# 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

March 4, 2015

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

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

## 1 Introduction

Between nAE and MRAE. With a typical nonce-based authenticated-encryption (nAE) scheme [45, 47], 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) [48]. 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 [48], 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 [25] 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) [14]. 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)

[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* [16]. Others use *IV* (initialization vector). We will stick with *nonce* for a value that nominally ought not repeat, yet might.

where the $i$th 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 [38] 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 $\mathsf{B}_n$ 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 $\mathcal{E}_K^{N,A}(L \parallel \cdot \parallel S)$, for an arbitrary, known $L$. See Section 3. The idea was introduced, in a different setting, with the BEAST attack [24].

While many real-world settings won't enable a CPSS attack, our own take is that, for a general-purpose tool, such a weakness effectively refutes any claim of misuse resistance. If the phrase is to mean anything, it should entail that the basic characteristics of nAE are maintained in the presence of nonce-reuse. An AE scheme satisfying nAE (employing non-repeating nonces) or MRAE (without that restriction) would certainly be immune to such an attack.

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 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 [54]. 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" [5]. 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 [40] 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,

| | **OAE1** (from FFL [25]) | **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 | $\tau$ bits per message (eg, $\tau = 128$) | $\tau$ 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 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.

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 two 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.

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) [54] expressed this insight and provided a definition based on it. TSS explain that AE à la Boldyreva and Taesombut [20] (or BBKN or FFL, for that matter) "processes and outputs ... blocks as soon as the next input block is received" [54, p. 4], whence they ask, "what if the input is smaller than a block?", even a bit,[6] or what "if the input is a [segment] ... of arbitrary length?" TSS maintain that such situations occur in practice, and they give examples [54, 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 only commonality is some motivation and syntax. Yet it

[6] It is in this sense that one might maintain that the OAE1 definition does not model even encryption being *online*; the encrypting party might be unable to output *anything*, as the current block is incomplete.

seems unfortunate that the TSS manuscript escaped greater notice; in our view, it seems more prescient than alternative lines. For further discussion of prior work, see Appendix B.

A real-world need. Netflix recently described a protocol of theirs, MSL, for streaming video [40]. 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 publicly asked for such a tool, complaining that with the current AE schemes one has to wait until the end of a ciphertext before one can release the decrypted message. 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?" [53].

## 2 OAE1 Definition

All OAE definitions of widespread use spring from FFL [25], who married the definition of an online cipher from Bellare, Boldyreva, Knudsen, and Namprempre [14] with the definition of authenticity of ciphertexts (also called integrity of ciphertexts) [15, 37, 47]. 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 $\mathcal{D}\colon\ \mathcal{K} \times \mathcal{H} \times \{0,1\}^* \to \mathsf{B}_n^* \cup \{\bot\}$. The set $\mathcal{H}$ associated to $\Pi$ 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 $\Pi$ is its *blocksize*. Note that the *message space* $\mathcal{M}$ of $\Pi$ 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, additionally, to require 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 $\pi$ on $\mathsf{B}_n^*$ where $i$th 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 $\pi \twoheadleftarrow \mathrm{OPerm}[n]$.

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 $\Pi$ and an adversary $\mathscr{A}$ the real number $\mathbf{Adv}_{\Pi}^{\mathrm{oae1}}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{Real1}} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{Ideal1}} \Rightarrow 1]$ 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 said to be OAE1 secure if $\mathbf{Adv}_{\Pi}^{\mathrm{oae1}}(\mathscr{A})$ 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 $n$.

```
initialize                                    Real1_Π
$K \twoheadleftarrow \mathcal{K}$

proc Enc($H, M$)
if $H \notin \mathcal{H}$ or $M \notin \mathsf{B}_n^*$ then
    return $\bot$
return $\mathcal{E}(K, H, M)$

proc Dec($H, C$)
if $H \notin \mathcal{H}$ then return $\bot$
return $\mathcal{D}(K, H, C)$
```

```
initialize                                    Ideal1_Π
for $H \in \mathcal{H}$ do $\pi_H \twoheadleftarrow \mathrm{OPerm}[n]$
for $(H, M) \in \mathcal{H} \times \mathsf{B}_n^*$ do $R_{H,M} \twoheadleftarrow \{0,1\}^\tau$

proc Enc($H, M$)
if $H \notin \mathcal{H}$ or $M \notin \mathsf{B}_n^*$ then return $\bot$
return $\pi_H(M) \parallel R_{H,M}$

proc Dec($H, C$)
return $\bot$
```

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 $\tau$. See the accompanying text for the definition of OPerm$[n]$.

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 $\tau$-bit random string, the authentication tag. But the original definition from FFL somehow omitted the second part [25, 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 [1, 26]. After discussions among ourselves and checking with one of the FFL authors [39], 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 $\mathcal{E}_K^H(M)$ depends 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 being an object computationally close to something with a property $P$ doesn't mean that it will *always* have 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 $\Pi$ as above and consider an adversary $\mathscr{A}$ that is given an oracle $\mathcal{E}_K(\cdot, \cdot)$ for $K \twoheadleftarrow \mathcal{K}$. 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 [24].

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 $M = P \parallel S$ 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 $\mathcal{E}_K^H(P \parallel S)$ for any $P$ it wants—except, to be realistic, we insist that $P$ be a multiple of $b$ bits. This is assumed for $S$ too. Typically $P$ and $S$ must be byte strings, whence $b = 8$; for concreteness, let us assume this. Also for concreteness, assume a blocksize of $n = 128$ bits. Assume that $\mathcal{E}$ can in fact operate on arbitrary byte-length strings, but suffers LCP leakage on block-aligned prefixes (this is what happens if one pads and then applies an OAE1-secure scheme). Finally, assume $|S|$ is a multiple of the blocksize.

To recover $S$, the adversary proceeds as follows. First it selects an arbitrary string $P_1$ whose byte length is one byte shorter than $p$ blocks, for an arbitrary $p \geq 1$. (For example, it would be fine to have $P_1 = 0^{120}$.) The adversary requests ciphertext $C_1 = \mathcal{E}_K^H(P_1 \parallel S)$. This will be used to learn $S_1$, the first byte of $S$. To do so, the adversary requests ciphertexts $C_{1,B} = \mathcal{E}_K^H(P_1 \parallel B \parallel S)$ for all 256 one-byte values $B$. Due to LCP leakage, exactly one of these values, the one with $B = S_1$, will agree with $C_1$ on the first $p$ blocks. At this point the adversary knows the first byte of $S$, and has spent 257 queries to get it. There is an obvious strategy to reduce this to 256 queries: omit one of the 256-possible byte values for $B$ and use this for $S_1$ if no other match is found.

Now the adversary wants to learn $S_2$, the second byte of $S$. It selects an arbitrary string $P_2$ that is *two* bytes short of $p$ blocks, for any $p \geq 1$. The adversary requests the ciphertext $C_2 = \mathcal{E}_K^H(P_2 \parallel S)$; and it requests ciphertexts $C_{2,B} = \mathcal{E}_K^H(P_2 \parallel S_1 \parallel B \parallel S)$ 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 $L \parallel R \parallel S$ 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 of the message $L$, 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 portion 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 $L \parallel R \parallel S$ 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 $i$th 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 $i$th *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 [49, 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* $\underline{\textit{all}}$ *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 $\{0,1\}^{\geq n}$ [49]. 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.

```
algorithm $\boldsymbol{\mathcal{E}}(K, N, A, \boldsymbol{M})$
$m \leftarrow |\boldsymbol{M}|$
if $m = 0$ then return $\Lambda$
$(M_1, \ldots, M_m) \leftarrow M$
$S_0 \leftarrow \mathcal{E}.\mathrm{init}(K, N, A)$
for $i \leftarrow 1$ to $m - 1$ do
  $(C_i, S_i) \leftarrow \mathcal{E}.\mathrm{next}(K, S_{i-1}, M_i)$
$C_m \leftarrow \mathcal{E}.\mathrm{last}(K, S_{m-1}, M_m)$
return $(C_1, \ldots, C_m)$
```

```
algorithm $\boldsymbol{\mathcal{D}}(K, N, A, \boldsymbol{C})$
$c \leftarrow |\boldsymbol{C}|$
if $c = 0$ then return $\Lambda$
$(C_1, \ldots, C_m) \leftarrow \boldsymbol{C}$
$S_0 \leftarrow \mathcal{D}.\mathrm{init}(K, N, A)$
for $i \leftarrow 1$ to $c - 1$ do
  if $\mathcal{D}.\mathrm{next}(K, S_{i-1}, C_i) = \bot$ then
    if $c = 1$ return $\Lambda$
    else return $(M_1, \ldots, M_{i-1})$
  else $(M_i, S_i) \leftarrow \mathcal{D}.\mathrm{next}(K, S_{i-1}, C_i)$
if $\mathcal{D}.\mathrm{last}(K, S_{c-1}, C_c) = \bot$ then
  return $(M_1, \ldots, M_{c-1})$
$M_c \leftarrow \mathcal{D}.\mathrm{last}(K, S_{c-1}, C_c)$
return $(M_1, \ldots, M_c)$
```

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

*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-based view, where the segmentation of a plaintext is determined by the caller. In an online scheme, we expect each segment be processed using only a constant amount of memory, and we expect that only a constant amount of state is retained between the processing of segments. Ciphertext segments may be longer than corresponding plaintext segments.

