Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance

Viet Tung Hoang[1,2] Reza Reyhanitabar[3] Phillip Rogaway[4] Damian Vizár[3]

[1] Dept. of Computer Science, Georgetown University, USA
[2] Dept. of Computer Science, University of Maryland, College Park, USA
[3] EPFL, Lausanne, Switzerland
[4] Dept. of Computer Science, University of California, Davis, USA

Abstract. A definition of *online authenticated-encryption* (OAE), call it OAE1, was given by Fleischmann, Forler, and Lucks (2012). It has become a popular definitional target because, despite allowing encryption to be online, security is supposed to be maintained even if nonces get reused. We argue that this expectation is effectively wrong. OAE1 security has also been claimed to capture best-possible security for any online-AE scheme. We claim that this understanding is wrong, too. So motivated, we redefine OAE-security, providing a radically different formulation, OAE2. The new notion effectively *does* capture best-possible security for a user's choice of plaintext segmentation and ciphertext expansion. It is achievable by simple techniques from standard tools. Yet even for OAE2, nonce-reuse can still be devastating. The picture to emerge is that no OAE definition can meaningfully tolerate nonce-reuse, but, at the same time, OAE security ought never have been understood to turn on this question.

Keywords: Authenticated encryption, CAESAR competition, misuse resistance, nonce reuse, online AE, symmetric encryption.

1 Introduction

Between nAE & MRAE. With typical nonce-based authenticated-encryption (nAE) schemes [52, 54], nonces must never repeat when encrypting a series of messages; if they do, it is possible—and routine—that all security will be forfeit. To create some breathing room around this rigid requirement, Rogaway and Shrimpton defined a stronger authenticated-encryption (AE) notion, which they called *misuse-resistant AE* (MRAE) [55]. In a scheme achieving this, repeating a nonce has no adverse impact on authenticity, while privacy is damaged only to the extent that an adversary can detect repetitions of (N, A, M) triples, these variables representing the nonce, associated data (AD), and plaintext.

While it's easy to construct MRAE schemes [55], any such scheme must share a particular inefficiency: *encryption can't be online*. When we speak of

encryption being *online* we mean that it can be realized with constant memory while making a single left-to-right pass over the plaintext M, writing out the ciphertext C, also left-to-right, during that pass. The reason an MRAE scheme can't have online encryption is simple: the definition entails that every bit of ciphertext depends on every bit of the plaintext, so one can't output the first bit of a ciphertext before reading the last bit of plaintext. Coupled with the constant-memory requirement, single-pass MRAE becomes impossible.

Given this efficiency/security tension, Fleischmann, Forler, and Lucks (FFL) put forward a security notion [28] that slots between nAE and MRAE. We call it OAE1. Its definition builds on the idea of an *online cipher* due to Bellare, Boldyreva, Knudsen, and Namprempre (BBKN) [15]. Both definitions depend on a constant n, the *blocksize*. Let $\mathsf{B}_n = \{0,1\}^n$ denote the set of n-bit strings, or *blocks*. An *online cipher* is a blockcipher $\mathcal{E}\colon \mathcal{K} \times \mathsf{B}_n^* \to \mathsf{B}_n^*$ (meaning each $\mathcal{E}(K,\cdot)$ is a length-preserving permutation) where the ith block of ciphertext depends only on the key and the first i blocks of plaintext. An OAE1-secure AE scheme is an AE scheme where encryption behaves like an (N,A)-tweaked [43] online cipher of blocksize n followed by a random, (N,A,M)-dependent tag.

Problems with OAE1. FFL assert that OAE1 supports online-AE and nonce-reuse security. We disagree with the second claim, and even the first.

To begin, observe that as the blocksize n decreases, OAE1 becomes weaker, in the sense that the ability to perform a chosen-plaintext attack (CPA) implies the ability to decrypt the ciphertext of an m-block plaintext with $(2^n-1)m$ encryption queries. Fix a ciphertext $C = C_1 \cdots C_m T$ with $C_i \in \mathsf{B}_n$, a nonce N, and an AD A. Using just an encryption oracle Enc, we want to recover C's plaintext $M = M_1 \cdots M_m$ with $M_i \in \mathsf{B}_n$. Here's an attack for $n = 1$. If $\mathrm{Enc}(N,A,0) = C_1$ set $M_1 = 0$; otherwise, set $M_1 = 1$. Next, if $\mathrm{Enc}(N,A,M_1\,0) = C_1C_2$ set $M_2 = 0$; otherwise, set $M_2 = 1$. Next, if $\mathrm{Enc}(N,A,M_1M_2\,0) = C_1C_2C_3$ set $M_3 = 0$; otherwise, set $M_3 = 1$. And so on, until, after m queries, one recovers M. For $n > 1$ generalize this by encrypting $M_1 \cdots M_{i-1}\,M_i$ (instead of $M_1 \cdots M_{i-1}\,0$) with M_i taking on values in B_n until one matches $C_1 \cdots C_i$ or there's only a single possibility remaining. The worst-case number of Enc queries becomes $(2^n-1)m$. We call this the *trivial* attack.

The trivial attack might suggest hope for OAE1 security as long as the blocksize is fairly large, like $n = 128$. We dash this hope by describing an attack, what we call a *chosen-prefix / secret-suffix* (CPSS) attack, that breaks any OAE1-secure scheme, for any n, in the sense of recovering S from given an oracle for $\mathcal{E}_K^{N,A}(L\|\cdot\|S)$, for an arbitrary, known L. See Section 3. The idea was introduced, in a different setting, with the BEAST attack [27].

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. If the phrase is to mean anything, it should entail that the basic characteristics of nAE are maintained in the presence of nonce-reuse. An AE scheme satisfying nAE (employing non-repeating nonces) or MRAE (without that restriction) would certainly be immune to such an attack.

	OAE1 (from FFL [28])	**OAE2** (this paper)
Definitional idea	Online cipher then a tag	Aencrypt each segment
Segmentation	Fixed-size blocks of scheme-determined lengths	Variable-size segments of user-determined lengths
Typical block/seg size	5–16 bytes	1–10000 bytes?
Ciphertext expansion	τ bits per message	τ bits per segment
Message space	$M \in \mathsf{B}_n^*$ for blocksize n	$M \in \{0,1\}^*$ or $\boldsymbol{M} \in \{0,1\}^{**}$
Decryption online?	No, not in general	Yes, automatically
Can aencrypt infinite streams?	No, msgs must end	Yes, msgs can be infinite
OK to repeat nonces?	No, attacks always possible	No, attacks always possible

Fig. 1: **Approaches to formulating online-AE**. It is a thesis of this paper that OAE1 misformulates the desired goal and wrongly promises nonce-reuse misuse-resistance.

We next pull back and take a more philosophical view. We argue that the definition of OAE1 fails in quite basic ways to capture the intuition for what secure online-AE (OAE) ought to do. First, 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 [61]. These two things are unrelated and ought to be disentangled. Second, OAE1 fails to define security for plaintexts that aren't a multiple of the blocksize. But constructions do just that, encrypting arbitrary bit strings or byte strings. Third, OAE1 measures privacy against an idealized object that's an online cipher followed by a tag. But having such a structure is not only unnecessary for achieving online encryption, but also undesirable for achieving good security. Finally, while OAE1 aims to ensure that encryption is online, it ignores decryption. The elision has engendered an additional set of definitions for RUP security, "releasing unverified plaintext" [7]. We question the utility of online 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.

An alternative: OAE2. There *are* environments where online encryption is needed. The designer of an FPGA or ASIC encryption/decryption engine might be unable to buffer more than a kilobyte of message. An application like SSH needs to send across a character interactively typed at the keyboard. Netflix needs to stream a film [46] that is "played" as it is received, never buffering an excessive amount or incurring excessive delays. A software library might want to support an incremental encryption and decryption API. Whatever the setting, we think of the plaintext and ciphertext as having been *segmented* into a sequence of *segments*. We don't control the size of segments—that's a user's purview—and different segments can have different lengths.

Thus the basic problem that OAE2 formalizes involves a (potentially long, even infinite) plaintext M that gets segmented by the user to $(M_1, \ldots, M_m)$. We must encrypt each segment M_i as soon as it arrives, carrying forward only a constant-size state. Thus M gets transformed into a segmented ciphertext $(C_1, \ldots, C_m)$. Each C_i must enable immediate recovery of M_i (the receiver can no more wait for C's completion than the sender can wait for M's). We don't

insist that $|C_i| = |M_i|$; in fact, the user will do better to grow each segment, $|C_i| > |M_i|$, to support expedient verification of what has come so far. See Fig. 1 for a brief comparison of OAE1 and OAE2.

After formulating OAE2, which we do in three approximately-equivalent ways, we describe simple means to achieve it. We don't view OAE2 as a goal for which one should design a fundamentally new AE scheme; the preferred approach is to use a conventional AE scheme and wrap it in a higher-level protocol. We describe two such protocols. The first, **CHAIN**, can be used to turn an MRAE scheme (e.g., SIV) into an OAE2 scheme. The second, **STREAM**, can be used to turn an nAE scheme (e.g., OCB) into a nonce-based OAE scheme. That aim, nOAE, is identical to OAE2 except for insisting that, on the encryption side, nonces don't repeat. Finally, we consider a weakening of OAE2, we call it dOAE, stronger than nOAE and achievable with online processing of each segment.

For reasons of length, the treatment of nOAE, dOAE, and **STREAM** appear only in the full version of this paper [33]. Also see the full version for proofs and a more complete discussion of related work.

We emphasize that moving from OAE1 to OAE2 does not enable one to safely repeat nonces; an OAE2-secure scheme will still be susceptible to CPSS attack, for example. In that light, we would not term an OAE2 scheme *misuse resistant*. What makes OAE2 "better" than OAE1 is not added robustness to nonce-reuse (at least none that we know how to convincingly formalize) but a better modeling of the problem at hand, and a more faithful delivery on the promise of achieving best-possible security for an online-AE scheme. In view of the fact that, with OAE2, one must still deprecate nonce reuse, we would view nOAE as the base-level aim for online-AE.

Related work. A crucial idea for moving beyond BBKN's and FFL's conceptions of online encryption is to sever the association of the blocksize of some underlying tool and the quantum of text a user is ready to operate on. A 2009 technical report of Tsang, Solomakhin, and Smith (TSS) [61] expressed this insight and provided a definition based on it. TSS explain that AE à la Boldyreva and Taesombut [23] (or BBKN or FFL, for that matter) "processes and outputs ... blocks as soon as the next input block is received" [61, p. 4], whence they ask, "what if the input is smaller than a block?", even a bit, or what "if the input is a [segment] ... of arbitrary length?" TSS maintain that such situations occur in practice, and they give examples [61, Section 8].

There are major difference in how TSS proceed and how we do. They insist on schemes in which there is ciphertext expansion only at the beginning and end, and their definition is oriented towards that assumption. They do not authenticate the segmented plaintext but the string that is their concatenation. Our formalization of OAE2 lets the adversary run multiple, concurrent sessions of online encryption and decryption, another novum. In the end, the main commonality is some motivation and syntax.

