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

June 25, 2018

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.

Table of Contents

1 Introduction

Between nAE & MRAE. With typical nonce-based authenticated-encryption (nAE) schemes [53,55], nonces must never repeat when encrypting a series of messages; if they do, it is possible—and routine—that all security will be forfeit.[5] 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) [56]. 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 [56], 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 [44] 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

[5] Throughout this paper we ignore an annoying discursive problem surrounding the word *nonce*. The word is usually understood to mean something that does not repeat (in some context); if it does repeat, it's not a nonce. This would make the phrase *nonce repetition* a logical absurdity. For a more neutral term, Bernstein has advocated *message number* [17]. Others use *IV* (initialization vector). We will stick with *nonce* for a value that nominally *ought* not repeat, yet might.

	OAE1 (from FFL [28])	**OAE2** (new to this paper)
Definitional idea	Online cipher followed by a tag	Aencrypt each segment
Segmentation	Fixed-size blocks of scheme-determined lengths	Variable-size segments of user-determined lengths
Typical block/segment size	5–16 bytes	1–10000 bytes? Not cryptographer's decision
Ciphertext expansion	τ bits per message (eg, $\tau = 128$)	τ bits per segment (eg, $\tau = 128$)
Message space	$M \in \mathsf{B}_n^*$ for blocksize n	$M \in \{0,1\}^*$ (one view) or $\boldsymbol{M} \in \{0,1\}^{**}$ (another)
Decryption also online?	No, not in general	Yes, automatically
Can aencrypt infinite streams?	No, messages must end	Yes, messages can be conceptually infinite
OK to repeat nonces?	No, attacks are always possible	No, attacks are 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.

nonce-reuse. An AE scheme satisfying nAE (employing non-repeating nonces) or MRAE (without that restriction) would certainly be immune to such an attack.

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 [62]. 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 [47] 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) [62] 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" [62, 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 [62, 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. See Appendix B for more details, and for further discussion of related work.

A real-world need. Netflix recently described a protocol of theirs, MSL, for streaming video [47]. 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.

Even before the Netflix announcement, practitioners had been publicly asking for such a tool. For example, Stephen Touset writes: "I asked DJB [Dan Bernstein] [if] he had any intent to add a streaming API to an authenticated cipher. His response was ... that one should never release a decrypted plaintext before verifying the authenticator. However, this got me to thinking. ... Is it possible, or even advisable, to mimic a streaming interface?" [61].

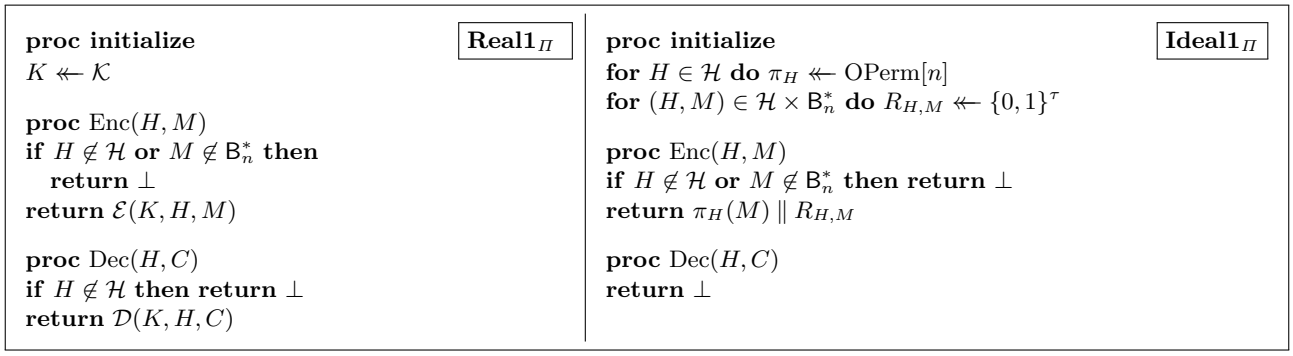

Fig. 2: **OAE1 security.** Defining security for a block-based AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with header space $\mathcal{H}$, blocksize n, and ciphertext expansion τ. See the accompanying text for the definition of OPerm$[n]$.

Errata. The CHAIN construction in the proceedings version [34] of our paper was buggy due to an improper domain separation constant. The problem has been corrected in the current paper. The specific details of the bugs are discussed at the end of Section 6.

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) [16,42,55]. 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 $|M|$.

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 $\pi \twoheadleftarrow$ 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 [45], 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 $\mathcal{E}_K^H(M)$ 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 something 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 $p \ge 1$. (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 $p \ge 1$. 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 Appendix C 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 [57, 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 all 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 $\{0,1\}^{\geq n}$ [57]. 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 $\boldsymbol{M} = \boldsymbol{\mathcal{D}}(K, N, \boldsymbol{A}, \boldsymbol{C})$.

CIPHERTEXT EXPANSION. We focus on segmented-AE schemes with constant segment-expansion, defined as follows: associated to Π is a number $\tau \ge 0$ 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

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

algorithm $\boldsymbol{\mathcal{D}}(K, N, \boldsymbol{A}, \boldsymbol{C})$
$m \leftarrow |\boldsymbol{C}|$
if $m = 0$ **or** $|\boldsymbol{A}| \neq |\boldsymbol{C}|$ **then return** 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 $m = 1$ **return** Λ
    else return $(M_1, \ldots, M_{i-1})$
  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 return** $(M_1, \ldots, M_{m-1})$
else return $(M_1, \ldots, M_m)$

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 $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$.

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 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 $\text{Inj}(\tau)$ denote the set of all τ-expanding injective functions—the set of all functions $f\colon \{0,1\}^* \to \{0,1\}^*$ that are injective and satisfy $|f(x)| = |x| + \tau$. Endow this set with the uniform distribution in the natural way. We write $f \twoheadleftarrow \text{Inj}(\tau)$ to denote uniformly sampling a random, τ-expanding injective function. Now define a distribution on functions $\text{IdealOAE}(\tau)$ 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**
  $f_{N,\boldsymbol{A},\boldsymbol{M},0} \twoheadleftarrow \text{Inj}(\tau)$; $f_{N,\boldsymbol{A},\boldsymbol{M},1} \twoheadleftarrow \text{Inj}(\tau)$
for $m \in \mathbb{Z}^+$, $\boldsymbol{A} \in (\{0,1\}^*)^m$, $\boldsymbol{X} \in (\{0,1\}^*)^m$, $\delta \in \{0,1\}$ **do**
  $F(N, \boldsymbol{A}, \boldsymbol{X}, \delta) \leftarrow (f_{N,\boldsymbol{A}[1..1],\Lambda,0}(\boldsymbol{X}[1]),\ f_{N,\boldsymbol{A}[1..2],\boldsymbol{X}[1..1],0}(\boldsymbol{X}[2]),$
    $f_{N,\boldsymbol{A}[1..3],\boldsymbol{X}[1..2],0}(\boldsymbol{X}[3]),\ \ldots,\ f_{N,\boldsymbol{A}[1..m-1],\boldsymbol{X}[1..m-2],0}(\boldsymbol{X}[m-1]),$
    $f_{N,\boldsymbol{A}[1..m],\boldsymbol{X}[1..m-1],\delta}(\boldsymbol{X}[m]))$
return F

Thus $F \twoheadleftarrow \text{IdealOAE}(\tau)$ 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.

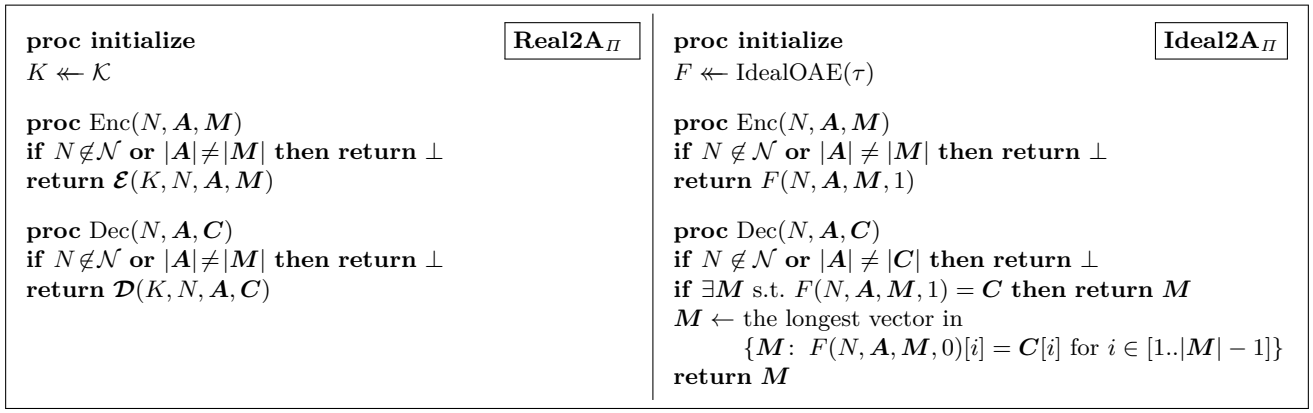

Fig. 4: **OAE2a security.** The segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has nonce space $\mathcal{N}$ and segment-expansion τ. It induces algorithms $\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}}$ as per Fig. 3. The distribution IdealOAE(τ) is described in the text.

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. This is our first measure of OAE2 security.

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 [35], 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.

Real2B$_{\Pi}$

proc initialize
$I, J \leftarrow 0$; $K \twoheadleftarrow \mathcal{K}$

proc Enc.init(N)
if $N \notin \mathcal{N}$ **then return** $\perp$
$I \leftarrow I + 1$; $S_I \leftarrow \mathcal{E}.\text{init}(K, N)$
return I

proc Enc.next(i, A, M)
if $i \notin [1..I]$ **or** $S_i = \perp$ **then return** $\perp$
$(C, S_i) \leftarrow \mathcal{E}.\text{next}(S_i, A, M)$
return C

proc Enc.last(i, A, M)
if $i \notin [1..I]$ **or** $S_i = \perp$ **then return** $\perp$
$C \leftarrow \mathcal{E}.\text{last}(S_i, A, M)$
$S_i \leftarrow \perp$; **return** C

proc Dec.init(N)
if $N \notin \mathcal{N}$ **then return** $\perp$
$J \leftarrow J + 1$; $S'_J \leftarrow \mathcal{D}.\text{init}(K, N)$
return J

proc Dec.next(j, A, C)
if $j \notin [1..J]$ **or** $S'_j = \perp$ **then return** $\perp$
$(M, S'_j) \leftarrow \mathcal{D}.\text{next}(S'_j, A, C)$
return M

proc Dec.last(j, A, C)
if $j \notin [1..J]$ **or** $S'_j = \perp$ **then return** $\perp$
$M \leftarrow \mathcal{D}.\text{last}(S'_j, A, C)$
$S'_j \leftarrow \perp$
return M

Ideal2B$_{\Pi}$

proc initialize
$I, J \leftarrow 0$; $F \twoheadleftarrow \text{IdealOAE}(\tau)$

proc Enc.init(N)
if $N \notin \mathcal{N}$ **then return** $\perp$
$I \leftarrow I + 1$; $N_I \leftarrow N$; $\boldsymbol{A}_I \leftarrow \Lambda$; $\boldsymbol{M}_I \leftarrow \Lambda$
return I

proc Enc.next(i, A, M)
if $i \notin [1..I]$ **or** $\boldsymbol{M}_i = \perp$ **then return** $\perp$
$\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \,\|\, A$; $\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \,\|\, M$; $m \leftarrow |\boldsymbol{M}_i|$
$\boldsymbol{C} \leftarrow F(N_i, \boldsymbol{A}_i, \boldsymbol{M}_i, 0)$; **return** $\boldsymbol{C}[m]$

proc Enc.last(i, A, M)
if $i \notin [1..I]$ **or** $\boldsymbol{M}_i = \perp$ **then return** $\perp$
$\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \,\|\, A$; $\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \,\|\, M$; $m \leftarrow |\boldsymbol{M}_i|$
$\boldsymbol{C} \leftarrow F(N_i, \boldsymbol{A}_i, \boldsymbol{M}_i, 1)$; $\boldsymbol{M}_i \leftarrow \perp$; **return** $\boldsymbol{C}[m]$

proc Dec.init(N)
if $N \notin \mathcal{N}$ **then return** $\perp$
$J \leftarrow J + 1$; $N'_J \leftarrow N$; $\boldsymbol{A}'_j \twoheadleftarrow \Lambda$; $\boldsymbol{C}_J \leftarrow \Lambda$
return J

proc Dec.next(j, A, C)
if $j \notin [1..J]$ **or** $\boldsymbol{C}_j = \perp$ **then return** $\perp$
$\boldsymbol{A}'_j \leftarrow \boldsymbol{A}_j \,\|\, A$; $\boldsymbol{C}_j \leftarrow \boldsymbol{C}_j \,\|\, C$; $m \leftarrow |\boldsymbol{C}_j|$
if $\exists \boldsymbol{M}$ s.t. $F(N'_j, \boldsymbol{A}'_j, \boldsymbol{M}, 0) = \boldsymbol{C}_j$
then return $\boldsymbol{M}[m]$
else $\boldsymbol{C}_j \leftarrow \perp$; **return** $\perp$; **fi**

proc Dec.last(j, A, C)
if $j \notin [1..J]$ **or** $\boldsymbol{C}_j = \perp$ **then return** $\perp$
$\boldsymbol{A}'_j \leftarrow \boldsymbol{A} \,\|\, A$; $\boldsymbol{C}_j \leftarrow \boldsymbol{C}_j \,\|\, C$; $m \leftarrow |\boldsymbol{C}_j|$
if $\exists \boldsymbol{M}$ s.t. $F(N'_j, \boldsymbol{A}'_j, \boldsymbol{M}_j, 1) = \boldsymbol{C}_j$
then $\boldsymbol{C}_j \leftarrow \perp$; **return** $\boldsymbol{M}[m]$
else $\boldsymbol{C}_j \leftarrow \perp$; **return** $\perp$ **fi**

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

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 Enc($N_i, \boldsymbol{A}_i\|\varepsilon, \boldsymbol{M}_i\|\varepsilon$) call, and returning its $|\boldsymbol{M}_i|$-th component. Calls of the form Enc.last(i, A, M) result in making an Enc($N_i, \boldsymbol{A}_i\|A, \boldsymbol{M}_i\|M$) 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 $\mathbf{Adv}^{\text{oae2a}}_{\Pi}(\mathscr{B}) = \mathbf{Adv}^{\text{oae2b}}_{\Pi}(\mathscr{A})$. 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.

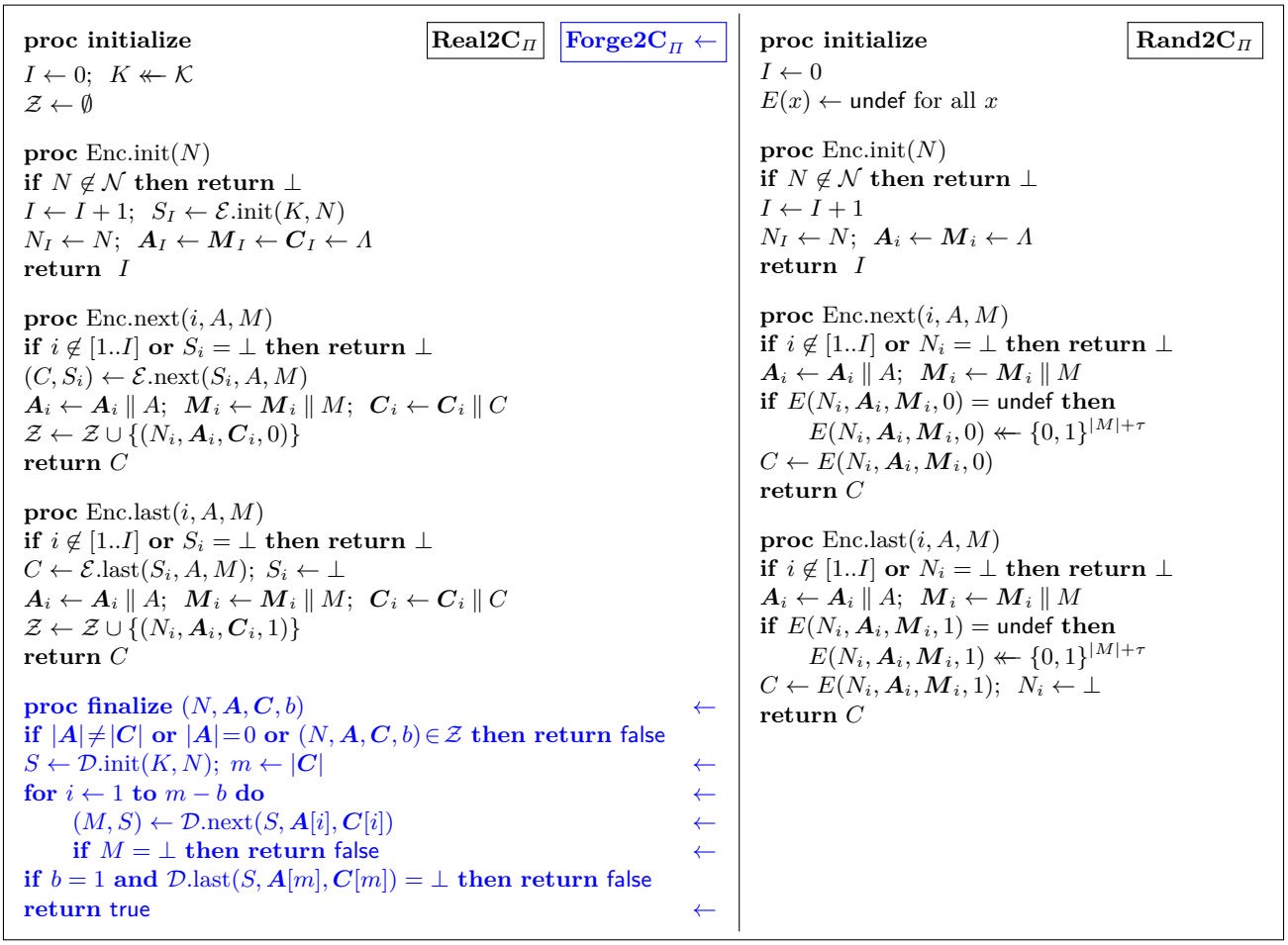

Fig. 6: **OAE2c security.** Privacy and authenticity are separately defined, the first by comparing games **Real2C** and **Rand2C**, and the second using game **Forge2C**, which includes the additional lines indicated.

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. We will return to this in Section 7.

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 Appendices E.2 and E.3.

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 $\mathbf{Adv}^{\text{oae2-priv}}_{\Pi}(\mathscr{A}_1) \le \mathbf{Adv}^{\text{oae2b}}_{\Pi}(\mathscr{B}_1) + q^2/2^\tau$ and $\mathbf{Adv}^{\text{oae2-auth}}_{\Pi}(\mathscr{A}_2) \le \mathbf{Adv}^{\text{oae2b}}_{\Pi}(\mathscr{B}_2) + \ell/2^\tau$, where q is the number of $\mathscr{A}_1$'s queries and 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; see Appendix D.