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 $\mathcal{E} = (\mathcal{E}.\mathrm{init}, \mathcal{E}.\mathrm{next}, \mathcal{E}.\mathrm{last})$ and decryption $\mathcal{D} = (\mathcal{D}.\mathrm{init}, \mathcal{D}.\mathrm{next}, \mathcal{D}.\mathrm{last})$ are specified by triples of deterministic algorithms. Associated to $\Pi$ are sets $\mathcal{A}, \mathcal{N}, \mathcal{S} \subseteq \{0,1\}^*$ called the *AD space*, *nonce space*, and *state space* that appear in the signature of the components of $\mathcal{E}$ and $\mathcal{D}$:

$$\begin{array}{ll}
\mathcal{E}.\mathrm{init}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \to \mathcal{S} & \mathcal{D}.\mathrm{init}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \to \mathcal{S} \\
\mathcal{E}.\mathrm{next}\colon \mathcal{K} \times \mathcal{S} \times \{0,1\}^* \to \{0,1\}^* \times \mathcal{S} & \mathcal{D}.\mathrm{next}\colon \mathcal{K} \times \mathcal{S} \times \{0,1\}^* \to (\{0,1\}^* \times \mathcal{S}) \cup \{\bot\} \\
\mathcal{E}.\mathrm{last}\colon \mathcal{K} \times \mathcal{S} \times \{0,1\}^* \to \{0,1\}^* & \mathcal{D}.\mathrm{last}\colon \mathcal{K} \times \mathcal{S} \times \{0,1\}^* \to \{0,1\}^* \cup \{\bot\}
\end{array}$$

Denote by $\{0,1\}^{**} = (\{0,1\}^*)^*$ the set of *segmented-strings*: each component of a segmented-string is a string. The segmented-string with zero components is the empty list $\Lambda$. This is different from the empty string $\varepsilon$. The number of components in a segmented-string $\boldsymbol{M}$ is denoted $|\boldsymbol{M}|$, while the $i$th component of $\boldsymbol{M}$, $i \in [1..|\boldsymbol{M}|]$, is denoted $\boldsymbol{M}[i]$. Note that indexing begins at 1.

Given a segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ there are *induced* encryption and decryption algorithms $\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^{**} \to \{0,1\}^{**}$ (note the change to bold font) that operate, all at

```
initialize                                                    REAL2_Π
$K \twoheadleftarrow \mathcal{K}$

proc Enc($N, A, \boldsymbol{M}$)
if $N \notin \mathcal{N}$ or $A \notin \mathcal{A}$ then return $\perp$
return $\boldsymbol{\mathcal{E}}(K, N, A, \boldsymbol{M})$

proc Dec($N, A, \boldsymbol{C}$)
if $N \notin \mathcal{N}$ or $A \notin \mathcal{A}$ then return $\perp$
return $\boldsymbol{\mathcal{D}}(K, N, A, \boldsymbol{C})$
```

```
initialize                                                    IDEAL2_Π
for $(N, A) \in \mathcal{N} \times \mathcal{A}$ do $\pi_{N,A} \twoheadleftarrow \mathbf{Inj}(\tau)$

proc Enc($N, A, \boldsymbol{M}$)
if $N \notin \mathcal{N}$ or $A \notin \mathcal{A}$ then return $\perp$
return $\pi_{N,A}(\boldsymbol{M})$

proc Dec($N, A, \boldsymbol{C}$)
if $N \notin \mathcal{N}$ or $A \notin \mathcal{A}$ then return $\perp$
if $\exists \boldsymbol{M}$ s.t. $\pi_{N,A}(\boldsymbol{M}, 1) = \boldsymbol{C}$ then return $\boldsymbol{M}$
$\boldsymbol{M} \leftarrow$ a longest vector in
        $\{\boldsymbol{M} \colon \pi_{N,A}(\boldsymbol{M}, 0)[i] = \boldsymbol{C}[i]$ for $i \in [1..|\boldsymbol{M}| - 1]\}$
return $\boldsymbol{M}$
```

Fig. 4: **OAE2 security: coarse-grained definition.** The segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has nonce-space $\mathcal{N}$, AD-space $\mathcal{A}$, and segment-expansion $\tau$. It induces algorithms $\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}}$ as per Fig. 3. See the accompanying text for the definition of $\mathbf{Inj}(\tau)$.

once, on segmented plaintexts and segmented ciphertexts. These maps are defined in Fig. 3. Observe how Dec$(N, A, \boldsymbol{C})$ returns a longest $\boldsymbol{M}$ whose encryption (using $N$ and $A$) 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 $K \in \mathcal{K}$, $N \in \mathcal{N}$, $A \in \mathcal{A}$, $\boldsymbol{M} \in \{0,1\}^{**}$, and $\boldsymbol{C} = \boldsymbol{\mathcal{E}}(K, N, A, \boldsymbol{M})$, then $\boldsymbol{M} = \boldsymbol{\mathcal{D}}(K, N, A, \boldsymbol{C})$.

Ciphertext expansion. We focus on segmented-AE schemes with constant segment-expansion, defined as follows: associated to $\Pi$ is a number $\tau \geq 0$ such that if $K \in \mathcal{K}$, $N \in \mathcal{N}$, $A \in \mathcal{A}$, $\boldsymbol{M} = (M_1, \ldots, M_m) \in \{0,1\}^{**}$, and $\boldsymbol{C} = (C_1, \ldots, C_m) = \boldsymbol{\mathcal{E}}(K, N, A, \boldsymbol{M})$, then $|C_i| = |M_i| + \tau$ for all $i \in [1..m]$. In words: each segment grows by $\tau$ bits, for some constant $\tau$. We call $\tau$ the *segment-expansion* of $\Pi$.

We favor constant segment-expansion because we think it runs contrary to the spirit of online-AE to furnish interior segments with an inferior authenticity guarantee than that afforded to the whole message. After all, much of the point of online-AE is to allow a decrypting party to safely act on $C_1 \cdots C_i$ as soon as $C_i$ completes; the decrypting party deserves proper assurance that, so far, authenticity looks fine. 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.

Security: coarse-grained definition. We shall give two, essentially equivalent definitions for online-AE. The first definition is more coarse-grained; the second, more fine-grained. We begin with the former, and with notation to be used throughout.

For $\boldsymbol{X}, \boldsymbol{Y} \in \{0,1\}^{**}$, let $\boldsymbol{X} \,\|\, \boldsymbol{Y} \in \{0,1\}^{**}$ be the components of $\boldsymbol{X}$, in order, followed by those of $\boldsymbol{Y}$, again in order. To concatenate a vector of strings and a string, treat the latter as a length-1 vector. Now consider a function $\pi\colon \{0,1\}^{**} \times \{0,1\} \to \{0,1\}^{**}$. We say that $\pi$ is $\tau$-*expanding* if $|\pi(\boldsymbol{X}, \delta)| = |\boldsymbol{X}|$ and $|(\pi(\boldsymbol{X}, \delta))[i]| = |\boldsymbol{X}[i]| + \tau$ for all $\boldsymbol{X} \in \{0,1\}^{**}$, $\delta \in \{0,1\}$, $m = |\boldsymbol{X}|$, and $i \in [1..m]$. We say that $\pi$ is *accretive* if $\pi(\boldsymbol{X}, 0) = \pi(\boldsymbol{X} \,\|\, \boldsymbol{Y}, 0)[1, m]$ for all $\boldsymbol{X} \in \{0,1\}^{**}$, $m = |\boldsymbol{X}|$. Finally, we say that $\pi$ is *sealing* if $\pi(\boldsymbol{X}, 0)[i] = \pi(\boldsymbol{X}, 1)[i]$ for all $\boldsymbol{X} \in \{0,1\}^{**}$, $m = |\boldsymbol{X}|$, and $i \in [1..m-1]$. Let $\mathbf{Inj}(\tau)$ denote the set of all injective, $\tau$-expanding, accretive, sealing functions from and to $\{0,1\}^{**}$. This set is endowed with the uniform distribution in the natural way. We write $\pi \twoheadleftarrow \mathbf{Inj}(\tau)$ to draw a random sample according to this distribution.

Let us explain what a random element $\pi \twoheadleftarrow \mathbf{Inj}(\tau)$ looks like by giving a recursive definition for how one could sample from it. First set $\pi(\Lambda, 0) = \Lambda = \pi(\Lambda, 1)$. Now, for increasing $m = 1, 2, \ldots$, independently for each $\boldsymbol{X} \in (\{0,1\}^*)^m$, select a pair of random $\tau$-expanding injective functions $f, \sigma : \{0,1\}^* \to \{0,1\}^*$ and extend $\pi$ by asserting that $\pi(\boldsymbol{X} \,\|\, Y, 0) = \pi(\boldsymbol{X}, 0) \,\|\, f(Y)$ and $\pi(\boldsymbol{X} \,\|\, Y, 1) = \pi(\boldsymbol{X}, 0) \,\|\, \sigma(Y)$.

Fig. 4 defines games $\mathbf{REAL2}_{\Pi}$ and $\mathbf{IDEAL2}_{\Pi}$ for a $\tau$-expanding segmented-AE scheme $\Pi$. Given an adversary $\mathscr{A}$ with access to oracles Enc and Dec determined by these games, define $\mathbf{Adv}_{\Pi}^{\mathrm{OAE2}}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{REAL2}_{\Pi}} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{IDEAL2}_{\Pi}} \Rightarrow 1]$ as the adversary's distinguishing advantage. This is the "coarse-grained" definition 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 $\boldsymbol{C} = (C_1, \ldots, C_m)$ under the control of $K, N, A$. He wants to do this *as well as possible* subject to the constraint that segments grow by exactly $\tau$ bits and $M_1 \cdots M_i$ must be recoverable from $K, N, A, C_1 \cdots C_i$. As with robust-AE [30], 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 $\mathbf{Inj}(\tau)$ and its inverse, where inversion is understood to invert as many components as possible, stopping at the first point where one cannot proceed.

The definition of $\mathbf{Inj}(\tau)$ 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 $M' = (A, B, C)$. This should 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 $\mathbf{Inj}(\tau)$, whose member functions get a second argument $\delta$ with semantics indicating if the first argument is *complete.* So $\pi((A, B, C), 0)$ is what $\boldsymbol{M} = (A, B, C)$ should map to if more segments are coming, while $\pi((A, B, C), 1)$ is what it should map to if $C$ is the final segment of $\boldsymbol{M}$.