Bertoni, Daemen, Peeters, and Van Assche (BDPV) present a mechanism, the duplex construction, to turn a cryptographic permutation f into an object very much like what we are calling an OAE2 scheme [19]. BDPV consider encrypting

and authenticating a sequence of messages $(B_1, B_2, \ldots)$ having corresponding headers $(A_1, A_2, \ldots)$. Asserting that it is "interesting to authenticate and encrypt a sequence of messages in such a way that the authenticity is guaranteed not only on each (A, B) but also on the sequence received so far" [19, p. 323], they encrypt each (A_i, B_i) to a ciphertext C_i and a tag T_i, the process depending on the prior (A_j, B_j) values. The authors explain that "Intermediate tags can also be useful in practice to be able to catch fraudulent transactions early" [19, p. 323]. BDPV provide a definition for the kind of AE they speak of [19, Section 2]. It resembles both OAE2 and nOAE, and inspired dOAE.

A REAL-WORLD NEED. Netflix recently described a protocol of theirs, MSL, for streaming video [46]. The movie is broken into variable-length segments and each segment is independently encrypted and authenticated, with the ordering of the segments itself authenticated. MSL is based on Encrypt-then-MAC composition, where the encryption is AES-CBC with PKCS#5 padding and the MAC is HMAC-SHA256. The choice suggests that even in real-time applications, use of a two-pass AE scheme for each segment can be fine, as long as segments are of appropriate length. MSL resembles an instantiation of **STREAM**. The current paper provides foundations for the problem that Netflix faced, offering definitions and generic solutions with good provable security.

2 OAE1 Definition

All OAE definitions of widespread use spring from FFL [28], who married the definition of an online cipher from Bellare, Boldyreva, Knudsen, and Namprempre [15] with the definition of authenticity of ciphertexts (also called integrity of ciphertexts) [16,41,54]. In this section we recall the FFL definition, staying true to the original exposition as much possible, but necessarily deviating to correct an error. We call the (corrected) definition OAE1.

SYNTAX. For any $n \geq 1$ let $\mathsf{B}_n = \{0,1\}^n$ denote the set of n-bit *blocks*. A *block-based AE scheme* is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where the *key space* $\mathcal{K}$ is a nonempty set with an associated distribution and where the *encryption algorithm* $\mathcal{E}$ and *decryption algorithm* $\mathcal{D}$ are deterministic algorithms with signatures $\mathcal{E}\colon \mathcal{K} \times \mathcal{H} \times \mathsf{B}_n^* \to \{0,1\}^*$ and $\mathcal{D}\colon \mathcal{K} \times \mathcal{H} \times \{0,1\}^* \to \mathsf{B}_n^* \cup \{\perp\}$. The set $\mathcal{H}$ associated to Π is the *header space*. FFL assumes that it is $\mathcal{H} = \mathsf{B}_n^+ = \mathcal{N} \times \mathcal{A}$ with $\mathcal{N} = \mathsf{B}_n$ and $\mathcal{A} = \mathsf{B}_n^*$ the *nonce space* and *AD space*. The value n associated to Π is its *blocksize*. Note that the *message space* $\mathcal{M}$ of Π must be $\mathcal{M} = \mathsf{B}_n^*$ and the blocksize n will play a central role in the security definition. We demand that $\mathcal{D}(K, N, A, \mathcal{E}(K, N, A, M)) = M$ for all $K \in \mathcal{K}$, $N \in \mathcal{N}$, $A \in \mathcal{A}$, and $M \in \mathsf{B}_n^*$.

To eliminate degeneracies it is important to demand that $|\mathcal{E}(K, H, M)| \geq |M|$ for all K, H, M and that $|\mathcal{E}(K, H, M)|$ depends on at most H and $|M|$. To keep things simple, we assume that the ciphertext expansion $|\mathcal{E}(K, H, M)| - |M|$ is a constant $\tau \geq 0$ rather than an arbitrary function of H and $|M|$.

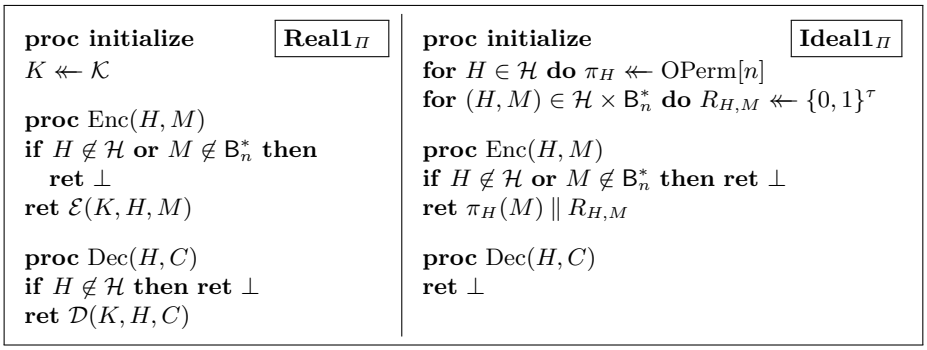

Fig. 2: **OAE1 security.** Defining security for a block-based AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with header space $\mathcal{H}$, blocksize n, and ciphertext expansion τ.

Security. Let OPerm$[n]$ be the set of all length-preserving permutations π on B_n^* where ith block of $\pi(M)$ depends only on the first i-blocks of M; more formally, a length-preserving permutation $\pi\colon \mathsf{B}_n^* \to \mathsf{B}_n^*$ is in OPerm$[n]$ if the first $|X|$ bits of $\pi(XY)$ and $\pi(XY')$ coincide for all $X, Y, Y' \in \mathsf{B}_n^*$. Despite its being infinite, one can endow OPerm$[n]$ with the uniform distribution in the natural way. To sample from this we write $\pi \xleftarrow{\$}$ OPerm$[n]$.

Fix a block-based AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with $\mathcal{E}\colon \mathcal{K} \times \mathcal{H} \times \mathsf{B}_n^* \to \{0,1\}^*$. Then we associate to Π and an adversary $\mathscr{A}$ the real number $\mathbf{Adv}^{\mathrm{oae1}}_{\Pi}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{Real1}} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{Ideal1}} \Rightarrow 1]$ where games **Real1** and **Ideal1** are defined in Fig. 2. Adversary $\mathscr{A}$ may not ask a Dec query (H, C) after an Enc query (H, M) returned C. Informally, $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is OAE1 secure if $\mathbf{Adv}^{\mathrm{oae1}}_{\Pi}(\mathscr{A})$ is small for any reasonable $\mathscr{A}$. Alternatively, we can speak of OAE1$[n]$ security to emphasize the central role in defining security of the scheme's blocksize n.

Discussion. The OAE1 definition effectively says that, with respect to privacy, a ciphertext must resemble the image of a plaintext under a random online permutation (tweaked by the nonce and AD) followed by a τ-bit random string (the authentication tag). But the original definition from FFL somehow omitted the second part [28, Definition 3]. The lapse results in a definition that makes no sense, as $\mathcal{E}$ must be length-increasing to provide authenticity. The problem was large enough that it wasn't clear to us what was intended. Follow-on work mostly replicated this [2, 29]. After discussions among ourselves and checking with one of the FFL authors [44], we concluded that the intended definition is the one we have given.

LCP leakage. Say that a block-based AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with blocksize n is LCP$[n]$ (for "longest common prefix") if for all K, H, M, and $i \le |M|/n$, the first i blocks of $\mathcal{E}^H_K(M)$ depend only on the first i blocks of M. While all schemes we know claiming to be OAE1$[n]$ are also LCP$[n]$, an OAE1$[n]$-secure scheme isn't *necessarily* LCP$[n]$. This is because the requirement for OAE1$[n]$ security is to be computationally close to an object that is LCP$[n]$, and some-

thing being computationally close to something with a property P doesn't mean it has property P. Indeed it is easy to construct an artificial counterexample; for example, starting with a OAE1[n]-secure scheme that is LCP[n], augment the key with n extra bits, K', and modify encryption so that when the first block of plaintext coincides with K', then reverse the bits of the remainder of the plaintext before proceeding. OAE1 security is only slightly degraded but the scheme is no longer LCP[n]. Still, despite such counterexamples, an OAE1[n]-secure scheme must be *close* to being LCP[n]. Fix Π as above and consider an adversary $\mathscr{A}$ that is given an oracle $\mathcal{E}_K(\cdot,\cdot)$ for $K \xleftarrow{} \mathcal{K}$. Consider $\mathscr{A}$ to be *successful* if it outputs $H \in \mathcal{H}$ and $X, Y, Y' \in \mathsf{B}_n^*$ such that the first $|X|/n$ blocks of $\mathcal{E}_K^H(XY)$ and $\mathcal{E}_K^H(XY')$ are different (i.e., the adversary found non-LCP behavior). Let $\mathbf{Adv}_{\Pi}^{\mathrm{lcp}}(\mathscr{A})$ be the probability that $\mathscr{A}$ is successful. Then it's easy to transform $\mathscr{A}$ into an equally efficient adversary $\mathscr{B}$ for which $\mathbf{Adv}_{\Pi}^{\mathrm{oae1}}(\mathscr{B}) = \mathbf{Adv}_{\Pi}^{\mathrm{lcp}}(\mathscr{A})$. Because of this, there is no real loss of generality, when discussing OAE1[n] schemes, to assume them LCP[n]. In the next section we will do so.

3 CPSS Attack

Section 1 described the *trivial* attack to break OAE1-secure schemes with too small a blocksize. We now describe a different fixed-header CPA attack, this one working for any blocksize. We call the attack a *chosen-prefix, secret-suffix* (CPSS) attack. The attack is simple, yet devastating. It is inspired by the well-known BEAST (Browser Exploit Against SSL/TLS) attack [27].

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a block-based AE scheme with blocksize n satisfying LCP[n]. We consider a setting where messages $M = P \,\|\, S$ that get encrypted can be logically divided into a prefix P that is controlled by an adversary, then a suffix S that is secret, fixed, and not under the adversary's control. The adversary wants to learn S. We provide it the ability to obtain an encryption of $\mathcal{E}_K^H(P \,\|\, S)$ for any P it wants—except, to be realistic, we insist that P be a multiple of b bits. This is assumed for S too. Typically P and S must be byte strings, whence $b = 8$; for concreteness, let us assume this. Also for concreteness, assume a blocksize of $n = 128$ bits. Assume that $\mathcal{E}$ can in fact operate on arbitrary byte-length strings, but suffers LCP leakage on block-aligned prefixes (this is what happens if one pads and then applies an OAE1-secure scheme). Finally, assume $|S|$ is a multiple of the blocksize.

To recover S, the adversary proceeds as follows. First it selects an arbitrary string P_1 whose byte length is one byte shorter than p blocks, for an arbitrary $p \geq 1$. (For example, it would be fine to have $P_1 = 0^{120}$.) The adversary requests ciphertext $C_1 = \mathcal{E}_K^H(P_1 \,\|\, S)$. This will be used to learn S_1, the first byte of S. To do so, the adversary requests ciphertexts $C_{1,B} = \mathcal{E}_K^H(P_1 \,\|\, B \,\|\, S)$ for all 256 one-byte values B. Due to LCP leakage, exactly one of these values, the one with $B = S_1$, will agree with C_1 on the first p blocks. At this point the adversary knows the first byte of S, and has spent 257 queries to get it. There is an obvious

strategy to reduce this to 256 queries: omit one of the 256-possible byte values for B and use this for S_1 if no other match is found.