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) [56]. The notion is close to MRAE [56], with a gap $q^2/2^{s+\tau} + q/2^\tau$

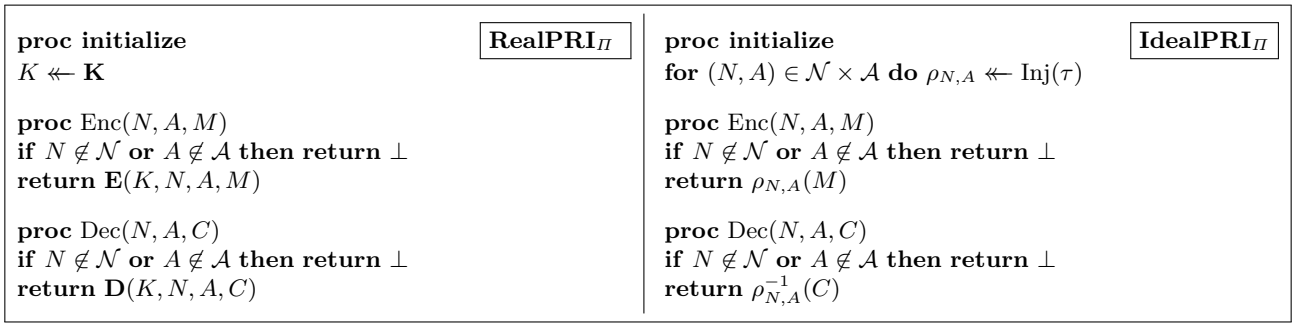

Fig. 7: **PRI security.** Defining security for an AE scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ with expansion τ, nonce space $\mathcal{N}$, and AD space $\mathcal{A}$. Here $\mathrm{Inj}(\tau)$ is the set of all injective functions $f\colon \{0,1\}^* \to \{0,1\}^*$ such that $|f(x)| = |x|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

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 [56] if τ is large, say $\tau = 128$, or AEZ scheme [35], 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 $\mathbf{D}_K^{N,A}(\mathbf{E}_K^{N,A}(M)) = M$. 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 $(A, d) \in \{0,1\}^* \times \{0,1,2,3,4,5\}$ to a string $\langle A, d \rangle \in \{0,1\}^*$. For example, one can represent d by a three-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 Appendix E.1.

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.

To achieve OAE2 with multivalued segment-expansion, use an RAE-secure underlying scheme [35], a generalization of PRI that allows one to select an arbitrary ciphertext-expansion for each query. The construction is modified in the natural way.

proc $\mathcal{E}$.init(K, N) — $\mathcal{E}$ algorithms
return $(K, N, 0)$

proc $\mathcal{E}$.next(S, A, M)
$(K, V, d) \leftarrow S$; $C \leftarrow \mathbf{E}_K(V, \langle A, d\rangle, M)$
if $|M| \geq n$ **then** $V \leftarrow (C[1,n] \oplus M[1,n])$
else $V \leftarrow (\mathbf{E}_K(V, \langle A, d+4\rangle, M \,\|\, 0^n))[1,n]$
return $(C, (K, V, 1))$

proc $\mathcal{E}$.last(S, A, M)
$(K, V, d) \leftarrow S$
if $d = 0$ **then** $d \leftarrow 3$ **else** $d \leftarrow 2$
return $\mathbf{E}_K(V, \langle A, d\rangle, M)$

proc $\mathcal{D}$.init(K, N) — $\mathcal{D}$ algorithms
return $(K, N, 0)$

proc $\mathcal{D}$.next(S, A, C)
$(K, V, d) \leftarrow S$; $M \leftarrow \mathbf{D}_K(V, \langle A, d\rangle, C)$
if $M = \perp$ **then return** $(\perp, \perp)$
if $|M| \geq n$ **then** $V \leftarrow C[1,n] \oplus M[1,n]$
else $V \leftarrow (\mathbf{E}_K(V, \langle A, d+4\rangle, M \,\|\, 0^n))[1,n]$
return $(M, (K, V, 1))$

proc $\mathcal{D}$.last(S, A, C)
$(K, V, d) \leftarrow S$
if $d = 0$ **then** $d \leftarrow 3$ **else** $d \leftarrow 2$
return $\mathbf{D}_K(V, \langle A, d\rangle, C)$

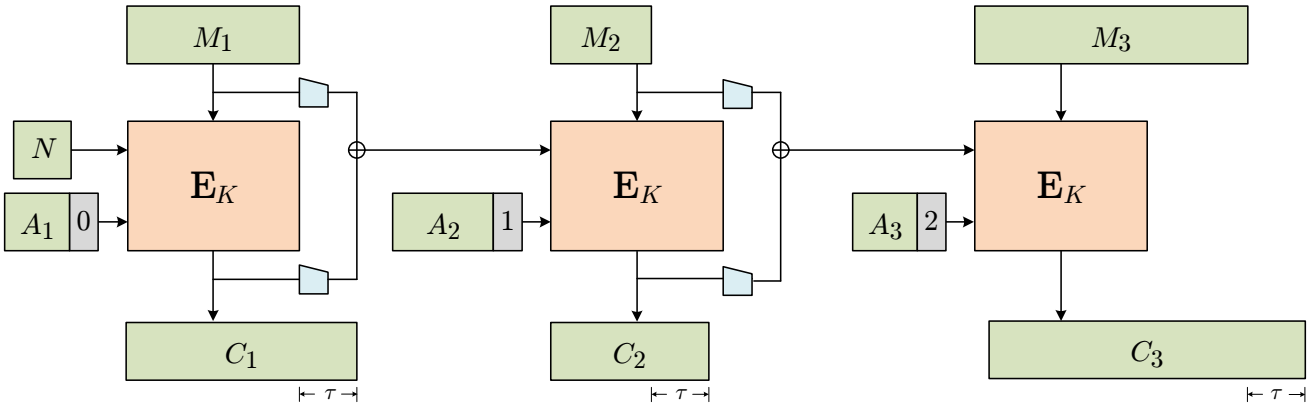

Fig. 8: **The CHAIN construction for OAE2. Top:** Encryption scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$, secure as a PRI with expansion τ, is turned into a segmented-AE scheme $\mathbf{CHAIN}[\Pi, \langle\cdot\rangle, n] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with $\mathcal{K} = \mathbf{K}$. **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.

Bugs in the proceedings version. Note that in the **CHAIN** construction, when we encrypt a segmented message $\boldsymbol{M} = (M_1, \ldots, M_m)$ with a segmented AD $\boldsymbol{A} = (A_1, \ldots, A_m)$, the first message segment M_1 is encrypted with AD $\langle A_1, 0\rangle$, whereas any middle message segment M_i (with $1 < i < m$) is encrypted with AD $\langle A_i, 1\rangle$. The distinction in the treatment of the first segment and the middle ones is crucial for security. In the proceedings version [34], we instead encrypted middle segments M_i with AD $\langle A_i, 0\rangle$, which leads to the following attack. The adversary would pick any arbitrary nonce N, and query $(N, \boldsymbol{A}, \boldsymbol{M})$ to the encryption oracle to get $\boldsymbol{C} = (C_1, C_2, C_3)$, where $\boldsymbol{M} = (0^n, 0^n, 0^n)$ and $\boldsymbol{A} = (\varepsilon, \varepsilon, \varepsilon)$. It then queries $(N', \boldsymbol{A}', \boldsymbol{M}')$ to the encryption oracle to get $\boldsymbol{C}' = (C'_1, C'_2)$, where $N' = C_1$, $\boldsymbol{A}' = (\varepsilon, \varepsilon)$ and $\boldsymbol{M}' = (0^n, 0^n)$. The adversary then outputs 1 if $C'_1 = C_2$, and outputs 0 otherwise. In game **Real2A** we always have $C'_1 = C_2$. In contrast, in game **Ideal2A**, the chance that $N' \neq N$ is at least $1 - 1/2^n$, and given $N' \neq N'$, the conditional probability that $C'_1 = C_2$ is at most $1/2^n$. In other words, in game **Ideal2A**, the chance that $C_1 = C_2$ is at most $2/2^n$. Hence the adversary wins with advantage at least $1 - 2/2^n$.

Note that when the segmented message has only one component, meaning $m = 1$, then we have to encrypt M_1 with $\langle A_1, 3\rangle$. If we instead use $\langle A_1, 2\rangle$ then one can attack the scheme as follows. The adversary would pick any arbitrary nonce N, and query $(N, \boldsymbol{A}, \boldsymbol{M})$ to the encryption oracle to get $\boldsymbol{C} = (C_1, C_2)$, where $\boldsymbol{M} = (0^n, 0^n)$ and $\boldsymbol{A} = (\varepsilon, \varepsilon)$. It then queries $(N', \boldsymbol{A}', \boldsymbol{M}')$ to the encryption oracle to get $\boldsymbol{C}' = (C'_1)$, where $N' = C_1$, $\boldsymbol{A}' = (\varepsilon)$ and $\boldsymbol{M}' = (0^n)$. The adversary then outputs 1 if $C'_1 = C_2$, and outputs 0 otherwise. Again this adversary wins with advantage at least $1 - 2/2^n$.

On the other hand, in processing a very short segment $\boldsymbol{M}[i]$ (meaning $|\boldsymbol{M}[i]| < n$), we update the state via $V \leftarrow (\mathbf{E}_K(V, \langle A, 4\rangle, \boldsymbol{M}[i] \,\|\, 0^n))[1,n]$ if $i = 1$, otherwise $V \leftarrow (\mathbf{E}_K(V, \langle A, 5\rangle, \boldsymbol{M}[i] \,\|\, 0^n))[1,n]$.

If we instead always use $V \leftarrow (\mathbf{E}_K(V, \langle A, 4\rangle, \boldsymbol{M}[i] \parallel 0^n))[1, n]$ for every $i \geq 1$ then one can break the scheme as follows. The adversary first picks an arbitrary nonce N and queries $(N, \boldsymbol{A}, \boldsymbol{M})$ to the encryption oracle, where $\boldsymbol{A} = (\varepsilon, \varepsilon, \varepsilon)$ and $\boldsymbol{M} = (0^n, 0, 0)$, to receive $\boldsymbol{C} = (C_1, C_2, C_3)$. Next, it queries $(N', \boldsymbol{A}', \boldsymbol{M}')$ to the encryption oracle, where $N' = C_1 \oplus \boldsymbol{M}[1]$, $\boldsymbol{A}' = (\varepsilon, \varepsilon)$, and $\boldsymbol{M}' = (0, 0)$, to receive $\boldsymbol{C}' = (C'_1, C'_2)$. The adversary outputs 1 if $C'_2 = C_3$, and outputs 0 otherwise. In the ideal world, the chance that the adversary outputs 1 is merely $1/2^{\tau+1}$. In contrast, in the real world:

- On the one hand, for the first query, the state V_1 that we produce after processing the first segment is $\boldsymbol{M}[1] \oplus C_1 = N'$, and $C_3 \leftarrow \mathbf{E}_K(V_2, \langle \varepsilon, 2\rangle, 0)$, where $V_2 \leftarrow (\mathbf{E}_K(V_1, \langle \varepsilon, 4\rangle, 0^{n+1}))[1, n]$.
- On the other hand, for the second query, the state V'_1 that we produce after processing the first segment is $(\mathbf{E}_K(N', \langle \varepsilon, 4\rangle, 0^{n+1}))[1, n] = V_2$, and C'_2 is $\mathbf{E}_K(V'_1, \langle \varepsilon, 2\rangle, 0) = C_3$.

Thus in the real world, the adversary always outputs 1. Hence the adversary wins with advantage $1 - 2^{-(\tau+1)}$.

The old proof of Theorem 1 missed the bugs above: it claims incorrectly that the last game in the game chain corresponds to the ideal game, and also incorrectly analyzes the collision of the state V for very short segments. Those mistakes are corrected in this version.

7 Weakening OAE2

The CPSS attack applies to OAE2-secure schemes as much as to OAE1-secure ones, suggesting that, even for OAE2, users should still be directed not to repeat a nonce. In this section we explore the extent to which OAE2 can be simplified if one insists on such a restriction.

Notions nOAE and dOAE. We consider two different relaxations of OAE2. The first, nOAE, effectively demands of the encrypting party that it not reuse any nonce. Apart from that, everything's the same as with OAE2. If nonces *are* reused, then nothing is guaranteed. Potentially all security is forfeit. The nOAE notion is adequate for settings where the encrypting party does what it's *supposed* to do: it never repeats a nonce. When saying that nonces don't get repeat we mean, informally, that the adversary can adaptively grow only one segmented plaintext (one *chain*, one might say) per nonce.

The second weakening of OAE2 we consider, dOAE, slots between nOAE2 and OAE2. Now nonces *may* be reused; what's required is that one not encrypt $(N, (A_1, \ldots, A_{m-1}, A_m), (M_1, \ldots, M_{m-1}, M_m))$ and then encrypt $(N, (A_1, \ldots, A_{m-1}, A_m), (M_1, \ldots, M_{m-1}, M'_m))$ with $M_m \neq M'_m$ and where either both messages are either *incomplete* (more segments are to come) or both are *complete* (the segmented string is over). It's as though the *effective-nonce* for plaintext segment M_m was defined to be N, $(A_1, \ldots, A_m)$, $(M_1, \ldots, M_{m-1})$, and an indication as to whether the message is complete; and, on encryption, no effective-nonce may repeat. This is a much more liberal requirement than demanding that nonces not repeat. We name this notion dOAE, as a nod to the idea coming from the paper on the duplex construction by Bertoni, Daemen, Peeters, and Van Assche [19].

The advantage of weakening OAE2 to either nOAE or dOAE is that simpler or more efficient constructions becomes possible. Our constructions will illustrate these efficiency advantages. More specifically, good OAE2-security requires that every plaintext segment be processed in full before any bit of that ciphertext segment can issue: online processing of segments is impossible. Plus, segments must be processed in order, and the AD for each segment must be processed prior to processing the plaintext segment. In contrast, for dOAE and nOAE, online processing of individual segments *is* possible. This facilitates use of more standard and efficient AE schemes can be used as building blocks. For dOAE, segments must still be processed sequentially, and the AD of a segment must still be processed before the segment. But there are nOAE-secure schemes (such as **STREAM**) in which these restrictions vanish.

The nOAE and dOAE definitions can be formalized using any of the three approaches from Section 5. This would give rise to approximately-equivalent notions nOAEa, nOAEb, nOAEc, and approximately-equivalent notions dOAEa, dOAEb, dOAEc. The reader will be relieved that we don't intend to give six

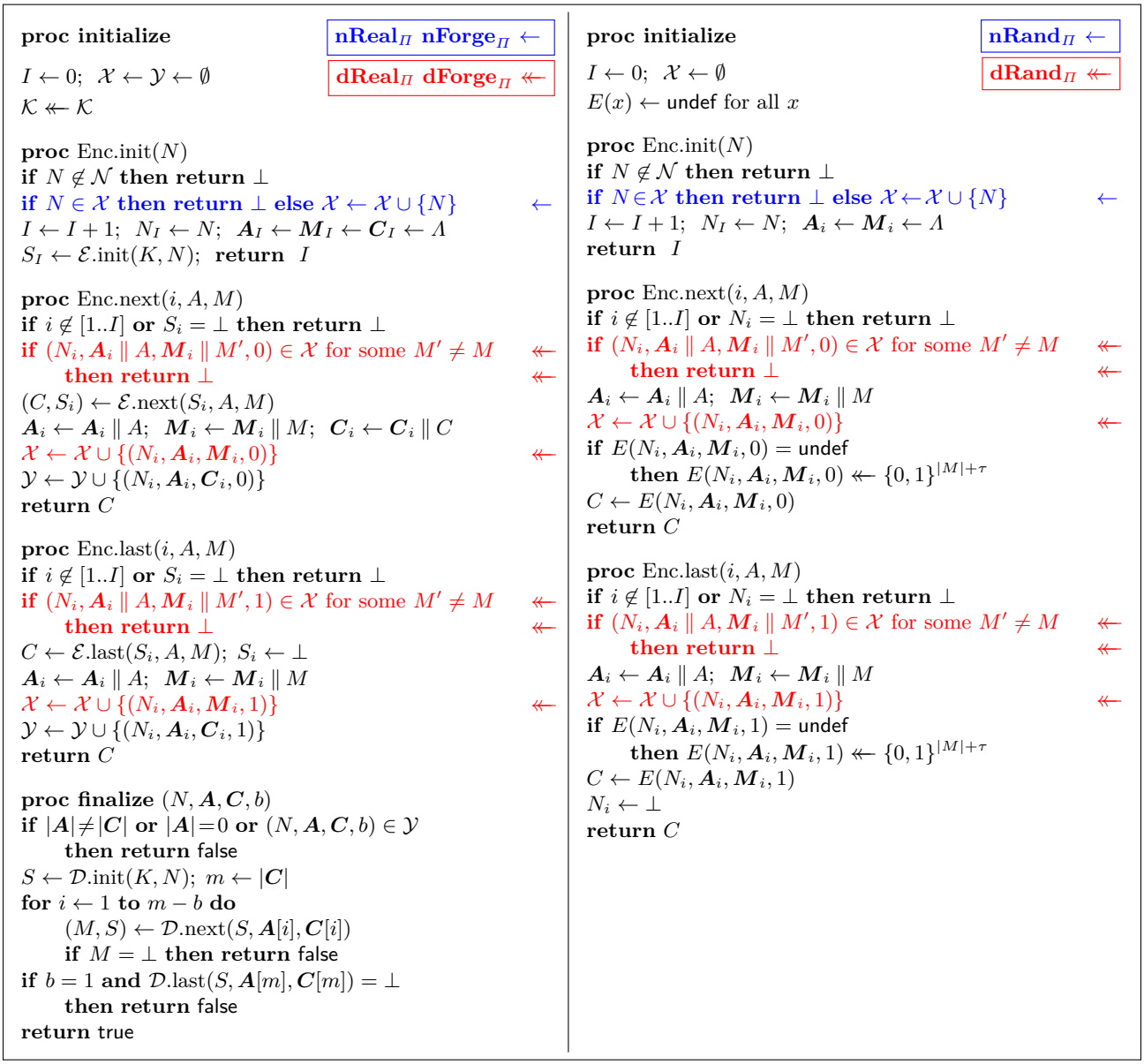

Fig. 9: **nOAE and dOAE security.** Games **nForge**/**dForge** include the **finalize** procedure; games **nReal**/**dReal** don't. Games **nReal**/**nForge**/**nRand** include lines marked like this ←; the other games include those marked like this ↞.

new advantage measures in this section. We choose the third (the 'c') approach for defining dOAE and nOAE, the definition that used separate privacy and authenticity parts. We make this choice because this approach is easiest to work with and because we *want* users to select a reasonably large segment expansion τ, which becomes an implicit requirement for good security.