Fine-grained formulation. Fig. 5 give a more fine-grained model for OAE. The adversary, instead of providing a vector $\boldsymbol{M}$ and getting a vector $\boldsymbol{C} = \mathrm{Enc}(N, A, \boldsymbol{M})$, can adaptively grow $\boldsymbol{M}$ one component at a time. Similarly, instead of providing a segmented ciphertext $\boldsymbol{C}$ and getting $\boldsymbol{M} = \mathrm{Dec}(N, A, \boldsymbol{C})$, it can adaptively grow $\boldsymbol{C}$. As before, we associate to a $\tau$-expanding segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and an adversary $\mathscr{A}$ the real number $\mathbf{Adv}_{\Pi}^{\mathrm{oae2}}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{Real2}_{\Pi}} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{Ideal2}_{\Pi}} \Rightarrow 1]$ that is its distinguishing advantage.

The coarse-grained and fine-grained definitions are essentially equivalent; when speaking of OAE2 security, we mean either one. 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 OAE2-style adversary $\mathscr{A}$ we can construct an equally effective oae2-style adversary $\mathscr{B}$: it translates each $\mathrm{Enc}(N, A, M_1, \ldots, M_m)$ 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 $\mathrm{Dec}(N, A, C_1, \ldots, C_m)$ calls into Dec.init, Dec.next, Dec.last calls. Adversary $\mathscr{B}$ gets exactly the oae2-advantage that $\mathscr{A}$ had as OAE2-advantage. It runs in almost the exact same time.

Simulation in the other direction is less efficient. Given an adversary $\mathscr{A}$ attacking the oae2-security of a $\Pi$, we construct an adversary $\mathscr{B}$ for attacking the OAE2-security of the same scheme. Adversary $\mathscr{B}$ maintains lists $N_i, A_i, \boldsymbol{M}_i$ 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$(i, M)$, when valid, result in appending $M$ to $\boldsymbol{M}_i$, making an $\mathrm{Enc}(N_i, A_i, \boldsymbol{M}_i \parallel \varepsilon)$ call, and returning its $|\boldsymbol{M}_i|$-th component. Calls of the form Enc.last$(i, M)$ result in making an $\mathrm{Enc}(N_i, A_i, \boldsymbol{M}_i \parallel M)$ call, returning its last component, resetting $\boldsymbol{M}_i$ to $\perp$ before doing so. Calls of the form Dec.init, Dec.next, and Dec.last are treated analogously, maintaining $N_i', A_i', \boldsymbol{C}_i$ values. Once again the simulation is perfect, so $\mathbf{Adv}_{\Pi}^{\mathrm{OAE2}}(\mathscr{B}) = \mathbf{Adv}_{\Pi}^{\mathrm{oae2}}(\mathscr{A})$. 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 coarse-grained definition is more compact, the improved concision for the adversary's queries in the fine-grained definition make it preferable—particularly as this concision better models the

```
initialize                                                        Real2_Π
    $I, J \leftarrow 0$
    $K \twoheadleftarrow \mathcal{K}$

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

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

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

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

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

proc Dec.last$(j, C)$
    if $j \notin [1..J]$ or $S'_j = \perp$ then return $\perp$
    $M \leftarrow \mathcal{D}.\mathrm{next}(K, S'_j, C)$
    $S'_j \leftarrow \perp$;  return $M$
```

```
initialize                                                        Ideal2_Π
    $I, J \leftarrow 0$
    for $(N, A) \in \mathcal{N} \times \mathcal{A}$ do $\pi_{N,A} \twoheadleftarrow \mathbf{Inj}(\tau)$

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

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

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

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

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

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

Fig. 5: **OAE2 security, re-expressed in a more granular manner.** The segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has nonce-space $\mathcal{N}$, AD-space $\mathcal{A}$, and segment-expansion $\tau$.

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

There are a couple of further reasons to favor the fine-grained formulation. One is that it more directly captures the possibility of "infinite" (non-terminating) plaintexts (also called "streams"). This is simply the setting where Enc.last and Dec.last are never called. Second, the fine-grained notion makes it easier to define nonce-respecting adversaries for the OAE setting. Such an adversary may adaptively grow a plaintext based on a single nonce, but it may grow only *one* plaintext for any given nonce. In the coarse-grained formulation this would be awkward to say; in the fine-grained formulation, it is natural. We will return to this Section 7.

Online computability. We have yet to speak of the memory requirements for an OAE scheme. We say that a segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has *online-encryption* if there are constants $s$ and $w$ such that $\mathcal{S} = \{0,1\}^{\leq s}$ and both $\mathcal{E}$.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 $\Pi$ has *online-decryption* if there are constants $s$ and $w$ such that $\mathcal{S} = \{0,1\}^{\leq s}$

and $\mathcal{D}$.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 underlying 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 again has nothing to do with the working memory $w$.

Our security definitions don't actually care if a segmented-AE scheme is online: these are efficiency requirements, not security requirements, so they don't enter into the definition. Yet the whole point of the segmented-AE syntax is to properly deal with schemes that have such efficiency constraints.

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, $\sigma$, for plaintext components other than the last, and a different expansion value, $\tau$, at the end. For such a $(\sigma, \tau)$-expanding scheme, the set $\mathbf{Inj}(\tau)$ that served as a reference object would be replaced by a set $\mathbf{Inj}(\sigma, \tau)$ whose members $\pi$ would now have to be $(\sigma, \tau)$-expanding: $|\pi(\boldsymbol{X}, \delta)| = |\boldsymbol{X}|$ and $|\pi(\boldsymbol{X}, 0)[i]| = |\boldsymbol{X}[i]|+\sigma$ for all $\boldsymbol{X} \in \{0,1\}^{**}$, $m = |\boldsymbol{X}|$, and $i \in [1..m-1]$, while $|\pi(\boldsymbol{X}, 1)[m]| = |\boldsymbol{X}[m]|+\tau$. Of course members of $\mathbf{Inj}(\sigma, \tau)$ would still need to be injective, accretive, and sealing, as before.

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 *not* be the same: 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 bandwidth; obviously one will use fewer 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 bandwidth 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.

## 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) [48]. The notion is close to MRAE [48], with a gap $q^2/2^{s+\tau} + q^2/2^s$ 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 [48] if $\tau$ is large, say $\tau = 128$, or the recent AEZ scheme [30], for arbitrary $\tau$. 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) $\mathbf{D} : \mathbf{K} \times \mathcal{N} \times \mathcal{A} \times \{0,1\}^* \to \{0,1\}^* \cup \{\bot\}$ 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 $\tau$ associated to $\Pi$, 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 $\mathbf{Adv}_{\Pi}^{\mathrm{pri}}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{RealPRI}_\Pi} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{IdealPRI}_\Pi} \Rightarrow 1]$ using Fig. 6'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 $i$th bit to the $j$th bit (inclusive). For a string $X$ such that $|X| < n$, we write $X10^*$ to denote $X \,\|\, 10^{n-1-|X|}$. Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be a conventional AE scheme of ciphertext-expansion $\tau$, nonce-space $\{0,1\}^*$, and AD-space $\{0,1\}^*$. Fig. 7 defines a segmented-AE

```
initialize                                   RealPRI_Π
    $K \twoheadleftarrow \mathbf{K}$

proc Enc($N, A, M$)
    if $N \notin \mathcal{N}$ or $A \notin \mathcal{A}$ then return $\bot$
    return $\mathbf{E}(K, N, A, M)$

proc Dec($N, A, C$)
    if $N \notin \mathcal{N}$ or $A \notin \mathcal{A}$ then return $\bot$
    return $\mathbf{D}(K, N, A, C)$
```

```
initialize                                   IdealPRI_Π
    for $(N, A) \in \mathcal{N} \times \mathcal{A}$ do $\rho_{N,A} \twoheadleftarrow \mathrm{Inj}(\tau)$

proc Enc($N, A, M$)
    if $N \notin \mathcal{N}$ or $A \notin \mathcal{A}$ then return $\bot$
    return $\rho_{N,A}(M)$

proc Dec($N, A, C$)
    if $N \notin \mathcal{N}$ or $A \notin \mathcal{A}$ then return $\bot$
    return $\rho^{-1}_{N,A}(C)$
```

Fig. 6: **PRI security.** Defining security for an AE scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ with expansion $\tau$, 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 $f^{-1}(y) = \bot$ otherwise.

scheme $\mathbf{CHAIN}[\Pi, n] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with segment expansion $\tau$, nonce space $\{0,1\}^*$, AD space $\{0,1\}^*$, and state space $\{0,1\}^n$. The proof of the following theorem is in Appendix E.

**Theorem 1.** Let $\Pi$, $n$, and $\mathbf{CHAIN}[\Pi, n]$ be as above. There is an explicit given reduction $R$ with the following property. For any adversary $\mathscr{A}$, adversary $\mathscr{B} = R(\mathscr{A})$ satisfies $\mathbf{Adv}^{\mathrm{oae2}}_{\mathbf{CHAIN}[\Pi,n]}(\mathscr{A}) \le \mathbf{Adv}^{\mathrm{pri}}_{\Pi}(\mathscr{B}) + 2q^2/2^n$ where $q$ is the number of segments in $\mathscr{A}$'s queries. Adversary $\mathscr{B}$ uses about the same running time as $\mathscr{A}$ and the total length of $\mathscr{B}$'s queries is the total length of $\mathscr{A}$'s queries, plus at most $8qn$ bits.

Discussion. In $\mathcal{E}$.next and $\mathcal{D}$.next, the state is computed via $M[1,n] \oplus C[1,n]$. We remark that 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 $\mathbf{CHAIN1}[\Pi, n]$. The method is insecure for small $\tau$. Here is an attack for the case $\tau = 0$. The adversary makes a single query $(N, A, \boldsymbol{C})$ to the decryption oracle, where $N$ and $A$ are arbitrary and $\boldsymbol{C} = (0^n, 0^n, 0^n)$. Let the answer be $\boldsymbol{M} = (M_1, M_2, M_3)$. The adversary will output 1 only if $M_2 = M_3$. In the **Ideal2** game the strings $M_2$ and $M_2$ are independent random strings. However, in game **Real2** we always have $M_2 = M_3 = \mathbf{D}_K^{0^n,\varepsilon}(0^n)$. Hence the adversary can win with advantage $1 - 2^{-n}$. In contrast to the above, for large $\tau$, the scheme $\mathbf{CHAIN1}[\Pi, n]$ *is* OAE2 secure.

To achieve OAE2 with multivalued segment-expansion, use an RAE-secure underlying scheme [30], a generalization of PRI that allows one to select an arbitrary ciphertext-expansion for each query. The construction is modified in the natural way.

## 7 Nonce-Based OAE

The CPSS attack applies to OAE2-secure schemes as much as to OAE1-secure schemes, demonstrating that, even for OAE2, nonces must not repeat. That is, while OAE2 defines what must happen when nonces repeat, absent a contravening analysis for some application, nonce-repetition must still be deprecated. This section thus asks and answers the question: *once we require that an adversary not repeat a nonce, can an OAE scheme be made better than what is specified in Section 6?*

We will answer affirmatively by showing that nonce-based OAE2, which we'll call nOAE, can be achieved with a simpler and more efficient tool: the underlying AE scheme we will use need only be nAE secure, not PRI or MRAE secure. Thus a scheme like OCB will work fine. This statement assumes that the segment-expansion $\tau$ is fairly large, like $\tau = 128$. We begin with the target definition.

Defining nonce-based OAE. One can either forbid adversaries from repeating nonces to encryption queries or one can make such repetitions pointless. We do the latter. Returning to Fig. 5, modify games