Now the adversary wants to learn S_2, the second byte of S. It selects an arbitrary string P_2 that is *two* bytes short of p blocks, for any $p \geq 1$. The adversary requests the ciphertext $C_2 = \mathcal{E}_K^H(P_2 \,\|\, S)$; and it requests ciphertexts $C_{2,B} = \mathcal{E}_K^H(P_2 \,\|\, S_1 \,\|\, B \,\|\, S)$ for all 256 one-byte values B. Due to LCP leakage and the fact that we have matched the first byte S_1 of S already, exactly one of these 256 values, call it S_2, will agree with C_2 on the first p blocks. At this point the adversary knows S_2, the second byte of S. It has used 257 more queries to get this. This can be reduced to 256 as before.

Continuing in this way, the adversary recovers all of S in $256\,|S|/8$ queries. In general, we need $2^b|S|/b$ queries to recover S. Note that the adversary has considerable flexibility in selecting the values that prefix S: rather than this being completely chosen by the adversary, it is enough that it be a known, fixed value, followed by the byte string that the adversary can fiddle with. That is, the CPSS attack applies when the adversary can manipulate a portion R of values $L \,\|\, R \,\|\, S$ that get encrypted, where L is known and S is not.

How practical? It is not uncommon to have protocols where there is a predictable portion L of a message, followed by an adversarially mutable portion R specifying details, followed by further information S, some or all of which is sensitive. This happens in HTTP, for example, where the first portion of the request specifies a method, such as `GET`, the second specifies a resource, such as `/img/scheme.gif/`, and the final portion encodes information such as the HTTP version number, an end-of-line character, and a session cookie. If an LCP-leaking encryption scheme is used in such a setting, one is asking for trouble.

Of course we do not suggest that LCP leakage will always foreshadow a real-world break. But the whole point of having general-purpose notions and provable-security guarantees is to avoid relying on application-specific characteristics of a protocol to enable security. If misuse comes as easily as giving adversaries the ability to manipulate a middle portion $L \,\|\, R \,\|\, S$ of plaintexts, one has strayed very far indeed from genuine misuse-resistance.

MRAE and CPSS. In the full version [33] we evidence that MRAE provides a modicum of misuse resistance that OAE1 lacks by establishing the rather obvious result that any MRAE-secure scheme resists CPSS attack.

4 Broader OAE1 Critique

The CPSS attack suggests that the OAE1 definition is "wrong" in the sense that it promises nonce-reuse security but compliant protocols are susceptible to realistic fixed-nonce attacks. In this section we suggest that OAE1's defects are more fundamental—that the definition fails to capture the intuition about what something called "online-AE" ought do. Our complaints are thus philosophical, but only in the sense that assessing the worth of a cryptographic definition always includes assessing the extent to which it delivers on some underlying intuition.

The blocksize should not be a scheme-dependent constant. A reasonable syntactic requirement for online-AE would say that the ith bit of ciphertext should depend only on the first i bits of plaintext (and, of course, the key, nonce, and AD). This would make online-AE something akin to a stream cipher. But the requirement above is not what OAE1 demands—it demands that the ith *block* depends only on the first i *blocks* of plaintext. Each of these blocks has a fixed *blocksize*, some number n associated to the scheme *and* its security definition. Thus implicit in the OAE1 notion is the idea that there is going to be *some* buffering of the bits of an incoming message before one can output the next block of bits. It is not clear if this fixed amount of buffering is done as a matter of efficiency, simplicity, or security. In schemes targeting OAE1-security, the blocksize is usually small, like 128 bits, the value depending on the width of some underlying blockcipher or permutation used in the scheme's construction.

That there's a blocksize parameter at all implies that, to the definition's architects, it is desirable, or at least acceptable, to buffer *some* bits of plaintext before acting on them—just not too many. But the number of bits that are reasonable to buffer is application-environment specific. One application might need to limit the blocksize to 128 KB, so as to fit comfortably within the L2 cache of some CPU. Another application might need to limit the blocksize to 1 KB, to fit compactly on some ASIC or FPGA. Another application might need to limit the blocksize to a single byte, to ensure bounded latency despite bytes arriving at indeterminate times. The problem is that the designer of a cryptographic scheme is in no position to know the implementation-environment's constraint that motivates the selection of a blocksize in the first place. By choosing some fixed blocksize, a scheme's designer simultaneously forecloses on an implementation's potential need to buffer less *and* an implementation's potential ability to buffer more. *Any* choice of a blocksize replaces a user's environment-specific constraint by a hardwired choice from a primitive's designer.

(Before moving on let us point out that, if it *is* the amount of memory available to an implementation that is an issue, the right constraint is not the blocksize n, where block C_i depends only on prior blocks, but the requirement that an implementation be achievable in one pass and n bits of memory. These are not the same thing [56, p. 241]. And the former is a poor substitute for the latter since context sizes vary substantially from scheme to scheme. While one *could* build an OAE notion by parameterizing its online memory requirement, we find it more appealing to eliminate any such parameter.)

Security must be defined for <u>all</u> plaintexts. The OAE1[n] notion only defines security when messages are a multiple of n bits. What should security mean when the message space is larger, like $\mathcal{M} = \{0,1\}^*$? Saying "we pad first, so needn't deal with strings that aren't multiples of the blocksize" is a complete non-answer, as it leaves unspecified what the goal *is* one is aiming to achieve by padding on the message space of interest—the one before padding is applied.

There are natural ways to try to extend OAE1[n] security to a larger message space; see, for example, the approach used for online ciphers on $\{0,1\}^{\geq n}$ [56]. This can be extended to OAE1. But it is not the only approach, and there will

still be issues for dealing with strings of fewer than n bits. In general, we think that an online-AE definition is not really meaningful, in practice, until one has specified what security means on the message space $\mathcal{M} = \{0,1\}^*$.

Decryption too must be online. If one is able to produce ciphertext blocks in an online fashion one had better be able to process them as they arrive. Perhaps the message was too long to store on the encrypting side. Then the same will likely hold on the decrypting side. Or perhaps there are timeliness constraints that one needs to act on a message fragment *now*, before the remainder of it arrives. Think back to the Netflix scenario. It would be pointless to encrypt the film in an online fashion only to have to buffer the entire thing at the receiver before it could play.

But online decryption is not required by OAE1 security, and it is routine that online decryption of each provided block would be fatal. We conjecture that it is an unusual scenario where it is important for encryption be computable online but irrelevant if decryption can be online as well.

The OAE1 reference object is not ideal. The reference object for OAE1$[n]$ security pre-supposes that encryption resembles an online-cipher followed by a random-looking tag. But it is wrong to think of this as capturing ideal behavior. First, it implicitly assumes that all authenticity is taken care of at the very end. But if a plaintext is long and one is interested in encryption being online to ensure timeliness, then waiting until the end of a ciphertext to check authenticity make no sense. If one is going to act on a prefix of a plaintext when it's recovered, it better be authenticated. Second, it is simply irrelevant, from a security point of view, if, prior to receipt of an authentication tag, encryption amounts to length-preserving permutation. Doing this may minimize ciphertext length, but that is an efficiency issue, not a basic goal. And achieving this particular form of efficiency is at odds with possible authenticity aims.

5 OAE2: Reformalizing Online-AE

We provide a new notion for online-AE. We will call it OAE2. To accurately model the underlying goal, not only must the security definition depart from that used by nAE and MRAE, but so too must a scheme's basic syntax. In particular, we adopt an API-motivated view in which the segmentation of a plaintext is determined by the caller.

After defining the syntax we offer three ways to quantify the advantage an adversary gets in attacking an OAE2 scheme. We term these advantage measures OAE2a, OAE2b, OAE2c. The notions are essentially equivalent. We provide quantitative results to make this *essentially* precise.

Why describe three different advantage measures of OAE2 security? We think it helps clarify just what OAE2 really *is*. The measures have different characteristics. The first, OAE2a, is a vector-oriented formulation. It employs a fairly easy-to-understand reference object. The second advantage measure, OAE2b, is

a string-oriented formulation. It employs a tighter and more realistic accounting of the adversary's actual resource expenditure. The third advantage measure, OAE2c, is more aspirational in character. Yet it is the easiest notion to work with, at least for proving schemes OAE2-secure. The OAE2c measure only makes sense if the segment-expansion τ is fairly large.

We begin with a bit of notation.

SEGMENTED STRINGS. Denote by $\{0,1\}^{**} = (\{0,1\}^*)^*$ the set of *segmented-strings*: a segmented string $\boldsymbol{X} \in \{0,1\}^{**}$ is a vector (or list) of strings. Each of its components, which we call a segment, is a string. The segmented-string with zero components is the empty list Λ. This is different from the empty string ε. The number of components in a segmented-string $\boldsymbol{X}$ is denoted $|\boldsymbol{X}|$, while the ith component of $\boldsymbol{X}$, $i \in [1..|\boldsymbol{X}|]$, is denoted $\boldsymbol{X}[i]$. Note that indexing begins at 1. For $\boldsymbol{X} \in \{0,1\}^{**}$ and $1 \le i \le j \le |\boldsymbol{X}|$, by $\boldsymbol{X}[i..j]$ we mean the $(j-i+1)$-vector $(\boldsymbol{X}[i], \boldsymbol{X}[i+1], \cdots \boldsymbol{X}[j])$. If $\boldsymbol{X} \in \{0,1\}^{**}$ and $X \in \{0,1\}^*$ then $\boldsymbol{X} \parallel X$ is the $|\boldsymbol{X}|+1$ vector consisting of the components of $\boldsymbol{X}$, in order, followed by X. Keep in mind that this is not concatenation of strings but, instead, appending a string to vector of strings to get a longer vector of strings. We emphasize that a segmented string is not a string.

SCHEME SYNTAX. A *segmented-AE scheme* is a tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where the *key space* $\mathcal{K}$ is a nonempty set with an associated distribution and both encryption $\mathcal{E} = (\mathcal{E}.\text{init}, \mathcal{E}.\text{next}, \mathcal{E}.\text{last})$ and decryption $\mathcal{D} = (\mathcal{D}.\text{init}, \mathcal{D}.\text{next}, \mathcal{D}.\text{last})$ are specified by triples of deterministic algorithms. Associated to Π are its *nonce space* $\mathcal{N} \subseteq \{0,1\}^*$ and its *state space* $\mathcal{S}$. For simplicity, a scheme's *AD space* $\mathcal{A} = \{0,1\}^*$, *message space* $\mathcal{M} = \{0,1\}^*$, and *ciphertext space* $\mathcal{C} = \{0,1\}^*$ are all strings. While an AD will be provided with each plaintext segment, a single nonce is provided for the entire sequence of segments. The signature of the components of $\mathcal{E}$ and $\mathcal{D}$ are as follows:

$$\begin{array}{ll} \mathcal{E}.\text{init}\colon \mathcal{K} \times \mathcal{N} \to \mathcal{S} & \mathcal{D}.\text{init}\colon \mathcal{K} \times \mathcal{N} \to \mathcal{S} \\ \mathcal{E}.\text{next}\colon \mathcal{S} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \times \mathcal{S} & \mathcal{D}.\text{next}\colon \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to (\mathcal{M} \times \mathcal{S}) \cup \{\perp\} \\ \mathcal{E}.\text{last}\colon \mathcal{S} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} & \mathcal{D}.\text{last}\colon \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\} \end{array}$$

When an algorithm takes or produces a point $S \in \mathcal{S}$ from its state space, it is understood that a fixed encoding of S is employed.

