The Memory-Tightness of Authenticated Encryption*

Ashrujit Ghoshal, Joseph Jaeger, and Stefano Tessaro

Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle, US
{ashrujit,jsjaeger,tessaro}@cs.washington.edu

Abstract. This paper initiates the study of the provable security of authenticated encryption (AE) in the memory-bounded setting. Recent works – Tessaro and Thiruvengadam (TCC '18), Jaeger and Tessaro (EUROCRYPT '19), and Dinur (EUROCRYPT '20) – focus on confidentiality, and look at schemes for which trade-offs between the attacker's memory and its data complexity are inherent. Here, we ask whether these results and techniques can be lifted to the full AE setting, which additionally asks for integrity.

We show both positive and negative results. On the positive side, we provide tight memory-sensitive bounds for the security of GCM and its generalization, CAU (Bellare and Tackmann, CRYPTO '16). Our bounds apply to a restricted case of AE security which abstracts the deployment within protocols like TLS, and rely on a new memory-tight reduction to corresponding restricted notions of confidentiality and integrity. In particular, our reduction uses an amount of memory which linearly depends on that of the given adversary, as opposed to only imposing a constant memory overhead as in earlier works (Auerbach et al., CRYPTO '17).

On the negative side, we show that a large class of black-box reductions cannot generically lift confidentiality and integrity security to a joint definition of AE security in a memory-tight way.

Keywords: Provable security, symmetric cryptography, time-memory trade-offs, memory-tightness

1 Introduction

Cryptographic attacks aim to use as little *memory* as possible. While some attacks are memoryless (e.g., for collision finding), others are subject to a *trade-off* – as the available memory decreases, the time and data complexities increase. A security proof (especially one in the spirit of *concrete* security) should tell us precisely *how* memory affects other complexity metrics. However, this is technically challenging, and consequently, security proofs ignored memory until recently.

This paper continues an ongoing line of works introducing memory limitations in provable security, and initiates the study of (nonce-based) *authenticated encryption* (AE) in the memory-bounded setting. Recent works [17,11,7] have shown memory-sensitive proofs of security for symmetric encryption, showing that trade-offs between memory and data complexities are inherent. These results, however, only deal with *confidentiality* of encryption – and one of the main contributions of this paper is to highlight the challenges of lifting them to the more complex setting of AE.

We discuss definitional aspects, and then shift our focus to *memory-tight reductions* [1] in the AE setting. We prove both *positive* and *negative* results. We introduce a new technique for memory-tight reductions to obtain tight memory-sensitive bounds for the AE-security of GCM in a setting that corresponds to its usage for establishing a secure channel. We also show that restricting AE security to specific settings is inherent for memory-tight reductions – indeed, we show that the common approach of lifting confidentiality and integrity guarantees into a combined notion of AE security (or of CCA security) fails in its most general form, at least with respect to a broad class of security reductions.

* A preliminary version of this paper appears in the proceedings of CRYPTO 2020. This is the full version.

1.1 Context: Time-memory Trade-offs for AE

Let us start by setting the context and highlighting some of the challenges. First off, existing results [11,7] can be combined to analyze the INDR security[1] of nonce-based encryption. For example, consider a toy scheme[2] SE based on a block cipher E with block length n which encrypts $M \in \{0,1\}^n$ with key K as

$$\mathsf{SE.E}(K, N, M) = \mathrm{E}_K(N) \oplus M\ .$$

Here, N is the nonce and INDR security should hold as long as no two messages are encrypted with the same nonce. One can show that for every adversary $\mathcal{A}$ with time, data, and memory complexities t, q, and S, respectively,

$$\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{SE}}(\mathcal{A}) \leqslant O\left(\frac{q \cdot S \cdot \log(q)}{2^n}\right) + \mathsf{Adv}^{\mathsf{prp}}_{\mathrm{E}}(\mathcal{B})\ , \tag{1}$$

where $\mathcal{B}$ is an adversary against the security of E as a pseudorandom permutation (PRP), which has time and memory complexities (roughly) t and S, respectively, and makes q queries. In particular, if $S < 2^{n/2}$, then SE achieves beyond-birthday security $q > 2^{n/2}$ with respect to data complexity.

Our goal, in more detail. However, INDR security is rarely sufficient on its own – we want *fully secure* AE schemes which also satisfy (ciphertext) *integrity* (or CTXT security, for short). Following [16], we adopt a *single* AE security definition that incorporates *both* INDR and CTXT, by measuring indistinguishability of two oracle pairs ($\textsc{Enc}_b, \textsc{Dec}_b$) for $b \in \{0,1\}$. For $b = 1$, $\textsc{Enc}_1$ returns real ciphertexts, and $\textsc{Dec}_1$ decrypts properly. For $b = 0$, instead, $\textsc{Enc}_0$ returns random ciphertexts, and $\textsc{Dec}_0$ decrypts only previous outputs from $\textsc{Enc}_0$. It is important to use a combined definition, as it captures settings such as chosen-ciphertext attacks and padding-oracle attacks [18], which use a decryption oracle to break confidentiality.[3]