```
proc $\mathcal{E}$.init$(K, N, A)$                                   $\mathcal{E}$ algorithms
$V \leftarrow \mathbf{E}_K(N, 1^n \parallel A, 0^n)$; return $V[1, n]$

proc $\mathcal{E}$.next$(K, S, M)$
$C \leftarrow \mathbf{E}_K(S, \varepsilon, M)$
if $|M| \geq n$ then $S' \leftarrow C[1, n] \oplus M[1, n]$
else
  $V \leftarrow \mathbf{E}_K(S, 010^*, M10^*)$; $S' \leftarrow V[1, n]$
return $(C, S')$

proc $\mathcal{E}$.last$(K, S, M)$
return $\mathbf{E}_K(S, 0^n, M)$
```

```
proc $\mathcal{D}$.init$(K, N, A)$                                   $\mathcal{D}$ algorithms
$V \leftarrow \mathbf{E}_K(N, 1^n \parallel A, 0^n)$; return $V[1, n]$

proc $\mathcal{D}$.next$(K, S, C)$
$M \leftarrow \mathbf{D}_K(S, \varepsilon, C)$
if $M = \perp$ then return $\perp$
if $|M| \geq n$ then $S' \leftarrow C[1, n] \oplus M[1, n]$
else
  $V \leftarrow \mathbf{E}_K(S, 010^*, M10^*)$; $S' \leftarrow V[1, n]$
return $(M, S')$

proc $\mathcal{D}$.last$(K, S, C)$
return $\mathbf{D}_K(S, 0^n, C)$
```

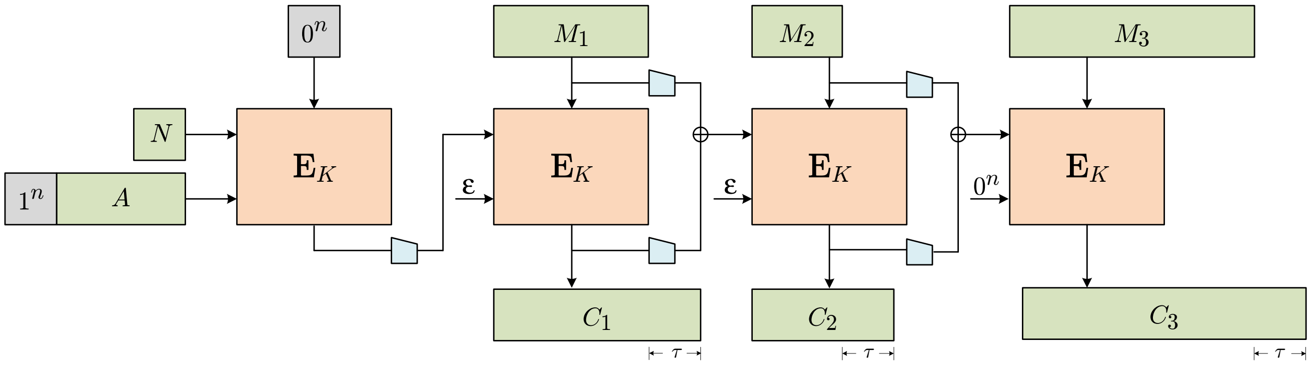

Fig. 7: **The CHAIN construction for OAE2. Top:** Encryption scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$, secure as a PRI with ciphertext-expansion $\tau$, is turned into a segmented-AE scheme $\mathbf{CHAIN}[\Pi, n] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with $\mathcal{K} = \mathbf{K}$. **Bottom:** Illustration of segmented encryption where each segment of $(M_1, M_2, M_3)$ has at least $n$ bits. The trapezoid represents truncation to $n$ bits.

**Real2** and **Ideal2** to make games **nReal** and **nIdeal** as follows. To the **initialize** procedure of both games, add in the statement $\mathscr{N} \leftarrow \emptyset$. To the Enc.init procedure of each game add in the statement: "**if** $N \in \mathscr{N}$ **then return** $\perp$ **else** $\mathscr{N} \leftarrow \mathscr{N} \cup \{N\}$" as the first or second line. With the games so modified, associate to a $\tau$-expanding segmented-AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and an adversary $\mathscr{A}$ the real number $\mathbf{Adv}_{\Pi}^{\text{noae}}(\mathscr{A}) = \Pr[\mathscr{A}^{\mathbf{nReal}_\Pi} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{nIdeal}_\Pi} \Rightarrow 1]$. In English: nOAE security is identical to OAE2 security except that the adversary may only explore one encryption "chain" for any given nonce. No such limits are placed on the decryption side of things.

Below we show how to combine an nAE scheme and a PRF to produce an nOAE-secure segmented-AE scheme. We must first recall the classic nAE and PRF notions.

Preliminaries. For a conventional AE scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$, its nAE security against an adversary $\mathscr{A}$ is defined as $\mathbf{Adv}_{\Pi}^{\text{nae}}(\mathscr{A}) = \Pr[K \twoheadleftarrow \mathbf{K} \colon \mathscr{A}^{\mathcal{E}_K(\cdot,\cdot,\cdot), \mathcal{D}_K(\cdot,\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathscr{A}^{\$(\cdot,\cdot,\cdot), \perp(\cdot,\cdot,\cdot)} \Rightarrow 1]$, 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.

For a function $F \colon \mathbf{K} \times \{0,1\}^* \to \{0,1\}^n$, its PRF security against an adversary $\mathscr{A}$ is defined as $\mathbf{Adv}_{F}^{\text{prf}}(\mathscr{A}) = \Pr[K \twoheadleftarrow \mathbf{K} \colon \mathscr{A}^{F_K(\cdot)} \Rightarrow 1] - \Pr[\mathscr{A}^{\$\$(\cdot)} \Rightarrow 1]$ where $\$\$(\cdot)$ is an oracle that, on each input, returns a random $n$-bit output. The adversary is prohibited from repeating a prior query.

Achieving nonce-based OAE. We now describe the **STREAM** construction to turn an nAE scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ into an nOAE-secure segmented-AE scheme $\mathbf{STREAM}[F, \Pi]$. This construction, as shown in Fig. 8, is based on an nAE scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ and a PRF $F \colon \mathbf{K}_0 \times \{0,1\}^* \to \{0,1\}^n$.

```
proc $\mathcal{E}$.init$(K, N, A)$                                $\mathcal{E}$ algorithms
$(K_0, K_1) \leftarrow K;\ V \leftarrow F_{K_0}(A)$
return $(N, V, 1)$

proc $\mathcal{E}$.next$(K, S, M)$
$(K_0, K_1) \leftarrow K;\ (N, V, i) \leftarrow S$
$C \leftarrow \mathbf{E}_{K_1}(\langle 0, N, i\rangle, V, M);\ S' \leftarrow (N, \varepsilon, i+1)$
return $(C, S')$

proc $\mathcal{E}$.last$(K, S, M)$
$(K_0, K_1) \leftarrow K;\ (N, V, i) \leftarrow S$
return $\mathbf{E}_{K_1}(\langle 1, N, i\rangle, V, M)$
```

