $c_i \leftarrow E_K(m_i, \cdot)$
104: end for
105: $C \leftarrow c_1 \,||\, \ldots \,||\, c_\ell$
106: $T \leftarrow$ GenerateTag($M$)
107: return $(C, T)$
```

```
DecryptAndVerify($C, T$)
201: $\ell \leftarrow |C|/n$
202: for $i \leftarrow 1, \ldots, \ell$ do
203:     $m_i \leftarrow E_K^{-1}(c_i, \cdot)$
204: end for
205: $M \leftarrow m_1 \,||\, \ldots \,||\, m_\ell$
206: $T' \leftarrow$ GenerateTag($M$)
207: if $T' = T$ then
208:     return $M$
209: else
210:     return $\perp$
211: end if
```

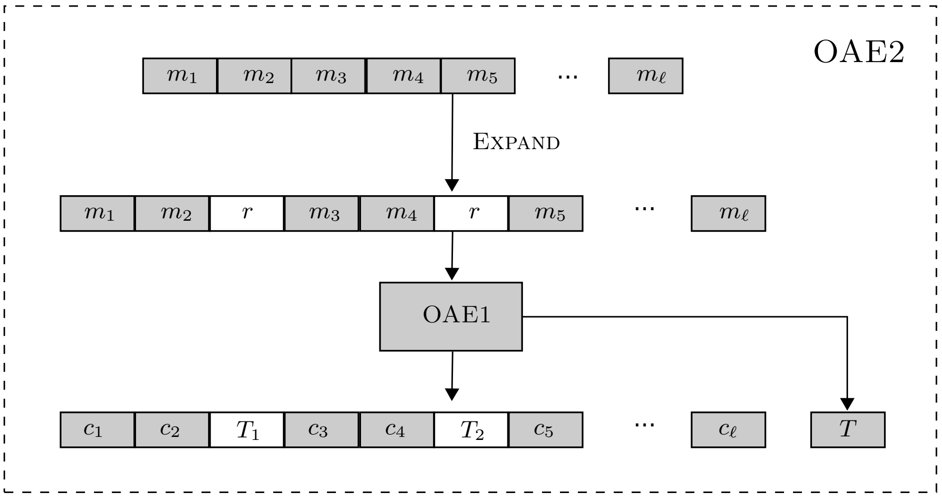
OAE2
m_1 m_2 m_3 m_4 m_5 ... m_ℓ
Expand
m_1 m_2 r m_3 m_4 r m_5 ... m_ℓ
OAE1
c_1 c_2 T_1 c_3 c_4 T_2 c_5 ... c_ℓ T

Fig. 3. Realizing intermediate tags by adding redundancy to the message.

returns a message which is a multiple of the block length (we do not provide an extra figure since it depends on the used OAE2 scheme), and the procedure **ExtractMessage** removes the redundancy of a valid message. We denote by r the so called redundancy block which is added to a message M in a way so that after each μ bits of M, a segment r with $|r| = \sigma$ is added. Thus, σ denotes the size of the intermediate tags in bits and μ determines their *frequency*. Note that if $|M|$ is a multiple of μ, r would be naturally added after the final message block m_ℓ with $\ell = |M|/n$. But, since each AE scheme is defined by a tag generation step after the message is fully processed, this last block r can be neglected. Therefore, a message consisting of $|M|$ bits is first expanded to a message M' of size $|M| + (\lceil |M|/\mu \rceil - 1) \cdot \sigma$ bits, and then encrypted to a ciphertext of the same size, consisting of $\lceil |C|/\mu \rceil - 1$ intermediate tags T_i. In this work, for simplicity we assume that $\sigma = n$ and that μ is a multiple of n, where n denotes the block size of the underlying primitive (see Figure 3 for an illustration). Based on the definition of OAE2 in [13], the procedure **DecryptAndVerify** returns all decrypted and valid segments of a ciphertext until the first intermediate tag is invalid (see Lines 406 and 409 of Algorithm 2).