Lifting trade-offs. We want to prove a bound analogous to that of (1) for AE security, preserving in particular the existing space-time trade-off. The usual approach is to prove INDR and CTXT *individually*, and then combine them to show AE security. This makes sense because (1) we *know* how to prove tight trade-offs for INDR security, and (2) we may be able to prove stronger bounds on CTXT easily, even without memory restrictions. The classical statement (originally in [16]) is that for every adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{SE}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{SE}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{ctxt}}_{\mathsf{SE}}(\mathcal{C})\ ,$$

for suitable adversaries $\mathcal{B}$ and $\mathcal{C}$, with similar time and query complexities as those of $\mathcal{A}$. However, this is only helpful towards our goal if the reduction is *memory-tight*, in the sense Auerbach et al. (ACKF) [1], i.e., $\mathcal{B}$ and $\mathcal{C}$'s memory costs must not noticeably exceed those of $\mathcal{A}$. This is fundamental to preserve a time-memory trade-off like the one from (1).

Unfortunately, the standard proof is not memory-tight with respect to the INDR adversary $\mathcal{B}$, as it needs to simulate $\textsc{Dec}_0$ which requires remembering prior ciphertexts. In a nutshell, we will show that the lack of memory-tightness is inherent, *but* the definition can be restricted enough for interesting deployment scenarios to actually allow for a memory-tight reduction.

Definitional issues. Several "without loss of generality" definitional equivalences are false in the memory-bounded setting. For example, INDR security holds as long as nonces do not repeat, but there are options to formalize this, e.g.: (A) The game enforces this by answering encryption queries repeating a nonce with $\perp$, unless the same message is re-encrypted, or (B) The adversary never repeats a nonce. If we do not care about memory, these two definitions are indeed equivalent, but if we do, then they are not. Indeed, the bound in (1) for our toy scheme can only be true for (B) – it is not hard to see that otherwise we can mount a memory-less distinguishing attack with $q \approx 2^{n/2}$ queries. (The attack also works if $\perp$ is returned even if we re-encrypt the same message.) We discuss definitions in detail in Section 3.

[1] Which measures the indistinguishability of ciphertexts from truly random ones.
[2] Our discussion can easily be extended to many schemes following the format of counter-mode encryption.
[3] While we target such a single definition of AE, we stress that our results would extend to considering CCA security as a target.

1.2 Positive Results

We provide a novel memory-tight reduction for the common case where AE is used to establish a secure communication *channel*, as in TLS. The key point is that in this setting, only certain restricted adversarial interactions can occur in the AE security game, i.e.:

(1) Nonces are *implicit* – they are incremented as a counter.

(2) The receiver *aborts upon the first decryption failure*. In particular, messages *need* to be delivered in the same order as they are encrypted.

Our memory-tight reduction is for an abstraction of this setting we refer to as a *channel*. (Although, for this introduction, we stick with the more conventional language of AE.) We apply our reduction to prove (tight) memory-sensitive bounds for a channel instantiated with the CAU scheme by Bellare and Tackmann [4], an abstraction of GCM [12].

The security game. When restricting AE security to this setting, we can assume that the adversary $\mathcal{A}$ can encrypt messages $M_1, M_2, \ldots$ and obtains ciphertexts $C_1, C_2, \ldots$ via an *encryption oracle* $\textsc{Enc}_b$, for $b \in \{0,1\}$. When $b = 1$, the C_i's are actual encryptions of the M_i's (with increasing nonces), whereas when $b = 0$, they are truly random ciphertexts. The adversary is also given access to a *decryption* oracle $\textsc{Dec}_b$. If $b = 1$, this just applies the decryption algorithm of the AE scheme, using increasing nonces. If decryption fails, $\textsc{Dec}_b$ responds to this and any future queries with $\perp$. For $b = 0$, the oracle responds with $M_1, M_2, \ldots$ as long as it is supplied the ciphertexts $C_1, C_2, \ldots$ *in the order they have been produced by* $\textsc{Enc}_0$. If the ciphertexts come in the wrong order, $\textsc{Dec}_0$ responds to this and any future queries with $\perp$. The goal here is to distinguish $(\textsc{Enc}_0, \textsc{Dec}_0)$ and $(\textsc{Enc}_1, \textsc{Dec}_1)$.

Proof idea. In this channel setting, to obtain a memory-tight reduction from AE security to CTXT and INDR security, we first use CTXT security to replace the oracles $(\textsc{Enc}_1, \textsc{Dec}_1)$ with $(\textsc{Enc}_1, \textsc{Dec}_0)$. (This step is easily seen to be memory-tight.) Next, we aim to use INDR security to replace $\textsc{Enc}_1$ with $\textsc{Enc}_0$. The catch here is that when doing so, we need to simulate the $\textsc{Dec}_0$ oracle in the INDR security game (which does not provide one). Again, this seems to require remembering all prior ciphertexts, thus preventing memory-tightness.

A key observation, however, is that ciphertexts are only accepted when arriving with the right order. For this reason, we will show (via an information-theoretic argument) that our reduction only needs to store the δ oldest ciphertexts which have not been delivered yet, for some δ – the key point here is that δ can be chosen to depend (roughly linearly) on the memory of the adversary used by the reduction, so the overall memory of the constructed adversary is of the same magnitude of that of the AE adversary.