```
proc $\mathcal{D}$.init$(K, N, A)$                                $\mathcal{D}$ algorithms
$(K_0, K_1) \leftarrow K;\ V \leftarrow F_{K_0}(A)$
return $(N, V, 1)$

proc $\mathcal{D}$.next$(K, S, C)$
$(K_0, K_1) \leftarrow K;\ (N, V, i) \leftarrow S;\ M \leftarrow \mathbf{D}_{K_1}(\langle 0, N, i\rangle, V, C)$
if $M = \bot$ then return $\bot$
$S' \leftarrow (N, \varepsilon, i+1)$; return $(M, S')$

proc $\mathcal{D}$.last$(K, S, M)$
$(K_0, K_1) \leftarrow K;\ (N, V, i) \leftarrow S$
return $\mathbf{D}_{K_1}(\langle 1, N, i\rangle, V, C)$
```

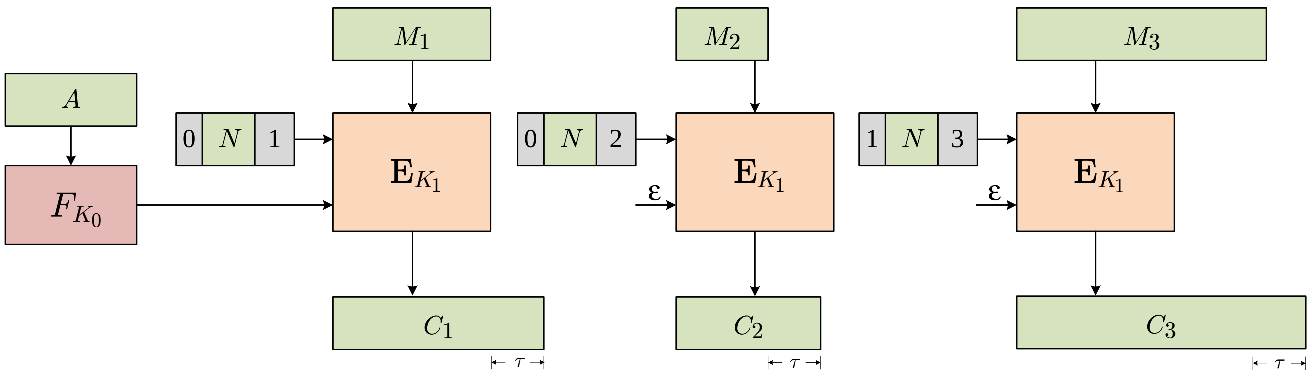

Fig. 8: **The STREAM construction for nOAE.** Encryption scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ secure as an nAE with ciphertext expansion $\tau$ is turned into a segmented-AE scheme $\Pi'' = (\mathcal{K}, \mathcal{E}, \mathcal{D}) = \mathbf{STREAM}[F, \Pi]$ with key space $\mathcal{K} = \mathbf{K}_0 \times \mathbf{K}$, where $F\colon\ \mathbf{K}_0 \times \{0,1\}^* \to \{0,1\}^n$ is a PRF. Here $\langle b, N, i\rangle$ is a string encoding of the triple $(b, N, i)$.

Theorem 2 below shows that $\mathbf{STREAM}[F, \Pi] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ achieves nOAE security. The proof is in Appendix F.

**Theorem 2.** Let $\Pi$ be a conventional AE scheme with ciphertext expansion $\tau$ and let $F\colon \mathbf{K}_0 \times \{0,1\}^* \to \{0,1\}^n$ be a PRF. There are explicitly given reductions $R_0$ and $R_1$ with the following property. For any adversary $\mathscr{A}$, adversaries $\mathscr{B}_0 = R_0(\mathscr{A})$ and $\mathscr{B}_1 = R_1(\mathscr{A})$ satisfy $\mathbf{Adv}^{\mathrm{noae}}_{\mathbf{STREAM}[F,\Pi]}(\mathscr{A}) \leq \mathbf{Adv}^{\mathrm{prf}}_{F}(\mathscr{B}_0) + \mathbf{Adv}^{\mathrm{nae}}_{\Pi}(\mathscr{B}_1) + q^2/2^{n+1} + q/2^{\tau}$ where $q$ is the number of segments in $\mathscr{A}$'s queries. Adversary $\mathscr{B}_0$ uses about the same running time as $\mathscr{A}$ plus the time to encrypt $\mathscr{A}$'s queries, and the total length of $\mathscr{B}_0$ queries is the total length of $\mathscr{A}$'s AD. Adversary $\mathscr{B}_1$ uses about the same running time as $\mathscr{A}$ and the total length of $\mathscr{B}_1$'s queries is the total length of $\mathscr{A}$'s queries, plus at most $q(\tau + \ell)$ bits, where $\ell$ bounds the length of the nonces of $\mathcal{E}$.

## 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 [48] never suggested that nonce-reuse was OK; it said that an MRAE scheme must do "as well as possible with whatever IV is provided" [48, 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" [48, p. 12–13].

| |
|---|
| **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 $\tau$-bit function of the nonce, AD, and plaintext. **Schemes1:** COPA [7], Deoxys [33], Joltik [34], KIASU [35], Marble [29], McOE [25], SHELL [56], POET [1], Prøst-COPA [18] **Schemes2:** ++AE [44] |
| **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 $\tau$-bit function of the nonce, AD, and plaintext. **Schemes1:** APE [4], ELmD [22], ELmE [23], Prøst-APE [18] |
| **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 $\tau$-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 [10] |
| **OAE1c** Leaks equality of any blocks at the same position. E.g., if ciphertexts $C$ and $C'$ arise from 4-block plaintexts $P = \mathrm{A} \parallel \mathrm{B} \parallel \mathrm{C} \parallel \mathrm{D}$ and $P' = \mathrm{E} \parallel \mathrm{B} \parallel \mathrm{F} \parallel \mathrm{D}$ then $C_2 = C'_2$ and $C_4 = C'_4$. Security is formalized by comparing $\mathcal{E}_K$ with: a function from $n$ bits to $n$ bits tweaked by the nonce and an integer, the position; followed by a random tag. **Schemes1:** Minalpher [52] |
| **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 $P = \mathrm{A} \parallel \mathrm{B} \parallel \mathrm{C} \parallel \mathrm{D}$ and $P' = \mathrm{A} \parallel \mathrm{B} \parallel \mathrm{E} \parallel \mathrm{F}$ 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 $P = \mathrm{A} \parallel \mathrm{B} \parallel \mathrm{C} \parallel \mathrm{D}$ and $P' = \mathrm{E} \parallel \mathrm{F} \parallel \mathrm{G} \parallel \mathrm{H}$ will have $C_1 \oplus C'_1 = \mathrm{A} \oplus \mathrm{E}$. **Schemes2:** Artemia [3] CBEAM [50], ICEPOLE [43], iFeed [59], Jambu [57], Keyak [17], MORUS [58], NORX [12], STRIBOB [51] |
| **OAE0a** 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 [31], SILC [32] |
| **OAE0b** 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 [12], Trivia-ck [21] **Schemes2:** OTR [41] |

Fig. 9: **A menagerie of OAE notions and schemes.** All of the schemes are CAESAR submissions except ElmE and McOE. **Schemes1** lists proposals that claim some flavor of nonce-reuse misuse resistance. **Schemes2** lists proposals that didn't, yet are or were marked as such in the AE Zoo [13] or AFL survey [2].

The FFL paper indicates that the authors wish "to achieve both simultaneously: security against nonce-reusing adversaries ... and support for on-line-encryption" [25, 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" [25, 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" [7, 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" [5, p. 9]. The authors add that "Often a deterministic AE scheme does not even have an IV input" [5, 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 [29, slide 2] and Lauridsen [19, Slides 4, 6].

We have thus seen a transformation in language, played out over eight years, taking us from a strong definition (MRAE) pitched as trying to capture the best one can do when a nonce gets reused to a comparatively weak definition (OAE1) nowadays pitched as being so strong so as to render nonces superfluous. This is a dramatic shift. 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" [6, p. 7]. Identical language is found in the COBRA submission [9, p. 7]. In our view, such claims are wrong; there would seem to be a significant gap between OAE1 and OAE2 security.

Weaker notions. Concurrent with the rhetoric for what OAE1 delivers being ratcheted up, weakened variants of OAE1 have proliferated. We document this trend in Fig. 9, which introduces a variety of OAE notions. They are all weaker than OAE1 except for OAE1a; by standard arguments, OAE1 and OAE1a are quantitatively close if the blocksize is reasonably large. In this race to the bottom, it may seem as though the scheme comes first and whatever properties it provides is branded as *some* form misuse resistance.

The number of different OAE definitions, and their precise meanings, has never been clear. The evolution of what's been indicated in the "Nonce-MR" column of the AE Zoo [13] illustrates the struggle of researchers doing their best at a nearly impossible task: trying to accurately summarize the extent of nonce-reuse misuse-resistance for tens of AE schemes. Our own attempt at sorting this out, Fig. 9, is far from 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 comments or corrections from Farzaneh Abed, Nasour Bagheri, Dan Bernstein, Danilo Gligoroski, Stefan Lucks, and Samuel Neves.

Some 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. F. Abed, S. Fluhrer, C. Forler, E. List, S. Lucks, D. McGrew, J. Wenzel. Pipelineable On-Line Encryption. Cryptology ePrint report 2014/591 (2014)
2. F. Abed, C. Forler, S. Lucks. General Overview of the First-Round CAESAR Candidates for Authenticated Encryption. Cryptology ePrint report 2014/792 (2014)
3. J. Alizadeh, M. R. Aref, N. Bagheri. Artemia v1. CAESAR submission (2014)
4. E. Andreeva, B. Bilgin, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha, K. Yasuda. APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In: *FSE 2014*. LNCS xxxx, Springer (2015)
5. E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha, K. Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. In: *ASIACRYPT (1) 2014*. LNCS 8873, pp. 105–125. Springer (2015)
6. E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, K. Yasuda. AES-COPA v.1. CAESAR submission (2014)
7. E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, K. Yasuda. Parallelizable and Authenticated Online Ciphers. In: *ASIACRYPT 2013*. LNCS 8269, pp. 424–443. Springer (2013)
8. E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, K. Yasuda. Parallelizable (Authenticated) Online Ciphers. DIAC presentation (2013)
9. E. Andreeva, A. Luykx, B. Mennink, K. Yasuda. AES-COBRA v1. CAESAR submission (2014)
10. E. Andreeva, A. Luykx, B. Mennink, K. Yasuda. COBRA: A Parallelizable Authenticated Online Cipher without Block Cipher Inverse. In: *FSE 2014*. LNCS xxxx, Springer (2014)
11. Anonymous. Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures. `https://web.archive.org/web/20120630143111/http://www.openssl.org/~bodo/tls-cbc.txt`.
12. J. P. Aumasson, P. Jovanovic, S. Neves. NORX v1. CAESAR submission (2014)
13. Authenticated Encryption Zoo, `https://aezoo.compute.dtu.dk`

14. M. Bellare, A. Boldyreva, A. Knudsen, C. Namprempre. Online Ciphers and the Hash-CBC Construction. In: *CRYPTO 2001*. LNCS 2139, pp. 292–309. Springer (2001)