Definitions for nOAE and dOAE. Let Π be a segmented-AE scheme with segment-expansion τ and nonce-space $\mathcal{N}$. Fig. 9 defines games **nReal**, **nForge**, **dReal**, **dForge**, **nRand**, and **dRand**. The games are only slightly tweaked from corresponding games **Real2C**, **Forge2C**, and **Rand2C**. Specifically, we now return $\perp$ if an adversary asks a *disallowed* queries, as implicitly defined by the code. Let

$$\mathbf{Adv}^{\text{noae-priv}}_{\Pi}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{nReal}_\Pi} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{nRand}_\Pi} \Rightarrow 1] \quad \text{and} \quad \mathbf{Adv}^{\text{noae-auth}}_{\Pi}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{nForge}_\Pi}]$$
$$\mathbf{Adv}^{\text{doae-priv}}_{\Pi}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{dReal}_\Pi} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{dRand}_\Pi} \Rightarrow 1] \quad \text{and} \quad \mathbf{Adv}^{\text{doae-auth}}_{\Pi}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{dForge}_\Pi}]$$

$\mathcal{E}$ algorithms	$\mathcal{D}$ algorithms
proc $\mathcal{E}.\text{init}(K, N)$ **return** $(K, N, 1)$	**proc** $\mathcal{D}.\text{init}(K, N)$ **return** $(K, N, 1)$
proc $\mathcal{E}.\text{next}(S, A, M)$ $(K, N, i) \leftarrow S$; $S \leftarrow (K, N, i+1)$ $C \leftarrow \mathbf{E}_K(\langle N, i, 0\rangle, A, M)$ **return** (C, S)	**proc** $\mathcal{D}.\text{next}(S, A, C)$ $(K, N, i) \leftarrow S$; $M \leftarrow \mathbf{D}_K(\langle N, i, 0\rangle, A, C)$ **if** $M = \perp$ **then return** $(\perp, \perp)$ $S \leftarrow (K, N, i+1)$; **return** (M, S)
proc $\mathcal{E}.\text{last}(S, A, M)$ $(K, N, i) \leftarrow S$; **return** $\mathbf{E}_K(\langle N, i, 1\rangle, A, M)$	**proc** $\mathcal{D}.\text{last}(S, A, M)$ $(K, N, i) \leftarrow S$; **return** $\mathbf{D}_K(\langle N, i, 1\rangle, A, C)$

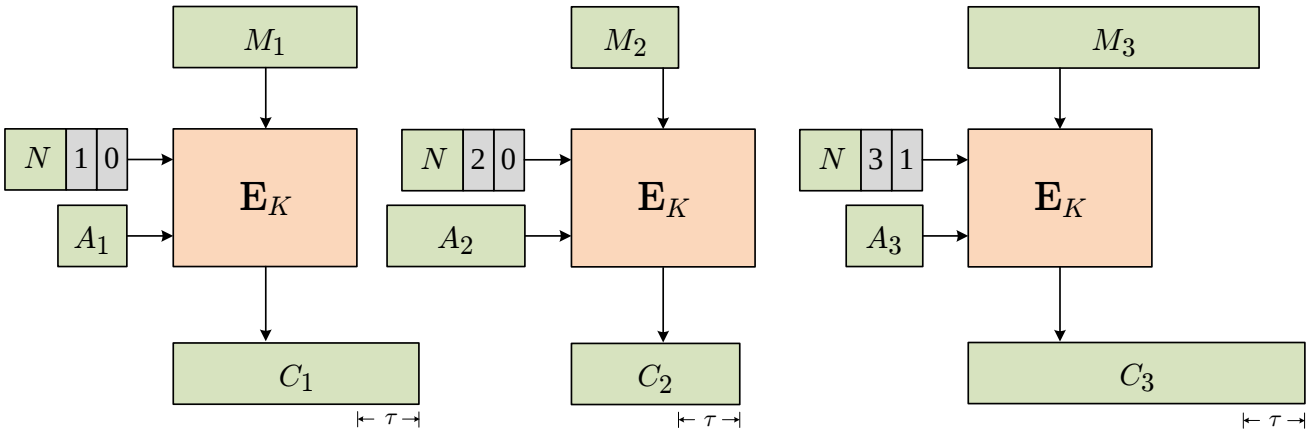

Fig. 10: **The STREAM construction for nOAE.** Encryption scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ secure as an nAE with ciphertext expansion τ is turned into a segmented-AE scheme $\Pi' = (\mathcal{K}, \mathcal{E}, \mathcal{D}) = \mathbf{STREAM}[\Pi, \langle\cdot\rangle]$ with key space $\mathcal{K} = \mathbf{K}$.

We remark that there are two ways to formalize the weakening of OAE2 to nOAE and dOAE: forbid the adversary from asking particular queries, or allow the adversary to ask any queries, but give it a useless value if it asks a disallowed query. We have elected the second approach because it result in more self-contained pseudocode.

We now show how to translate the classic nAE to nOAE. We must first recall the nAE notion.

nAE security. A nonce-based AE scheme (an nAE scheme) is a triple $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ where $\mathbf{K}$ is a nonempty set endowed with a distribution and $\boldsymbol{\mathcal{E}}\colon\ \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^* \to \{0,1\}^*$ is the encryption algorithm and $\boldsymbol{\mathcal{D}}\colon\ \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ is the decryption algorithm. There must be a number τ, the ciphertext expansion, such that if $C = \mathcal{E}_K^{N,A}(M) = E_K(N, A, M) = \mathcal{E}(K, N, A, M)$ then $|C| = |M| + \tau$. Encryption and decryption reverse one another: $\mathcal{D}_K(N, A, C) = M \in \{0,1\}^*$ iff $\mathcal{E}_K(N, A, M) = C \in \{0,1\}^*$. We assume that $\varepsilon \in \mathcal{A}$. Define the nAE security of Π against an adversary $\mathscr{A}$ as $\mathbf{Adv}^{\text{nae}}_{\Pi}(\mathscr{A}) = \Pr[K \twoheadleftarrow \mathbf{K}\colon\ \mathscr{A}^{\mathcal{E}_K(\cdot,\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathscr{A}^{\$(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1]$, where $\$(\cdot,\cdot,\cdot)$ is an oracle that returns $C \twoheadleftarrow \{0,1\}^{|M|+\tau}$ for any input (N, A, M) and $\perp(\cdot,\cdot,\cdot)$ is an oracle that returns $\perp$ for any input. The adversary is prohibited from repeating a nonce N in queries to its first oracle and from asking (N, A, C) to its second oracle after receiving C from a prior query (N, A, M) to its first oracle.

Constructing an nOAE-secure scheme. Fix an encoding function $\langle\cdot\rangle$ that maps a tuple of strings $(N, i, d) \in \mathcal{N}' \times \mathcal{I} \times \{0,1\}$ to a string $\langle N, i, d\rangle$. Here $\mathcal{I} = \mathbb{N}$ or else $\mathcal{I} = \{1, 2, \ldots, \mathsf{max}\}$ for some $\mathsf{max} \in \mathbb{N}$ (the maximum number of segments in any message). Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be an nAE scheme with AD space $\mathcal{A}$, and nonce space $\mathcal{N}$ that includes all possible values of $\langle N, i, d\rangle$. Such an nAE scheme is said to be *compatible* with the encoding function. We now describe the **STREAM** construction, defined and illustrated in Fig. 10, to turn Π and $\langle\cdot\rangle$ into an nOAE-secure segmented-AE scheme $\mathbf{STREAM}[\Pi, \langle\cdot\rangle]$ whose AD space is $\mathcal{A}$ and whose nonce space is $\mathcal{N}'$. The theorem below shows

that $\Pi' = \mathbf{STREAM}[\Pi, \langle\cdot\rangle] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ achieves nOAE security. The proof is in Appendix E.4. For it we assume that, if $\mathcal{N}' = [1..\mathsf{max}]$, then the adversary asks no query resulting in an $\langle N, i, d\rangle$ value with $i > \mathsf{max}$.

Theorem 2. Fix an encoding scheme $\langle\cdot\rangle$ and a compatible nAE scheme Π. There are explicitly 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 $\mathbf{Adv}^{\text{noae-priv}}_{\mathbf{STREAM}[\Pi,\langle\cdot\rangle]}(\mathscr{A}_1) \le \mathbf{Adv}^{\text{nae}}_{\Pi}(\mathscr{B}_1)$ and $\mathbf{Adv}^{\text{noae-auth}}_{\mathbf{STREAM}[\Pi,\langle\cdot\rangle]}(\mathscr{A}_2) \le \mathbf{Adv}^{\text{nae}}_{\Pi}(\mathscr{B}_2)$. For each $i \in \{1, 2\}$, adversary $\mathscr{B}_i$ uses about the same running time as $\mathscr{A}_i$, and the total length of its queries is at most that of $\mathscr{A}_i$'s queries plus the total length of induced nonces of $\mathcal{E}$ on running $\mathscr{A}_i$'s queries.

Discussion. The state space for $\Pi' = \mathbf{STREAM}[\Pi, \langle\cdot\rangle]$ is the set $\mathcal{S}$ of points in $\mathcal{K} \times \mathcal{N}' \times \mathcal{I}$. This set is infinite when $\mathcal{I} = \mathbb{N}$, the setting where one places no limit on the number of possible segments. It is also infinite if $\mathcal{N}'$ is. This means that, depending on these choices, the segmented-AE scheme Π' might not qualify as "online" under the definition given in Section 5.

In practice, however, the construction either is online or can easily be made so. One normally *would* set a limit max on the number of possible segments, and one would usually limit the length of nonces, too. We have therefore elected to present **STREAM** in what we regard as its simplest form, even if the algorithm, in that form, would not qualify as online for some encoding schemes.

As for the nonce space $\mathcal{N}'$ of Π', we note that the usual reason that nAE schemes sport a fairly large nonce space, like $\mathcal{N}' = \{0,1\}^{96}$, is that there may be a large number of messages encrypted under a single key. In the nOAE setting, we doubt that this will routinely be so. For here one needs one nonce for each *stream* (segmented message), and, in many settings, the number of streams associate to a key will be one—or at least few. As a concrete example, if one were starting with an nAE scheme with nonce space of 15 or fewer bytes [43], one might select an encoding function $\langle N, i, d\rangle$ that concatenates an 8-byte N, a 6-byte encoding of i, and a 1-byte encoding of d.

Using STREAM for Netflix. Recall that Netflix is currently using an nAE scheme to encrypt each segment of a movie independently, with the segment positioning used as AD. Compared to this solution, **STREAM** is more efficient. If the nAE scheme is OCB then we save one blockcipher call per segment, since we don't have to process the positioning as AD. The saving may be even more, since for Netflix's solution, if a user is viewing several movies at the same time[6], then the AD of each segment must contain the movie ID. In contrast, for **STREAM**, we only need to specify the movie ID in the AD of the *first* segment of each movie; subsequent ciphertext segments of the same nonce will be associated to the same movie. Moreover, in **STREAM**, we only have to generate a nonce per movie, whereas Netflix has to generate a nonce per segment.

On the other hand, suppose that one uses **STREAM** to encrypt movies. If a user skips around—say she's jumping to the 100th segment after watching the first one, and then goes back to the 20th one and keeps watching till the end of the movie—then it's tempting to use the same nonce to encrypt the requested segments. But this reveals user's watching pattern: an adversary will know that the user watched the 100th segment twice, since the same ciphertext segment is transmitted twice. We instead suggest that one should treat this as a request for three segmented-strings: $(M_1, M_2, \cdots), (M_{100}, M_{101}, \cdots)$, and $(M_{20}, M_{21}, \cdots)$ and use three distinct nonces.

Constructing a dOAE-secure scheme. If we consider only one-segment strings, then the dOAE notion degenerates to nAE0, a variant of nAE that guarantees security only if one never reuses (N, A). Formally, let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be a nonce-based AE scheme with ciphertext expansion τ. Define the nAE0 security of Π against an adversary $\mathscr{A}$ as

$$\mathbf{Adv}^{\text{nae0}}_{\Pi}(\mathscr{A}) = \Pr[K \xleftarrow{} \mathbf{K} \colon \mathscr{A}^{\mathcal{E}_K(\cdot,\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathscr{A}^{\$(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1],$$

[6] Actually, Netflix forbids users from opening two movies at the same time, but this is allowed by other movie providers such as Amazon Instant Video.

where $\$(\cdot,\cdot,\cdot)$ is an oracle that returns $C \xleftarrow{\$} \{0,1\}^{|M|+\tau}$ for any input (N,A,M) and $\perp(\cdot,\cdot,\cdot)$ is an oracle that returns $\perp$ for any input. The adversary is prohibited from repeating a pair (N,A) in queries to its first oracle and from asking (N,A,C) to its second oracle after receiving C from a prior query (N,A,M) to its first oracle. An nAE0-secure AE scheme can be easily realized; for example, apply a PRF to distill an effective nonce N' from (N,A), then use N' as a nonce on an nAE-secure scheme. Theorem 3 below shows that the $\mathbf{CHAIN}[\Pi,\langle\cdot\rangle,n]$ construction of Section 6 is dOAE-secure if Π is nAE0-secure. The proof is in Appendix E.5.

Theorem 3. Let Π, $\langle\cdot\rangle, n$, and $\mathbf{CHAIN}[\Pi,\langle\cdot\rangle,n]$ be as in Section 6. There are explicit reductions R_1 and R_2 with the following property. For any adversaries $\mathscr{A}_1$ and $\mathscr{A}_2$, adversaries $\mathscr{B}_1 = R(\mathscr{A}_1)$ and $\mathscr{B}_2 = R_2(\mathscr{A}_2)$ satisfy $\mathbf{Adv}^{\text{doae-priv}}_{\mathbf{CHAIN}[\Pi,\langle\cdot\rangle,n]}(\mathscr{A}_1) \le 2\,\mathbf{Adv}^{\text{nae0}}_{\Pi}(\mathscr{B}_1) + p^2/2^n$ and $\mathbf{Adv}^{\text{doae-auth}}_{\mathbf{CHAIN}[\Pi,\langle\cdot\rangle,n]}(\mathscr{A}_2) \le 2\,\mathbf{Adv}^{\text{nae0}}_{\Pi}(\mathscr{B}_2) + q^2/2^n$, where p is the number of $\mathscr{A}_1$'s queries and q is the number of $\mathscr{A}_2$'s queries plus the number of segments of $\mathscr{A}_2$'s output. For each $i \in \{1,2\}$, adversary $\mathscr{B}_i$ uses about the same running time as $\mathscr{A}_i$. The total length of $\mathscr{B}_1$'s queries is that of $\mathscr{A}_1$ plus at most $5np$ bits, whereas the total length of $\mathscr{B}_2$'s queries is that $\mathscr{A}_2$ plus at most $5nq$ bits.

Remark. As mentioned in Section 6, a prior version of the **CHAIN** construction is buggy, susceptible to OAE2-attacks. One can also mount the same attacks to break the dOAE security of that buggy version. The old proof of Theorem 3 missed those bugs, committing similar mistakes as the old proof of Theorem 1; those mistakes are corrected in this version.

8 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 [56] never suggested that nonce-reuse was OK; it said that an MRAE scheme must do "as well as possible with whatever IV is provided" [56, 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" [56, 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

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 [38], Joltik [39], KIASU [40], Marble [32], McOE [28], SHELL [64], POET [1,2], Prøst-COPA [20] Schemes2: ++AE [52]
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 $P = \mathrm{A} \,\|\, \mathrm{B} \,\|\, \mathrm{C} \,\|\, \mathrm{D}$ and $P' = \mathrm{E} \,\|\, \mathrm{B} \,\|\, \mathrm{F} \,\|\, \mathrm{D}$ 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 [60]
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 [58], ICEPOLE [51], iFeed [67], Jambu [65], Keyak [18], MORUS [66], NORX [13], STRIBOB [59]
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 [36], SILC [37]
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], Trivia-ck [24] Schemes2: OTR [48]

Fig. 11: **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].

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. 11, 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. 11, 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.

9 Concluding Remarks

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.

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 comments and corrections from Farzaneh Abed, Nasour Bagheri, Dan Bernstein, Danilo Gligoroski, Stefan Lucks, Samuel Neves, and Kenny Paterson. We thank Eik List for pointing out a bug in the **CHAIN** construction's treatment of very short segments.

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. https://www.uni-weimar.de/fileadmin/user/fak/medien/professuren/Mediensicherheit/Research/Drafts/nonce-misuse-oae.pdf
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., Reyhanitabar, R., Rogaway, P., Vizár, D: Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. In: CRYPTO 2015, LNCS, vol. 9215, pp. 493–517. Springer (2015)
35. Hoang, V. T., Krovetz, T., Rogaway, P.: Robust Authenticated Encryption: AEZ and the Problem that it Solves. In: EUROCRYPT 2015. LNCS, vol. 9056, pp. 15–44. Springer (2015)
36. Iwata, I., Minematsu, K., Guo, J., Morioka, S.: CLOC: Compact Low-Overhead CFB. CAESAR submission. (2014)
37. Iwata, T., Minematsu, K., Guo, J., Morioka, S., Kobayashi, E.: SILC: SImple Lightweight CFB. CAESAR submission. (2014)
38. Jean, J., Nikolić, I., Peyrin, T.: Deoxys v1. CAESAR submission. (2014)
39. Jean, J., Nikolić, I., Peyrin, T.: Joltik v1. CAESAR submission. (2014)
40. Jean, J., Nikolić, I., Peyrin, T.: KIASU v1. CAESAR submission. (2014)
41. 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)
42. Katz, J., Yung, M.: Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In: FSE 2000. LNCS, vol. 1978, pp. 284–299. Springer (2001)
43. Krovetz, T., Rogaway, P.: The OCB Authenticated-Encryption Algorithm. RFC 7253. Internet Research Task Force (IRTF) and Crypto Forum Research Group (CFRG) (2014)
44. Liskov, M., Rivest, R., Wagner, D.: Tweakable Block Ciphers. J. of Cryptology, 24(3), pp. 588–614. Springer (2011)
45. Lucks, S.: Personal communication (2014)
46. 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)
47. Miaw, W.: Netflix / msl. (2014). https://github.com/Netflix/msl/wiki
48. Minematsu, K.: AES-OTR v1. CAESAR submission (2014)
49. Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. Cryptology ePrint Archive, Report 2013/628 (2013)
50. Möller, B.: Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures. https://web.archive.org/web/20120630143111/http://www.openssl.org/~bodo/tls-cbc.txt
51. Morawiecki, P., Gaj, K., Homsirikamol, E., Matusiewicz, K., Pieprzyk, J., Rogawski, M., Srebrny, M., Wójcik, M.: ICEPOLE v1. CAESAR submission (2014)
52. Recacha, F.: ++AE v1.0. CAESAR submission (2014)
53. Rogaway, P.: Authenticated-Encryption with Associated-Data. In: ACM CCS 2002. ACM Press, pp. 98–107 (2002)
54. Rogaway, P.: Problems with Proposed IP Cryptography. Manuscript (1995)
55. 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)
56. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer (2006)
57. Rogaway, P., Zhang, H.: Online Ciphers from Tweakable Blockciphers. In: CT-RSA 2011. LNCS, vol. 6558, pp. 237–249. Springer (2011)
58. Saarinen, M.-J. O.: The CBEAMr1 Authenticated Encryption Algorithm. CAESAR submission (2014)
59. Saarinen, M.-J. O.: The STRIBOBr1 Authenticated Encryption Algorithm. CAESAR submission (2014)
60. Sasaki, Y., Todo, Y., Aoki, K., Naito, Y., Sugawara, T., Murakami, Y., Matsui, M., Hirose, S.: Minalpher v1. CAESAR submission (2014)
61. Touset, S.: Streaming API to Authenticated Encryption. Cryptography Stack Exchange (16 Jan 2013). Also see response by "fgrieu" (23 Jan 2013) http://crypto.stackexchange.com/questions/6008
62. Tsang, P., Solomakhin, R., Smith, S.: Authenticated Streamwise On-line Encryption. Dartmouth Computer Science Technical Report TR2009-640 (2009)

63. 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)
64. Wang, L.: SHELL v1. CAESAR submission (2014)
65. Wu, H., Huang, T.: JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU (v1). CAESAR submission (2014)
66. Wu, H., Huang, T.: The Authenticated Cipher MORUS (v1). CAESAR submission (2014)
67. Zhang, L, Wu, W., Sui, H., Wang, P.: iFeed[AES] v1. CAESAR submission (2014)

A Anticipated Objections

Criticizing a definition that many people are invested in might be contentious. We would like to anticipate some possible objections to our work.

Objection 1. What OAE2 is formalizing is fine, but it isn't closely related to what OAE1 tried to do. OAE1 aims to capture misuse-resistant online-AE. The blocksize is fixed because our tools are like that. OAE2 aims to capture something quite different, what one might call *streaming* AE. It's an utterly different problem, and criticizing OAE1 because it doesn't solve that which OAE2 solves is completely unfair.

The critique goes wrong by conflating our normative claim with the objection's descriptive one. The descriptive claim is that OAE1 and OAE2 are very different notions. Agreed; they certainly are. The normative claim is that OAE1 solves the "wrong" problem. Wrong in the sense that something deservedly called nonce-reuse misuse-resistance is impossible for online schemes, and wrong in the sense that OAE1 fails to capture the real-world characteristics of the problem that practitioners actually need to have solved.

Objection 2. OAE1 is only trying to achieve a *reasonable* level of security. It doesn't aim to achieve a super-strong notion like MRAE, which ends up too inefficient for some applications. It is only a "second line of defense" [4, p. 8], and one that is certainly better than nothing.

The objection amounts to saying that it's fine that a notion is weak if the authors didn't intend otherwise. We agree—assuming it is clearly communicated how strong or weak the notion is. Section 8 documents the opposite, at least in the years after FFL. In general, 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. This is of course subjective, due to the word *comparable*, yet we think that OAE1—as well as OAE2—fail this test.

Objection 3. AE that is not online is completely useless for a bunch of applications. If you really think that the users of these applications don't deserve as much misuse resistance as they can get, then say so.