This is in contrast to existing memory-tight reductions in the literature which are (near) "memory-less", i.e., the reduction adds a small memory overhead, *independent* of the memory of the adversary. Our reduction is the first example where the reduction uses memory in addition to that of the adversary, but the size of this memory is bounded in terms of the adversary's memory complexity.

Application to CAU. We apply our memory-tight reduction to show bounds for CAU (and hence GCM) in the communication channel setting. We refer to the resulting channel as NCH, and it is based on a block cipher E. We show that for every adversary $\mathcal{A}$, there exists $\mathcal{B}$ such that

$$\mathsf{Adv}^{\mathsf{ch\text{-}ae}}_{\mathsf{NCH}}(\mathcal{A}) \leqslant 4 \cdot \mathsf{Adv}^{\mathsf{prp}}_{\mathrm{E}}(\mathcal{B}) + O\left(\frac{pqS}{2^n}\right) , \tag{2}$$

where $O(\cdot)$ hides a small constant, q and S are the data and memory complexities of $\mathcal{A}$, and p is an upper bound on the length of ciphertexts. Further, $\mathcal{B}$ makes $q \cdot p$ queries, and has time complexity similar to that of $\mathcal{A}$. Instrumental to our result here is Dinur's Switching Lemma [7]. The main challenge is to prove a bound for CTXT security – our proof relies once again on similar techniques to our memory-tight reduction.

1.3 Negative Results

A meaningful question is whether we can give a memory-tight reduction beyond the setting of channels, and reduce AE security to INDR and CTXT security in the most general sense. Here, we show that this is unlikely by giving impossibility results for black-box reductions.

We consider reductions to INDR and CTXT which are restricted, but note that all prior impossibility results on memory-tight reductions [1,19,9] make similar or stronger restrictions. In particular, we require the reductions to simulate their encryption oracles "faithfully" to an AE adversary, i.e., if they answer an encryption query with a ciphertext C, the same query (1) has been asked to the encryption oracle available to the reduction and (2) it has returned C. This restriction is natural, and we are not aware of any reductions evading it.

Straightline reductions. Our first result builds an (inefficient) adversary $\mathcal{A}$ against AE security which no straightline reduction can use to (1) break CTXT security (regardless of the memory available to the reduction) or, more importantly, to (2) break INDR security (unless the reduction uses an amount of memory proportional to the query complexity of the adversary). Moreover, $\mathcal{A}$ uses little memory, and thus our result implies impossibility even for "weakly memory-tight reductions" which adapt their memory usage (such as the one we give in this paper). This is unlike recent works [19,9], which only rule out reductions with memory independent of that of the adversary.

At a high level, $\mathcal{A}$ forces the reduction to complete a memory-hard task before being useful. If the reduction succeeds, $\mathcal{A}$ executes an (inefficient) procedure to break INDR security. (And importantly, this procedure does not help in breaking CTXT security!) More in detail, the first part of $\mathcal{A}$'s execution consists of *challenge rounds*. In each of these rounds, $\mathcal{A}$ encrypts random plaintexts $M_1, \ldots, M_u$, which result in ciphertexts $C_1, \ldots, C_u$, and also picks a random index $i^* \in [u]$. It then asks for the decryption of C_{i^*}, and checks whether the response equals M_{i^*}. If so, it moves to the next round, if not it aborts by doing something useless. Only if all rounds are successful $\mathcal{A}$ proceeds to break INDR security. We use techniques borrowed from the setting of random oracles with auxiliary input (AI-ROM) [5] to prove that the probability that all rounds are successful decays exponentially as long as the reduction's memory does not fit all of $M_1, \ldots, M_u$.

Full Rewinding. The restriction to straightline reductions seem too restrictive: After all, a reduction could (1) wait for a decryption query C_{i^*}, then (2) rewind the adversary to re-ask $M_1, M_2, \ldots$ until M_{i^*} is asked. The caveat is that our definition of INDR security does not allow for *re-asking* encryption queries (again, as pointed out above, such a notion would prevent us from using the results of [11,7]). Therefore, if we assume that all the reduction can do is remember (say) S plaintext-ciphertext pairs, the above adversary $\mathcal{A}$ will fail to pass a challenge round with probability at least $1 - S/u$.

Still, this does not mean that rewinding cannot help when allowing more general adversarial strategies. While handling arbitrary rewinding appears to be out of reach, we make partial progress by extending our proof (and our construction of $\mathcal{A}$) to show that "full" rewinding (i.e., re-running $\mathcal{A}$ from the beginning) does not help. This is the same rewinding model considered in prior memory-tightness lower bounds [1]. However, in those results, one obtains a rewinding-memory trade-offs (in that reducing memory would require more rewinding). Here, our result is absolute, in the sense that if memory is too small, no amount of rewinding can help.

Paper overview. In Section 2, we introduce our notation, basic definitions and cover some cryptographic background necessary for the paper. In Section 3, we recall the standard definitions for the security notions of nonce-based encryption. We point out several nuances while defining security in the memory bounded setting. We conclude the section by giving a time-memory tradeoff for the INDR security of CAU. In Section 4, we show that memory-tight reductions can be given for the combined confidentiality and integrity security of cryptographic channels. Using the result from Section 3, we prove the security of a channel based on CAU. The resulting channel can be viewed as (a simplification of) the channel obtained when using GCM in TLS 1.3. In Section 5, we give impossibility results (for a natural restricted class of black-box reductions) for giving a memory-tight reduction from AE security to INDR and CTXT security. This establishes that our move to the channel setting for Section 4 was necessary for our positive result.