15. M. Bellare, P. Rogaway. Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: *ASIACRYPT 2000*. LNCS 1976, pp. 317–330. Springer (2000)
16. D. Bernstein. Cryptographic competitions: CAESAR. `http://competitions.cr.yp.to`
17. G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, R. V. Keer. CAESAR submission: Keyak v1. CAESAR submission (2014)
18. E. Bilge Kavun, M. Lauridsen, G. Leander, C. Rechberger, P. Schwabe, T. Yalçın. Prøst v1.1. CAESAR submission (2014)
19. A. Bogdanov, M. Lauridsen, E. Tischhauser. AES-Based Authenticated Encryption Modes in Parallel High-Performance Software. DIAC presentation (2014)
20. A. Boldyreva, N. Taesombut. Online Encryption Schemes: New Security Notions and Constructions. In: *CT-RSA 2014*. LNCS 2964, pp. 1–14, Springer (2004). Full version on the first authors' webpage.
21. A. Chakraborti, M. Nandi. TriviA-ck-v1. CAESAR submission. (2014)
22. N. Datta, M. Nandi. ELmD v1.0. CAESAR submission. (2014)
23. N. Datta, M. Nandi. ELmE: A Misuse Resistant Parallel Authenticated Encryption. In: ACISP 2014. LNCS 8544, pp. 306–321. Springer (2014)
24. T. Duong, J. Rizzo. Here Come The $\oplus$ Ninjas. Manuscript (2011).
25. E. Fleischmann, C. Forler, S. Lucks. McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In: *FSE 2012*. LNCS 7549, pp. 196–215. Springer (2012)
26. E. Fleischmann, C. Forler, S. Lucks, J. Wenzel. McOE: A Foolproof On-line Authenticated Encryption Scheme. Full version of [25]. Cryptology ePrint report 2011/644 (2013)
27. P.-A. Fouque, A. Joux, G. Martinet, F. Valette. Authenticated On-Line Encryption. In: *SAC 2003*. LNCS 3006, pp. 145–159, Springer (2003)
28. P.-A. Fouque, G. Martinet, G. Poupard. Practical Symmetric On-line Encryption. In: *FSE 2003*. LNCS 3006, pp.145–159. Springer (2003)
29. J. Guo. Marble Specification Version 1.0. CAESAR submission (2014). Also DIAC presentation (2014)
30. V.T. Hoang, T. Krovetz, P. Rogaway. Robust authenticated encryption: AEZ and the problem that it solves. Cryptology ePrint report 2014/793.
31. T. Iwata, K. Minematsu, J. Guo, S. Morioka. CLOC: Compact Low-Overhead CFB. CAESAR submission. (2014)
32. T. Iwata, K. Minematsu, J. Guo, S. Morioka, E. Kobayashi. SILC: SImple Lightweight CFB. CAESAR submission. (2014)
33. J. Jean, I. Nikolić, T. Peyrin. Deoxys v1. CAESAR submission. (2014)
34. J. Jean, I. Nikolić, T. Peyrin. Joltik v1. CAESAR submission. (2014)
35. J. Jean, I. Nikolić, T. Peyrin. KIASU v1. CAESAR submission. (2014)
36. A. Joux, G. Martinet, F. Valette. Blockwise Adaptive Attakers: Revisiting the (In)seurity of Some Provably Secure Encryption Modes: CBC, GEM, IACBC. In: *Crypto 2002*. LNCS 2442, pp. 17–30. Springer (2002)
37. J. Katz, M. Yung: Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In: *FSE 2000*. LNCS 1978, pp. 284–299, Springer (2001)
38. M. Liskov, R. Rivest, D. Wagner. Tweakable Block Ciphers. *J. of Cryptology*, 24(3), pp. 588–6143. Springer (2011)
39. S. Lucks. Personal communication (2014)
40. W. Miaw. Netflix / msl. `https://github.com/Netflix/msl/wiki`. June 10, 2014. Visited 2015.02.18.
41. K. Minematsu. AES-OTR v1. CAESAR submission (2014)
42. K. Minematsu. Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. Cryptology ePrint Archive, Report 2013/628 (2013)
43. P. Morawiecki, K. Gaj, E. Homsirikamol, K. Matusiewicz, J. Pieprzyk, M. Rogawski, M. Srebrny, M. Wójcik. ICEPOLE v1. CAESAR submission (2014)
44. F. Recacha. ++AE v1.0. CAESAR submission (2014)
45. P. Rogaway. Authenticated-Encryption with Associated-Data. In: *ACM CCS 2002*, pp. 98–107 (2002)
46. P. Rogaway. Problems with Proposed IP Cryptography. Manuscript (1995)
47. P. Rogaway, M. Bellare, J. Black, T. Krovetz: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In: *ACM CCS 2001*, pp. 196–205 (2001)
48. P. Rogaway, T. Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In: *EUROCRYPT 2006*. LNCS 4004, pp. 373–390. Springer (2006)
49. P. Rogaway, H. Zhang. Online Ciphers from Tweakable Blockciphers. In: *CT-RSA 2011*. LNCS 6558, pp 237-249. Springer (2011)
50. M.-J. O. Saarinen. The CBEAMr1 Authenticated Encryption Algorithm. CAESAR submission (2014)
51. M.-J. O. Saarinen. The STRIBOBr1 Authenticated Encryption Algorithm. CAESAR submission (2014)
52. Y. Sasaki, Y. Todo, K. Aoki, Y. Naito, T. Sugawara, Y. Murakami, M. Matsui, S. Hirose. Minalpher v1. CAESAR submission (2014)
53. S. Touset. Streaming API to Authenticated Encryption. Cryptography Stack Exchange (Jan 16, 2013). Also see response by "fgrieu" (Jan 23, 2013)
`http://crypto.stackexchange.com/questions/6008`

54. P. Tsang, R. Solomakhin, S. Smith. Authenticated Streamwise On-line Encryption. Dartmouth Computer Science Technical Report TR2009-640. (2009)
55. S. Vaudenay. CBC padding: Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS, .... In: *EUROCRYPT 2002*. LNCS 2332, pp. 534–545. Springer (2002)
56. L. Wang. SHELL v1. CAESAR submission (2014)
57. H. Wu, T. Huang. JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU (v1). CAESAR submission (2014)
58. H. Wu, T. Huang. The Authenticated Cipher MORUS (v1). CAESAR submission (2014)
59. L. Zhang, W. Wu, H. Sui, P. Wang. 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" [2, 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 fails 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 Other OAE Notions

There are at least other OAE notions in the literature that predate FFL's [25]. We briefly sketch them, and compare them with OAE1 and OAE2. We also mention a more recent definitional approach that aims to strengthen OAE1-security by demanding online decryptability.

FJMV notions. Back in 2003, Fouque, Joux, Martinet, and Valette (FJMV) put forward two notions for OAE, one for privacy and another for authenticity [27]. 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 $C_1 \cdots C_m \parallel T$.

To define privacy, we 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 $b \twoheadleftarrow \{0,1\}$. The adversary will make several oracle queries, each of the form $(M_1, M_2)$, where $M_0, M_1 \in \{0,1\}^n \cup \{\perp\}$. If $M_0, M_1 \in \{0,1\}^n$ then the oracle computes $M \leftarrow M \parallel M_b$, 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 seems like an attempt to recast 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 $C_1 \parallel T_1$, and then query $(N, A, M_1 M_2)$ to get $C_1 C_2 \parallel T_2$, and so on.

BT notion. Also in 2003, Boldyreva and Taesombut (BT) [20] 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 [20, Section 6] 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, insofar 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 $R \parallel M \parallel R$ 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, below, work that is by far the closest prior notion to OAE2.

TSS notion. 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) [54] is the unique 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 the Encrypt-then-MAC fashion.

We consider it unfortunate that the TSS manuscript has been so uninfluential up until now, as it seems to us more definitionally prescient than 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.

RUP-related notions. Andreeva, Bogdanov, Luykx, Mennink, Mouha, and Yasuda (ABLMMY) [5] 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 [4] meets their PA1 definition. This scheme does not, and in some sense cannot [5, 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 $\tau$, 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 rigorous implications or separations.

## 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 $\tau$ associated to $\Pi$, 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 $\Pi$ is defined as

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

where $\$(\cdot,\cdot,\cdot)$ is an oracle that returns $C \twoheadleftarrow \{0,1\}^{|M|+\tau}$ 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 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 $\mathbf{Adv}_{\mathcal{S}}^{\mathrm{guess}}$ denote the min-entropy of the distribution generated by $\mathcal{S}$. Let $\mathscr{A}$ be a CPSS adversary attacking an AE scheme $\Pi$. Consider the following MRAE adversary $\mathscr{B}$ attacking $\Pi$. It generates the suffix $S \twoheadleftarrow \mathcal{S}$ 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 $P \parallel S$ 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 $\mathbf{Adv}_{\mathcal{S}}^{\mathrm{guess}}$, since it only receives random answers independent of $S$. Hence the chance that $\mathscr{A}$ can guess $S$ in the CPSS attack against $\Pi$ is at most $\mathbf{Adv}_{\Pi}^{\mathrm{mrae}}(\mathscr{B}) + \mathbf{Adv}_{\mathcal{S}}^{\mathrm{guess}}$.

## 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. 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 $\Pi$ 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 [7] and McOE [25]. 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 $\Pi$ 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 $\mathbf{Adv}_{\Pi}^{\mathrm{oae1}}(\mathscr{B})$. 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 $\tau$ 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 $\tau$-expanding