We concur that there *are* users who need online-AE, and they *do* deserve to have cryptographers attending to this need. Where we disagree is more specific, and turns on the question of whether OAE1 is the right definition for satisfying users' needs, and if it actually *does* provide users as much nonce-reuse misuse-resistance as they can get.

We would hasten to add in that the same users also deserve clear discourse about what the definitions do and don't imply. Echoing our response to Objection 2, 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.

Objection 4. The paper gives two constructions, one for OAE2 (**CHAIN**) and another for nOAE (**STREAM**). Attending to the former setting suggests that the authors believe that nonces *might* get reused in some contexts, and, when this happens, that OAE2 and **CHAIN** are appropriate. But this contradicts the authors' claim that all formulations of OAE are useless if nonces get reused.

First, we try to avoid words like *useless*; a more accurate description of our claim would be that, in the presence of nonce-reuse, online-AE schemes are incapable of achieving what we'd regard as a desirable security notion. But to more directly answer the question: the OAE2 definition *is* stronger than nOAE, and, unlike nOAE, it does capture a reasonable approximation of best-possible security for an OAE scheme (even in the presence of nonce-reuse). For this reason we believe OAE2 worth formalizing and investigating, which includes giving a construction. But our doing so is not license to reuse the nonce. We discourage users from doing so.

B Related Work

There are several definitions in the literature that relate to those developed here. In this section we sketch some of them and compare them with OAE1, OAE2, or nOAE.

BDPV. In advance of proving security for their *duplex* construction, Bertoni, Daemen, Peeters, and Van Assche (BDPV) [19] provided novel syntax and security definitions for AE. They did not name their notion, so we will call it *duplex-AE*. It resembles both OAE2 and nOAE, and inspired dOAE.

Beginning with syntax, the encryption algorithm $\mathcal{E}$ of a duplex-AE scheme takes a key $K \in \mathcal{K}$ and segmented strings $\boldsymbol{A} \in \{0,1\}^{**}$ and $\boldsymbol{M} \in \{0,1\}^{**}$ having an equal number of components. The algorithm returns, deterministically, $C \parallel T = \mathcal{E}_K(\boldsymbol{A}, \boldsymbol{M})$. This is the ciphertext and tag that the *final plaintext segment* of $\boldsymbol{M}$, namely $M = \boldsymbol{M}[|\boldsymbol{M}|]$, encrypts to. Ciphertext C must have the same length as M while tag T is a τ-bit string (for some scheme-dependent constant τ). The decryption syntax is similar: given $\boldsymbol{A}$ and $\boldsymbol{C}$ with the same number of components, $\mathcal{D}_K(\boldsymbol{A}, \boldsymbol{C}, T)$ is either a plaintext $M \in \{0,1\}^*$ of the same length as $\boldsymbol{C}$'s final segment $C = \boldsymbol{C}[|\boldsymbol{C}|]$, or else it's a distinguished *error* value.

To be considered secure, a duplex-AE scheme must satisfy privacy and authenticity requirements. The first says that the map from $(\boldsymbol{A}, \boldsymbol{M})$ to $\mathcal{E}_K(\boldsymbol{A}, \boldsymbol{M})$ should resemble a random oracle—one that always returns $|\boldsymbol{M}[|\boldsymbol{M}|]| + \tau$ random bits. This requirement is levied only for adversaries that meet the following *nonce-requirement*: if an adversary queries $(\boldsymbol{A} \parallel A,\ \boldsymbol{M} \parallel M)$ then it makes no subsequent query of $(\boldsymbol{A} \parallel A,\ \boldsymbol{M} \parallel M')$ with $M \neq M'$. (Here $\boldsymbol{A}$ and $\boldsymbol{M}$ are segmented strings with an equal number of components, while A, M, M' are strings.) The authenticity requirement, on the other hand, says that reasonable adversaries can only rarely forge, where forging means outputting an $(\boldsymbol{A}, \boldsymbol{C}, T)$ for which $\mathcal{D}_K(\boldsymbol{A}, \boldsymbol{C}, T) \in \{0,1\}^*$ (not an *error*) and the adversary made no prior encryption query $\mathcal{E}_K(\boldsymbol{A}, \boldsymbol{M})$ that returned $C \parallel T$ with $\boldsymbol{C}[|\boldsymbol{C}|] = C$.

Comparing duplex-AE to OAE2 and nOAE is made difficult by the multitude of syntactic differences. First, there is no nonce with duplex-AE: the role played by a nonce in nOAE is effective subsumed by the nonce-requirement. While one might imagine that the first segment of AD, $\boldsymbol{A}[1]$, routinely encodes a nonce, nothing like this is mandated. Second, ciphertexts are required to consist a ciphertext core and a tag. While AE schemes with this structure are common, some AE schemes don't work this way. The duplex-AE syntax forbids the decryption of a ciphertext segment to depend on prior tags. Third, the duplex-AE syntax does not explicitly model state. The absence is problematic if one wants to define online computability. Relatedly, for an adversary to acquire the ciphertext of an m-segment $\boldsymbol{M}$, it will have to query $\Omega(m^2)$ segments. As we argued with respect to our coarse-grained OAE2 definition, this blowup doesn't seem to capture a realistic adversary's actual efficiency, which may be better. Fourth, segmented messages are not explicitly terminated (some sort of "end-of-message") in duplex-AE. Thus no distinction is drawn between (M_1, M_2) as a completed plaintext and (M_1, M_2) as the first two segments of a longer (additional component) segmented plaintext. Forging one is forging the other. While one could use the final AD segment to mark the final plaintext segment, that would leave undefined just what one is attempting to accomplish by such annotation.

Yet if one can overlook the differences above, the gap between duplex-AE and OAE2/nOAE is not so large, at least for large τ. For all of these notions, a sequence of plaintext and AD values are transformed into a corresponding sequence of ciphertext values, each ensuring authenticity not only of the latest

segment but, also, of the entire sequence that has come before. Intuitively, duplex-AE slots between OAE2 and nOAE—and dOAE *does* slot between those two notions. The BDPV nonce-requirement is less demanding than the OAE2 requirement insofar as an encryption query $(N, \boldsymbol{A}, \boldsymbol{M})$ in the second setting can be followed by anything, including a query that changes only the last segment of $\boldsymbol{M}$. One must achieve random-oracle-like behavior even then. On the other hand, the nonce-requirement is more demanding than the nOAE requirement insofar as the adversary, for duplex-AE, is permitted all sorts of queries where one is *not* just growing at-most-one-chain for each nonce/$\boldsymbol{A}[1]$ value.

We regard BDPV's nonce-requirement as a natural restriction if one aims to have accretive processing of segments and bits within segments. As an example, for duplex-AE security, the first bit of $\boldsymbol{C}[10]$ needs to depend on all bits of $\boldsymbol{A}[1], \ldots, \boldsymbol{A}[10]$ and $\boldsymbol{M}[1], \ldots, \boldsymbol{M}[9]$, but it need *not* depend on later bits of $\boldsymbol{M}[10]$. In contrast, for OAE2 security, the first bit of $\boldsymbol{C}[10]$ needs to depend on all bits of N, $\boldsymbol{A}[1], \ldots, \boldsymbol{A}[10]$, and $\boldsymbol{M}[1], \ldots, \boldsymbol{M}[10]$. In the other direction, for nOAE security, $\boldsymbol{C}[10]$ might depend on as little as N, $\boldsymbol{A}[10]$, and $\boldsymbol{M}[10]$, since a nonce-respecting adversary will be unable to "see" if, for example, $\boldsymbol{C}[10]$ really does depend on $\boldsymbol{M}[1]$. While the reference object has that behavior, an adversary can't verify it if nonces don't repeat.

The definition of duplex-AE, along with an email from the Keyak team calling our attention to this work, motivated us to support vector-valued AD and to formalize OAE2c and dOAE, and to provide the related results.

FJMV. Back in 2003, Fouque, Joux, Martinet, and Valette (FJMV) put forward two notions for OAE, one for privacy and another for authenticity [30]. In their setting, encryption is online and probabilistic, but decryption is viewed as an atomic operation. In both notions, the adversary can mount blockwise-adaptive queries on the encryption oracle. But FJMV only give a sketchy explanation of what this actually means. Here we give a rigorous description of FJMV's privacy notion, according to our interpretation. The authenticity notion can be defined analogously.

If $\mathscr{M}$ is a probabilistic algorithm, we write $\mathscr{M}(x_1, \cdots; r)$ to denote the running of $\mathscr{M}$ with arguments $x_1, \cdots$ under coins r. Let n be the blocksize. The encryption scheme is pair of probabilistic algorithms $(\mathcal{E}.\mathrm{enc}, \mathcal{E}.\mathrm{tag})$, with $\mathcal{E}.\mathrm{enc}\colon \mathcal{K} \times (\{0,1\}^n)^* \to \{0,1\}^n$ and $\mathcal{E}.\mathrm{tag}\colon \mathcal{K} \times (\{0,1\}^n)^* \to \{0,1\}^\tau$. To encrypt a message $M_1 \cdots M_m$ under coins r, with $|M_i| = n$, we'll compute $C_i \leftarrow \mathcal{E}.\mathrm{enc}(K, M_1 \cdots M_i; r)$ for every $i \le m$, and $T \leftarrow \mathcal{E}.\mathrm{tag}(K, M_1 \cdots M_m; r)$, and then output $C_1 \cdots C_m \,\|\, T$.

To define privacy, let the adversary play the following game. It begins by sampling the coins r and key K at random, initializing $M \leftarrow \varepsilon$, and choosing a challenge bit $b \xleftarrow{\$} \{0,1\}$. The adversary will make several oracle queries, each of the form (M_0, M_1), where $M_0, M_1 \in \{0,1\}^n \cup \{\bot\}$. If $M_0, M_1 \in \{0,1\}^n$ then the oracle computes $M \leftarrow M \,\|\, M_b$, and returns $\mathcal{E}.\mathrm{enc}(K, M; r)$. Otherwise, it computes $T \leftarrow \mathcal{E}.\mathrm{tag}(K, M; r)$, resets $M \leftarrow \varepsilon$, re-samples r at random, and returns T.

OAE1 resembles a recasting of FJMV's notions with nonce-based AE syntax, where $\mathcal{E}.\mathrm{enc}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times (\{0,1\}^n)^* \to \{0,1\}^n$ and $\mathcal{E}.\mathrm{tag}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times (\{0,1\}^n)^* \to \{0,1\}^\tau$ are deterministic algorithms. Instead of providing a message $M_1 \cdots M_m$ incrementally, since the algorithms $\mathcal{E}.\mathrm{enc}$ and $\mathcal{E}.\mathrm{tag}$ are deterministic, one can have the adversary query (N, A, M_1) to the encryption oracle to get $C_1 \,\|\, T_1$, and then query $(N, A, M_1 M_2)$ to get $C_1 C_2 \,\|\, T_2$, and so on.

BT. Also in 2003, Boldyreva and Taesombut (BT) [23] considered a setting in which the adversary can mount blockwise-adaptive queries on both encryption and decryption oracles. Messages are again regarded as sequences of blocks, but the adversary can provide these blocks incrementally. Their focus is on probabilistic, chosen-ciphertext-attack (CCA) security, but the full version of their paper [23, Section 6] also speaks of AE. Despite major difference (including probabilistic vs. nonce-encryption, and whether messages are regarded as sequence of blocks), BT's motivation seems closer in spirit to OAE2 than OAE1 is, as they insist, from the beginning, that decryption be incremental. The scheme BT suggest to encrypt a plaintext M is to apply an online cipher to $R \,\|\, M \,\|\, R$ where R is a random

block and M is a sequence of blocks. The BT paper served as motivation for the authors of the TSS paper described below.

TSS. The notions of FJMV and BT, like OAE1 and its many variants, are ill-defined for messages whose lengths are not multiple of the blocksize. Moreover, authenticity is only verified at the very end, making those notions unsuitable for streaming applications such as the Netflix example. The blocksize is a fixed, small value, not a partitioning the user may select. The 2009 paper Tsang, Solomakhin, and Smith (TSS) [62] is an exception to all this. Its syntax is similar to OAE2, permitting variable-length segmentation. But it has no associated data, and two segmented-strings $(M_1, \cdots, M_m)$ and $(M'_1, \ldots, M'_k)$ are considered the same if $M_1 \cdots M_m = M'_1 \cdots M'_k$—that is, one doesn't authenticate the segmentation, but, instead, the underlying message. TSS demand that nonces be unique, which our work effectively justifies as a sensible choice, as no OAE scheme achieves desirable security characteristics when nonces are reused. The TSS model does not allow interleaving of queries with different nonces; it handles one encryption, and one decryption, until they are completed. Authenticity for TSS is only verified at the very beginning and at the very end; the intermediate segments are required to have zero ciphertext expansion. TSS impose this restriction to avoid trivial attacks, but those attacks might be possible in real applications. In the Netflix example, an adversary could have users watch the first few scenes from the correct movie, and then switch to whatever it wants. Authenticity would only fail at the end of the entire film. TSS give a construction that achieves their notion, by composing a stream cipher and a MAC, in an Encrypt-then-MAC fashion.

We consider it unfortunate that the TSS manuscript was so little noticed, as it seems to us more definitionally prescient than most alternative lines. Perhaps one reason the paper wasn't more noticed was that it never made it beyond being a Dartmouth technical report. Sadly, the lead author died an early death, and the paper never appeared in any conference or journal.

ABLMMY. Andreeva, Bogdanov, Luykx, Mennink, Mouha, and Yasuda (ABLMMY) [7] study OAE definitions and schemes that are meant to withstand the release of unverified plaintext (RUP). Their motivations overlap with our own—a desire to support decrypting devices with insufficient memory to store the entire plaintext, or prefixes of the decrypted plaintext being needed *now*, to processed real-time needs. The authors define a variety of new security notions, INT-RUP, INT-RUP1, PA1, PA2, and DI, as well as IND-CPA, IND-CCA, INT-CTXT notions. They investigate implications and separations among different combinations and under different assumptions on the IV. The PA notions demand the existence of a *plaintext extractor* as a way to evidence the valuelessness of the decryption oracle. Among other results, ABLMMY show that APE [6] meets their PA1 definition. This scheme does not, and in some sense cannot [7, Prop. 1], support online decryption.

It would take us far afield of our focus to undertake a careful analysis of the ABLMMY definitions. Despite intersecting motivations, our definitional endpoint and theirs are radically different. Unlike ABLMMY, our own work captures user-selectable segmentation, levies no requirements for knowledge extractors, shuns definitional proliferation, and keeps the focus on encryption *and* decryption being online. When OAE2 is used in its intended manner, with reasonable segment-expansion τ, the notion is simultaneously stronger than RUP notions in the sense of ensuring that all decrypted segments *are* authentic; weaker than RUP notions in the sense that plaintext-extractors are in no way mandated; and incomparable in the sense that the extensive syntactic mismatch renders infeasible any direct implications or separations.

BDPS. Boldyreva, Degabriele, Paterson, and Stam (BDPS) [22] consider a setting in which a sender wants to (probabilistically) encrypt a vector of messages $(M_1, \ldots, M_m)$ and send the ciphertexts in a stream $C_1 \| \cdots \| C_m$. An adversary fragments this stream in an arbitrary way, and the recipient receives the fragments $(F_1, \ldots, F_s)$ that satisfies $F_1 \| \cdots \| F_s = C_1 \| \cdots \| C_m$. The adversary may also send fragments of its choice to a decryption oracle. The recipient, without knowing the length $|C_1|, \ldots, |C_m|$,

has to buffer and decide when a ciphertext segment is complete. In addition to the privacy of messages, BDPS aim to hide the boundaries of the ciphertexts in the stream to frustrate traffic analysis. In their syntax, both the encryption and decryption algorithms are stateful: each maintains a short state in processing the segments.

The main overlap with our own works is in attending to security for sequences of messages (what we call segments) rather than atomic messages. But we do not aim to hide the segment lengths or boundaries; in fact, these things need to be recoverable by the decrypting party.

AFLLW. Soon after we made public an earlier version of the present paper, Abed, Forler, List, Lucks, and Wenzel (AFLLW) responded by way of a manuscript [3] (2015.03.10). While responding to a response to a response to a definition begins to get a bit *too* dialectical, we think we had better make public some of the feedback we provided to AFLLW (2015.03.17).

The AFLLW manuscript describes new definitions, OAE1$^+$ and OAE2$^+$. But the message space remains $\mathsf{B}_n^* = (\{0,1\}^n)^*$. One of our central complaints about OAE1 is that it is unreasonable to define security only on B_n^* when one actually wants to operate on a larger space.

The syntax AFLLW ascribe to OAE2 is not ours: they split ciphertexts into a ciphertext core and tag, which we do not do. There are several reasons for our choice (e.g., there's no tag if one uses AEZ to build an OAE2 scheme). Regardless, having introduced the syntax they do, AFLLW fail to use it: Section 5 aims to build an OAE2 scheme but doesn't construct the needed tuple of algorithms and doesn't describe, even informally, what to do with arbitrary-length segments. One goal of OAE2 is to be capable of dealing with arbitrary segmentation (whether a user employs this capability or not); if one hasn't done so, it's not yet an OAE2 scheme.

Syntactic problems make much of the manuscript incoherent, at least to us. For example, AFLLW write that "the combination of decryption robustness and OAE1 security (called OAE1$^+$) implies OAE2 security." But an OAE1 scheme and an OAE2 scheme are different *kinds* of objects. Or consider the OAE1$^+$ definition, where the ideal $\mathcal{E}_K(H,\cdot)$ and $\mathcal{D}_K(H,\cdot)$ are supposed to be online permutations π and its inverse π^{-1}. But the former is not a permutation when $\tau > 0$. This syntactic mismatch is not easily fixed by, for example, using an injective function ρ instead of a permutation π, since AFLLW want the reference object $\mathcal{D}_K(H,\cdot)$ to return non-$\perp$ results on *all* inputs, to model privacy under release of unverified plaintext. Similar problems recur with OAE2$^+$ definition: with Π an OAE2 scheme, as formalized in this paper, it can't be OPERM-CCA or INT-RUP.

Somehow trying to ignore the errant syntax, the sort of thing AFLLW attempt to turn an OAE1-secure scheme into an OAE2-secure one can't work. Consider the case where messages have a single segment. In this setting OAE2 essentially coincides with a PRI (or an MRAE scheme, assuming the segment-expansion is large). (The "essentially" belies the fact that a formal statement would have to deal with the syntactic mismatch.) But the PRI/MRAE goal is unachievable by any online scheme.

It is our view that what AFLLW have in mind for OAE2$^+$ doesn't model anything very meaningful. Having re-conceptualized OAE2 encryption to involve the transmission of a sequence of tagged internal segments and then a final tagged segment, the authors want to provide the adversary an oracle that produces the speculative plaintext for a segmented message regardless of its tags. This is to be done to model the release of unverified plaintext. But why is it worthwhile to model the release of unverified plaintexts in the first place? It is because a ciphertext might be so long, or have timeliness requirements so stringent, that that the ciphertext needs to be acted on before the decryption apparatus gets it all. But the OAE2 definition was *designed* to model this setting. When using an OAE2-secure scheme where segments can be regarded as having authentication tags, if a tag is invalid, decryption should stop. So the OAE2$^+$ definition, once corrected, would capture something like: "Yes, there is an authentication tag for each segment, but suppose that the decrypting party fails to perform the required checks and releases all speculative plaintext segments anyway. Would such misbehavior then invalidate subsequent *authenticity*?" This is not a well-motivated side-channel.

One thing we agree with in the AFLLW manuscript is that the term *misuse-resistant* is also questionable for MRAE [56]. It is true that having an oracle for $(N, A, \text{Mask}) \mapsto \mathcal{E}_K(N, A, M\&\text{Mask})$ lets one recover a secret M, for any deterministic $\mathcal{E}$, and one could reasonably maintain that this precludes deservedly calling $\mathcal{E}$ *misuse-resistant*. One could also say, more simply, that leaking repetitions in (N, A, M) precludes true misuse-resistance.