2 Definitions

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$. For $D \in \mathbb{N}$, let $[D] = \{1, 2, \ldots, D\}$. If S and S' are finite sets, then $\mathsf{Fcs}(S, S')$ denotes the set of all functions $F : S \to S'$ and $\mathsf{Perm}(S)$ denotes the set of all permutations on S. Picking an element uniformly at random from S and assigning it to s is denoted by $s \leftarrow\!\!\$\ S$. The set of finite vectors with entries in S is S^* or $(S)^*$. Thus $\{0,1\}^*$ is the set of finite length strings.

If $x \in \{0,1\}^*$ is a string, then $|x|$ denotes its bitlength. If $n \in \mathbb{N}$ and $x \in \{0,1\}^*$, then $|x|_n = \max\{1, \lceil |x|/n \rceil\}$. We let $x_1 \dots x_\ell \leftarrow_n x$ denote setting $\ell \leftarrow |x|_n$ and parsing x into ℓ blocks of length n (except x_ℓ which may have $|x_\ell| < n$). We let $x[: n]$ denote the first n bits of x and $x[i : n]$ denote the i-th (exclusive) through n-th (inclusive) bits of x. We adopt the convention that if $|x| < |x'|$ then $x \oplus x' = x \oplus x'[: |x|]$. The empty string is ε.

We will make use of queues which operate in first-in, first-out order. If Q is a queue then $Q.\mathsf{add}(M)$ adds M to the back of the queue and $M \leftarrow Q.\mathsf{dq}()$ removes the first element of the queue and assigns it to M. If the queue is empty, then M is assigned the value $\perp \notin \{0,1\}^*$ which is used to represent rejection or uninitialized values.

Algorithms are randomized when not specified otherwise. If $\mathcal{A}$ is an algorithm, then $y \leftarrow \mathcal{A}^{\mathrm{O}_1,\cdots}(x_1,\dots;r)$ denotes running $\mathcal{A}$ on inputs $x_1,\dots$ with coins r and access to the oracles $\mathrm{O}_1,\dots$ to produce output y. Performing this execution with a random r is denoted $y \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{O}_1,\cdots}(x_1,\dots)$. The set of all possible outputs of $\mathcal{A}$ when run with inputs $x_1,\dots$ is $[\mathcal{A}(x_1,\dots)]$. The notation $y \leftarrow \mathrm{O}(x_1,\dots)$ is used for calling oracle O with inputs $x_1,\dots$ and assigning its output to y. (Note, the code run by the oracle is not necessarily deterministic.)

We make regular use of pseudocode games inspired by the code-based framework of [3]. Examples of games can be found in Fig. 1. We let $\mathsf{Pr}[\mathsf{G}]$ denote the probability that a game G outputs `true`. Booleans are implicitly initialized to `false`, integers to 0, and all other types to $\perp$.

Complexity conventions. When measuring the efficiency of an adversary we follow the standard convention used in studying memory-tightness [1] on measuring the local complexity of an adversary and not included the complexity of whatever game it interacts with. We primarily focus on the worst-case runtime (i.e. how much computation it performs in between making oracle queries) and memory complexity (i.e. how many bits of state it stores for local computation) of adversaries. Note that while these exclude the time and memory used within whatever oracles the adversary may call, we do include the time and memory used to write down an oracle query and receive the response.

2.1 Cryptographic Background

Function family. A function family is an efficiently computable function $\mathsf{F} : A \times B \to C$, where A, B, and C are sets. A hash function is a family of functions. We often write $F_K(\cdot)$ in place of $F(K,\cdot)$.

Pseudorandom function/permutation. Let $\mathrm{E} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ be a function family. If $n = m$ and $\mathrm{E}_K(\cdot)$ is a permutation for each $K \in \{0,1\}^k$, then we say that E is a block-cipher. The primary security notions of interest for such functions are PRF and PRP security. The former is typically more useful in applications, but when E is a block-cipher we prefer to assume PRP security and use that to deduce PRF security.

These security notions are defined by games shown in Fig. 1. In $\mathsf{G}^{\mathsf{prp}}$, the adversary is given access to either $\mathrm{E}_K(\cdot)$ for a random key or a random permutation $P : \{0,1\}^n \to \{0,1\}^n$. Game $\mathsf{G}^{\mathsf{prf}}$ is defined similarly except a random function $F : \{0,1\}^n \to \{0,1\}^m$ is used in place of the permutation. For $x \in \{\mathsf{prp}, \mathsf{prf}\}$, we define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^x_{\mathrm{E}}(\mathcal{A}) = \mathsf{Pr}[\mathsf{G}^x_{\mathrm{E},1}(\mathcal{A})] - \mathsf{Pr}[\mathsf{G}^x_{\mathrm{E},0}(\mathcal{A})]$.