Given a segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ there are *induced* encryption and decryption algorithms $\boldsymbol{\mathcal{E}}$, $\boldsymbol{\mathcal{D}}\colon \mathcal{K} \times \mathcal{N} \times \{0,1\}^{**} \times \{0,1\}^{**} \to \{0,1\}^{**}$ (note the change to bold font) that operate, all at once, on vectors of plaintext, ciphertext, and AD. These maps are defined in Fig. 3. Observe how $\text{Dec}(K, N, \boldsymbol{A}, \boldsymbol{C})$ returns a longest $\boldsymbol{M}$ whose encryption (using K, N, and $\boldsymbol{A}$) is a prefix of $\boldsymbol{C}$; in essence, we stop at the first decryption failure, so $|\boldsymbol{C}| = |\boldsymbol{M}|$ if and only if $\boldsymbol{C}$ is entirely valid. We require the following validity condition for any segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with induced $(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}})$: if $K \in \mathcal{K}$, $N \in \mathcal{N}$, $\boldsymbol{A} \in \{0,1\}^{**}$, $\boldsymbol{M} \in \{0,1\}^{**}$, and $\boldsymbol{C} = \boldsymbol{\mathcal{E}}(K, N, \boldsymbol{A}, \boldsymbol{M})$, then $\boldsymbol{M} = \boldsymbol{\mathcal{D}}(K, N, \boldsymbol{A}, \boldsymbol{C})$.

algorithm $\boldsymbol{\mathcal{E}}(K, N, \boldsymbol{A}, \boldsymbol{M})$
$m \leftarrow |\boldsymbol{M}|$; **if** $m = 0$ **or** $|\boldsymbol{A}| \neq |\boldsymbol{M}|$ **then ret** Λ
$(A_1, \ldots, A_m) \leftarrow \boldsymbol{A}$
$(M_1, \ldots, M_m) \leftarrow \boldsymbol{M}$
$S_0 \leftarrow \mathcal{E}.\text{init}(K, N)$
for $i \leftarrow 1$ **to** $m - 1$ **do**
  $(C_i, S_i) \leftarrow \mathcal{E}.\text{next}(S_{i-1}, A_i, M_i)$
$C_m \leftarrow \mathcal{E}.\text{last}(S_{m-1}, A_m, M_m)$
ret $(C_1, \ldots, C_m)$

algorithm $\boldsymbol{\mathcal{D}}(K, N, \boldsymbol{A}, \boldsymbol{C})$
$m \leftarrow |\boldsymbol{C}|$
if $m = 0$ **or** $|\boldsymbol{A}| \neq |\boldsymbol{M}|$ **then ret** Λ
$(A_1, \ldots, A_m) \leftarrow \boldsymbol{A}$; $(C_1, \ldots, C_m) \leftarrow \boldsymbol{C}$
$S_0 \leftarrow \mathcal{D}.\text{init}(K, N)$
for $i \leftarrow 1$ **to** $m - 1$ **do**
  if $\mathcal{D}.\text{next}(S_{i-1}, A_i, C_i) = \perp$ **then**
    if $m = 1$ **ret** Λ
    else ret $(M_1, \ldots, M_{i-1})$
  else $(M_i, S_i) \leftarrow \mathcal{D}.\text{next}(S_{i-1}, A_i, C_i)$
$M_m \leftarrow \mathcal{D}.\text{last}(S_{m-1}, A_m, C_m)$
if $M_m = \perp$ **then ret** $(M_1, \ldots, M_{m-1})$
else ret $(M_1, \ldots, M_m)$

Fig. 3: **Operating on segmented strings.** The figure shows the algorithms $\boldsymbol{\mathcal{E}}$ and $\boldsymbol{\mathcal{D}}$ that are *induced* by the segmented encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$.

Ciphertext expansion. We focus on segmented-AE schemes with constant segment-expansion, defined as follows: associated to Π is a number $\tau \geq 0$ such that if $K \in \mathcal{K}$, $N \in \mathcal{N}$, $\boldsymbol{A} \in \{0,1\}^{**}$, $\boldsymbol{M} \in \{0,1\}^{**}$, $m = |\boldsymbol{A}| = |\boldsymbol{M}|$, and $\boldsymbol{C} = \boldsymbol{\mathcal{E}}(K, N, \boldsymbol{A}, \boldsymbol{M})$, then $|\boldsymbol{C}[i]| = |\boldsymbol{M}[i]| + \tau$ for all $i \in [1..m]$. Thus each segment grows by exactly τ bits, for some constant τ. We call τ the *segment-expansion* of Π.

We favor constant segment-expansion because we think it runs contrary to the spirit of online-AE to furnish interior segments with an inferior authenticity guarantee than that afforded to the whole message. After all, much of the point of online-AE is to allow a decrypting party to safely act on a ciphertext segment as soon as its available. Still, there is an obvious efficiency cost to expanding every segment. See the heading "Multivalued segment-expansion" for the case where the amount of segment-expansion is position dependent.

Online computability. We say that a segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has *online-encryption* if its state space $\mathcal{S}$ is finite and there's a constant w such that $\mathcal{E}$.next and $\mathcal{E}$.last use at most w bits of working memory. The value w excludes memory used for storing an algorithm's inputs or output; we elaborate below. Similarly, scheme Π has *online-decryption* if its state space $\mathcal{S}$ is finite and there's a constant w such that $\mathcal{D}$.next and $\mathcal{D}$.last use at most w bits of working memory. A segmented-AE scheme is *online* if it has online-encryption and online-decryption. In accounting for memory above, the model of computation provides input values on a read-only input tape; the input's length is not a part of the working memory accounted for by w. Similarly, algorithms produce output by writing to a write-only output tape in a left-to-right fashion. The number of bits written out has nothing to do with the working memory w.

Our security definitions don't care if a segmented-AE scheme is online: that's an efficiency requirement, not a security requirement. Yet a good part of the

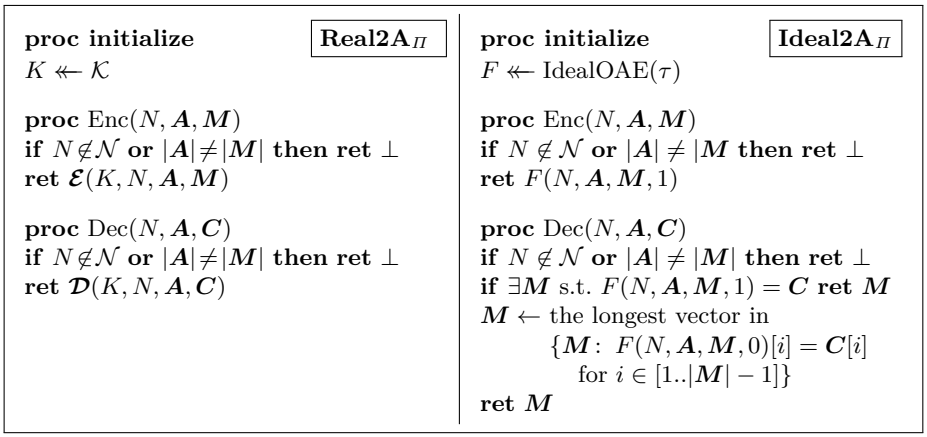

Fig. 4: **OAE2a security.** The segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has nonce space $\mathcal{N}$ and segment-expansion τ. It induces algorithms $\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}}$ as per Fig. 3. The distribution IdealOAE(τ) is described in the text.

purpose of the segmented-AE syntax is to properly deal with schemes that have such efficiency constraints.

First OAE2 definition: OAE2a. We begin by defining the *ideal* behavior for an OAE scheme. Let $\mathrm{Inj}(\tau)$ denote the set of all τ-expanding injective functions—the set of all functions $f : \{0,1\}^* \to \{0,1\}^*$ that are injective and satisfy $|f(x)| = |x| + \tau$. Endow this set with the uniform distribution in the natural way. We write $f \xleftarrow{\$} \mathrm{Inj}(\tau)$ to denote uniformly sampling a random, τ-expanding injective function. Now define a distribution on functions IdealOAE(τ) as follows:

for $m \in \mathbb{Z}^+$, $N \in \{0,1\}^*$, $\boldsymbol{A} \in (\{0,1\}^*)^m$, $\boldsymbol{M} \in (\{0,1\}^*)^{m-1}$ **do**
  $f_{N,\boldsymbol{A},\boldsymbol{M},0} \xleftarrow{\$} \mathrm{Inj}(\tau)$; $f_{N,\boldsymbol{A},\boldsymbol{M},1} \xleftarrow{\$} \mathrm{Inj}(\tau)$
for $m \in \mathbb{Z}^+$, $\boldsymbol{A} \in (\{0,1\}^*)^m$, $\boldsymbol{X} \in (\{0,1\}^*)^m$, $\delta \in \{0,1\}$ **do**
  $F(N, \boldsymbol{A}, \boldsymbol{X}, \delta) \leftarrow (f_{N,\boldsymbol{A}[1..1],A,0}(\boldsymbol{X}[1]),\ f_{N,\boldsymbol{A}[1..2],\boldsymbol{X}[1..1],0}(\boldsymbol{X}[2]),$
  $f_{N,\boldsymbol{A}[1..3],\boldsymbol{X}[1..2],0}(\boldsymbol{X}[3]),\ \ldots,\ f_{N,\boldsymbol{A}[1..m-1],\boldsymbol{X}[1..m-2],0}(\boldsymbol{X}[m-1]),$
  $f_{N,\boldsymbol{A}[1..m],\boldsymbol{X}[1..m-1],\delta}(\boldsymbol{X}[m]))$
ret F

Thus $F \xleftarrow{\$} \mathrm{IdealOAE}(\tau)$ grows by accretion, the ith component of $F(N, \boldsymbol{A}, \boldsymbol{X}, 0)$ depending on N, $\boldsymbol{A}[1..i]$, and $\boldsymbol{X}[1..i]$. It must be decryptable (hence the injectivity) and have the mandated length. The final input to F, the flag δ, indicates if the argument $\boldsymbol{X}$ is complete: a 1 means it is, a 0 means it's not.

Fig. 4 defines games $\mathbf{Real2A}_\Pi$ and $\mathbf{Ideal2A}_\Pi$ for a τ-expanding segmented-AE scheme Π. Given an adversary $\mathscr{A}$ with oracles Enc and Dec determined by these games, let $\mathbf{Adv}^{\mathrm{oae2a}}_\Pi(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{Real2A}_\Pi} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{Ideal2A}_\Pi} \Rightarrow 1]$ be the adversary's distinguishing advantage.

Discussion. The security notion may be described as follows. A user wants to encrypt a segmented message $\boldsymbol{M} = (M_1, \ldots, M_m)$ into a ciphertext $\boldsymbol{C} =$

$(C_1, \ldots, C_m)$ using $K, N, \boldsymbol{A}$. He wants to do this *as well as possible* subject to the constraint that segments grow by exactly τ bits and $M_1 \cdots M_i$ are recoverable from $K, N, (A_1, \ldots, A_i), (C_1, \ldots, C_i)$. As with robust-AE [34], the phrase "as well as possible" targets an achievable (instead of aspirational) goal. Specifically, it is formalized by comparing the real object to a random element from IdealOAE(τ) and its inverse, the later understood to invert as many components as possible, stopping at the first point one can't proceed.