In emails subsequent to the AFLLW manuscript, Lucks suggests that the OAE2 definition goes wrong by allowing users to choose different lengths for different segments: revelation of these differing lengths can reveal crucial information, which users might not understand. While the second claim is true, the first doesn't follow from it. Conventional definitions of symmetric encryption always anticipate that the length of a ciphertext reveals the length of its plaintext; in the OAE setting, the analogous expectation is that the length of a ciphertext segment reveals the length of the corresponding plaintext segment. While definitionally permitted disclosure of lengths can be dangerously revelatory, that is always so: we don't see anything new in this regard with respect to OAE.

Finally, we would make a discursive warning about AFLLW's use of the term *robust* or *robustness* has no obvious relation to the technical meaning for robust-AE defined in the RAE/AEZ paper [35].

C MRAE Resists CPSS

We evidence that MRAE-secure schemes, unlike OAE1-secure schemes, resist CPSS attack. The MRAE notion is much stronger still, but a result like what we give is a starting point.

We first recall the MRAE notion. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a conventional AE scheme, meaning that (i) The key space $\mathcal{K}$ is a nonempty set with an associated distribution, and (ii) $\mathcal{E}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^* \to \{0,1\}^*$ is the encryption scheme, and (iii) $\mathcal{D}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ is the decryption scheme. Both $\mathcal{E}$ and $\mathcal{D}$ are deterministic, and decryption reverses encryption, meaning that for every $N \in \mathcal{N}$, $A \in \mathcal{A}$, $M \in \{0,1\}^*$, and $K \in \mathcal{K}$, we have $\mathcal{D}_K^{N,A}(\mathcal{E}_K^{N,A}(M)) = M$. We insist that there be a constant τ associated to Π, its *ciphertext expansion*, where $|\mathcal{E}_K^{N,A}(M)| = |M| + \tau$ for every $N \in \mathcal{N}$, $A \in \mathcal{A}$, $M \in \{0,1\}^*$, $K \in \mathcal{K}$. The advantage of an adversary $\mathscr{A}$ attacking the MRAE security of Π is defined as

$$\mathbf{Adv}^{\text{mrae}}_{\Pi}(\mathscr{A}) = \Pr[K \xleftarrow{\$} \mathcal{K}\colon \mathscr{A}^{\mathcal{E}_K(\cdot,\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathscr{A}^{\$(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1]$$

where $\$(\cdot,\cdot,\cdot)$ is an oracle that returns $C \xleftarrow{\$} \{0,1\}^{|M|+\tau}$ for any input (N, A, M) and $\perp(\cdot,\cdot,\cdot)$ is an oracle that returns $\perp$ for any input. The adversary is prohibited from querying (N, A, M) to the first oracle to get C and then querying (N, A, C) to the second oracle, or repeating a prior query to the first oracle.

We now explain why a scheme secure in the MRAE sense will always resist CPSS attack. Suppose that the secret suffix S is generated by an efficient sampler $\mathcal{S}$. Let $\mathbf{Adv}^{\text{guess}}_{\mathcal{S}}$ denote the min-entropy of the distribution generated by $\mathcal{S}$. Let $\mathscr{A}$ be a CPSS adversary attacking an AE scheme Π. Consider the following MRAE adversary $\mathscr{B}$ attacking Π. It generates the suffix $S \xleftarrow{\$} \mathcal{S}$ and runs $\mathscr{A}$. For each prefix P that $\mathscr{A}$ produces, if $\mathscr{A}$ repeats a prior prefix then $\mathscr{B}$ gives the consistent answer. Otherwise, it queries $P \,\|\, S$ to its encryption oracle, and returns the answer to $\mathscr{A}$. If $\mathscr{A}$ can reproduce S then $\mathscr{B}$ outputs 1; otherwise it outputs 0. If $\mathscr{B}$'s oracle implements $\$(\cdot)$ then $\mathscr{A}$ can guess S with probability at most $\mathbf{Adv}^{\text{guess}}_{\mathcal{S}}$, since it only receives random answers independent of S. Hence the chance that $\mathscr{A}$ can guess S in the CPSS attack against Π is at most $\mathbf{Adv}^{\text{mrae}}_{\Pi}(\mathscr{B}) + \mathbf{Adv}^{\text{guess}}_{\mathcal{S}}$.

D Separating OAE1[n] and OAE2[$0, n$]

OAE1 is weaker than OAE2 in the sense that the former does not support arbitrary segmentation or demand security over arbitrary strings. This brief section argues a more specific claim: that 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. (To address the syntactic mismatch, we'll assume that every $\boldsymbol{A}$ satisfies $\boldsymbol{A}[i] = \varepsilon$ for every $1 < i \le |\boldsymbol{A}|$, so that OAE2 can be viewed as operating on a single AD string, instead of an AD vector.) This is true even if we fix a reasonably large value of n, say $n = 128$. We ignore mundane matters of mismatched syntax that would have to be dealt with in a more formal treatment (i.e., that OAE1 and OAE2 schemes are very different *kinds* of objects).

So consider an OAE1-secure scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ of blocksize n. Assume that all keys output by $\mathbf{K}$ also have n bits, which is the most common case for $n = 128$. Suppose that for scheme Π one can recover the m-th block of the (putative) plaintext from K, N, A, and the first m blocks of ciphertext, which again holds for typical schemes, like COPA [9] and McOE [28]. Now consider the following scheme $\tilde{\Pi} = (\mathbf{K}, \tilde{\mathbf{E}}, \tilde{\mathbf{D}})$. For any $X \in (\{0,1\}^*)^n$, let $\tilde{\mathbf{E}}_K^{N,A}(K \parallel X) = \mathbf{E}_K^{N,A}(M_0 \parallel X)$, where M_0 is the first block of the putative plaintext obtained from decrypting 0^{2n} under key K with nonce N and AD A; and let $\tilde{\mathbf{E}}_K^{N,A}(M_0 \parallel X) = \mathbf{E}_K^{N,A}(K \parallel X)$. Let $\tilde{\mathbf{E}}_K^{N,A}$ coincide with $\mathbf{E}_K^{N,A}$ on all other points. Then the scheme $\tilde{\Pi}$ ought still to be OAE1-secure. For given an adversary $\mathscr{A}$ attacking $\tilde{\Pi}$, one can transform it to an equally efficient adversary $\mathscr{B}$ attacking Π that outputs 1 if the first block of some ciphertext is 0^n, and the probability that $\mathscr{A}$ can query some $M_0 \parallel X$ to the encryption oracle is at most $\mathbf{Adv}^{\mathrm{oae1}}_{\Pi}(\mathscr{B})$. But $\tilde{\Pi}$ is not OAE2-secure, as an adversary can query 0^{2n} to the decryption oracle to learn K. In brief, we can adjust an OAE1-secure scheme to fail miserably in the presence of a decryption capability, the adjustment irrelevant for OAE1 security.

We emphasize that the above counterexample does not imply that common OAE1-secure schemes with expansion τ fail to be OAE2-secure with expansion $(0, \tau)$ once reconceptualized and restricted. Such a determination would have to be made on a case-by-case basis, asking if supplementing the adversary's capabilities with an online decryption oracle would violate indistinguishability. We haven't carried out such investigations because even when an OAE1 scheme *is* OAE2-secure once restricted and reconceptualized, we are not suggesting this would make it a desirable way to address online-AE: in particular, expansion parameters of $(0, \tau)$ are likely to be a poor choice in most settings, since they provide no authenticity assurance until a ciphertext's end. This is why our focus has been on τ-expanding schemes, where all segments are afforded the same authenticity guarantees. It also seems undesirable to insist on segmenting messages along n-bit boundaries for some small, fixed n, and to fail to define security for messages that are not blocksize multiples.

E Deferred Proofs

E.1 Proof of Theorem 1

The reduction R creates from $\mathscr{A}$ adversary $\mathscr{B}$ as follows. The latter runs the former and simulates game $\mathbf{Real2B}_{\mathbf{CHAIN}}[\Pi, \langle\cdot\rangle, n]$, but each call to $\mathbf{E}_K(\cdot)$ or $\mathbf{D}_K(\cdot)$ is replaced by the corresponding query to the first or second oracle of $\mathscr{B}$, respectively. Adversary $\mathscr{B}$ then outputs the same guess as $\mathscr{A}$.

Consider games G_1–G_4 in Fig. 12. Game G_1 corresponds to game $\mathbf{Real2B}_{\mathbf{CHAIN}}[\Pi, \langle\cdot\rangle, n]$, and game G_4 corresponds to game $\mathbf{Ideal2B}_{\mathbf{CHAIN}}[\Pi, \langle\cdot\rangle, n]$. We now explain the game chain. Initially, we'll sample $\pi_{V,\langle A', d\rangle} \twoheadleftarrow \mathrm{Inj}(\tau)$ for every $V \in \{0,1\}^n$ and every $A' \in \{0,1\}^*$, $d \in \{0,1,2,3,4,5\}$, and $\rho_{N,\boldsymbol{A},\boldsymbol{M},\delta} \twoheadleftarrow \mathrm{Inj}(\tau)$ for every $N \in \{0,1\}^n, \delta \in \{0,1\}$, and $\boldsymbol{A}, \boldsymbol{M} \in \{0,1\}^{**}$ such that $|\boldsymbol{A}| = |\boldsymbol{M}| + 1$. Game G_2 is identical to game G_1, except that instead of using $\mathbf{E}_K^{V,A}$ and $\mathbf{D}_K^{V,A}$, we'll use an injective function $\pi_{V,A}$ and its inverse $\pi^{-1}_{V,A}$ respectively. Then

$$\Pr[\mathscr{A}^{G_1} \Rightarrow 1] - \Pr[\mathscr{A}^{G_2} \Rightarrow 1] = \mathbf{Adv}^{\mathrm{pri}}_{\Pi}(\mathscr{B}) \ .$$

Game G_3 is identical to game G_2 except that, we want the state V' to be never repeated, but still maintain the following consistency: (i) calling Map with the same (V, A, M) always results in the same (C, V'), (ii) calling MapInv with the same (V, A, C) always result in the same (M, V'), and (iii) if

proc Enc.init(N)
$I \leftarrow I + 1$; $\boldsymbol{M}_I \leftarrow \Lambda$
$\boldsymbol{A}_I \leftarrow \Lambda$; $S_I \leftarrow (K, N, 0)$; $N_i \leftarrow N$
return I

proc Enc.next(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$(K, V, d) \leftarrow S_i$; $\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \parallel A$
$(N, \boldsymbol{A}, \boldsymbol{M}, \delta) \leftarrow (N_i, \boldsymbol{A}_i, \boldsymbol{M}_i, 0)$
$(C, V) \leftarrow \mathrm{Map}(V, \langle A, d\rangle, M)$
$S_i \leftarrow (K, V, 1)$; $\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \parallel M$
return C

proc Enc.last(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$(K, V, d) \leftarrow S_i$; $\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \parallel A$
if $d = 0$ **then** $d \leftarrow 3$ **else** $d \leftarrow 2$
$(N, \boldsymbol{A}, \boldsymbol{M}, \delta) \leftarrow (N_i, \boldsymbol{A}_i, \boldsymbol{M}_i, 1)$
$(C, V) \leftarrow \mathrm{Map}(V, \langle A, d\rangle, M)$; $S_i \leftarrow \bot$; **return** C

proc Dec.init(N)
$J \leftarrow J + 1$; $\boldsymbol{M}'_J \leftarrow \Lambda$
$\boldsymbol{A}'_J \leftarrow \Lambda$; $S'_J \leftarrow (K, N, 0)$; $N'_j \leftarrow N$
return J

proc Dec.next(j, A, C)
if $j > J$ **or** $S'_j = \bot$ **then return** $\bot$
$(K, V, d) \leftarrow S'_j$; $\boldsymbol{A}'_j \leftarrow \boldsymbol{A}'_j \parallel A$
$(N, \boldsymbol{A}, \boldsymbol{M}, \delta) \leftarrow (N'_j, \boldsymbol{A}'_j, \boldsymbol{M}'_j, 0)$
$(M, V) \leftarrow \mathrm{MapInv}(V, \langle A, d\rangle, C)$
if $M \neq \bot$ **then** $S'_j \leftarrow (K, V, 1)$ **else** $S'_j \leftarrow \bot$
$\boldsymbol{M}'_j \leftarrow \boldsymbol{M}'_j \parallel M$; **return** M

proc Dec.last(j, A, C)
if $j > J$ **or** $S_j = \bot$ **then return** $\bot$
$(K, V, d) \leftarrow S'_j$; $\boldsymbol{A}_j \leftarrow \boldsymbol{A}_j \parallel A$
$(N, \boldsymbol{A}, \boldsymbol{M}, \delta) \leftarrow (N'_j, \boldsymbol{A}'_j, \boldsymbol{M}'_j, 1)$
if $d = 0$ **then** $d \leftarrow 3$ **else** $d \leftarrow 2$
$(M, V) \leftarrow \mathrm{MapInv}(V, \langle A, d\rangle, C)$; $S'_j \leftarrow \bot$; **return** M

proc Map(V, A, M) — Game G_1
$C \leftarrow \mathbf{E}_K^{V,A}(M)$; $\langle A', d\rangle \leftarrow A$
if $d \notin \{0, 1\}$ **then** $V' \leftarrow \bot$
elsif $H[V, A, M] \neq \bot$ **then** $V' \leftarrow H[V, A, M]$
else $V' \leftarrow \mathrm{Ev}(V, A, M, C)$; $H[V, A, M] \leftarrow V'$
return (C, V')

proc MapInv(V, A, C)
$M \leftarrow \mathbf{D}_K^{V,A}(C)$; $\langle A', d\rangle \leftarrow A$
if $M = \bot$ **then return** $(\bot, \bot)$
if $d \notin \{0, 1\}$ **then** $V' \leftarrow \bot$
elsif $H[V, A, M] \neq \bot$ **then** $V' \leftarrow H[V, A, M]$
else $V' \leftarrow \mathrm{Ev}(V, A, M, C)$; $H[V, A, M] \leftarrow V'$
return (M, V')

proc Ev(V, A, M, C)
if $|M| \geq n$ **then** $V' \leftarrow C[1, n] \oplus M[1, n]$
elsif $A = \langle A', d\rangle$ **then**
  $L \leftarrow \mathbf{E}_K(V, \langle A', d + 4\rangle, M \parallel 0^n)$; $V' \leftarrow L[1, n]$
$\mathrm{Dom} \leftarrow \mathrm{Dom} \cup \{V'\}$; **return** V'

proc Map(V, A, M) — Games G_2, $\boxed{G_3}$
$C \leftarrow \pi_{V,A}(M)$; $\langle A', d\rangle \leftarrow A$
if $d \notin \{0, 1\}$ **then** $V' \leftarrow \bot$
elsif $H[V, A, M] \neq \bot$ **then** $V' \leftarrow H[V, A, M]$
else $V' \leftarrow \mathrm{Ev}(V, A, M, C)$; $H[V, A, M] \leftarrow V'$
return (C, V')

proc MapInv(V, A, C)
$M \leftarrow \pi_{V,A}^{-1}(C)$; $\langle A', d\rangle \leftarrow A$
if $M = \bot$ **then return** $(\bot, \bot)$
if $d \notin \{0, 1\}$ **then** $V' \leftarrow \bot$
elsif $H[V, A, M] \neq \bot$ **then** $V' \leftarrow H[V, A, M]$
else $V' \leftarrow \mathrm{Ev}(V, A, M, C)$; $H[V, A, M] \leftarrow V'$
return (M, V')

proc Ev(V, A, M, C)
if $|M| \geq n$ **then** $V' \leftarrow C[1, n] \oplus M[1, n]$
elsif $A = \langle A', d\rangle$ **then**
  $L \leftarrow \pi_{V,\langle A', d+4\rangle}(M \parallel 0^n)$; $V' \leftarrow L[1, n]$
if $(V' \in \mathrm{Dom})$ **then** $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{V' \twoheadleftarrow \{0,1\}^n \backslash \mathrm{Dom}}$
$\mathrm{Dom} \leftarrow \mathrm{Dom} \cup \{V'\}$; **return** V'

proc Map(V, A, M)
$C \leftarrow \rho_{N,\boldsymbol{A},\boldsymbol{M},\delta}(M)$
return $(C, \bot)$

proc MapInv(V, A, C) — Game G_4
$M \leftarrow \rho_{N,\boldsymbol{A},\boldsymbol{M},\delta}^{-1}(C)$
return $(M, \bot)$

Fig. 12: Games G_1–G_4 used in the proof of Theorem 1. Game G_3 contains the corresponding boxed statement but game G_2 doesn't. The games share the common procedures Enc.init, Enc.next, Enc.last, Dec.init, Dec.next, and Dec.last, and each game maintains local procedures Map, MapInv, and Ev that are inaccessible to the adversary. In each game, there is an implicit procedure **initialize**() that initializes $\mathrm{Dom} \leftarrow \emptyset$ and $I, J \leftarrow 0$, and samples $K \twoheadleftarrow \mathbf{K}$, $\pi_{V,\langle A', d\rangle} \twoheadleftarrow \mathrm{Inj}(\tau)$ for every $V \in \{0,1\}^n$ and every $A' \in \{0,1\}^*$, $d \in \{0, 1, 2, 3, 4, 5\}$, and $\rho_{N,\boldsymbol{A},\boldsymbol{M},\delta} \twoheadleftarrow \mathrm{Inj}(\tau)$ for every $N \in \{0,1\}^n, \delta \in \{0, 1\}$, and $\boldsymbol{A}, \boldsymbol{M} \in \{0,1\}^{**}$ such that $|\boldsymbol{A}| = |\boldsymbol{M}| + 1$.

$(C, V') \leftarrow \mathrm{Map}(V, A, M)$ then $\mathrm{MapInv}(V, A, C)$ returns (M, V'), and (iv) if $(M, V') \leftarrow \mathrm{MapInv}(V, A, C)$ and $M \neq \bot$ then $\mathrm{Map}(V, A, M)$ returns (C, V'). The two games are identical-until-bad, and thus

$$\Pr[\mathscr{A}^{G_2} \Rightarrow 1] - \Pr[\mathscr{A}^{G_3} \Rightarrow 1] \leq \Pr[G_2 \text{ sets } \mathsf{bad}] \ .$$

We now bound the chance that game G_2 sets bad. Terminate the game immediately when bad gets set; it doesn't change the probability that G_2 sets bad. Observe that

(1) In $\mathrm{Map}(V, A, M)$ and $\mathrm{MapInv}(V, A, C)$, we'll have A of the form $\langle A', d\rangle$, with $d \in \{0, 1, 2, 3\}$.

(2) In $\mathrm{Map}(V, A, M)$, we call $C \leftarrow \pi_{V,A}(M)$ and invoke Ev to compute $L \leftarrow \pi_{V,\langle A', d+4\rangle}(M \parallel 0^n)$. The second call is made only if $|M| < n$, $d \in \{0, 1\}$, and there is no prior call $\mathrm{Map}(V, A, M)$ or $\mathrm{MapInv}(V, A, C)$.
(3) In $\mathrm{MapInv}(V, A, C)$, we call $M \leftarrow \pi^{-1}_{V,A}(C)$ and invoke Ev to compute $L \leftarrow \pi_{V,\langle A', d+4\rangle}(M \parallel 0^n)$. The second call is made only if $|M| < n$, $d \in \{0, 1\}$, and there is no prior call $\mathrm{Map}(V, A, M)$ or $\mathrm{MapInv}(V, A, C)$.

Wlog, assume that $1 < q \le 2^{n-1}$; otherwise the bound is trivial. The flag bad is triggered only if the state V' repeats one of its prior values. Consider the following cases.

Case 1: $V' \leftarrow L[1, n]$, where $L \leftarrow \pi_{V,\langle A', d+4\rangle}(M \parallel 0^n)$. Due to the domain separation in the use of associated data, there is no prior call to $\pi^{-1}_{V,\langle A', d+4\rangle}(L)$. Also, due to the one-to-one correspondence between $(V, A, M) = (V, \langle A', d\rangle, M)$ and $(V, \langle A', d+4\rangle, M\|0^n)$, there is no prior call to $\pi_{V,\langle A', d+4\rangle}(M\|0^n)$. Then L is chosen uniformly random from a subset of $\{0, 1\}^{n+\tau}$ that has at least $2^{n+\tau} - q$ elements. This case triggers bad with probability at most

$$\sum_{i=0}^{q-1} \frac{i \cdot 2^\tau}{2^{n+\tau} - q} \le \frac{0.5q^2}{2^n - q} \le \frac{q^2}{2^n};$$

the last inequality is due to the assumption that $q \le 2^{n-1}$.