Switching Lemma. A classic result in cryptography is the "switching lemma" which bounds how well an adversary can distinguish between a random function and a random permutation. Consider the game $\mathsf{G}^{\mathsf{sl}}_{D,b}$ shown in Fig. 1. In it, the adversary is given oracle access to either a random function or a random permutation with domain/range $[D]$ and is trying to figure out which. We define $\mathsf{Adv}^{\mathsf{sl}}_D(\mathcal{A}) = \mathsf{Pr}[\mathsf{G}^{\mathsf{sl}}_{D,b}(\mathcal{A})] - \mathsf{Pr}[\mathsf{G}^{\mathsf{sl}}_{D,b}(\mathcal{A})]$.

The classic switching lemma shows $\mathsf{Adv}^{\mathsf{sl}}_D(\mathcal{A}) \in O(q^2/D)$ where q is the number of queries made by $\mathcal{A}$. In general, bounding the memory-complexity of the attacker cannot be used to meaningfully improve this bound because a low-memory collision-finding attack (e.g., using Pollard's ρ-method [13,14]) achieves advantage $\mathsf{Adv}^{\mathsf{sl}}_D(\mathcal{A}) \in \Omega(q^2/D)$. However, as originally observed by Jaeger and Tessaro we *can* obtain better results when restricting attention to adversaries that never repeat any queries.

```
Game $\mathsf{G}^{\mathsf{prp}}_{\mathrm{E},b}(\mathcal{A})$
    $K \leftarrow\$ \{0,1\}^k$
    $P \leftarrow\$ \mathsf{Perm}(\{0,1\}^n)$
    $b' \leftarrow\$ \mathcal{A}^{\textsc{Eval}_b}$
    Return $b' = 1$

Oracle $\textsc{Eval}_b(x)$
    $y_1 \leftarrow \mathrm{E}_K(x)$
    $y_0 \leftarrow P(x)$
    Return $y_b$
```

```
Game $\mathsf{G}^{\mathsf{prf}}_{\mathrm{E},b}(\mathcal{A})$
    $K \leftarrow\$ \{0,1\}^k$
    $F \leftarrow\$ \mathsf{Fcs}(\{0,1\}^n, \{0,1\}^m)$
    $b' \leftarrow\$ \mathcal{A}^{\textsc{Eval}_b}$
    Return $b' = 1$

Oracle $\textsc{Eval}_b(x)$
    $y_1 \leftarrow \mathrm{E}_K(x)$
    $y_0 \leftarrow F(x)$
    Return $y_b$
```

```
Game $\mathsf{G}^{\mathsf{sl}}_{D,b}(\mathcal{A})$
    $F \leftarrow\$ \mathsf{Fcs}([D],[D])$
    $P \leftarrow\$ \mathsf{Perm}([D])$
    $b' \leftarrow\$ \mathcal{A}^{\textsc{Eval}_b}$
    Return $b' = 1$

Oracle $\textsc{Eval}_b(x)$
    $y_1 \leftarrow F(x)$
    $y_0 \leftarrow P(x)$
    Return $y_b$
```

Fig. 1. Security games for PRF and PRP security of E and the switching lemma.

```
Game $\mathsf{G}^{\mathsf{axu}}_{\mathrm{H}}(\mathcal{X})$
    $((A_1, C_1), (A_2, C_2), Z) \leftarrow\$ \mathcal{X}$
    $L \leftarrow\$ \{0,1\}^k$
    If $(A_1, C_1) = (A_2, C_2)$ then return false
    Return $\mathrm{H}_L(A_1, C_1) \oplus \mathrm{H}_L(A_2, C_2) = Z$
```

Fig. 2. Security game for AXU security of H.

Let $\mathsf{Adv}^{\mathsf{sl}}_D(q, S)$ denote the maximal value of $\mathsf{Adv}^{\mathsf{sl}}_D(\mathcal{A})$ for all $\mathcal{A}$ that are S-bounded and make q non-repeating queries to their oracle. Jaeger and Tessaro [11] showed that $\mathsf{Adv}^{\mathsf{sl}}_D(q, S) \leqslant \sqrt{Sq/D}$ under a combinatorial conjecture. Later, Dinur [7] improved this to show that $\mathsf{Adv}^{\mathsf{sl}}_D(q, S) \in O(Sq\log(q)/D)$.

An immediate application of the switching lemma is that if $\mathcal{A}$ is an S-bounded adversary which makes q non-repeating queries to its oracle, then $|\mathsf{Adv}^{\mathsf{prf}}_{\mathrm{E}}(\mathcal{A}) - \mathsf{Adv}^{\mathsf{prp}}_{\mathrm{E}}(\mathcal{A})| \leqslant \mathsf{Adv}^{\mathsf{sl}}_D(q, S)$ for any block-cipher E whose range has size D.