The definition of IdealOAE(τ) is complex enough that an example may help. Consider encrypting a segmented plaintext $M = (A, B, C, D)$ with a fixed key, nonce, and AD. Let (U, V, X, Y) be the result. Now encrypt $M' = (A, B, C)$. We want this to give (U, V, Z), not (U, V, X), as the final segment is special: processed by $\mathcal{E}$.last instead of $\mathcal{E}$.next, it is as though $M = (A, B, C, D)$ means $(A, B, C, D\$)$, while $M = (A, B, C)$ means $(A, B, C\$)$, where the \$-symbol is an end-of-message sentinel. Written like this, it is clear that the two segmented ciphertexts should agree on the first two components but not the third. Correspondingly, possession of (U, V, X, Y) ought not enable a forgery of (U, V, X). All of this understanding gets quietly embedded into the definition of IdealOAE(τ), whose member functions get a final argument δ with semantics indicating if the message is *complete*. Thus $F(N, \boldsymbol{A}, (A, B, C), 0)$ is what $\boldsymbol{M} = (A, B, C)$ should map to if more segments are to come, while $F(N, \boldsymbol{A}, (A, B, C), 1)$ is what it should map to if C is the final segment of $\boldsymbol{M}$.

Second OAE2 definition: OAE2b. Fig. 5 gives a more fine-grained and string-oriented measure for OAE2 security. The adversary, instead of providing $N, \boldsymbol{A}, \boldsymbol{M}$ and getting a vector $\boldsymbol{C} = \mathrm{Enc}(N, \boldsymbol{A}, \boldsymbol{M})$, can adaptively grow $\boldsymbol{A}$ and $\boldsymbol{M}$ one component at a time. Similarly, instead of providing a segmented ciphertext $N, \boldsymbol{A}, \boldsymbol{C}$ and getting $\boldsymbol{M} = \mathrm{Dec}(N, \boldsymbol{A}, \boldsymbol{C})$, it can adaptively grow $\boldsymbol{A}, \boldsymbol{C}$. As before, we associate to a τ-expanding segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and an adversary $\mathscr{A}$ the real number $\mathbf{Adv}^{\mathrm{oae2b}}_{\Pi}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{Real2B}_{\Pi}} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{Ideal2B}_{\Pi}} \Rightarrow 1]$ that is its distinguishing advantage.

The OAE2a and OAE2b measures are essentially equivalent. The *essentially* of this sentence entails a simple result explaining how to convert an adversary for one definition into an adversary for the other. First, given an oae2a-style adversary $\mathscr{A}$ we can construct an equally effective oae2b-style adversary $\mathscr{B}$: it translates each $\mathrm{Enc}(N, (A_1, \ldots, A_m), (M_1, \ldots, M_m))$ asked by adversary $\mathscr{A}$ into an Enc.init, then $m - 1$ Enc.next calls, then an Enc.last call, assembling the answers into a segmented ciphertext $(C_1, \ldots, C_m)$. Similarly, it translates $\mathrm{Dec}(N, (A_1, \ldots, A_m), (C_1, \ldots, C_m))$ calls into Dec.init, Dec.next, Dec.last calls. Adversary $\mathscr{B}$ gets exactly the oae2b-advantage that $\mathscr{A}$ had as oae2a-advantage. It runs in almost the exact same time.

Simulation in the other direction is less efficient. Given an adversary $\mathscr{A}$ attacking the oae2b-security of a Π, we construct an adversary $\mathscr{B}$ for attacking the oae2a-security of the same scheme. Adversary $\mathscr{B}$ maintains lists $N_i, \boldsymbol{A}_i, \boldsymbol{M}_i$ that are initialized in the natural way with each Enc.init call (incrementing i, initially zero, with each Enc.init). Calls of the form Enc.next(i, A, M), when valid, result in appending A to $\boldsymbol{A}_i$ and M to $\boldsymbol{M}_i$, making an $\mathrm{Enc}(N_i, \boldsymbol{A}_i \parallel \varepsilon, \boldsymbol{M}_i \parallel \varepsilon)$

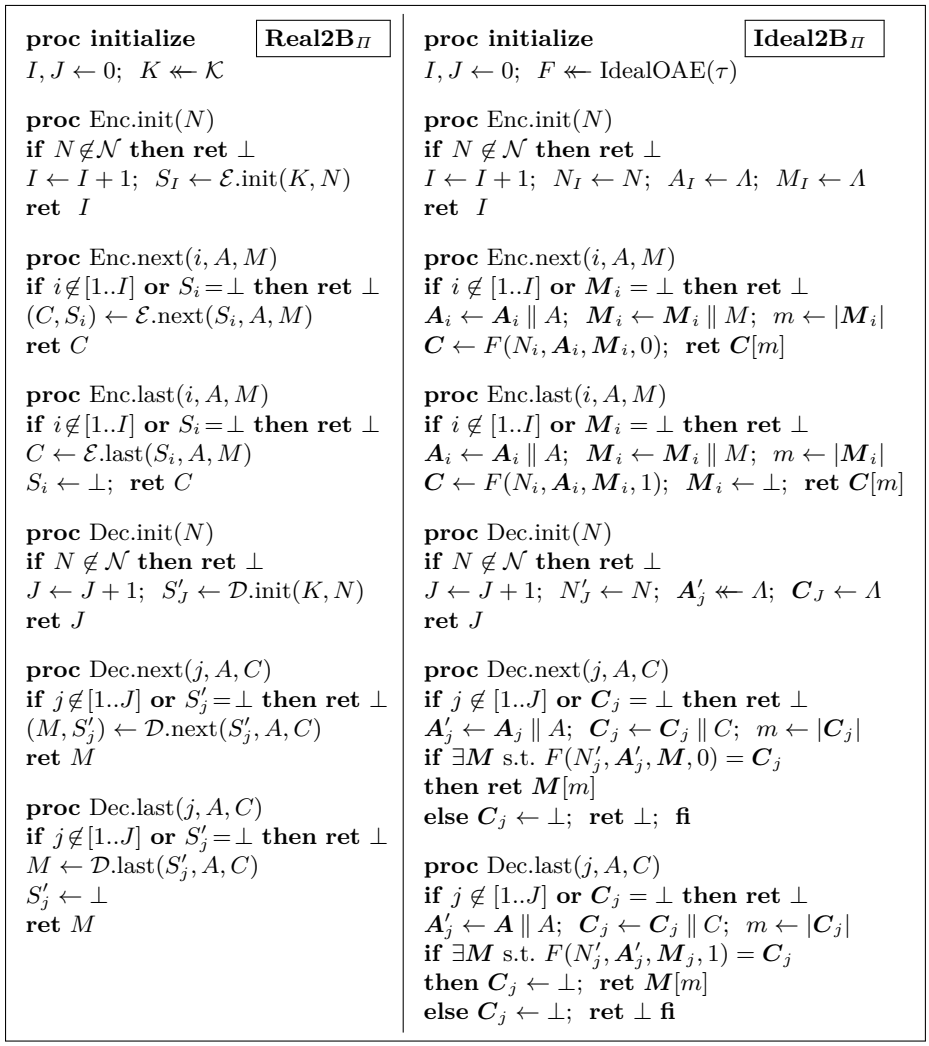

Fig. 5: **OAE2b security.** The segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has nonce space $\mathcal{N}$ and segment-expansion τ.

call, and returning its $|\boldsymbol{M}_i|$-th component. Calls of the form Enc.last(i, A, M) result in making an Enc($N_i, \boldsymbol{A}_i \parallel A, \boldsymbol{M}_i \parallel M$) call, returning its last component, resetting $\boldsymbol{M}_i$ to $\perp$ before doing so. Calls of the form Dec.init, Dec.next, and Dec.last are treated analogously, maintaining $N'_i, A'_i, \boldsymbol{C}_i$ values. Once again the simulation is perfect, so $\mathbf{Adv}^{\mathrm{oae2a}}_{\Pi}(\mathscr{B}) = \mathbf{Adv}^{\mathrm{oae2b}}_{\Pi}(\mathscr{A})$. But now there is a quadratic slowdown in running time: the argument lists can grow long, as can return values, only one component of which is used with each call.

While the OAE2a definition is more compact, the improved concision for the adversary's queries in the OAE2b definition ultimately make it preferable,

particularly as this concision better models the real-world semantics, where an adversary might be able to incrementally grow a plaintext or ciphertext with the unwitting cooperation of some encrypting or decrypting party. We note that we could achieve greater concision still by introducing a shorthand that would allow the adversary to grow a tree and not just a chain. But this would not seem to model anything meaningful in the real-world.

There are a couple of further reasons to favor OAE2b. One is that it more directly captures the possibility of "infinite" (non-terminating) plaintexts (an infinite "stream" of messages). This is simply the setting where Enc.last and Dec.last are never called. Second, the OAE2b definition makes it easier to define nonce-respecting adversaries for the OAE setting. Such adversaries may adaptively grow a plaintext based on a single nonce, but it may grow only *one* plaintext for any given nonce. Building on the OAE2a formulation this is awkward to say, but building on the OAE2b formulation, it is natural.

THIRD OAE2 DEFINITION: OAE2c. Let Π be a segmented-AE scheme with segment-expansion τ and nonce-space $\mathcal{N}$. Our final formulation of OAE2 security uses a two-part definition, separately defining privacy and authenticity requirements. With games defined in Fig. 6, we let $\mathbf{Adv}^{\text{oae2-priv}}_{\Pi}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{Real2C}_{\Pi}} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{Rand2C}_{\Pi}} \Rightarrow 1]$. Similarly, define $\mathbf{Adv}^{\text{oae2-auth}}_{\Pi}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{Forge2C}_{\Pi}}]$, meaning the probability that $\mathscr{A}$ returns a value that, when provided as input to the procedure **finalize**, evaluates to true. Informally, OAE2c security for a scheme Π means that reasonable adversaries get small oae2-priv advantage *and* small oae2-auth advantage.

Definition OAE2c is simpler than prior games in the sense that, for privacy, no decryption oracles are provided and the reference experiment simply returns the right number of uniformly random bits. For the authenticity portion of the definition, forgeries are defined to allow any $(N, \boldsymbol{A}, \boldsymbol{C})$ that the adversary does not trivially know to be valid, the adversary marking in $\boldsymbol{C}$ has terminated ($b = 1$) or not ($b = 0$). Set $\mathcal{Z}$ records the tuples that the trivially adversary knows by virtue of encryption queries.

The following propositions show that OAE2b and OAE2c are close, assuming that the segment-expansion τ is fairly large. The proofs are in the full version [33].

Proposition 1 (oae2c ⇒ oae2b). Let Π be a segmented-AE scheme with ciphertext expansion τ. There are explicit given reductions R_1 and R_2 with the following property. For any adversary $\mathscr{A}$, adversaries $\mathscr{B}_1 = R_1(\mathscr{A})$ and $\mathscr{B}_2 = R_2(\mathscr{A})$ satisfy $\mathbf{Adv}^{\text{oae2b}}_{\Pi}(\mathscr{A}) \le \mathbf{Adv}^{\text{oae2-priv}}_{\Pi}(\mathscr{B}_1) + p \cdot \mathbf{Adv}^{\text{oae2-auth}}_{\Pi}(\mathscr{B}_2) + q^2/2^{\tau}$, where p and q are the number of decryption chains and the number of queries of $\mathscr{A}$, respectively. For each $i \in \{1, 2\}$, adversary $\mathscr{B}_i$ uses about the same running time as $\mathscr{A}$, and the length of its queries is also at most that of $\mathscr{A}$'s queries.