Note that the values μ and σ are parameters of the OAE2 scheme. Moreover, to remain secure in the sense of OAE2, the scheme has to provide domain separation between different parameter sets of μ and σ. If the adversary could encrypt messages under μ and σ, and request the decryption under $\mu' \neq \mu$ or $\sigma' \neq \sigma$, using the same key and the same header, trivial attacks could become possible. A possible approach is shown in Section 6 for POET, where sessions keys are derived from a single master key, and depend on σ and μ.

It is easy to see from Algorithm 2 that one can transform an OAE1$^+$ scheme, based on an underlying fixed-output-length primitive, into an OAE2$^+$ scheme that encrypts/decrypts segments of arbitrary size with intermediate tags for each segment.

6 POET is an OAE1$^+$ Scheme

In [2], Abed et al. have shown that POET is an OAE scheme which provides OPERM-CCA and INT-CTXT security. Figure 4 recalls the message-encryption and tag-generation step of POET. In this section, we argue that POET also provides INT-RUP security, which renders POET an OAE1$^+$ scheme. For the sake of simplicity, we only provide arguments for the version of POET where message lengths are a multiple of the block length n

Algorithm 2 The procedures **Encrypt** and **DecryptAndVerify** of $\Pi_{\mathsf{OAE2}} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$.

```
Encrypt($M$)
101: $M' \leftarrow$ ExpandMessage($M$)
102: $M' \leftarrow$ PadMessage($M'$)
103: $\ell \leftarrow |M'|/n$
104: for $i \leftarrow 1, \ldots, \ell$ do
105:     $c_i \leftarrow E_K(m_i, \cdot)$
106: end for
107: $C \leftarrow c_1 \,||\, \ldots \,||\, c_\ell$
108: $T \leftarrow$ GenerateTag($M'$)
109: return $(C, T)$

ExpandMessage($M$)
201: $b \leftarrow 1$
202: $M' \leftarrow \emptyset$
203: $\ell \leftarrow \lceil |M|/\mu \rceil - 1$
204: for $i \leftarrow 1, \ldots, \ell$ do
205:     $M' \leftarrow M' \,||\, M[b, \ldots, b + \mu - 1] \,||\, r$
206:     $b \leftarrow b + \mu$
207: end for
208: return $M'$

ExtractMessage($M$)
301: $b \leftarrow 1$
302: $\ell \leftarrow \lceil |M|/(\mu + \sigma) \rceil - 1$
303: $M' \leftarrow \emptyset$
304: for $i \leftarrow 1, \ldots, \ell$ do
305:     $M' \leftarrow M' \,||\, M[b, \ldots, b + \mu - 1]$
306:     $b \leftarrow b + \mu + \sigma$
307: end for
308: return $M'$

DecryptAndVerify($C, T$)
401: $\ell \leftarrow |C|/n$
402: $M \leftarrow \emptyset$, $S \leftarrow \emptyset$
403: $s \leftarrow \lceil (\mu + \sigma)/n \rceil$
404: for $i \leftarrow 1, \ldots, \ell$ do
405:     $m_i \leftarrow E_K^{-1}(c_i, \cdot)$
406:     $S \leftarrow S \,||\, m_i$
407:     if $i \bmod s = 0$ then
408:         if not CheckSegment($S$) and $(i \neq \ell)$ then
409:             return ExtractMessage($M$)
410:         end if
411:         $M \leftarrow M \,||\, S$
412:         $\ell_S \leftarrow |S|$
413:         $S \leftarrow \emptyset$
414:     end if
415: end for
416: $T' \leftarrow$ GenerateTag($M$)
417: if $T' = T$ then
418:     return ExtractMessage($M$)
419: else
420:     return ExtractMessage($M[1, \ldots, \ell - \ell_S]$)
421: end if

CheckSegment($S$)
501: return $S[\mu, \ldots, \mu + \sigma - 1] = r$
```

of the cipher. Finally, we briefly discuss that POET can also be transformed into an OAE2$^+$ scheme by applying the transformation explained in Section 5.