AXU HASH FUNCTION. Let $\mathrm{H} : \{0,1\}^k \times (\{0,1\}^* \times \{0,1\}^*) \to \{0,1\}^n$ be a hash function. Its almost XOR-universal (AXU) security is defined by the game $\mathsf{G}^{\mathsf{axu}}_{\mathrm{H}}$ shown in Fig. 2. In it, an adversary $\mathcal{X}$ attempts to guess the xor of the output of H on two distinct inputs of its choosing for a random key L. We define $\mathsf{Adv}^{\mathsf{axu}}_{\mathrm{H}}(\mathcal{X}) = \mathsf{Pr}[\mathsf{G}^{\mathsf{axu}}_{\mathrm{H}}(\mathcal{X})]$. Typically one makes use of a *c-AXU* hash which for all $\mathcal{X}$ satisfy $\mathsf{Adv}^{\mathsf{axu}}_{\mathrm{H}}(\mathcal{X}) \leqslant c \cdot (N_1 + N_2)/2^n$ where N_1 (resp. N_2) is the maximum block length of any A (resp. C) output by $\mathcal{X}$. Note this is unconditional, so we will not have to worry about memory complexity when reducing to AXU security.

3 Nonce-based Encryption and Memory-boundedness

In this section we recall known definitions and results for nonce-based encryption [15]. We carefully consider how these change when we move to the memory-bounded setting. For example, as was previously noted by Auerbach, et al. [1], definitions which are tightly equivalent when the memory usage of adversaries is not bounded do not necessarily remain so with bounds on memory. So we will consider several variants of the definitions we are recalling and try to reason about which is the "correct" one to use. We additionally note some results which can be extended to give appealing time-memory tradeoffs in the memory-bounded setting and some for which this does not seem to be possible.

In Section 3.1, we discuss INDR security which measures the indistinguishability of ciphertexts from truly random ones. This security notion requires that the adversary be disallowed from repeating nonces. We discuss three conventions for capturing this which are tightly equivalent when ignoring memory restrictions, but observe they are no longer tightly equivalent with these restrictions. Based on these discussions, the rest of the paper focuses on the restricted class of adversaries that will never repeat nonces in their queries to encryption oracles. In Section 3.2, we discuss CTXT (integrity of ciphertexts) and AE security (combined INDR and CTXT) security. For these, the adversary must be disallowed from trivially winning by forwarding ciphertexts from its encryption oracle to its decryption oracle. Again we discuss several conventions for this which are tightly equivalent when ignoring memory restrictions. Based on these discussions, the rest of the paper will use the convention that if an adversary queries (N, C) to its decryption oracle after receiving C from an encryption query for (N, M), the oracle will respond with M. With our chosen conventions, it does not appear to be possible to prove that AE security is implied by INDR and CTXT security with a memory-tight reduction. The rest of the paper will focus on this (im)possibility. Section 4 shows it *is* possible in the restricted setting of secure channels while Section 5 shows it is not possible for general nonce-based encryption if the reduction behaves in a black-box manner.

Finally, in Section 3.3 we recall the CAU scheme by Bellare and Tackmann [4], an abstraction of GCM [12]. Following existing proofs [4,10,12] and using [11,7], we show that INDR security of CAU can be proven by a memory-tight reduction to PRP security with an appealing time-memory tradeoff and we informally discuss why such reductions seem impossible for CTXT or AE security.

Syntax and correctness. A (nonce-based) encryption scheme NE is defined by algorithms $\mathsf{NE.Kg}$, $\mathsf{NE.D}$, and $\mathsf{NE.E}$. Additionally it is associated with message space $\mathsf{NE.M} \subseteq \{0,1\}^*$ and nonce space $\mathsf{NE.N}$.

The syntax of the algorithms is shown in Fig. 3. The key generation algorithm $\mathsf{NE.Kg}$ takes no input and returns key K. The encryption algorithm $\mathsf{NE.E}$ takes key K, nonce $N \in \mathsf{NE.N}$, and message $M \in \mathsf{NE.M}$. It returns ciphertext C. The decryption algorithm $\mathsf{NE.D}$ takes key K, nonce $N \in \mathsf{NE.N}$, and ciphertext C. It returns message $M \in \mathsf{NE.M} \cup \{\bot\}$. When $M = \bot$, the ciphertext is rejected as invalid.

We additionally assume there is a ciphertext-length function $\mathsf{NE.cl} : \mathbb{N} \to \mathbb{N}$ such that for any $K \in [\mathsf{NE.Kg}]$, $N \in \mathsf{NE.N}$, and $M \in \mathsf{NE.M}$ we have $|C| = \mathsf{NE.cl}(|M|)$ whenever $C \leftarrow \mathsf{NE.E}(K, N, M)$. Typically, a nonce-based encryption scheme also takes associated data as input which is authenticated during encryption. Associated data does not meaningfully effect our results, so we have omitted it for simplicity of notation.

Correctness of an encryption scheme requires for all $K \in [\mathsf{NE.Kg}]$, $N \in \mathsf{NE.N}$, and $M \in \mathsf{NE.M}$ that $\mathsf{NE.D}(K, N, \mathsf{NE.E}(K, N, M)) = M$.

```
NE Syntax
$K \leftarrow\!\!\$\ \mathsf{NE.Kg}$
$C \leftarrow \mathsf{NE.E}(K, N, M)$
$M \leftarrow \mathsf{NE.D}(K, N, C)$
```

Fig. 3. Syntax of nonce-based encryption scheme.

3.1 Indistinguishability From Random (INDR) Security

The first security notion we will consider requires that ciphertexts output by the encryption scheme cannot be distinguished from ciphertexts chosen at random.