Proposition 2 (oae2b ⇒ oae2c). Let Π be a segmented-AE scheme with ciphertext expansion τ. There are explicit given reductions R_1 and R_2 with the following property. For any adversaries $\mathscr{A}_1$ and $\mathscr{A}_2$, adversaries $\mathscr{B}_1 = R_1(\mathscr{A}_1)$ and $\mathscr{B}_2 = R_2(\mathscr{A}_2)$ satisfy $\mathbf{Adv}^{\text{oae2-priv}}_{\Pi}(\mathscr{A}_1) \le \mathbf{Adv}^{\text{oae2b}}_{\Pi}(\mathscr{B}_1)$ and $\mathbf{Adv}^{\text{oae2-auth}}_{\Pi}(\mathscr{A}_2) \le$

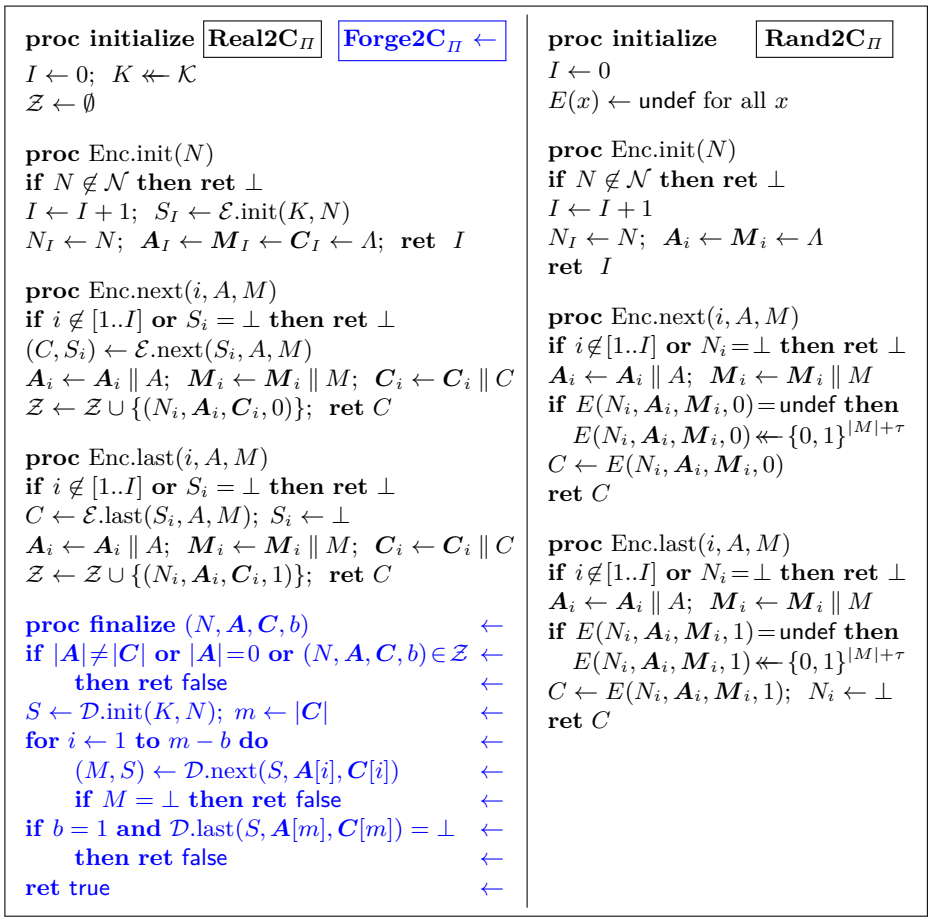

Fig. 6: **OAE2c security.** Privacy and authenticity are separately defined, the first by comparing games **Real2C** and **Rand2C**, and the second using game **Forge2C**, which includes the additional lines indicated.

$\mathbf{Adv}^{\mathrm{oae2b}}_{\Pi}(\mathscr{B}_2) + \ell/2^{\tau}$, where ℓ is the number of segments in $\mathscr{A}_2$'s output. For each $i \in \{1, 2\}$, adversary $\mathscr{B}_i$ uses about the same running time as $\mathscr{A}_i$, and the length of its queries is at most that of $\mathscr{A}_i$'s queries.

Multivalued segment-expansion. It is easy to extend the definitions of this section to schemes for which the segment-expansion varies according to segment position. In particular, one could use one expansion value, σ, for plaintext components other than the last, and a different expansion value, τ, at the end. For such a (σ, τ)-expanding scheme, distribution IdealOAE(τ) would be adjusted to IdealOAE(σ, τ) in the natural way.

The main reason for considering multivalued segment-expansion is to clarify how OAE2 security relates to prior notions in the literature. In particular, OAE2

resembles OAE1 where the segment-expansion is $(0, \tau)$ and where all segments are required to have some fixed length n. Yet even then the definitions would be very different: the OAE2 version would be stronger, since an online decryption capability is not allowed to compromise OAE2 security, whereas the capability may compromise OAE1 security. It is easy to give a separating example [33].

Another potential reason to consider multivalued segment-expansion is as a way to save on bits; obviously one will use fewer total bits, over a sequence of two or more segments, if only the last is expanded. But we suspect that this benefit is rarely worth its cost. If segments are 1 KByte (which is fairly short) and tags are 128 bits (which is fairly long), the difference (in total number of needed bits) between authenticating every segment and authenticating only the last one will always be less than 2%. This seems a small price to pay to have each and every segment properly authenticated.

Why vector-valued AD? In modeling OAE it is unclear if one ought think of the AD as a fixed string that is known before the plaintext begins to arrive, or if, instead, one should think of the AD as vector-valued, its ith segment available when the ith segment of plaintext is. We adopted the second view (switching from the first at the urging of the Keyak team) for closer concordance with prior work [19] and for greater generality: a string-valued AD of A can be regarded as a vector-valued AD of $\boldsymbol{A} = (A, \varepsilon, \varepsilon, \ldots)$. More philosophically, the two conceptions correspond to whether one thinks of breaking up a fixed plaintext $\boldsymbol{M}$ into a sequence of segments M_i or one regards the M_i values as more autonomous, each encrypted when available, each with its own associated context. With plaintexts and AD both vector-valued, one conceptually extends across time a channel that securely transmit pairs of strings, one component with privacy and both with authenticity. All that said, the authors are uncertain of the actual utility of vector-valued over string-valued AD.

6 Achieving OAE2

In the special case that each segmented-string has only one component, OAE2 degenerates to the notion of a *pseudorandom injection* (PRI) [55]. The notion is close to MRAE [55], with a gap $q^2/2^{s+\tau} + q/2^{\tau}$ where q is the number of queries and s is the length of the shortest plaintext queried. Below we construct an OAE2-secure scheme *from* a PRI-secure scheme. The scheme could be SIV [55] if τ is large, say $\tau = 128$, or AEZ scheme [34], for arbitrary τ. We begin by recalling the PRI notion.

Pseudorandom injections. Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be a conventional AE scheme, meaning that (i) the key space $\mathbf{K}$ is a nonempty set with an associated distribution, (ii) $\mathbf{E}\colon\ \mathbf{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^* \to \{0,1\}^*$ is the encryption scheme, and (iii) $\mathbf{D} : \mathbf{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ is the decryption scheme. Both $\mathbf{E}$ and $\mathbf{D}$ are deterministic, and decryption reverses encryption, meaning that for every $N \in \mathcal{N}$, $A \in \mathcal{A}$, $M \in \{0,1\}^*$, and $K \in \mathbf{K}$, we have $\mathbf{D}_K^{N,A}(\mathbf{E}_K^{N,A}(M)) = M$.

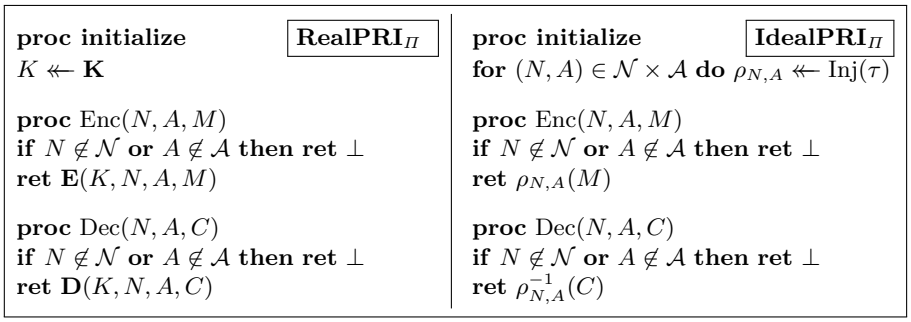

Fig. 7: **PRI security.** Defining security for an AE scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ with expansion τ, nonce space $\mathcal{N}$, and AD space $\mathcal{A}$. Here $\mathrm{Inj}(\tau)$ is the set of all injective functions $f\colon \{0,1\}^* \to \{0,1\}^*$ such that $|f(x)| = |x| + \tau$ for all $x \in \{0,1\}^*$. For each $y \in \{0,1\}^*$ let $f^{-1}(y) = x$ if there's an $x \in \{0,1\}^*$ such that $f(x) = y$, and $f^{-1}(y) = \perp$ otherwise.

We insist there be a constant τ associated to Π, its *ciphertext-expansion*, where $|\mathbf{E}_K^{N,A}(M)| = |M| + \tau$ for all $N \in \mathcal{N}$, $A \in \mathcal{A}$, $M \in \{0,1\}^*$, $K \in \mathbf{K}$. Define $\mathbf{Adv}^{\mathrm{pri}}_{\Pi}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{RealPRI}_\Pi} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{IdealPRI}_\Pi} \Rightarrow 1]$ using Fig. 7's games.

Achieving OAE2 security. Fix integers $n \geq \tau \geq 0$. For a string $X \in \{0,1\}^*$ and $1 \leq i \leq j \leq |X|$, let $X[i,j]$ denote the substring of X from the ith bit to the jth bit (inclusive). Let $\langle\cdot\rangle$ denote an encoding that maps a pair $(A, d) \in \{0,1\}^* \times \{0,1,2\}$ to a string $\langle A, d\rangle \in \{0,1\}^*$. For example, one can represent d by a two-bit string, and append this to A. Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be a conventional AE scheme of ciphertext-expansion τ, nonce space $\{0,1\}^n$, and AD space $\{0,1\}^*$. Fig. 8 defines a segmented-AE scheme $\mathbf{CHAIN}[\Pi, \langle\cdot\rangle, n] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with segment expansion τ, nonce space $\{0,1\}^n$, AD space $\{0,1\}^*$, and state space $\mathbf{K} \times \{0,1\}^n$. The proof of the following theorem is in the full version [33].

Theorem 1. Let Π, $\langle\cdot\rangle, n$, and $\mathbf{CHAIN}[\Pi, \langle\cdot\rangle, n]$ be as above. There is an explicit reduction R with the following property. For any adversary $\mathscr{A}$, adversary $\mathscr{B} = R(\mathscr{A})$ satisfies $\mathbf{Adv}^{\mathrm{oae2b}}_{\mathbf{CHAIN}[\Pi,\langle\cdot\rangle,n]}(\mathscr{A}) \leq \mathbf{Adv}^{\mathrm{pri}}_{\Pi}(\mathscr{B}) + 2q^2/2^n$ where q is the number of $\mathscr{A}$'s queries. Adversary $\mathscr{B}$ uses about the same running time as $\mathscr{A}$ and the total length of $\mathscr{B}$'s queries is that of $\mathscr{A}$ plus at most $5qn$ bits.