POET Is INT-RUP-Secure (Proof Sketch). The gap between INT-CTXT and INT-RUP security results from the fact that the decryption oracle always returns the would-be plaintext in the latter setting. So, the advantage of an adversary in the INT-CTXT in comparison to that in the INT-RUP setting can only be increased by this additional knowledge about invalid plaintexts from the decryption oracle. Since POET is OPERM-CCA-secure, the only thing an adversary can observe from such invalid decryptions is the longest common prefix (LCP) of header and message with other queries. Starting from the first block beyond the LCP, every message block cannot be distinguished from the output of a random on-line permutation. Let $\mathcal{A}$ be an INT-RUP adversary that asks at most q queries of total length at most ℓ blocks to its available oracles; $\mathcal{A}$ maintains a query history $\mathcal{Q}$ where it stores its queries together with the oracles corresponding responses as tuples (H, M, C, T). $\mathcal{A}$ can win the INT-RUP game only if it can construct a *winning* query $Q = (H, *, C, T)$ that is accepted by the verification oracle. We have to consider the following cases for its winning query:

- **H is fresh, i.e., $(H, *, *, *) \notin \mathcal{Q}$:**
  - We let an adversary $\mathcal{A}$ win if it finds two distinct headers H, H' such that **ProcessHeader**(H) = **ProcessHeader**(H'). Since the **ProcessHeader** function of POET is PMAC1, the success probability of $\mathcal{A}$ in this case is upper bounded by $\ell^2/2^n$ [18]. Thus, in all following cases, we assume that the header is old.
  - $\mathcal{A}$ finds a collision between the final input block X_ℓ and any other input block X_i. Since the top-row chaining function $F_{K_1}(X_\ell) \oplus \tau \oplus L_T$ is an ϵ-AU family of hash functions, the probability for this event can be upper bounded by $\epsilon\ell^2/2$.
  - If X_ℓ is fresh, the output Y_ℓ is a random value. Thus, $\mathcal{A}$ can only win if it guesses the tag T, which on the other hand can be upper bound by $q/(2^n - (\ell + 2q))$.
- **(H, C) is fresh, i.e., $(H, C, *, *) \notin \mathcal{Q}$:**
  - $\mathcal{A}$ finds a collision between the last input block X_ℓ and any other input block X_i. Since the top-row chaining forms an ϵ-AU family of hash functions, this event can be upper bounded by $\ell^2/2\epsilon$.
  - If X_ℓ is fresh, the output Y_ℓ is a random value. Thus, $\mathcal{A}$ can only win if it guesses the tag T, which on the other hand can be upper bound by $q/(2^n - (\ell + 2q))$.

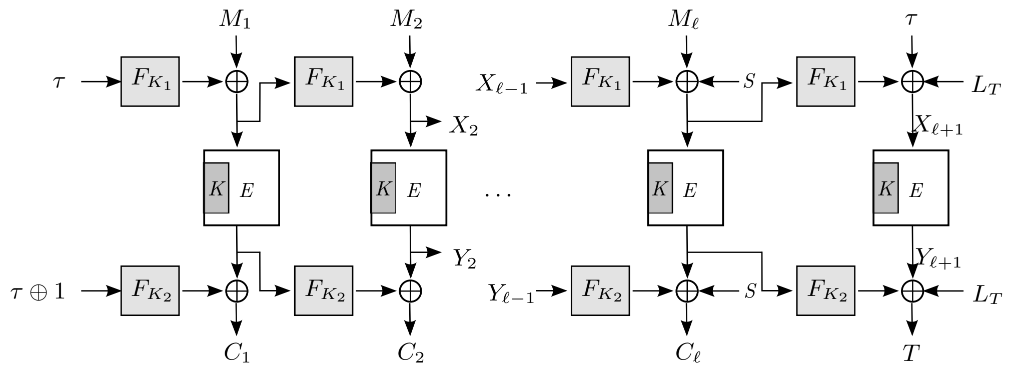

Fig. 4. The encryption and authentication process for an ℓ-block message M with POET. τ denotes the result of processing the header, K_1, K_2, K, L_T are derived from encrypting distinct constants under the secret key, and $S = E_K(|M|)$ is the encrypted message length.

– **(H, C) is old, i.e., $(H, C, *, *) \in \mathcal{Q}$:**
  - In this case, the tag T has to be fresh. Since POET matches each pair (H, C) to a unique tag value, the success probability of $\mathcal{A}$ is 0 in this case.