Definitions. Consider the game $\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}$ shown in Fig. 4. Here an adversary $\mathcal{A}$ is given access to an encryption oracle Enc to which it can query a pair (N, M) and receive back either the encryption of message M with nonce N $(b = 1)$ or a random string of the appropriate length $(b = 0)$. The adversary outputs a bit trying to guess which of these two views it was given. We define $\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},0}(\mathcal{A})]$.

In defining security we must address how to handle the possibility of $\mathcal{A}$ making multiple queries with the same nonce. Encryption schemes are typically designed under the assumption that the same nonce will not be used multiple times and may become completely insecure in the face of such nonce repetition. The primary convention we will adopt is to restrict attention to adversaries that will never repeat nonces in their encryption queries. We use the phrase "nonce-respecting INDR" to refer to security with respect to such adversaries.

An alternate approach would be to modify the code of the game to respond appropriately to queries where nonces repeat. One version of this, which we will refer to as INDR-R, would restrict attention to adversaries that will only repeat nonces when they also repeat the message queried to encryption. For this the game would be modified to keep track of all encryption queries that have been made so far. When it receives a repeated (N, M) pair, it simply returns the same C that it returned last time it saw that pair. A second version of this, which we will refer to as INDR-B, makes no restriction on the queries of the adversary. Instead, the game is modified to return $\bot$ whenever the adversary makes a query with a nonce it has already used.

```
Game $\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}(\mathcal{A})$
    $K \leftarrow\!\!\$\ \mathsf{NE.Kg}$
    $b' \leftarrow \mathcal{A}^{\textsc{Enc}_b}$
    Return $b' = 1$

Oracle $\textsc{Enc}_b(N, M)$
    $C_1 \leftarrow \mathsf{NE.E}(K, N, M)$
    $C_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{NE.cl}(|M|)}$
    $M[N, C_b] \leftarrow M$
    Return $C_b$

Game $\mathsf{G}^{\mathsf{ctxt}\text{-}w}_{\mathsf{NE},b}(\mathcal{A})$
    $K \leftarrow\!\!\$\ \mathsf{NE.Kg}$
    $b' \leftarrow \mathcal{A}^{\textsc{Enc}_1, \textsc{Dec}^w_b}$
    Return $b' = 1$

Game $\mathsf{G}^{\mathsf{ae}\text{-}w}_{\mathsf{NE},b}(\mathcal{A})$
    $K \leftarrow\!\!\$\ \mathsf{NE.Kg}$
    $b' \leftarrow \mathcal{A}^{\textsc{Enc}_b, \textsc{Dec}^w_b}$
    Return $b' = 1$

Oracle $\textsc{Dec}^w_b(N, C)$
    If $M[N, C] \neq \perp$ then
        Return $M[N, C]$ if $w = 1$
        Return $\diamond$ if $w = 2$
        Return $\perp$ if $w = 3$
    $M_1 \leftarrow \mathsf{NE.D}(K, N, C)$
    $M_0 \leftarrow \perp$
    Return $M_b$
```

Fig. 4. Games defining INDR, CTXT-w, and AE-w security of NE for $w \in \{1, 2, 3\}$.

Discussion. When memory is not an issue, all of these variants would be equivalent. Proving this follows by noting that an adversary can just remember all prior queries it has made and thus never need to repeat. This proof strategy is no longer available to us when we want to preserve the memory usage of adversaries. We focus on nonce-respecting INDR because it hits the sweet spot of being strong enough for common applications, yet weak enough that we know how to give provable time-memory trade-offs.

Because nonce-respecting INDR considers a strictly smaller class of adversaries than the other two and all of the games behave identically for this class of adversary it is tightly implied by the others. In fact, using ideas from [11,7] we can see that nonce-respecting INDR is strictly weaker. The toy encryption scheme SE considered in the introduction built from a block-cipher with block length n is vulnerable to low-memory collision-finding attacks with advantage $\Omega(q^2/2^n)$ in the INDR-R and INDR-B settings, but no attacks can have advantage better than $O(qs/2^n)$ in the nonce-respecting INDR setting. Here q and s refer to the number of queries and amount of memory used by the attackers, respectively. This underlies why the ideas of Jaeger and Tessaro [11] can be used to prove nonce-respecting INDR (but not INDR-R or INDR-B) time-memory trade-offs for natural counter-mode based encryption schemes. In most common uses of nonce-based encryption the nonces are incremented as a counter or picked uniformly at random. In the former case, nonces clearly never repeat so nonce-respecting INDR suffices (we will see this formally in Section 4). Nonces may repeat in the latter case, but we can follow [11,7] here and replace the uniform random values with random, non-repeating values so again nonce-respecting INDR suffices.

3.2 Security Beyond Confidentiality

INDR security only guarantees confidentiality of the messages against passive attackers. However, in practice, attackers may actively modify ciphertexts in transit. As such, it is important to consider security definition that take this into account. We will consider integrity definitions and authenticated encryption definitions which simultaneously asked for integrity and confidentiality.