```
proc Enc.init($N, A$)
$I \leftarrow I + 1$; $(C, S_I) \leftarrow \mathrm{Map}(N, 1^n \parallel A, 0^n)$; return $I$

proc Enc.next($i, M$)
if $i > I$ or $S_i = \perp$ then return $\perp$
$(C, S_i) \leftarrow \mathrm{Map}(S_i, \varepsilon, M)$; return $C$

proc Enc.last($i, M$)
if $i > I$ or $S_i = \perp$ then return $\perp$
$(C, S_i) \leftarrow \mathrm{Map}(S_i, 0^n, M)$; return $C$
```

```
proc Dec.init($N, A$)
$J \leftarrow J + 1$; $(C, S'_J) \leftarrow \mathrm{Map}(N, 1^n \parallel A, 0^n)$; return $J$

proc Dec.next($j, C$)
if $j > J$ or $S'_j = \perp$ then return $\perp$
$(M, S'_j) \leftarrow \mathrm{MapInv}(S'_j, \varepsilon, C)$; return $M$

proc Dec.last($j, C$)
if $j > J$ or $S_j = \perp$ then return $\perp$
$(M, S'_j) \leftarrow \mathrm{MapInv}(S'_j, 0^n, C)$; return $M$
```

```
proc Map($N, A, M$)                                  Game $G_1$
$C \leftarrow \mathbf{E}_K^{N,A}(M)$
if $H[N, A, M] \neq \perp$ then
    $S \leftarrow H[N, A, M]$; return $(C, S)$
$S \leftarrow \mathrm{Ev}(N, A, M, C)$; return $(C, S)$

proc MapInv($N, A, C$)
$M \leftarrow \mathbf{D}_K^{N,A}(C)$
if $M = \perp$ then return $(\perp, \perp)$
if $H[N, A, M] \neq \perp$ then
    $S \leftarrow H[N, A, M]$; return $(M, S)$
$S \leftarrow \mathrm{Ev}(N, A, M, C)$; return $(M, S)$

proc Ev($N, A, M, C$)
$S \leftarrow \perp$
if $|M| \geq n$ then $S \leftarrow C[1, n] \oplus M[1, n]$
elsif $A = \varepsilon$ then
    $V \leftarrow \mathbf{E}_K(N, 010^*, M10^*)$; $S \leftarrow V[1, n]$
if $(S \neq \perp)$ then $\mathrm{Dom} \leftarrow \mathrm{Dom} \cup \{S\}$
$H[N, A, M] \leftarrow S$; return $S$
```

```
proc Map($N, A, M$)                                  Games $G_2$, $\boxed{G_3}$
$C \leftarrow \rho_{N,A}(M)$
if $H[N, A, M] \neq \perp$ then
    $S \leftarrow H[N, A, M]$; return $(C, S)$
$S \leftarrow \mathrm{Ev}(N, A, M, C)$; return $(C, S)$

proc MapInv($N, A, C$)
$M \leftarrow \rho_{N,A}^{-1}(C)$
if $M = \perp$ then return $(\perp, \perp)$
if $H[N, A, M] \neq \perp$ then
    $S \leftarrow H[N, A, M]$; return $(M, S)$
$S \leftarrow \mathrm{Ev}(N, A, M, C)$; return $(M, S)$

proc Ev($N, A, M, C$)
$S \leftarrow \perp$
if $|M| \geq n$ then $S \leftarrow C[1, n] \oplus M[1, n]$
elsif $A = \varepsilon$ then
    $V \leftarrow \rho_{N,010^*}(M10^*)$; $S \leftarrow V[1, n]$
if $(S \neq \perp)$ then
    if $(S \in \mathrm{Dom})$ then bad $\leftarrow$ true; $\boxed{S \twoheadleftarrow \{0,1\}^n \backslash \mathrm{Dom}}$
    $\mathrm{Dom} \leftarrow \mathrm{Dom} \cup \{S\}$
$H[N, A, M] \leftarrow S$; return $S$
```

Fig. 10: Games $G_1$–$G_3$ in the proof of Theorem 1. Game $G_3$ contains the corresponding boxed statements 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 samples $K \twoheadleftarrow \mathbf{K}$ and $\rho_{N,A} \twoheadleftarrow \mathrm{Inj}(\tau)$ for every $N, A \in \{0,1\}^*$, and initializes $\mathrm{Dom} \leftarrow \emptyset$ and $I, J \leftarrow 0$.

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 Proof of Theorem 1

The reduction $R$ creates from $\mathscr{A}$ adversary $\mathscr{B}$ as follows. The latter runs the former and simulates game $\mathbf{Real2}_{\mathbf{CHAIN}[\Pi,n]}$, 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 $G_1$–$G_3$ in Fig. 10. Game $G_1$ corresponds to game $\mathbf{Real2}_{\mathbf{CHAIN}[\Pi,n]}$. We now explain the game chain. 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 a random injective function $\rho_{N,A} \twoheadleftarrow \mathrm{Inj}(\tau)$ and its inverse $\rho_{N,A}^{-1}$ respectively. Then

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

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

Map$(N, A, M)$ then MapInv$(N, A, C)$ returns $(M, S)$, and (iv) if $(M, S) \leftarrow$ MapInv$(N, A, C)$ and $M \neq \bot$ then Map$(N, A, M)$ returns $(C, S)$. 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 Map$(N, A, M)$ and MapInv$(N, A, C)$, we'll have either $A \in \{\varepsilon, 0^n\}$, or $A[1, n] = 1^n$.
(2) In Map$(N, A, M)$, we call $C \leftarrow \rho_{N,A}(M)$ and $V \leftarrow \rho_{N,010^*}(M10^*)$. The second call is made only if $|M| < n$ and $A = \varepsilon$, and there is no prior call Map$(N, A, M)$ or MapInv$(N, A, C)$.
(3) In MapInv$(N, A, C)$, we call $M \leftarrow \rho^{-1}_{N,A}(C)$ and $V \leftarrow \rho_{N,0^n}(M10^*)$. The second call is made only if $|M| < n$ and $A = \varepsilon$, and there is no prior call Map$(N, A, M)$ or MapInv$(N, A, C)$.

Wlog, assume that $1 < q \leq 2^{n-1}$; otherwise the bound is trivial. The flag bad is triggered only if the state $S$ belongs to a set Dom contains $0^n$ and all prior values of the state. Consider the following cases.

Case 1: $S \leftarrow V[1, n]$, where $V \leftarrow \rho_{N,010^*}(M10^*)$. From (1), (2), and (3), since there is a one-to-one correspondence between $M$ and $M10^*$, there is no prior call to $\rho_{N,010^*}(M10^*)$ or $\rho^{-1}_{N,0^n}(V)$. Then $V$ 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

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

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

Case 2: $S \leftarrow C[1, n] \oplus M[1, n]$. There must be no prior Map call of the same $(N, A, M)$ or MapInv call of the same $(N, A, C)$, 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 $\rho_{N,A}(M)$ or $\rho^{-1}_{N,A}(C)$. 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 $i2^{s-n}/(2^s - q) \leq i/(2^n - q) \leq 2i/2^n$. 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 $2q^2/2^n$.

We now argue that game $G_3$ is equivalent to game $\mathbf{Ideal2}_{\mathbf{CHAIN}[\Pi,n]}$, and thus

$$\mathbf{Adv}^{\mathrm{oae2}}_{\mathbf{CHAIN}[\Pi,n]}(\mathscr{A}) = \Pr[\mathscr{A}^{G_1} \Rightarrow 1] - \Pr[\mathscr{A}^{G_3} \Rightarrow 1] \leq \mathbf{Adv}^{\mathrm{pri}}_{\Pi}(\mathscr{B}) + \frac{2q^2}{2^n}$$

as claimed. Note that in $G_3$, each state $S_i$ never repeats its prior value. Moreover, $S_i = S_j$ if only if $(\boldsymbol{M}_i, N_i, A_i) = (\boldsymbol{M}_j, N_j, A_j)$. Each query Enc.next$(i, M)$ returns $C \leftarrow \rho_{S_i,\varepsilon}(M)$, effectively implementing $\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \parallel M$; $\boldsymbol{C} \leftarrow \pi_{N_i,A_i}(\boldsymbol{M}_i, 0)$; **return** $\boldsymbol{C}[|\boldsymbol{M}_i|]$. Likewise, each query Enc.last$(i, M)$ returns $C \leftarrow \rho_{S_i,0^n}(M)$, effectively implements $\boldsymbol{M}_i \leftarrow \boldsymbol{M}_i \parallel M$; $\boldsymbol{C} \leftarrow \pi_{N_i,A_i}(\boldsymbol{M}_i, 1)$; **return** $\boldsymbol{C}[|\boldsymbol{M}_i|]$. The analogous claims hold for Dec.next and Dec.last. Hence $G_3$ is equivalent to $\mathbf{Ideal2}_{\mathbf{CHAIN}[\Pi,n]}$.

## F Proof of Theorem 2

Without loss of generality, suppose that $\mathscr{A}$ never repeats a prior $(N, A)$ to Enc.init. The reduction $R_0$ creates from $\mathscr{A}$ adversary $\mathscr{B}_0$ as follows. The latter runs the former and simulates game $\mathbf{nReal}_{\mathbf{STREAM}[F,\Pi]}$, but each $F_{K_0}(x)$ is replaced by querying $x$ to $\mathscr{B}_0$'s oracle. (If there is a prior

```
proc Enc.init($N, A$)
$I \leftarrow I + 1$; $V \leftarrow \mathrm{PRF}(A)$
$S_I \leftarrow (N, V, 1)$; return $I$

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

proc Enc.last($i, M$)
if $i > I$ or $S_i = \perp$ then return $\perp$
$(N, V, v) \leftarrow S_i$; $S_i \leftarrow \perp$
$C \leftarrow \mathrm{Map}(\langle 1, N, v\rangle, V, M)$; return $C$
```