This means that an adversary cannot gain any additional advantage from the would-be plaintexts that increases its advantage from the INT-CTXT to the INT-RUP game. Note that this observation is specific for POET; as we show in Appendix A, one cannot generically assume that OPERM-CCA and INT-CTXT imply INT-RUP security.

POET Can Be Transformed Into an OAE2$^+$ Scheme. In this part we have argued that POET satisfies the properties of an OAE1$^+$ scheme. Moreover, in Section 5 we have shown that each OAE1$^+$ scheme can be transformed into an OAE2$^+$ scheme by adding well-formed redundancy to the input message under the restriction that the scheme provides domain separation for different sets of the parameters μ and σ. For POET, we realize this domain separation by using five guaranteed-to-be-distinct session keys which are derived from the secret key. More detailed, the sessions keys are generated by $K_i \leftarrow E_K(const_i)$ with $0 \le i \le 4$, where K denotes the secret key, K_i the i-th session key, E the underlying primitive, e.g., the AES, and we set $const_i = i + \mu \cdot 8 + 2^{64} \cdot \sigma$ for $\mu \in \{0, \dots, 2^{61} - 1\}$ and $\sigma \in \{0. \dots, 128\}$.

7 Robustness – The Cryptographic Safety Belt

Robustness to nonce-reuse must never be considered a free ticket to deliberate nonce-reuse – neither for on-line nor for off-line AE schemes. For this reason, the word *misuse-resistant* has been a dangerous misnomer, from early on. We propose to call this property *robust* instead: an even clearer wording would be *damage limitation*.

This implies that under accidental nonce-reuse, there are still good reasons to expect an acceptable level of security. Users of AE schemes which miss this form of robustness will, most likely, suffer a more severe damage. The fact that some protocols will break under nonce-reuse, even if one is using a scheme known to be nonce-reuse robust, does not disqualify a notion of robustness – not even OAE1. Using a robust scheme is similar to the case of wearing a safety belt; people get killed in traffic accidents, and sometimes even people wearing a safety belt get killed. But that does not disprove the fact that wearing a safety belt when a traffic accident occurs significantly reduces the risks.

When deciding about which AE scheme to use, we suggest to apply Algorithm 3.

We believe that the notion of *robustness* is of vital importance for the further development of practically useful and theoretically sound cryptosystems, and we expect the current discussion to terminate in a better understanding of *robustness*. We are thankful to Viet Tung, Reza, Phillip, and Damian for their inspiring work, which led us to reconsider our view on robustness, to revisit the formalism we worked with, and even to revisit our CAESAR candidate POET. It turned out that the formal security claims we made could be strengthened: We claimed and proved INT-CTXT, where INT-RUP notion by Andreeva et al. was more appropriate. Fortunately, it turned out that without much work, the INT-CTXT proof can be modified to actually show the stronger INT-RUP security. We are currently working on a minor tweak for POET to improve its performance, and will provide a full-fledged proof that the tweaked POET also provides INT-RUP-security.

Finally, we support the concluding remark from Hoang et al.:

> "The definitional enterprise in cryptography has always been a dialectical one. It should be understood that there is no insult in critiquing a definition; in fact, critique is acknowledgment that something has become significant enough to attend to." [p. 17]

Algorithm 3 Choosing an AE scheme, when considering nonce-reuse.

```
while you think nonce-reuse is OK do
    think again!
end while
if you worry about nonces being reused accidentally then
    don't panic!
    if you know the maximum message size, and you have no latency or storage issues with large messages then
        use a robust off-line AE scheme ("MRAE")
    else if you can afford a linear ciphertext expansion then
        use an $\mathsf{OAE2}^+$ scheme
    else
        use an $\mathsf{OAE1}^+$ scheme
    end if
else
    choose an AE scheme without considering robustness to nonce-reuse
end if
```

References

1. Farzanah Abed, Scott Fluhrer, Christian Forler, Eik List, Stefan Lucks, David McGrew, and Jakob Wenzel. Pipelinable On-Line Encryption. In *FSE*, 2014.