Case 2: $V' \leftarrow C[1, n] \oplus M[1, n]$. There must be no prior Map call of the same (V, A, M) or MapInv call of the same (V, A, C), otherwise we'll return the consistent state, and bad won't be triggered. First suppose that Map triggers bad. Let $s = |C|$. From (1), (2), and (3), there is no prior call to $\pi_{V,A}(M)$ or $\pi^{-1}_{V,A}(C)$. Suppose that there are i prior queries to Map or MapInv. Since we sample C uniformly from a subset of $\{0, 1\}^s$ that has at least $2^s - q$ elements, this case happens with probability at most $i2^{s-n}/(2^s - q) \le i/(2^n - q) \le 2i/2^n$. Next, consider the case that MapInv triggers bad. Suppose that there are i prior queries to Map or MapInv. Let $s = |M|$. By using the same analysis as above, the bound is again at most $2i/2^n$. Summing up for $i = 0, 1, \ldots, q - 1$ gives us the bound $q^2/2^n$.

Hence, totally, the chance that G_2 sets bad is at most $2q^2/2^n$, and thus

$$\Pr[\mathscr{A}^{G_2} \Rightarrow 1] - \Pr[\mathscr{A}^{G_3} \Rightarrow 1] \le \Pr[\mathscr{A}^{G_2} \text{ sets bad}] \le \frac{2q^2}{2^n} \ .$$

Note that in each game, right before we call $\mathrm{Map}(V, A, M)$ or $\mathrm{MapInv}(V, A, C)$, we maintain a tuple $(N, \boldsymbol{A}, \boldsymbol{M}, \delta)$. We claim that in game G_3, different tuples $(N, \boldsymbol{A}, \boldsymbol{M}, \delta)$ will correspond to different pairs (V, A); we will justify this claim later. In game G_4, instead of using $\pi_{V,A}$ or its inverse, we use $\rho_{N,\boldsymbol{A},\boldsymbol{M},\delta}$ and its inverse, respectively.[7] Since in game G_3, different tuples $(N, \boldsymbol{A}, \boldsymbol{M}, \delta)$ will correspond to different pairs (V, A), this is simply an internal implementation choice for $\pi_{V,A}$, and thus doesn't affect the input/output behavior of the game. Hence

$$\Pr[\mathscr{A}^{G_4} \Rightarrow 1] = \Pr[\mathscr{A}^{G_3} \Rightarrow 1] \ .$$

Summing up,

$$\mathbf{Adv}^{\mathrm{oae2b}}_{\mathbf{CHAIN}[\Pi,\langle\cdot\rangle,n]}(\mathscr{A}) = \Pr[\mathscr{A}^{G_1} \Rightarrow 1] - \Pr[\mathscr{A}^{G_4} \Rightarrow 1] \le \mathbf{Adv}^{\mathrm{pri}}_{\Pi}(\mathscr{B}) + \frac{2q^2}{2^n} \ .$$

[7] The code in Fig. 12 actually shows the simplified code of G_4: since the outputs that the adversary receives are independent of the state V', we can simplify Map and MapInv by (i) eliminating the useless code that computes V' in Map and MapInv, and (ii) having Map and MapInv instead return $\perp$ for V'.

proc Enc.init(N)
$I \leftarrow I + 1$; $N_I \leftarrow N$; $S_I \leftarrow \varepsilon$; $\boldsymbol{M}_I, \boldsymbol{A}_I, \boldsymbol{C}_I \leftarrow \Lambda$
return Enc.$\overline{\text{init}}(N)$

proc Enc.next(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$C \leftarrow$ Enc.$\overline{\text{next}}(i, A, M)$
$\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \,\|\, M$; $\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \,\|\, A$; $\boldsymbol{C}_i \leftarrow \boldsymbol{C}_i \,\|\, C$
$H[N_i, \boldsymbol{A}_i, \boldsymbol{C}_i, 0] \leftarrow \boldsymbol{M}_i$; **return** C

proc Enc.last(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$C \leftarrow$ Enc.$\overline{\text{last}}(i, A, M)$; $S_i \leftarrow \bot$
$\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \,\|\, M$; $\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \,\|\, A$; $\boldsymbol{C}_i \leftarrow \boldsymbol{C}_i \,\|\, C$
$H[N_i, \boldsymbol{A}_i, \boldsymbol{C}_i, 1] \leftarrow \boldsymbol{M}_i$; **return** C

proc Dec.init(N)
$J \leftarrow J + 1$; $N'_J \leftarrow N$; $S'_J \leftarrow \varepsilon$; $\boldsymbol{A}'_J, \boldsymbol{C}'_J \leftarrow \Lambda$
return J

proc Dec.next(j, A, C)
if $j > J$ **or** $S'_j = \bot$ **then return** $\bot$
$\boldsymbol{A}'_j \leftarrow \boldsymbol{A}'_j \,\|\, A$; $\boldsymbol{C}'_j \leftarrow \boldsymbol{C}$
if $H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 0] \neq \bot$ **then**
  $\boldsymbol{M} \leftarrow H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 0]$; **return** $\boldsymbol{M}[|\boldsymbol{M}|]$
$\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 0)\}$; $S'_j \leftarrow \bot$; **return** $\bot$

proc Dec.last(j, A, C)
if $j > J$ **or** $S'_j = \bot$ **then return** $\bot$
$\boldsymbol{A}'_j \leftarrow \boldsymbol{A}'_j \,\|\, A$; $\boldsymbol{C}'_j \leftarrow \boldsymbol{C}$; $S'_j \leftarrow \bot$
if $H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 0] \neq \bot$ **then**
  $\boldsymbol{M} \leftarrow H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 1]$; **return** $\boldsymbol{M}[|\boldsymbol{M}|]$
$\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 1)\}$; **return** $\bot$

Fig. 13: The simulated game that $\mathscr{B}_i$ (with $i \in \{1, 2\}$) in the proof of Proposition 1 use to run $\mathscr{A}$. The procedures Enc.$\overline{\text{init}}$, Enc.$\overline{\text{next}}$, Enc.$\overline{\text{last}}$ are the encryption oracles of $\mathscr{B}_i$. There is an implicit procedure **initialize**() that initializes $I, J \leftarrow 0$ and $Z \leftarrow \emptyset$.

What remains is to justify the claim above. Suppose that in game G_3, there is at least one pair of two different tuples $(N_1, \boldsymbol{A}_1, \boldsymbol{M}_1, \delta_1)$ and $(N_2, \boldsymbol{A}_2, \boldsymbol{M}_2, \delta_2)$ that correspond to the same pair (V, A). Among such pairs of tuples, consider the one that minimizes $|\boldsymbol{M}_1|$. Consider the following cases.

Case 1: $|\boldsymbol{M}_1|, |\boldsymbol{M}_2| > 0$. Let $\boldsymbol{M}_1^*$ be the prefix of $\boldsymbol{M}_1$ that consists of $|\boldsymbol{M}_1| - 1$ components. Define $\boldsymbol{M}_2^*$ for $\boldsymbol{M}_2$, $\boldsymbol{A}_1^*$ for $\boldsymbol{A}_1$, and $\boldsymbol{A}_2^*$ for $\boldsymbol{A}_2$ analogously. Due to the unique sampling of the state V', the tuples $(N_1, \boldsymbol{A}_1^*, \boldsymbol{M}_1^*, 0)$ and $(N_2, \boldsymbol{A}_2^*, \boldsymbol{M}_2^*, 0)$ must correspond to the same pair (V^*, A^*), contradicting the minimum of $|\boldsymbol{M}_1|$.

Case 2: $|\boldsymbol{M}_1| = 0$ and $|\boldsymbol{M}_2| > 0$. But then this is a contradiction: (1) since $(N_1, \boldsymbol{A}_1, \boldsymbol{M}_1, \delta_1)$ corresponds to (V, A), it means that A is of the form $\langle A', 0\rangle$ or $\langle A', 3\rangle$, but (2) since $(N_2, \boldsymbol{A}_2, \boldsymbol{M}_2, \delta_2)$ corresponds to (V, A), it means that A is of the form $\langle A'', 1\rangle$ or $\langle A'', 2\rangle$.

Case 3: $|\boldsymbol{M}_1| > 0$ and $|\boldsymbol{M}_2| = 0$. This is similar to Case 2.

Case 4: $|\boldsymbol{M}_1| = |\boldsymbol{M}_2| = 0$, meaning that $\boldsymbol{M}_1 = \boldsymbol{M}_2 = \Lambda$. Note that in this case, since $(N_1, \boldsymbol{A}_1, \boldsymbol{M}_1, \delta_1)$ corresponds to (V, A) and $|\boldsymbol{M}_1| = 0$, we must have $V = N_1$. Likewise, since $(N_2, \boldsymbol{A}_2, \boldsymbol{M}_2, \delta_2)$ corresponds to (V, A) and $|\boldsymbol{M}_2| = 0$, we must have $V = N_2$. In other words, $N_1 = N_2$. Let $d_1 = 0$ if $\delta_1 = 0$, and $d_1 = 3$ otherwise. Define d_2 for δ_2 analogously. Since $(N_1, \boldsymbol{A}_1, \boldsymbol{M}_1, \delta_1)$ corresponds to (V, A) and $|\boldsymbol{M}_1| = 0$, we have $A = \langle \boldsymbol{A}_1[1], d_1\rangle$. Likewise, since $(N_2, \boldsymbol{A}_2, \boldsymbol{M}_2, \delta_2)$ corresponds to (V, A) and $|\boldsymbol{M}_2| = 0$, we have $A = \langle \boldsymbol{A}_2[1], d_2\rangle$. In other words, $\boldsymbol{A}_1 = \boldsymbol{A}_2$, and $\delta_1 = \delta_2$. Hence the two tuples $(N_1, \boldsymbol{A}_1, \boldsymbol{M}_1, \delta_1)$ and $(N_2, \boldsymbol{A}_2, \boldsymbol{M}_2, \delta_2)$ are the same, which is a contradiction.

E.2 Proof of Proposition 1

The reduction R_1 creates from $\mathscr{A}$ adversary $\mathscr{B}_1$ as follows. The latter runs the former as indicated in Fig. 13, and outputs the same guess as $\mathscr{A}$. Next, the reduction R_2 creates from $\mathscr{A}$ adversary $\mathscr{B}_2$ as follows. The latter runs the former as indicated in Fig. 13. When $\mathscr{A}$ terminates, $\mathscr{B}_2$ will process the resulting set $\mathcal{Z}$. For $(N, \boldsymbol{A}, \boldsymbol{C}, 0)$ and $(N, \boldsymbol{A}', \boldsymbol{C}', \delta)$ in $\mathcal{Z}$, we'll delete the former vector if $m = |\boldsymbol{A}| < |\boldsymbol{A}'|$, and $\boldsymbol{A}[i] = \boldsymbol{A}'[i]$ and $\boldsymbol{C}[i] = \boldsymbol{C}'[i]$ for every $i \leq m$. Now the set $\mathcal{Z}$ will have only p elements that corresponds to the p decryption chains. Adversary $\mathscr{B}_2$ then outputs a random element of $\mathcal{Z}$.

Consider games G_1–G_3 in Fig. 14. Game G_1 corresponds to game $\mathbf{Real2B}_{\Pi}$. Game G_2 is identical to game G_1, except that Dec.next and Dec.last always return $\bot$. The two games are identical-until-bad,

proc Enc.init(N)
$I \leftarrow I + 1$; $N_I \leftarrow N$; $S_I \leftarrow \mathcal{E}.\text{init}(K, N)$
$\boldsymbol{M}_I, \boldsymbol{A}_I, \boldsymbol{C}_I \leftarrow \Lambda$; **return** I

proc Enc.next(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$C \leftarrow \mathcal{E}.\text{next}(S_i, A, M)$
$\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \parallel M$; $\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \parallel A$; $\boldsymbol{C}_i \leftarrow \boldsymbol{C}_i \parallel C$
$H[N_i, \boldsymbol{A}_i, \boldsymbol{C}_i, 0] \leftarrow \boldsymbol{M}_i$; **return** C

proc Enc.last(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$C \leftarrow \mathcal{E}.\text{last}(S_i, A, M)$; $S_i \leftarrow \bot$
$\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \parallel M$; $\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \parallel A$; $\boldsymbol{C}_i \leftarrow \boldsymbol{C}_i \parallel C$
$H[N_i, \boldsymbol{A}_i, \boldsymbol{C}_i, 1] \leftarrow \boldsymbol{M}_i$; **return** C

proc Dec.init(N) Games G_1, $\boxed{G_2}$
$J \leftarrow J + 1$; $N'_J \leftarrow N$; $S'_J \leftarrow \mathcal{D}.\text{init}(K, N)$
$\boldsymbol{M}'_J, \boldsymbol{A}'_J, \boldsymbol{C}'_J \leftarrow \Lambda$; **return** J

proc Dec.next(j, A, C)
if $j > J$ **or** $S'_j = \bot$ **then return** $\bot$
$\boldsymbol{A}'_j \leftarrow \boldsymbol{A}'_j \parallel A$; $\boldsymbol{C}'_j \leftarrow \boldsymbol{C}'_j \parallel C$
if $H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 0] \neq \bot$ **then**
  $\boldsymbol{M} \leftarrow H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 0]$; **return** $\boldsymbol{M}[|\boldsymbol{M}|]$
$(M, S'_j) \leftarrow \mathcal{D}.\text{next}(S'_j, A, C)$
if $M \neq \bot$ **then** bad $\leftarrow$ true; $\boxed{S'_j, M \leftarrow \bot}$
return M

proc Dec.last(j, A, C)
if $j > J$ **or** $S'_j = \bot$ **then return** $\bot$
$\boldsymbol{A}'_j \leftarrow \boldsymbol{A}'_j \parallel A$; $\boldsymbol{C}'_j \leftarrow \boldsymbol{C}'_j \parallel C$
if $H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 0] \neq \bot$ **then**
  $\boldsymbol{M} \leftarrow H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 1]$; $S'_j \leftarrow \bot$; **return** $\boldsymbol{M}[|\boldsymbol{M}|]$
if $M \neq \bot$ **then** bad $\leftarrow$ true; $\boxed{M \leftarrow \bot}$
$S'_j \leftarrow \bot$; **return** M

proc Enc.init(N)
$I \leftarrow I + 1$; $N_I \leftarrow N$; $S_I \leftarrow \varepsilon$
$\boldsymbol{M}_I, \boldsymbol{A}_I, \boldsymbol{C}_I \leftarrow \Lambda$; **return** I

proc Enc.next(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \parallel A$; $C \leftarrow \rho_{N_i, \boldsymbol{A}_i, \boldsymbol{M}_i, 0}(M)$
$\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \parallel M$; $\boldsymbol{C}_i \leftarrow \boldsymbol{C}_i \parallel C$
$H[N_i, \boldsymbol{A}_i, \boldsymbol{C}_i, 0] \leftarrow \boldsymbol{M}_i$; **return** C

proc Enc.last(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \parallel A$; $C \leftarrow \rho_{N_i, \boldsymbol{A}_i, \boldsymbol{M}_i, 1}(M)$; $S_i \leftarrow \bot$
$\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \parallel M$; $\boldsymbol{C}_i \leftarrow \boldsymbol{C}_i \parallel C$
$H[N_i, \boldsymbol{A}_i, \boldsymbol{C}_i, 1] \leftarrow \boldsymbol{M}_i$; **return** C

proc Dec.init(N) Game G_3
$J \leftarrow J + 1$; $N'_J \leftarrow N$; $S'_J \leftarrow \varepsilon$
$\boldsymbol{M}'_J, \boldsymbol{A}'_J, \boldsymbol{C}'_J \leftarrow \Lambda$; **return** J

proc Dec.next(j, A, C)
if $j > J$ **or** $S'_j = \bot$ **then return** $\bot$
$\boldsymbol{A}'_j \leftarrow \boldsymbol{A}'_j \parallel A$; $\boldsymbol{C}'_j \leftarrow \boldsymbol{C}'_j \parallel C$
if $H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 0] \neq \bot$ **then**
  $\boldsymbol{M}'_j \leftarrow \boldsymbol{M} \leftarrow H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 0]$; **return** $\boldsymbol{M}[|\boldsymbol{M}|]$
$M \leftarrow \rho^{-1}_{N'_j, \boldsymbol{A}'_j, \boldsymbol{M}'_j, 0}(C)$
if $M = \bot$ **then** $S'_j \leftarrow \bot$ **else** $\boldsymbol{M}'_j \leftarrow \boldsymbol{M}'_j \parallel M$
return M

proc Dec.last(j, A, C)
if $j > J$ **or** $S'_j = \bot$ **then return** $\bot$
$\boldsymbol{A}'_j \leftarrow \boldsymbol{A}'_j \parallel A$; $\boldsymbol{C}'_j \leftarrow \boldsymbol{C}$; $S'_j \leftarrow \bot$
if $H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 0] \neq \bot$ **then**
  $\boldsymbol{M} \leftarrow H[N'_j, \boldsymbol{A}'_j, \boldsymbol{C}'_j, 1]$; **return** $\boldsymbol{M}[|\boldsymbol{M}|]$
$M \leftarrow \rho^{-1}_{N'_j, \boldsymbol{A}'_j, \boldsymbol{M}'_j, 1}(C)$; **return** M

Fig. 14: Games G_1–G_3 in the proof of Proposition 1. Game G_2 contains the corresponding boxed statements, but game G_1 doesn't. There is an implicit procedure **initialize**() that initializes $I, J \leftarrow 0$ and $Z \leftarrow \emptyset$, and samples $\rho_{N, \boldsymbol{A}, \boldsymbol{M}, \delta} \twoheadleftarrow \text{Inj}(\tau)$ for every $N \in \mathcal{N}, \delta \in \{0,1\}$, and $\boldsymbol{A}, \boldsymbol{M} \in \{0,1\}^{**}$ such that $|\boldsymbol{A}| = |\boldsymbol{M}| + 1$.

and thus

$$\Pr[A^{G_1} \Rightarrow 1] - \Pr[A^{G_2} \Rightarrow 1] \leq \Pr[A^{G_2} \text{ sets } \mathsf{bad}] \leq p \cdot \Pr[B_2^{\mathbf{Forge2C}_{\Pi}}] = p \cdot \mathbf{Adv}^{\text{oae2-auth}}_{\Pi}(\mathscr{B}_2) \ .$$