```
proc Dec.init($N, A$)
$J \leftarrow J + 1$; $V \leftarrow \mathrm{PRF}(A)$
$S'_J \leftarrow (N, V, 1)$; return $J$

proc Dec.next($j, C$)
if $j > J$ or $S'_j = \perp$ then return $\perp$
$(N, V, v) \leftarrow S'_j$; $S'_j \leftarrow (N, \varepsilon, v + 1)$
$M \leftarrow \mathrm{MapInv}(\langle 0, N, v\rangle, \varepsilon, C)$
if $M = \perp$ then $S'_j \leftarrow \perp$
return $M$

proc Dec.last($j, C$)
if $j > J$ or $S_j = \perp$ then return $\perp$
$(N, V, v) \leftarrow S'_j$; $S'_j \leftarrow \perp$
$M \leftarrow \mathrm{MapInv}(\langle 1, N, v\rangle, V, C)$; return $M$
```

```
proc PRF($X$)                                Game $G_1$
if $R[X] \neq \perp$ then return $R[X]$
$R[X] \leftarrow F_{K_0}(X)$; return $R[X]$

proc Map($N, A, M$)
$C \leftarrow \mathbf{E}^{N,A}_{K_1}(M); H[N, A, C] \leftarrow M$; return $C$

proc MapInv($N, A, C$)
if $H[N, A, C] \neq \perp$ then return $H[N, A, C]$
$M \leftarrow \mathbf{D}^{N,A}_{K_1}(C)$
if $M \neq \perp$ then return $M$
return $\perp$
```

```
proc PRF($X$)                                Games $G_2$, $\boxed{G_3}$
if $R[X] \neq \perp$ then return $R[X]$
$R[X] \twoheadleftarrow \{0,1\}^n$
if $R[X] \in \mathrm{Dom}$ then
    bad $\leftarrow$ true; $\boxed{R[X] \twoheadleftarrow \{0,1\}^n \backslash \mathrm{Dom}}$
$\mathrm{Dom} \leftarrow \mathrm{Dom} \cup \{R[x]\}$; return $R[X]$

proc Map($N, A, M$)
$C \leftarrow \mathbf{E}^{N,A}_{K_1}(M); H[N, A, C] \leftarrow M$; return $C$

proc MapInv($N, A, C$)
if $H[N, A, C] \neq \perp$ then return $H[N, A, C]$
$M \leftarrow \mathbf{D}^{N,A}_{K_1}(C)$
if $M \neq \perp$ then return $M$
return $\perp$
```

```
proc PRF($X$)
if $R[X] \neq \perp$ then return $R[X]$
$R[X] \twoheadleftarrow \{0,1\}^n \backslash \mathrm{Dom}$
$\mathrm{Dom} \leftarrow \mathrm{Dom} \cup \{R[x]\}$; return $R[X]$
```

```
proc Map($N, A, M$)                          Games $G_4$
$C \twoheadleftarrow \{0,1\}^{|M|+\tau}$; $H[N, A, C] \leftarrow M$
return $C$

proc MapInv($N, A, C$)
return $\perp$
```

Fig. 11: Games $G_1$–$G_4$ used in the proof of Theorem 2. Game $G_3$ contains the corresponding boxed statements 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 PRF, Map, MapInv, and Ev that are inaccessible to the adversary. In each game there is an implicit procedure **initialize** () that samples $K_0 \twoheadleftarrow \mathbf{K}_0, K_1 \twoheadleftarrow \mathbf{K}$ and initializes $\mathrm{Dom} \leftarrow \emptyset$ and $I, J \leftarrow 0$.

$F_{K_0}(x)$ then $\mathscr{B}_0$ returns the consistent answer, instead of querying its oracle.) Adversary $\mathscr{B}_0$ then outputs the same guess as $\mathscr{A}$.

The reduction $R_1$ creates from $\mathscr{A}$ adversary $\mathscr{B}_1$ as follows. The latter runs the former and simulates game $\mathbf{nReal}_{\mathbf{STREAM}[F,\Pi]}$, but each call to $\mathbf{E}_{K_1}(\cdot)$ or $\mathbf{D}_{K_1}(\cdot)$ is replaced by the corresponding query to the first or second oracle of $\mathscr{B}_1$, respectively. Adversary $\mathscr{B}_1$ then outputs the same guess as $\mathscr{A}$.

Consider games $G_1$–$G_6$ in Figures 11 and 12. Game $G_1$ corresponds to game $\mathbf{nReal}_{\mathbf{STREAM}[F,\Pi]}$, and game $G_6$ corresponds to $\mathbf{nIdeal}_{\mathbf{STREAM}[F,\Pi]}$. We now explain the game chain. Game $G_2$ is identical to game $G_1$, except that instead of calling $F_{K_0}(\cdot)$, we'll return a random $n$-bit string. Then

$$\Pr[\mathscr{A}^{G_1} \Rightarrow 1] - \Pr[\mathscr{A}^{G_2} \Rightarrow 1] = \mathbf{Adv}^{\mathrm{prf}}_{F}(\mathscr{B}_0) \ .$$

In game $G_3$, instead of sampling the answers for procedures PRF at random, we make sure that the answers are distinct. Games $G_2$ and $G_3$ are identical-until-$\mathsf{bad}$, and thus

$$\Pr[\mathscr{A}^{G_2} \Rightarrow 1] - \Pr[\mathscr{A}^{G_3} \Rightarrow 1] \leq \Pr[\mathscr{A}^{G_2} \text{ sets } \mathsf{bad}] \leq q^2/2^{n+1} \ .$$

```
Games $G_5$, $\boxed{G_6}$

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

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

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

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

proc Dec.next($j, C$)
if $j > J$ or $\boldsymbol{C}_j = \bot$ then return $\bot$
$\boldsymbol{C}_j \leftarrow \boldsymbol{C}_j \,\|\, C$; $m \leftarrow |\boldsymbol{C}_j|$
if $H[N'_j, A'_j, 0, \boldsymbol{C}_j] \neq \bot$ then return $H[N'_j, A'_j, 0, \boldsymbol{C}_j]$
if $\exists \boldsymbol{M}$ s.t. $\pi_{N'_j, A'_j}(\boldsymbol{M}, 0) = \boldsymbol{C}_j$ then
    bad $\leftarrow$ true; $\boxed{\textbf{return } \boldsymbol{M}[m]}$
$\boldsymbol{C}_j \leftarrow \bot$; return $\bot$

proc Dec.last($j, C$)
if $j > J$ or $\boldsymbol{C}_j = \bot$ then return $\bot$
$\boldsymbol{C}_j \leftarrow \boldsymbol{C}_j \,\|\, C$; $m \leftarrow |\boldsymbol{C}_j|$; $M \leftarrow \bot$
if $H[N'_j, A'_j, 1, \boldsymbol{C}_j] \neq \bot$ then return $H[N'_j, A'_j, 1, \boldsymbol{C}_j]$
if $\exists \boldsymbol{M}$ s.t. $\pi_{N'_j, A'_j}(\boldsymbol{M}, 1) = \boldsymbol{C}_j$ then
    bad $\leftarrow$ true; $\boxed{M \leftarrow \boldsymbol{M}[m]}$
$\boldsymbol{C}_j \leftarrow \bot$; return $M$
```

Fig. 12: Games $G_5$ and $G_6$ used in the proof of Theorem 2. Game $G_6$ contains the corresponding boxed statements but game $G_5$ doesn't. In each game, there is an implicit **initialize** () procedure that samples $\pi_{N,A} \twoheadleftarrow \mathbf{Inj}(\tau)$ for every $N, A \in \{0,1\}^*$.

Game $G_4$ is identical to game $G_3$, except that instead of using $\mathbf{E}_{K_1}^{N,A}$ and $\mathbf{D}_{K_1}(\cdot,\cdot,\cdot)$, we'll use $\$(\cdot,\cdot,\cdot)$ and $\bot(\cdot,\cdot,\cdot)$ respectively. (But if one queries $C \leftarrow \mathbf{E}_{K_1}^{N,A}(M)$ and then queries $\mathbf{D}_{K_1}^{N,A}(C)$ then we'll give the consistent answer stored in the array $H$.) Then

$$\Pr[\mathscr{A}^{G_3} \Rightarrow 1] - \Pr[\mathscr{A}^{G_4} \Rightarrow 1] = \mathbf{Adv}_{\Pi}^{\mathrm{nae}}(\mathscr{B}_1) \ .$$

Game $G_5$ is equivalent to game $G_4$: if one only uses $\pi_{N,A} \twoheadleftarrow \mathbf{Inj}(\tau)$ for each $N, A \in \{0,1\}^*$ at most once then $\pi_{N,A}(\boldsymbol{M}, b)$ is to sample $C_1 \twoheadleftarrow \{0,1\}^{\tau + |\boldsymbol{M}[1]|}, \ldots, C_m \twoheadleftarrow \{0,1\}^{\tau + |\boldsymbol{M}[m]|}$, where $m = |\boldsymbol{M}|$. Game $G_6$ is identical to game $G_5$, but in Dec.next and Dec.last, we'll return a non-$\bot$ value if there exists $\boldsymbol{M}$ such that $\pi_{N'_j, A'_j}(\boldsymbol{M}, b) = \boldsymbol{C}_j$ (with $b = 0$ for Dec.next and $b = 1$ for Dec.last). Games $G_5$ and $G_6$ are identical-until-bad, and thus

$$\Pr[\mathscr{A}^{G_5} \Rightarrow 1] - \Pr[\mathscr{A}^{G_6} \Rightarrow 1] \le \Pr[\mathscr{A}^{G_5} \text{ sets } \mathsf{bad}] \le q/2^{\tau} \ .$$

Hence

$$\mathbf{Adv}_{\mathbf{STREAM}[F,\Pi]}^{\mathrm{noae}}(\mathscr{A}) = \Pr[\mathscr{A}^{G_1} \Rightarrow 1] - \Pr[\mathscr{A}^{G_6} \Rightarrow 1] \le \mathbf{Adv}_F^{\mathrm{prf}}(\mathscr{B}_0) + \mathbf{Adv}_{\Pi}^{\mathrm{nae}}(\mathscr{B}_1) + \frac{q^2}{2^{n+1}} + \frac{q}{2^{\tau}} \ .$$