2. Farzaneh Abed, Scott Fluhrer, John Foley, Christian Forler, Eik List, Stefan Lucks, David McGrew, and Jakob Wenzel. The POET Family of On-Line Authenticated Encryption Schemes. `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
3. Farzaneh Abed, Christian Forler, and Stefan Lucks. Overview of the First-Round Candidates in the CAESAR Competition for Authenticated Encryption. Cryptology ePrint Archive, Report 2014/792, 2014. `http://eprint.iacr.org/`.
4. Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. pages 1–31, 2014.
5. Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. Online Ciphers and the Hash-CBC Construction. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 292–309. Springer, 2001.
6. Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. Online Ciphers and the Hash-CBC Construction. Cryptology ePrint Archive, Report 2007/197; full version of [5], 2007. `http://eprint.iacr.org/`.
7. Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.
8. Dan J. Bernstein. CAESAR Call for Submissions, Final, January 27 2014. `http://competitions.cr.yp.to/caesar-call.html`.
9. Nilanjan Datta and Mridul Nandi. ELmD. `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
10. Ewan Fleischmann, Christian Forler, and Stefan Lucks. McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In *FSE*, pages 196–215, 2012.
11. Ewan Fleischmann, Christian Forler, Stefan Lucks, and Jakob Wenzel. McOE: A Foolproof On-Line Authenticated Encryption Scheme. Cryptology ePrint Archive, Report 2011/644, 2011. `http://eprint.iacr.org/`.
12. Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust Authenticated-Encryption: AEZ and the Problem that it Solves. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8xxx, page 793, 2015. To appear.
13. Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. Cryptology ePrint Archive, Report 2015/189, 2015. `http://eprint.iacr.org/`.
14. Stefan Kölbl, Martin M. Lauridsen, Christian Rechberger, and Tyge Tiessen. Authenticated Encryption Zoo, 2014. `https://aezoo.compute.dtu.dk/`.
15. Reza Reyhanitabar and Damian Vizár. Careful with Misuse Resistance of online AEAD. Unpublished manuscript distributed on the crypto-competitions mailing list, August 24 2014.
16. Philip Rogaway. The Evolution of Authenticated Encryption. 10 January 2013.
17. Philip Rogaway. CAESAR candidate AEZ. In *Directions in Authenticated Ciphers (DIAC) 2014. Santa Barbara, USA*, 23–24 August 2014.
18. Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.
19. Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006.
20. Patrick P Tsang, Rouslan V Solomakhin, and Sean W Smith. Authenticated Streamwise On-Line Encryption. 2009.

A OPERM-CCA- + INT-CTXT-Security does not imply INT-RUP-Security

The question of how to provide authenticated on-line encryption (without stating that name) has already been studied in [6], the revised and full version of [5]. One of the ideas presented there was to prepend and append a random W to a message M and then to perform the on-line encryption of $(W \mid\mid M \mid\mid W)$. (For simplicity, assume W to be a single-block value and the length of M to be a multiple of the block size.) The final block of the ciphertext is, de-facto, the authentication tag. In a nonce-based setting, we would assume W is pseudorandomly generated from the nonce. It is easy to see that this construction is INT-CTXT-secure if the online permutation is OPERM-CPA. However, even if the on-line permutation is OPERM-CCA-secure, this scheme is not INT-RUP-secure under nonce-reuse. Consider the following attack:

1. Ask for the encryption of some message M. This will be a ciphertext C, corresponding to applying an on-line permutation to the input $(W \mid\mid M \mid\mid W)$, with W being unknown to the adversary.
2. Ask for the decryption of some ciphertext $(C \mid\mid C')$. This will *fail*, but in an INT-RUP attack, the adversary will learn W from the *unverified plaintext*.
3. Ask for the encryption of some message $(X \mid\mid W)$. Use the same nonce as in the first query. This will give some ciphertext C'', corresponding to applying the OPERM on $(W \mid\mid X \mid\mid W \mid\mid W)$.
4. Truncate C'' by removing the final block. Applying the inverse OPERM to C'' gives $(W \mid\mid X \mid\mid W)$, and this eventually decrypts to the *valid* plaintext X.

We stress that we do not invalidate any security claims in [6] who certainly did not propose their scheme with security under nonce and decryption misuse in their minds. However, our example shows that one cannot generically combine OPERM-CCA and INT-CTXT security i order to have INT-RUP security.