Discussion. In $\mathcal{E}$.next and $\mathcal{D}$.next, the state is computed via $M[1,n] \oplus C[1,n]$. One might instead xor the n-bit suffix of M and C; this makes no difference. On the other hand, suppose one uses *just* $C[1,n]$, eliminating the xor with $M[1,n]$. Call this variant $\mathbf{CHAIN1}[\Pi, \langle\cdot,\rangle, n]$. The method is insecure for small τ. Here is an attack for the case $\tau = 0$. The adversary makes a single query $(N, \boldsymbol{A}, \boldsymbol{C})$ to the decryption oracle, where N is arbitrary, $\boldsymbol{A} = (\varepsilon, \varepsilon, \varepsilon)$ and $\boldsymbol{C} = (0^n, 0^n, 0^n, 0^n)$. Let the answer be $\boldsymbol{M} = (M_1, M_2, M_3, M_4)$. The adversary will output 1 only if $M_2 = M_3$. In the **Ideal2B** game the strings M_2 and M_3 are independent random strings. However, in game **Real2B** we always have $M_2 = M_3 = \mathbf{D}_K(0^n, \langle\varepsilon, 0\rangle, 0^n)$. Hence the adversary can win with advantage $1 - 2^{-n}$. In contrast, for large τ, scheme $\mathbf{CHAIN1}[\Pi, \langle\cdot,\rangle, n]$ *is* OAE2 secure.

$\mathcal{E}$ algorithms	$\mathcal{D}$ algorithms
proc $\mathcal{E}.\text{init}(K, N)$ **ret** (K, N)	**proc** $\mathcal{D}.\text{init}(K, N)$ **ret** (K, N)
proc $\mathcal{E}.\text{next}(S, A, M)$ $(K, V) \leftarrow S$; $C \leftarrow \mathbf{E}_K(V, \langle A, 0\rangle, M)$ **if** $\lvert M\rvert \geq n$ **then** $V \leftarrow (C[1,n] \oplus M[1,n])$ **else** $V \leftarrow (\mathbf{E}_K(V, \langle A, 2\rangle, M \,\|\, 0^n))[1,n]$ **ret** $(C, (K, V))$	**proc** $\mathcal{D}.\text{next}(S, A, C)$ $(K, V) \leftarrow S$; $M \leftarrow \mathbf{D}_K(V, \langle A, 0\rangle, C)$ **if** $M = \perp$ **then ret** $(\perp, \perp)$ **if** $\lvert M\rvert \geq n$ **then** $V \leftarrow C[1,n] \oplus M[1,n]$ **else** $V \leftarrow (\mathbf{E}_K(V, \langle A, 2\rangle, M \,\|\, 0^n))[1,n]$ **ret** $(M, (K, V))$
proc $\mathcal{E}.\text{last}(S, A, M)$ $(K, V) \leftarrow S$; **ret** $\mathbf{E}_K(V, \langle A, 1\rangle, M)$	**proc** $\mathcal{D}.\text{last}(S, A, C)$ $(K, V) \leftarrow S$; **ret** $\mathbf{D}_K(V, \langle A, 1\rangle, C)$

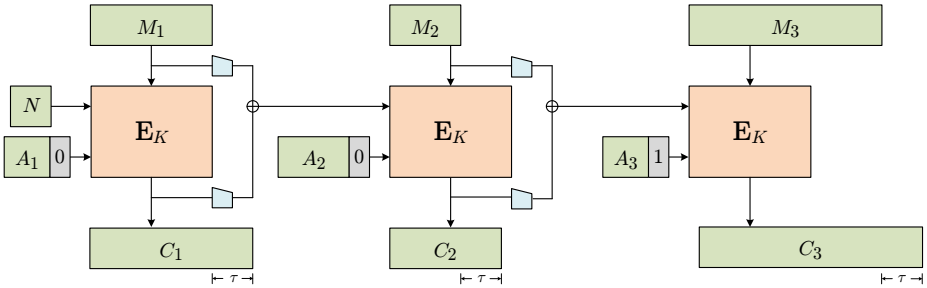

Fig. 8: **The CHAIN construction for OAE2. Top:** Encryption scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$, secure as a PRI with expansion τ, is turned into a segmented-AE scheme $\mathbf{CHAIN}[\Pi, \langle\cdot\rangle, n] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with $\mathcal{K} = \mathbf{K}$. **Bottom:** Illustration of the scheme. Each segment of (M_1, M_2, M_3) has at least n bits. Trapezoids represent truncation to n bits.

To achieve OAE2 with multivalued segment-expansion, use an RAE-secure underlying scheme [34], a generalization of PRI that allows one to select an arbitrary ciphertext-expansion for each query. The construction is modified in the natural way.

7 Escalating Claims, Diminishing Guarantees

A survey of the literature shows increasingly strong rhetoric surrounding nonce-reuse security of online schemes. We document this trend. In doing so we identify some of the notions (all quite weak, in our view) that have come to be regarded as nonce-reuse misuse-resistant.

Shifting language. The paper defining MRAE [55] never suggested that nonce-reuse was OK; it said that an MRAE scheme must do "as well as possible with whatever IV is provided" [55, p. 1]. Elaborating, the authors "aim for an AE scheme in which if the IV is a nonce then one achieves the usual notion for nonce-based AE; and if the IV does get repeated then authenticity remains and privacy is compromised only to the extent that [one reveals] if this plaintext is

equal to a prior one, and even that ... only if both the message and its header have been used with this particular IV" [55, p. 12–13].

The FFL paper indicates that the authors wish "to achieve both simultaneously: security against nonce-reusing adversaries ... and support for on-line-encryption" [28, p. 197]. While the authors understood that they were weakening MRAE, they saw the weakening as relatively inconsequential: they say that their scheme, McOE, "because of being on-line, satisfies a *slightly weaker* security definition against nonce-reusing adversaries" [28, p. 198] (emphasis ours). The paper did not investigate the definitional consequences of this weakening.

An early follow-on to FFL, the COPA paper, asserts that OAE1 schemes are distinguished by "not relying on the non-reuse of a nonce" [9, p. 438]. Andreeva *et al.* classify AE schemes according to the type of initialization vector (IV) one needs: either *random*, *nonce*, or *arbitrary*. A scheme satisfying OAE1 is understood to be an arbitrary-IV scheme, where "no restrictions on the IV are imposed, thus an adversary may choose any IV for encryption" [7, p. 9]. The authors add that "Often a deterministic AE scheme does not even have an IV input" [7, p. 9]. The linguistic progression reaches its logical conclusion in the rebranding of OAE1-secure schemes as *nonce-free*, as seen, for example, in recent talks of Guo [32, slide 2] and Lauridsen [21, Slides 4, 6].

We have thus seen a transformation in language, played out over eight years, taking us from a strong definition (MRAE) pitched as trying to capture the best one can do when a nonce gets reused to a comparatively weak definition (OAE1) nowadays pitched as being so strong so as to render nonces superfluous. Meanwhile, the best-one-can-do positioning of MRAE was mirrored in the online setting. The COPA authors indicate that their mode achieves "the maximum attainable for single pass schemes" [8, p. 7]. Identical language is found in the COBRA submission [11, p. 7]. In our view, such claims are wrong; there would seem to be a significant gap between OAE1 and OAE2 security.

Weaker notions. Concurrent with the rhetoric for what OAE1 delivers being ratcheted up, weakened variants of OAE1 have proliferated. We document this trend in Fig. 9, which introduces a variety of OAE notions. They are all weaker than OAE1 except for OAE1a; by standard arguments, OAE1 and OAE1a are quantitatively close if the blocksize is reasonably large. 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.

The number of different OAE definitions, and their meanings, has never been clear. The evolution of what's been indicated in the Nonce-MR column of the AE Zoo [14] illustrates the struggle of researchers trying to accurately summarize the extent of nonce-reuse misuse-resistance for tens of AE schemes. Our own attempt at sorting this out, Fig. 9, is not definitive. We do not formalize the notions in this table except for OAE1. (Some of the definitions are obvious, some are not.) The table is based on both author assertions (Schemes1) and assertions of others (Schemes2). The OAE1x notions only consider security for messages that are blocksize multiples.

OAE1 Leaks equality of block-aligned prefixes, formalized by comparing $\mathcal{E}_K$ with: a random n-bit-blocksize online permutation tweaked by the nonce, AD and plaintext; followed by a random τ-bit function of the nonce, AD, and plaintext. Schemes1: COPA [9], Deoxys [37], Joltik [38], KIASU [39], Marble [32], McOE [28], SHELL [63], POET [1,2], Prøst-COPA [20] Schemes2: ++AE [51]
OAE1a Leaks equality of block-aligned prefixes, formalized by comparing $\mathcal{E}_K$ with: a random n-bit-blocksize online *function* tweaked by the nonce, AD and plaintext; followed by a random τ-bit function of the nonce, AD, and plaintext. Schemes1: APE [6], ELmD [25], ELmE [26], Prøst-APE [20]
OAE1b Leaks equality of block-aligned prefixes, formalized by comparing $\mathcal{E}_K$ with: a random n-bit-blocksize online *function* tweaked by the nonce and plaintext (but *not* the AD); followed by a random τ-bit function of the nonce, AD, and plaintext. The relaxation enables a compliant scheme to process the plaintext before the AD is presented. However it also renders a compliant scheme vulnerable to CCA, CPSS, and NM attacks even if AD values are unique. Schemes1: COBRA [12]
OAE1c Leaks equality of any blocks at the same position. E.g., if ciphertexts C and C' arise from 4-block plaintexts $P = \mathrm{A} \parallel \mathrm{B} \parallel \mathrm{C} \parallel \mathrm{D}$ and $P' = \mathrm{E} \parallel \mathrm{B} \parallel \mathrm{F} \parallel \mathrm{D}$ then $C_2 = C'_2$ and $C_4 = C'_4$. Security is formalized by comparing $\mathcal{E}_K$ with: a function from n bits to n bits tweaked by the nonce and an integer, the position; followed by a random tag. Schemes1: Minalpher [59]
OAE1d Leaks equality of block-aligned prefixes and the XOR of the block directly following this prefix. E.g., if C, C' arise from 4-block plaintexts $P = \mathrm{A}\,\|\,\mathrm{B}\,\|\,\mathrm{C}\,\|\,\mathrm{D}$ and $P' = \mathrm{A}\|\mathrm{B}\|\mathrm{E}\|\mathrm{F}$ we always have $C_1 = C'_1$, $C_2 = C'_2$, and $C_3 \oplus C'_3 = \mathrm{C} \oplus \mathrm{E}$. Ciphertexts C, C' arising from 4-block plaintexts $P = \mathrm{A}\,\|\,\mathrm{B}\,\|\,\mathrm{C}\,\|\,\mathrm{D}$ and $P' = \mathrm{E}\,\|\,\mathrm{F}\,\|\,\mathrm{G}\,\|\,\mathrm{H}$ will have $C_1 \oplus C'_1 = \mathrm{A} \oplus \mathrm{E}$. Schemes2: Artemia [5] CBEAM [57], ICEPOLE [50], iFeed [66], Jambu [64], Keyak [18], MORUS [65], NORX [13], STRIBOB [58]
NAE1 Retains full security as long as all (N, A) pairs are unique among the encryption queries. If a pair repeats, all privacy is lost, but authenticity remains unchanged. Schemes1: CLOC [35], SILC [36]
NAE0 Retains full security as long as all (N, A) pairs are unique among the encryption queries. If a pair repeats, all security is forfeit. Schemes1: NORX [13], Triviack [24] Schemes2: OTR [47]

Fig. 9: **A menagerie of OAE notions and schemes.** All of the schemes are CAESAR submissions except ElmE and McOE. Schemes1 lists proposals that claim some flavor of nonce-reuse misuse resistance. Schemes2 lists proposals that didn't, yet are or were marked as such in the AE Zoo [14] or AFL survey [4].