Game G_3 is identical to game G_2, except that instead of calling Enc.next($i, A, \cdot$) and Enc.last($i, A, \cdot$), we use $\rho_{N_i, \boldsymbol{A}_i, \boldsymbol{M}_i, 0} \twoheadleftarrow \text{Inj}(\tau)$ and $\rho_{N_i, \boldsymbol{A}_i, \boldsymbol{M}_i, 1} \twoheadleftarrow \text{Inj}(\tau)$ respectively. Moreover, Dec.next($i, A, \cdot$) and Dec.last($i, A, \cdot$) are also replaced by $\rho^{-1}_{N_i, \boldsymbol{A}_i, \boldsymbol{M}_i, 0}$ and $\rho^{-1}_{N_i, \boldsymbol{A}_i, \boldsymbol{M}_i, 1}$ respectively. Then $\Pr[A^{G_2} \Rightarrow 1] = \Pr[B_1^{\mathbf{Real2C}} \Rightarrow 1]$. In addition, $|\Pr[B_1^{\mathbf{Rand2C}} \Rightarrow 1] - \Pr[A^{G_3} \Rightarrow 1]|$ is exactly the gap between PRI and MRAE, which is at most $q^2/2^{\tau+1} + 4q/2^{\tau} \leq q^2/2^{\tau}$, by [56, Theorem 7]. Finally, game G_3 coincides with game $\mathbf{Ideal2B}_{\Pi}$. Summing up,

$$\mathbf{Adv}^{\text{oae2b}}_{\Pi}(\mathscr{A}) = \Pr[A^{G_1} \Rightarrow 1] - \Pr[A^{G_3} \Rightarrow 1] \leq \mathbf{Adv}^{\text{oae2-priv}}_{\Pi}(\mathscr{B}_1) + p \cdot \mathbf{Adv}^{\text{oae2-auth}}_{\Pi}(\mathscr{B}_2) + q^2/2^{\tau} \ .$$

proc Enc.init(N)
$I \leftarrow I + 1$; $S_I \leftarrow (K, N, 1)$; **return** I

proc Enc.next(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$(K, N, v) \leftarrow S_i$; $S_i \leftarrow (K, N, v + 1)$
$C \leftarrow \mathrm{Map}(\langle N, v, 0\rangle, A, M)$
return C

proc Enc.last(i, A, M) Games G_1, $\boxed{G_2}$
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$(K, N, v) \leftarrow S_i$; $S_i \leftarrow \bot$
$C \leftarrow \mathrm{Map}(\langle N, v, 1\rangle, A, M)$; **return** C

proc Map(N, A, M)
$C \leftarrow \mathbf{E}_K^{N,A}(M)$; $\boxed{C \twoheadleftarrow \{0,1\}^{|M|+\tau}}$
return C

proc Enc.init(N)
$I \leftarrow I + 1$; $\boldsymbol{M}_I \leftarrow \Lambda$; $\boldsymbol{A}_I \leftarrow \Lambda$
$S_I \leftarrow (K, N, 1)$; **return** I

proc Enc.next(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$(K, N, v) \leftarrow S_i$; $S_i \leftarrow (K, N, v + 1)$
$C \leftarrow \mathrm{Map}(\langle N, v, 0\rangle, A, M)$
$\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \parallel M$; $\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \parallel A$
$\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(N, \boldsymbol{A}, \boldsymbol{M}, 0)\}$; **return** C

proc Enc.last(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$(K, N, v) \leftarrow S_i$; $S_i \leftarrow \bot$
$C \leftarrow \mathrm{Map}(\langle N, v, 1\rangle, A, M)$
$\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \parallel M$; $\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \parallel A$
$\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(N, \boldsymbol{A}, \boldsymbol{M}, 1)\}$; **return** C

proc Map(N, A, M) Games P_1, $\boxed{P_2}$
$C \leftarrow \mathbf{E}_K^{N,A}(M)$; $\boxed{C \twoheadleftarrow \{0,1\}^{|M|+\tau}}$
$H[N, A, C] \leftarrow M$; **return** C

proc MapInv(N, A, C)
if $H[N, A, C] \neq \bot$ **then return** $H[N, A, C]$
$M \leftarrow \mathbf{D}_K^{N,A}(C)$; $\boxed{M \leftarrow \bot}$
return M

proc finalize $(N, \boldsymbol{A}, \boldsymbol{C}, b)$
if $|\boldsymbol{A}| \neq |\boldsymbol{C}|$ **or** $(N, \boldsymbol{A}, \boldsymbol{C}, b) \in \mathcal{Z}$ **then return** false
$m \leftarrow |\boldsymbol{C}|$
if $m = 0$ **then return** false
for $i \leftarrow 1$ **to** $m - b$ **do**
  $M \leftarrow \mathrm{MapInv}(\langle N, i, 0\rangle, \boldsymbol{A}[i], \boldsymbol{C}[i])$
  if $M = \bot$ **then return** false
if $b = 0$ **then return** true
else return $(\mathrm{MapInv}(\langle N, m, 1\rangle, \boldsymbol{A}[m], \boldsymbol{C}[m]) \neq \bot)$

Fig. 15: Games G_1, G_2, P_1, P_2 used in the proof of Theorem 2. Games G_2 and P_2 contain the corresponding boxed statements, but games G_1 and P_1 do not. In each game there is an implicit procedure **initialize**() that samples $K \twoheadleftarrow \mathbf{K}$, and initializes $I \leftarrow 0$ and $\mathcal{Z} \leftarrow \emptyset$.

E.3 Proof of Proposition 2

Constructing $\mathscr{B}_1$ is trivial: it ignores its decryption oracles and runs $\mathscr{A}_1$ on its encryption oracles. Then $\Pr[\mathscr{B}_1^{\mathbf{Real2B}_\Pi} \Rightarrow 1] = \Pr[\mathscr{A}_1^{\mathbf{Real2C}_\Pi} \Rightarrow 1]$, and $|\Pr[\mathscr{B}_1^{\mathbf{Ideal2B}} \Rightarrow 1] - \Pr[\mathscr{A}_1^{\mathbf{Rand2C}} \Rightarrow 1]|$ is exactly the gap between PRI and MRAE, which is at most $q^2/2^{\tau+1} + 4q/2^\tau \leq q^2/2^\tau$. Hence $\mathbf{Adv}_\Pi^{\text{oae2b}}(\mathscr{B}_1) = \mathbf{Adv}_\Pi^{\text{oae2-priv}}(\mathscr{A}_1) + q^2/2^\tau$. The reduction R_2 creates from $\mathscr{A}_2$ adversary $\mathscr{B}_2$ as follows. The latter runs the former on its encryption oracles and maintains the set $\mathcal{Z}$ of the partial decryption chains $(N_i, \boldsymbol{A}_i, \boldsymbol{C}_i, \delta_i)$ as in game $\mathbf{Forge2C}_\Pi$. When $\mathscr{A}_2$ outputs $(N, \boldsymbol{A}, \boldsymbol{C}, b)$, adversary $\mathscr{B}_2$ runs the following code:

if $|\boldsymbol{A}| \neq |\boldsymbol{C}|$ **or** $(N, \boldsymbol{A}, \boldsymbol{C}, b) \in \mathcal{Z}$ **or** $|\boldsymbol{C}| = 0$ **then return** 0
Dec.init(N); $m \leftarrow |\boldsymbol{C}|$
for $i \leftarrow 1$ **to** $m - b$ **do**
  $(M, S) \leftarrow \mathrm{Dec.next}(1, \boldsymbol{A}[i], \boldsymbol{C}[i])$
  if $M = \bot$ **then return** 0
if $b = 1$ **and** $\mathrm{Dec.last}(1, \boldsymbol{A}[m], \boldsymbol{C}[m]) = \bot$ **then return** 0
return 1

Then $\Pr[\mathscr{B}_2^{\mathbf{Ideal2B}_\Pi} \Rightarrow 1] \leq \ell/2^\tau$ and $\mathbf{Adv}_\Pi^{\text{oae2-auth}}(\mathscr{A}_2) = \Pr[\mathscr{B}_2^{\mathbf{Real2B}_\Pi} \Rightarrow 1]$. Hence $\mathbf{Adv}_\Pi^{\text{oae2-auth}}(\mathscr{A}_2) \leq \mathbf{Adv}_\Pi^{\text{oae2b}}(\mathscr{B}_2) + \ell/2^\tau$.

E.4 Proof of Theorem 2

The reduction R_1 creates from $\mathscr{A}_1$ adversary $\mathscr{B}_1$ as follows. The latter runs the former and simulates game $\mathbf{nReal}_{\mathbf{STREAM}[\Pi,\langle\cdot\rangle]}$, but each call to $\mathbf{E}_K(\cdot)$ is replaced by the corresponding query to the first oracle of $\mathscr{B}_1$, respectively. Adversary $\mathscr{B}_1$ then outputs the same guess as $\mathscr{A}_1$. Consider games G_1 and G_2 in Fig. 15. Game G_1 corresponds to game $\mathbf{nReal}_{\mathbf{STREAM}[\Pi,\langle\cdot\rangle]}$, and game G_2 corresponds to

proc Enc.init(N)
$I \leftarrow I + 1$; $S_I \leftarrow (K, N, 0)$; **return** I

proc Enc.next(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$(K, V, d) \leftarrow S_i$; $(C, V) \leftarrow \mathrm{Map}(V, \langle A, d\rangle, M)$
$S_i \leftarrow (K, V, 1)$; **return** C

proc Enc.last(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$(K, V, d) \leftarrow S_i$
if $d = 0$ **then** $d \leftarrow 3$ **else** $d \leftarrow 2$
$(C, V) \leftarrow \mathrm{Map}(V, \langle A, d\rangle, M)$; $S_i \leftarrow \bot$; **return** C

proc Map(V, A, M)
$C \leftarrow \pi(V, A, M)$
if $H[V, A, M] = \bot$ **then** $H[V, A, M] \leftarrow \mathrm{Ev}(V, A, M, C)$
$V' \leftarrow H[V, A, M]$; **return** (C, V')

proc Ev(V, A, M, C) Games G_1, $\boxed{G_2}$
$A = \langle A', d\rangle$
if $d \notin \{0, 1\}$ **then return** $\bot$
elsif $|M| \geq n$ **then** $V' \leftarrow C[1, n] \oplus M[1, n]$
else $L \leftarrow \pi(V, \langle A', d + 4\rangle, M \,\|\, 0^n)$; $V' \leftarrow L[1, n]$
if $(V' \in \mathrm{Dom})$ **then** bad $\leftarrow$ true; $\boxed{V' \twoheadleftarrow \{0,1\}^n \backslash \mathrm{Dom}}$
$\mathrm{Dom} \leftarrow \mathrm{Dom} \cup \{V'\}$; **return** V'

proc $\pi(V, A, M)$
if $X[V, A, M] \neq \bot$ **then return** $X[V, A, M]$
$C \leftarrow \mathbf{E}_K^{V,A}(M)$; $X[V, A, M] \leftarrow C$; **return** C

proc Ev(V, A, M, C) Game G_3
if $|M| \geq n$ **then** $V' \leftarrow C[1, n] \oplus M[1, n]$
elsif $A = \langle A', d\rangle$ **then**
  $L \leftarrow \pi(V, \langle A', d + 4\rangle, M \,\|\, 0^n)$; $V' \leftarrow L[1, n]$
if $(V' \in \mathrm{Dom})$ **then** bad $\leftarrow$ true; $V' \twoheadleftarrow \{0,1\}^n \backslash \mathrm{Dom}$
$\mathrm{Dom} \leftarrow \mathrm{Dom} \cup \{V'\}$; **return** V'

proc $\pi(V, A, M)$
if $X[V, A, M] \neq \bot$ **then return** $X[V, A, M]$
$C \twoheadleftarrow \{0,1\}^{|M|+\tau}$; $X[V, A, M] \leftarrow C$; **return** C

Fig. 16: Games G_1–G_3 used in the proof of Theorem 3. Game G_2 contains the corresponding boxed statements but game G_1 doesn't. Each game maintains local procedures Map, Ev, π that are inaccessible to the adversary. The games share the common procedures Enc.init, Enc.next, Enc.last, Map. In each game, there is an implicit procedure **initialize**() that initializes $\mathrm{Dom} \leftarrow \emptyset$ and $I, J \leftarrow 0$, and samples $K \twoheadleftarrow \mathbf{K}$.

$\mathbf{nRand}_{\mathbf{STREAM}[\Pi,\langle\cdot\rangle]}$. Game G_2 is identical to game G_1, except that instead of using $\mathbf{E}_K^{N,A}$, we'll use $\$(\cdot,\cdot,\cdot)$. Then

$$\mathbf{Adv}^{\text{noae-priv}}_{\mathbf{STREAM}[\Pi,\langle\cdot\rangle]}(\mathscr{A}_1) = \Pr[\mathscr{A}_1^{G_1} \Rightarrow 1] - \Pr[\mathscr{A}_1^{G_2} \Rightarrow 1] = \mathbf{Adv}^{\text{nae}}_{\Pi}(\mathscr{B}_1) \ .$$

The reduction R_2 creates from $\mathscr{A}_2$ adversary $\mathscr{B}_2$ as follows. The latter runs the former and simulates game P_1, but each call to $\mathbf{E}_K(\cdot)$ or $\mathbf{D}_K(\cdot)$ is replaced by the corresponding query to the first or second oracle of $\mathscr{B}_2$, respectively. It outputs 1 if the simulated **finalize** returns true, and outputs 0 otherwise. Consider games P_1 and P_2 in Fig. 15. Game P_1 corresponds to game $\mathbf{nForge}_{\mathbf{STREAM}[\Pi,\langle\cdot\rangle]}$. Game G_2 is identical to game G_1, except that instead of using $\mathbf{E}_K^{N,A}$ and $\mathbf{D}_K^{N,A}$, we'll use $\$(\cdot,\cdot,\cdot)$ and $\bot(\cdot,\cdot,\cdot)$. Then

$$\Pr[\mathscr{A}_2^{P_1}] - \Pr[\mathscr{A}_2^{P_2}] = \mathbf{Adv}^{\text{nae}}_{\Pi}(\mathscr{B}_2) \ .$$

On the other hand, $\Pr[A_2^{P_2}] = 0$, and thus $\mathbf{Adv}^{\text{noae-auth}}_{\mathbf{STREAM}[\Pi,\langle\cdot\rangle]}(\mathscr{A}_2) \leq \mathbf{Adv}^{\text{nae}}_{\Pi}(\mathscr{B}_2)$.

E.5 Proof of Theorem 3

Wlog, assume that adversaries do not make invalid queries that result in $\bot$-answers. For privacy, consider games G_1–G_3 in Fig. 16. Game G_1 coincides with game $\mathbf{dReal}_{\mathbf{CHAIN}[\Pi,\langle\cdot\rangle,n]}$. Game G_2 is identical to game G_1, except that we want the state V' to be never repeated, but still maintain the following consistency: calling Map with the same (V, A, M) always results in the same (C, V'). The two games are identical-until-bad, and thus

$$\Pr[\mathscr{A}_2^{G_1} \Rightarrow 1] - \Pr[\mathscr{A}_2^{G_2} \Rightarrow 1] \leq \Pr[G_2 \text{ sets bad}] \ .$$

The reduction R_1 creates from $\mathscr{A}_1$ adversary $\mathscr{B}_1$ as follows. The latter runs the former and simulates game G_2, but each call to $\mathbf{E}_K(\cdot)$ is replaced by the corresponding query to the encryption oracle of $\mathscr{B}_1$.

The adversary $\mathscr{B}_2$ then samples $b \twoheadleftarrow \{0,1\}$. If $b = 1$, it returns 1 if its simulated game G_2 sets bad (and also terminates the simulated game immediately), and returns 0 otherwise. Otherwise, it returns the same guess of $\mathscr{A}$. Note that in game G_2,

(a) Calling Enc.next(i, A_i, M_i) and Enc.last(j, A_j, M_j) will never result in the same Map$(V, A, \cdot)$, due to the domain separation.
(b) Calling Enc.next(i, A_i, M_i) and Enc.next(j, A_j, M_j) result in the same Map$(V, A, \cdot)$ if and only if $(N_i, \boldsymbol{A}_i, \boldsymbol{M}_i) = (N_j, \boldsymbol{A}_j, \boldsymbol{M}_j)$ and $A_i = A_j$, where $\boldsymbol{A}_k$ and $\boldsymbol{M}_k$ are the segmented-AD and segmented-message of the k-th chain prior to A_k and M_k respectively, for $k \in \{i, j\}$. We will justify this claim later. The similar claim holds for Enc.last as well.

Then adversary $\mathscr{B}_1$ never repeats a prior (N, A) to its encryption oracle. Moreover, $\Pr[\mathscr{A}_1^{G_2} \Rightarrow 1] = \Pr[\mathscr{B}_1^{\mathbf{E}_K, \mathbf{D}_K} \Rightarrow 1 \mid b = 0]$ and $\Pr[\mathscr{A}_1^{G_2} \text{ sets } \mathsf{bad}] = \Pr[\mathscr{B}_1^{\mathbf{E}_K, \mathbf{D}_K} \Rightarrow 1 \mid b = 1]$. Hence

$$\Pr[\mathscr{A}_1^{G_1} \Rightarrow 1] \le \Pr[\mathscr{A}_1^{G_2} \Rightarrow 1] + \Pr[\mathscr{A}_1^{G_2} \text{ sets } \mathsf{bad}] = 2\Pr[\mathscr{B}_1^{\mathbf{E}_K, \mathbf{D}_K} \Rightarrow 1] \ .$$

To justify the claim (b) above, terminate the game G_2 as soon as we have two calls Enc.next(i, A_i, M_i) and Enc.next(j, A_j, M_j) that result in the same Map$(V, A, \cdot)$, but either $(N_i, \boldsymbol{A}_i, \boldsymbol{M}_i) \neq (N_j, \boldsymbol{A}_j, \boldsymbol{M}_j)$ or $A_i \neq A_j$. Since calling Enc.next(i, A_i, M_i) results in Map$(V, A, \cdot)$, it means that $A = \langle A_i, d\rangle$. Likewise, since calling Enc.next(j, A_j, M_j) results in Map$(V, A, \cdot)$, it means that $A = \langle A_j, d'\rangle$. In other words, we must have $A_i = A_j$. Hence $(N_i, \boldsymbol{A}_i, \boldsymbol{M}_i) \neq (N_j, \boldsymbol{A}_j, \boldsymbol{M}_j)$. We consider the following cases.

Case 1: $|\boldsymbol{M}_i|, |\boldsymbol{M}_j| > 0$. Let $\boldsymbol{M}'_i$ be the prefix of $\boldsymbol{M}_i$ that consists of $|\boldsymbol{M}_i| - 1$ components. Define $\boldsymbol{A}'_i$ for $\boldsymbol{A}_i$, $\boldsymbol{M}'_j$ for $\boldsymbol{M}_j$, and $\boldsymbol{A}'_j$ for $\boldsymbol{A}_j$ similarly. Let $A'_i = \boldsymbol{A}_i[|\boldsymbol{A}_i| - 1]$ and $M'_i = \boldsymbol{M}_i[|\boldsymbol{M}_i| - 1]$. Define A'_j and M'_j similarly. Then due to the unique sampling of the state V', earlier, the calls Enc.next(i, A'_i, M'_i) and Enc.next(j, A'_j, M'_j) would result in the same Map$(V', A', \cdot)$. Since the game didn't terminate then, it means that $(N_i, \boldsymbol{M}'_i, \boldsymbol{A}_i) = (N_j, \boldsymbol{M}'_j, \boldsymbol{A}_j)$. Because $(N_i, \boldsymbol{A}_i, \boldsymbol{M}_i) \neq (N_j, \boldsymbol{A}_j, \boldsymbol{M}_j)$, we must have $M_i \neq M_j$, meaning that either Enc.next(i, A_i, M_i) or Enc.next(j, A_j, M_j) is invalid, contradicting our assumption.

Case 2: $|\boldsymbol{M}_i| > 0$ and $|\boldsymbol{M}_j| = 0$. Since calling Enc.next(i, A_i, M_i) results in Map$(V, A, \cdot)$, it means that $A = \langle A_i, 1\rangle$. However, since calling Enc.next(j, A_j, M_j) results in Map$(V, A, \cdot)$, it means that $A = \langle A_j, 0\rangle$, which is a contradiction.

Case 3: $|\boldsymbol{M}_i| = 0$ and $|\boldsymbol{M}_j| > 0$. This is similar to Case 2.

Case 4: $|\boldsymbol{M}_i| = |\boldsymbol{M}_j| = 0$, meaning that $\boldsymbol{M}_i = \boldsymbol{M}_j = \Lambda$. Since calling Enc.next$(i, A_i, M_i)$ results in Map$(V, A, \cdot)$, it means that $V = N_i$. Likewise, since calling Enc.next(j, A_j, M_j) results in Map$(V, A, \cdot)$, it means that $V = N_j$. Because $(N_i, \boldsymbol{A}_i, \boldsymbol{M}_i) \neq (N_j, \boldsymbol{A}_j, \boldsymbol{M}_j)$, we must have $M_i \neq M_j$, meaning that either Enc.next(i, A_i, M_i) or Enc.next(j, A_j, M_j) is invalid, contradicting our assumption.

Next, game G_3 is identical to game G_2 except that, we use $\$(\cdot)$ instead of $\mathbf{E}_K$. Note that claims (a) and (b) above also hold for game G_3, and thus game G_3 coincides with game $\mathbf{dRand}_{\mathbf{CHAIN}[\Pi, \langle\cdot\rangle, n]}$. Then $\Pr[\mathscr{A}_1^{G_3} \Rightarrow 1] = \Pr[\mathscr{B}_1^{\$(\cdot), \perp(\cdot)} \Rightarrow 1 \mid b = 0]$ and $\Pr[\mathscr{A}_1^{G_3} \text{ sets } \mathsf{bad}] = \Pr[\mathscr{B}_1^{\$(\cdot), \perp(\cdot)} \Rightarrow 1 \mid b = 1]$. Hence

$$\Pr[\mathscr{A}_1^{G_3} \Rightarrow 1] = 2\Pr[\mathscr{B}_1^{\$(\cdot), \perp(\cdot)} \Rightarrow 1] - \Pr[\mathscr{A}_1^{G_3} \text{ sets } \mathsf{bad}],$$

Hence

$$\mathbf{Adv}^{\text{doae-priv}}_{\mathbf{CHAIN}[\Pi, \langle\cdot\rangle, n]}(\mathscr{A}_1) = \Pr[\mathscr{A}_1^{G_1} \Rightarrow 1] - \Pr[\mathscr{A}_1^{G_3} \Rightarrow 1] \le 2\,\mathbf{Adv}^{\text{nae0}}_{\Pi}(\mathscr{B}_1) + \Pr[\mathscr{A}_1^{G_3} \text{ sets } \mathsf{bad}] \ .$$

What's left is to show that $\Pr[\mathscr{A}_1^{G_3} \text{ sets } \mathsf{bad}] \le p^2/2^n$. Terminate the game immediately when bad gets set; it doesn't change the probability that G_3 sets bad. The flag bad is triggered only if the state V' repeats one of its prior values. Consider the following cases.

proc Enc.init(N)
$I \leftarrow I + 1$; $\boldsymbol{A}_I \leftarrow \Lambda$; $\boldsymbol{C}_I \leftarrow \Lambda$; $N_I \leftarrow N$
$S_I \leftarrow (K, N, d)$; **return** I

proc Enc.next(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$(K, V, d) \leftarrow S_i$; $(C, V) \leftarrow \text{Map}(V, \langle A, d\rangle, M)$
$\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \parallel A$; $\boldsymbol{C}_i \leftarrow \boldsymbol{C}_i \parallel C$
$\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(N_i, \boldsymbol{A}_i, \boldsymbol{C}_i, 0)\}$; $S_i \leftarrow (K, V, 1)$
return C

proc Enc.last(i, A, M)
if $i > I$ **or** $S_i = \bot$ **then return** $\bot$
$(K, V, d) \leftarrow S_i$
if $d = 0$ **then** $d \leftarrow 3$ **else** $d \leftarrow 2$
$(C, V) \leftarrow \text{Map}(V, \langle A, d\rangle, M)$
$\boldsymbol{A}_i \leftarrow \boldsymbol{A}_i \parallel A$; $\boldsymbol{C}_i \leftarrow \boldsymbol{C}_i \parallel C$
$\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(N_i, \boldsymbol{A}_i, \boldsymbol{C}_i, 1)\}$
$S_i \leftarrow \bot$; $C \leftarrow \pi(V, A, M)$; **return** C

proc Map(V, A, M)
$C \leftarrow \pi(V, A, M)$
if $H[V, A, M] = \bot$ **then** $H[V, A, M] \leftarrow \text{Ev}(V, A, M, C)$
$V' \leftarrow H[V, A, M]$; **return** (C, V')

proc MapInv(V, A, C)
$M \leftarrow \pi^{-1}(V, A, C)$
if $M \neq \bot$ **then** $V' \leftarrow H[V, A, M]$ **else** $V' \leftarrow \bot$
return (M, V')

proc finalize $(N, \boldsymbol{A}, \boldsymbol{C}, \delta)$
if $|\boldsymbol{A}| \neq |\boldsymbol{C}|$ **or** $(N, \boldsymbol{A}, \boldsymbol{C}, \delta) \in \mathcal{Z}$ **then return** false
if $|\boldsymbol{C}| = 0$ **then return** false
$m \leftarrow |\boldsymbol{C}|$; $V \leftarrow N$
for $i \leftarrow 1$ **to** $m - \delta$ **do**
  if $i = 1$ **then** $d \leftarrow 0$ **else** $d \leftarrow 1$
  $(M, V) \leftarrow \text{MapInv}(V, \langle \boldsymbol{A}[i], d\rangle, \boldsymbol{C}[i])$
  if $M = \bot$ **then return** false
if $(\delta = 1) \wedge (|\boldsymbol{C}| = 1)$ **then** $d \leftarrow 3$ **else** $d \leftarrow 2$
if $\delta = 0$ **then return** true
else return $(\text{MapInv}(V, \langle \boldsymbol{A}[m], d\rangle, \boldsymbol{C}[m]) \neq \bot)$

proc Ev(V, A, M, C) Games P_1, $\boxed{P_2}$
if $|M| \geq n$ **then** $V' \leftarrow C[1, n] \oplus M[1, n]$
elsif $A = \langle A', d\rangle$ **then**
  $L \leftarrow \pi(V, \langle A', d + 4\rangle, M \parallel 0^n)$; $V' \leftarrow L[1, n]$
if $(V' \in \text{Dom})$ **then** bad $\leftarrow$ true; $\boxed{V' \twoheadleftarrow \{0,1\}^n \backslash \text{Dom}}$
$\text{Dom} \leftarrow \text{Dom} \cup \{V'\}$; **return** V'

proc $\pi(V, A, M)$
if $X[V, A, M] \neq \bot$ **then return** $X[V, A, M]$
$C \leftarrow \mathbf{E}_K^{V,A}(M)$; $X[V, A, M] \leftarrow C$; $Y[V, A, C] \leftarrow M$
return C

proc $\pi^{-1}(V, A, C)$
if $Y[V, A, C] \neq \bot$ **then return** $Y[V, A, C]$
return $\mathbf{D}_K^{V,A}(C)$

proc Ev(V, A, M, C) Game P_3
if $|M| \geq n$ **then** $V' \leftarrow C[1, n] \oplus M[1, n]$
elsif $A = \langle A', d\rangle$ **then**
  $L \leftarrow \pi(V, \langle A', d + 4\rangle, M \parallel 0^n)$; $V' \leftarrow L[1, n]$
if $(V' \in \text{Dom})$ **then** $V' \twoheadleftarrow \{0,1\}^n \backslash \text{Dom}$
$\text{Dom} \leftarrow \text{Dom} \cup \{V'\}$; **return** V'

proc $\pi(V, A, M)$
if $X[V, A, M] \neq \bot$ **then return** $X[V, A, M]$
$C \twoheadleftarrow \{0,1\}^{|M|+\tau}$; $X[V, A, M] \leftarrow C$; $Y[V, A, C] \leftarrow M$
return C

proc $\pi^{-1}(V, A, C)$
if $Y[V, A, C] \neq \bot$ **then return** $Y[V, A, C]$
return $\bot$

Fig. 17: Games P_1–P_3 used in the proof of Theorem 3. Game P_2 contains the corresponding boxed statements but game $_1$ doesn't. Each game maintains local procedures Map, MapInv, Ev, π, and π^{-1} that are inaccessible to the adversary. The games share the common procedures Enc.init, Enc.next, Enc.last, Map, MapInv, and **finalize**. In each game, there is an implicit procedure **initialize**() that initializes $\mathcal{Z}, \text{Dom} \leftarrow \emptyset$ and $I, J \leftarrow 0$, and samples $K \twoheadleftarrow \mathbf{K}$.

Case 1: $V' \leftarrow L[1, n]$, where L is created by calling $\pi(V, \langle A', d + 4\rangle, M \parallel 0^n)$. There must be no prior call to $\text{Map}(V, A, M)$, where $A = \langle A', d\rangle$ otherwise we'll return the consistent value $H[V, A, M]$, and this line of code is not triggered. Due to the one-to-one correspondence between (V, A, M) and $(V, \langle A', d + 4\rangle, M \parallel 0^n)$, there is no prior call to $\pi(V, \langle A', d + 4\rangle, M \parallel 0^n)$. Hence L is a fresh random string, and thus the chance that V' repeats a prior value is at most $p/2^n$.

Case 2: $V' \leftarrow C[1, n] \oplus M[1, n]$. Again, there must be no prior call to $\text{Map}(V, A, M)$, otherwise we'll return the consistent value $H[V, A, M]$, and this line of code is not triggered. Then C is a fresh random string, and thus the chance that V' repeats a prior value is at most $p/2^n$.

Summing up over p queries, the chance that G_3 sets bad is at most $p^2/2^n$.

For authenticity, consider games P_1–P_3 in Fig. 17. Game P_1 coincides with game $\mathbf{dForge}_{\mathbf{CHAIN}[\Pi,\langle\cdot\rangle,n]}$. Game P_2 is identical to game P_1, except that we want the state V' to be never repeated, but still maintain the following consistency: calling Map with the same (V, A, M) always results in the same

(C, V'). The two games are identical-until-bad, and thus

$$\Pr[\mathscr{A}_2^{G_1} \Rightarrow 1] - \Pr[\mathscr{A}_2^{G_2} \Rightarrow 1] \leq \Pr[G_2 \text{ sets } \mathsf{bad}] \ .$$

The reduction R_2 creates from $\mathscr{A}_2$ adversary $\mathscr{B}_2$ as follows. The latter runs the former and simulates game P_2, but each call to $\mathbf{E}_K(\cdot)$ or $\mathbf{D}_K(\cdot)$ is replaced by the corresponding query to the first and second oracles of $\mathscr{B}_2$ respectively. The adversary $\mathscr{B}_2$ then samples $b \xleftarrow{} \{0,1\}$. If $b = 1$, it returns 1 if its simulated game P_2 sets bad (and terminates the simulated game immediately), and returns 0 otherwise. Otherwise, it returns 1 if the simulated **finalize** returns true, and returns 0 otherwise. Again, note that in game P_2,

(i) Calling Enc.next(i, A_i, M_i) and Enc.last(j, A_j, M_j) will never result in the same Map$(V, A, \cdot)$, due to the domain separation.
(ii) Calling Enc.next(i, A_i, M_i) and Enc.next(j, A_j, M_j) result in the same Map$(V, A, \cdot)$ if and only if $(N_i, \boldsymbol{A}_i, \boldsymbol{M}_i) = (N_j, \boldsymbol{A}_j, \boldsymbol{M}_j)$ and $A_i = A_j$, where $\boldsymbol{A}_k$ and $\boldsymbol{M}_k$ are the segmented-AD and segmented-message of the k-th chain prior to A_k and M_k respectively, for $k \in \{i, j\}$. The similar claim holds for Enc.last as well.

Then adversary $\mathscr{B}_2$ never repeats a prior (N, A) to its encryption oracle. Moreover, $\Pr[\mathscr{A}_2^{P_2} \Rightarrow 1] = \Pr[\mathscr{B}_2^{\mathbf{E}_K, \mathbf{D}_K} \Rightarrow 1 \mid b = 0]$ and $\Pr[\mathscr{A}_2^{P_2} \text{ sets } \mathsf{bad}] = \Pr[\mathscr{B}_2^{\mathbf{E}_K, \mathbf{D}_K} \Rightarrow 1 \mid b = 1]$. Hence

$$\Pr[\mathscr{A}_2^{P_1}] \leq \Pr[\mathscr{A}_2^{P_2}] + \Pr[\mathscr{A}_2^{P_2} \text{ sets } \mathsf{bad}] = 2\Pr[\mathscr{B}_2^{\mathbf{E}_K, \mathbf{D}_K} \Rightarrow 1] \ .$$

Next, game P_3 is identical to game P_2 except that, we use $\$(\cdot)$ and $\perp(\cdot)$ instead of $\mathbf{E}_K$ and $\mathbf{D}_K(\cdot)$. Then $\Pr[\mathscr{A}_2^{P_3}] = \Pr[\mathscr{B}_2^{\$(\cdot), \perp(\cdot)} \Rightarrow 1 \mid b = 0]$ and $\Pr[\mathscr{A}_2^{P_3} \text{ sets } \mathsf{bad}] = \Pr[\mathscr{B}_2^{\$(\cdot), \perp(\cdot)} \Rightarrow 1 \mid b = 1]$. Hence

$$\Pr[\mathscr{A}_2^{P_3}] = 2\Pr[\mathscr{B}_2^{\$(\cdot), \perp(\cdot)} \Rightarrow 1] - \Pr[\mathscr{A}_2^{P_3} \text{ sets } \mathsf{bad}] \ .$$

We now show that $\Pr[\mathscr{A}_2^{P_3} \text{ sets } \mathsf{bad}] \leq q^2/2^n$. Terminate the game immediately when bad gets set; it doesn't change the probability that P_3 sets bad. The flag bad is triggered only if the state V' repeats one of its prior values. Consider the following cases.

Case 1: $V' \leftarrow L[1, n]$, where L is created by calling $\pi(V, \langle A, d + 4\rangle, M \parallel 0^n)$. There must be no prior call to Map(V, A, M), otherwise we'll return the consistent value $H[V, A, M]$, and this line of code is not triggered. Due to the one-to-one correspondence between (V, A, M) and $(V, \langle A', d + 4\rangle, M \parallel 0^n)$, there is no prior call to $\pi(V, \langle A', d + 4\rangle, M \parallel 0^n)$. Hence L is a fresh random string, and thus the chance that V' repeats a prior value is at most $q/2^n$.

Case 2: $V' \leftarrow C[1, n] \oplus M[1, n]$. Again, there must be no prior call to Map(V, A, M), otherwise we'll return the consistent value $H[V, A, M]$, and this line of code is not triggered. Then C is a fresh random string, and thus the chance that V' repeats a prior value is at most $q/2^n$.

Summing up over q queries, the chance that P_3 sets bad is at most $q^2/2^n$.

We now show that $\Pr[\mathscr{A}_2^{P_3}] = 0$, and thus

$$\mathbf{Adv}^{\text{doae-auth}}_{\mathbf{CHAIN}[\Pi, \langle\cdot\rangle, n]}(\mathscr{A}_2) = \Pr[\mathscr{A}_2^{P_1}] \leq 2\,\mathbf{Adv}^{\text{nae0}}_{\Pi}(\mathscr{B}_2) + \Pr[\mathscr{A}_2^{P_3} \text{ sets } \mathsf{bad}] \leq 2\,\mathbf{Adv}^{\text{nae0}}_{\Pi}(\mathscr{B}_2) + q^2/2^n \ .$$

To justify the claim above, suppose that the query **finalize**$(N, \boldsymbol{A}, \boldsymbol{C}, \delta)$ returns true. Let $\boldsymbol{A}^*$ be the segmented string such that $|\boldsymbol{A}^*| = |\boldsymbol{A}|$ and $\boldsymbol{A}^*[k]$ is of the form $\langle \boldsymbol{A}[k], d_k\rangle$ for every $1 \leq k \leq |\boldsymbol{A}|$, where (1) $d_k = 0$ if $k = 1$ and ($\delta = 0$ or $|\boldsymbol{A}| > 1$), (2) $d_k = 1$ if $1 < k < |\boldsymbol{A}|$ or ($k = |\boldsymbol{A}|$ and $\delta = 0$), (3) $d_k = 2$ if $\delta = 1$, $k = |\boldsymbol{A}|$, and $|\boldsymbol{A}| > 1$, and (4) $d_k = 3$ otherwise. Recall that the **finalize** query internally calls MapInv$(N, \boldsymbol{A}^*[1], \boldsymbol{C}[1])$. Because **finalize**$(N, \boldsymbol{A}, \boldsymbol{C}, \delta)$ returns true, the call to MapInv above must not return $\perp$. Due to the domain separation in the use of AD, this means that there is some

$(N, \boldsymbol{A}_i, \boldsymbol{C}_i, \delta_i) \in \mathcal{Z}$ of the same nonce N such that $\boldsymbol{A}_i^*[1] = \boldsymbol{A}^*[1]$ and $\boldsymbol{C}_i[1] = \boldsymbol{C}[1]$, where $\boldsymbol{A}_i^*$ is defined from $\boldsymbol{A}_i$ as $\boldsymbol{A}^*$ was defined from $\boldsymbol{A}$ above. Let ℓ be the biggest number such that $\boldsymbol{A}_i^*[k] = \boldsymbol{A}^*[k]$ and $\boldsymbol{C}_i[k] = \boldsymbol{C}[k]$ for every $k \leq \ell$. Above, if there are many possible choices for tuples $(N, \boldsymbol{A}_i, \boldsymbol{C}_i, \delta_i)$, we would pick the one that maximizes ℓ. We consider the following cases.

Case 1: $|\boldsymbol{A}^*| = \ell$ and $|\boldsymbol{A}_i^*| = \ell$, meaning that $\boldsymbol{A}^* = \boldsymbol{A}_i^*$ and $\boldsymbol{C}_i = \boldsymbol{C}$. In particular, since $\boldsymbol{A}^*$ and $\boldsymbol{A}_i^*$ agree in their last components, due to the domain separation in the use of AD, we must have $\delta_i = \delta$. But then $(N, \boldsymbol{A}, \boldsymbol{C}, \delta) \in \mathcal{Z}$, and thus the answer for the **finalize** query would be false, which is a contradiction.

Case 2: $|\boldsymbol{A}^*| = \ell$ and $|\boldsymbol{A}_i^*| > \ell$. Let $\boldsymbol{A}'$ be the prefix of $\boldsymbol{A}_i^*$ of ℓ components, and define $\boldsymbol{C}'$ for $\boldsymbol{C}_i$ similarly. Then $(N, \boldsymbol{A}, \boldsymbol{C}, \delta) = (N, \boldsymbol{A}', \boldsymbol{C}', 0) \in \mathcal{Z}$, and thus the answer for the **finalize** query would be false, which is a contradiction.

Case 3: $|\boldsymbol{A}^*| > \ell$. Recall that **finalize**$(N, \boldsymbol{A}, \boldsymbol{C}, \delta)$ invokes MapInv$(V, \boldsymbol{A}^*[\ell+1], \boldsymbol{C}[\ell+1])$. Since **finalize** returns true, this call to MapInv must not return $\perp$, meaning there must be some $(N_j, \boldsymbol{A}_j, \boldsymbol{C}_j, \delta_j) \in \mathcal{Z}$ that runs MapInv$(V, \boldsymbol{A}^*[\ell+1], \cdot)$ such that $\boldsymbol{A}_j^*[\ell+1] = \boldsymbol{A}^*[\ell+1]$ and $\boldsymbol{C}_j[\ell+1] = \boldsymbol{C}[\ell+1]$, where $\boldsymbol{A}_j^*$ is defined from $\boldsymbol{A}_j$ as $\boldsymbol{A}^*$ was defined from $\boldsymbol{A}$ above. Let $\boldsymbol{M}_i$ and $\boldsymbol{M}_j$ be the corresponding segmented messages of $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ respectively. Since both $(N, \boldsymbol{A}_i, \boldsymbol{C}_i, \delta_i)$ and $(N_j, \boldsymbol{A}_j, \boldsymbol{C}_j, \delta_j)$ result in the same state V in their ℓth queries, from (1) the way we re-sample states to avoid collision and (2) the fact that claims (i) and (ii) also hold for game P_3, we must have $N_j = N$, $\boldsymbol{A}_j^*[k] = \boldsymbol{A}_i^*[k]$ and $\boldsymbol{C}_j[k] = \boldsymbol{C}_i[k]$ for every $k \leq \ell$. However, now $\boldsymbol{A}_j^*[t] = \boldsymbol{A}[t]$ and $\boldsymbol{C}_j[t] = \boldsymbol{C}[t]$ for every $t \leq \ell + 1$, violating the maximality of $(N, \boldsymbol{A}_i, \boldsymbol{C}_i, \delta_i)$.