Acknowledgments

The authors appreciate the excellent comments received from the Keyak/Ketje team: Joan Daemen, Guido Bertoni, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Their feedback called our attention to the duplexing-the-sponge paper [19] and led to our decision to generalize to vector-valued AD and to remove the key K from .next and .last calls. We appreciate further com-

ments and corrections from Farzaneh Abed, Nasour Bagheri, Dan Bernstein, Danilo Gligoroski, Stefan Lucks, Samuel Neves, and Kenny Paterson.

Much of the work on this paper was done while Phil Rogaway was visiting Ueli Maurer's group at ETH Zürich. Many thanks to Ueli for hosting that sabbatical. Rogaway was also supported by NSF grants CNS-1228828 and CNS-1314885. Reyhanitabar and Vizár were partially supported by Microsoft Research under the Swiss Joint Research Centre MRL Contract No. 2014-006 (DP1061305).

References

1. Abed, F., Fluhrer, S., Foley, J., Forler, C., List, E., Lucks, S., McGrew, D., Wenzel, J.: The POET Family of On-Line Authenticated Encryption Schemes (Version 1.01). CAESAR submission (2014)
2. Abed, F., Fluhrer, S., Forler, C., List, E., Lucks, S., McGrew, D., Wenzel, J.: Pipelineable On-Line Encryption. Cryptology ePrint report 2014/297 (2014) Also FSE 2014. LNCS, vol. 8540. Springer (2015)
3. Abed, F., Forler, C., List, E., Lucks, S., Wenzel, J.: Don't Panic! The Cryptographer's Guide to Robust (On-line) Encryption: Draft, March 11, 2015.
4. Abed, F., Forler, C., Lucks, S.: General Overview of the First-Round CAESAR Candidates for Authenticated Encryption. Cryptology ePrint report 2014/792 (2014)
5. Alizadeh, J., Aref, M. R., Bagheri, N.: Artemia v1. CAESAR submission (2014)
6. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In: FSE 2014. LNCS, vol. 8540. Springer (2015)
7. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to Securely Release Unverified Plaintext in Authenticated Encryption. In: ASIACRYPT (1) 2014. LNCS, vol. 8873, pp. 105–125. Springer (2015)
8. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: AES-COPA v.1. CAESAR submission (2014)
9. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and Authenticated Online Ciphers. In: ASIACRYPT 2013. LNCS, vol. 8269, pp. 424–443. Springer (2013)
10. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable (Authenticated) Online Ciphers. DIAC presentation (2013)
11. Andreeva, E., Luykx, A., Mennink, B., Yasuda, K.: AES-COBRA v1. CAESAR submission (2014)
12. Andreeva, E., Luykx, A., Mennink, B., Yasuda, K.: COBRA: A Parallelizable Authenticated Online Cipher without Block Cipher Inverse. In: FSE 2014. LNCS, vol. 8540. Springer (2015)
13. Aumasson, J. P., Jovanovic, P., Neves, S.: NORX v1. CAESAR submission (2014)
14. Authenticated Encryption Zoo. https://aezoo.compute.dtu.dk
15. Bellare, M., Boldyreva, A., Knudsen, L., Namprempre, C.: Online Ciphers and the Hash-CBC Construction. In: CRYPTO 2001. LNCS, vol. 2139, pp. 292–309. Springer (2001)
16. Bellare, M., Rogaway, P.: Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: ASIACRYPT 2000. LNCS, vol. 1976, pp. 317–330. Springer (2000)
17. Bernstein, D.: Cryptographic competitions: CAESAR. http://competitions.cr.yp.to

18. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: CAESAR submission: Keyak v1. CAESAR submission (2014)
19. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: SAC 2011 (Selected Areas in Cryptography). LNCS, vol. 7118, pp. 320–337. Springer (2012). Earlier version in: The Second SHA-3 Candidate Conference (2010)
20. Bilge Kavun, E., Lauridsen, M., Leander, G., Rechberger, C., Schwabe, P., Yalçın, T.: Prøst v1.1. CAESAR submission (2014)
21. Bogdanov, A., Lauridsen, M., Tischhauser, E.: AES-Based Authenticated Encryption Modes in Parallel High-Performance Software. DIAC presentation (2014)
22. Boldyreva, A., Degabriele, J.P., Paterson, K., Stam, M.: Security of Symmetric Encryption in the Presence of Ciphertext Fragmentation. EUROCRYPT 2012. LNCS, vol. 7237, pp. 682–699. Springer (2012)
23. Boldyreva, A., Taesombut, N.: Online Encryption Schemes: New Security Notions and Constructions. In: CT-RSA 2014. LNCS, vol. 2964, pp. 1–14. Springer (2004). Full version on the first authors' webpage.
24. Chakraborti, A., Nandi, M.: TriviA-ck-v1. CAESAR submission. (2014)
25. Datta, N., Nandi, M.: ELmD v1.0. CAESAR submission. (2014)
26. Datta, N., Nandi, M.: ELmE: A Misuse Resistant Parallel Authenticated Encryption. In: ACISP 2014. LNCS, vol. 8544, pp. 306–321. Springer (2014)
27. Duong, T., Rizzo, J.: Here Come The ⊕ Ninjas. Manuscript (2011).
28. Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In: FSE 2012. LNCS, vol. 7549, pp. 196–215. Springer (2012)
29. Fleischmann, E., Forler, C., Lucks, S., Wenzel, J.: McOE: A Foolproof On-line Authenticated Encryption Scheme. Cryptology ePrint report 2011/644 (2013)
30. Fouque, P.-A., Joux, A., Martinet, G., Valette, F.: Authenticated On-Line Encryption. In: SAC 2003. LNCS, vol. 3006, pp. 145–159. Springer (2003)
31. Fouque, P.-A., Martinet, G., Poupard, G.: Practical Symmetric On-line Encryption. In: FSE 2003. LNCS, vol. 3006, pp.145–159. Springer (2003)
32. Guo, J.: Marble Specification Version 1.0. CAESAR submission (2014). Also DIAC presentation (2014)
33. Hoang, V. T., Reyhanitabar, R., Rogaway, P., Vizár, D: Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. Cryptology ePrint Archive, Report 2015/189 (2015).
34. Hoang, V. T., Krovetz, T., Rogaway, P.: Robust Authenticated Encryption: AEZ and the Problem that it Solves. In: EUROCRYPT 2015. LNCS, vol. 9056. Springer (2015)
35. Iwata, I., Minematsu, K., Guo, J., Morioka, S.: CLOC: Compact Low-Overhead CFB. CAESAR submission. (2014)
36. Iwata, T., Minematsu, K., Guo, J., Morioka, S., Kobayashi, E.: SILC: SImple Lightweight CFB. CAESAR submission. (2014)
37. Jean, J., Nikolić, I., Peyrin, T.: Deoxys v1. CAESAR submission. (2014)
38. Jean, J., Nikolić, I., Peyrin, T.: Joltik v1. CAESAR submission. (2014)
39. Jean, J., Nikolić, I., Peyrin, T.: KIASU v1. CAESAR submission. (2014)
40. Joux, A., Martinet, G., Valette, F.: Blockwise Adaptive Attakers: Revisiting the (In)security of Some Provably Secure Encryption Modes: CBC, GEM, IACBC. In: CRYPTO 2002. LNCS, vol. 2442, pp. 17–30. Springer (2002)
41. Katz, J., Yung, M.: Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In: FSE 2000. LNCS, vol. 1978, pp. 284–299. Springer (2001)

42. Krovetz, T., Rogaway, P.: The OCB Authenticated-Encryption Algorithm. RFC 7253. Internet Research Task Force (IRTF) and Crypto Forum Research Group (CFRG) (2014)
43. Liskov, M., Rivest, R., Wagner, D.: Tweakable Block Ciphers. J. of Cryptology, 24(3), pp. 588–614. Springer (2011)
44. Lucks, S.: Personal communication (2014)
45. McGrew, D., Fluhrer, S., Lucks, S., Forler, C., Wenzel, J., Abed, F., List, E.: Pipelineable On-Line Encryption. In: FSE 2014. LNCS, vol. 8540. Springer (2015)
46. Miaw, W.: Netflix / msl. (2014). https://github.com/Netflix/msl/wiki
47. Minematsu, K.: AES-OTR v1. CAESAR submission (2014)
48. Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. Cryptology ePrint Archive, Report 2013/628 (2013)
49. Möller, B.: Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures. web.archive.org/web/20120630143111/http://www.openssl.org/~bodo/tls-cbc.txt
50. Morawiecki, P., Gaj, K., Homsirikamol, E., Matusiewicz, K., Pieprzyk, J., Rogawski, M., Srebrny, M., Wójcik, M.: ICEPOLE v1. CAESAR submission (2014)
51. Recacha, F.: ++AE v1.0. CAESAR submission (2014)
52. Rogaway, P.: Authenticated-Encryption with Associated-Data. In: ACM CCS 2002. ACM Press, pp. 98–107 (2002)
53. Rogaway, P.: Problems with Proposed IP Cryptography. Manuscript (1995)
54. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In: ACM CCS 2001, pp. 196–205. ACM Press (2001)
55. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer (2006)
56. Rogaway, P., Zhang, H.: Online Ciphers from Tweakable Blockciphers. In: CT-RSA 2011. LNCS, vol. 6558, pp. 237–249. Springer (2011)
57. Saarinen, M.-J. O.: The CBEAMr1 Authenticated Encryption Algorithm. CAESAR submission (2014)
58. Saarinen, M.-J. O.: The STRIBOBr1 Authenticated Encryption Algorithm. CAESAR submission (2014)
59. Sasaki, Y., Todo, Y., Aoki, K., Naito, Y., Sugawara, T., Murakami, Y., Matsui, M., Hirose, S.: Minalpher v1. CAESAR submission (2014)
60. Touset, S.: Streaming API to Authenticated Encryption. Cryptography Stack Exchange (16 Jan 2013). http://crypto.stackexchange.com/questions/6008
61. Tsang, P., Solomakhin, R., Smith, S.: Authenticated Streamwise On-line Encryption. Dartmouth Computer Science Technical Report TR2009-640 (2009)
62. Vaudenay, S.: CBC padding: Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS, ... In: EUROCRYPT 2002. LNCS, vol. 2332, pp. 534–545. Springer (2002)
63. Wang, L.: SHELL v1. CAESAR submission (2014)
64. Wu, H., Huang, T.: JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU (v1). CAESAR submission (2014)
65. Wu, H., Huang, T.: The Authenticated Cipher MORUS (v1). CAESAR submission (2014)
66. Zhang, L, Wu, W., Sui, H., Wang, P.: iFeed[AES] v1. CAESAR submission (2014)