Definitions. Consider the other two games shown in Fig. 4. We will first focus on $\mathsf{G}^{\mathsf{ae}\text{-}w}_{\mathsf{NE},b}$ which defines three variants of authenticated encryption security parameterized by $w \in \{1, 2, 3\}$. In this game, the adversary is given access to an encryption oracle and a decryption oracle. Its goal is to distinguish between a "real" and "ideal" world. In the real world ($b = 1$) the oracles uses NE to encrypt messages and decrypt ciphertexts. In the ideal world ($b = 0$) encryption returns random messages of the appropriate length and decryption returns $\perp$. For simplicity, we will restrict attention nonce-respecting adversaries which do not repeat nonces across encryption queries (as in nonce-respecting INDR security). Note there is no restriction placed on nonces used for decryption queries. Integrity of ciphertext security is defined by $\mathsf{G}^{\mathsf{ctxt}\text{-}w}_{\mathsf{NE},b}$ which behaves similarly except the adversary is always given access to the real encryption algorithm.

The decryption oracle needs to prevent trivial attacks. If the adversary receives C from a query of $\textsc{Enc}(N, M)$ and then queries $\textsc{Dec}(N, C)$ it would receive M in the real world and $\perp$ in the ideal world, making them easy to distinguish. We must adopt some convention for how the oracles behave when such a query is made to prevent this type of trivial attack. Towards this, the decryption oracle is parameterized by the value $w \in \{1, 2, 3\}$ corresponding to three different security notions. In all three, we use a table $M[\cdot, \cdot]$ to detect when the adversary forwards encryption queries on to its decryption oracle. When $w = 1$, the decryption oracle returns $M[N, C]$ in this case. When $w = 2$, it returns a special symbol $\diamond$. When $w = 3$, it returns the symbol $\perp$ which is also used by the encryption scheme to represent rejection. For $x \in \{\mathsf{ae}, \mathsf{ctxt}\}$ and $w \in \{1, 2, 3\}$ we define the advantage of an adversary $\mathcal{A}$ by $\mathsf{Adv}^{x\text{-}w}_{\mathsf{NE}}(\mathcal{A}) = \Pr[\mathsf{G}^{x\text{-}w}_{\mathsf{NE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{x\text{-}w}_{\mathsf{NE},0}(\mathcal{A})]$. The corresponding security notions are referred to as AE-w and CTXT-w.

Discussion. When memory usage is not an issue, the choice of w does not matter. We can without loss of generality assume that the adversary never makes one of these trivial attack queries because it could simply store the table $M[\cdot, \cdot]$ for itself and simulate any such queries.[4] It's not clear that this equivalence holds if we do not assume that storing $M[\cdot, \cdot]$ is "free" for the adversary.

The only memory-tight implication we are aware of between these is that security for $w = 2$ tightly implies security for $w = 3$. This follows because an adversary with access to $\textsc{Dec}^2_b$ can simulate $\textsc{Dec}^3_b$ with low memory. If $\textsc{Dec}^2_b$ returns $M = \diamond$ the adversary returns $\perp$, otherwise it does not modify M. All of the other implications we might want to show seem to require remembering all prior encryption queries to properly simulate $\textsc{Dec}$.

Ultimately, for heuristic reasons, we believe that $w = 1$ is the "correct" choice and will focus on it in our later sections. The typical motivation behind chosen-ciphertext security notions is that in practice an attacker can often observe the behavior of the decrypting party to learn something about the message they received. There is no reason to think an attacker should only be able to do that for ciphertexts that have been modified, but not ciphertexts that have been unmodified. This is best captured by $w = 1$. The $w = 2$ definition seems to posit that the adversary can distinguish between ciphertexts it forwarded on and ciphertexts that it modified (whether they were accepted or rejected) by observing the decrypting party's behavior. The $w = 3$ definition seems to posit that the adversary cannot learn anything about ciphertexts it forwards on unmodified, but can learn about other modified ciphertexts by observing the decrypting party's behavior.

Revisiting a classic result. A classic result, which has been shown for numerous styles of encryption, is that confidentiality and integrity together imply authenticated encryption [16]. However, this becomes more difficult for nonce-based encryption when we consider memory-tightness.

The classic proof that INDR and CTXT-1 security imply AE-1 security first replaces real decryption with $\perp$ via a reduction to CTXT-1 security and then replace real encryption with random using INDR security. However, in this second step the reduction adversary would have to simulate the oracle $\textsc{Dec}^1_0$ which seems to require storing the table $M[\cdot, \cdot]$.[5] This potentially requires using much more memory than the AE-1 adversary, losing the benefit of time-memory tradeoffs for INDR-R. The rest of the paper is dedicated to understanding this reduction. In Section 4.2, we make it memory tight when restricting attention to secure channels which only accept ciphertexts if they are received in order. In Section 5, we give negative results showing that for nonce-based encryption this reduction cannot be made memory tight (using a black-box reduction).

3.3 Security of the CAU Encryption Scheme

We conclude this section by considering the specific encryption scheme CAU for which we can prove INDR security with a time-memory tradeoff. We will use this scheme in Section 4 to show a time-memory tradeoff for the authenticated encryption security of a channel instantiated with it.

One of the most widely deployed encryption schemes is Galois Counter-Mode (GCM) [12]. Bellare and Tackmann [4] generalized it to the scheme CAU which constructs an encryption scheme from a block cipher E and hash function H. Using the techniques of Jaeger and Tessaro [11] we obtain a proof of security for its nonce-respecting INDR security with an appealing time-memory tradeoff.

[4] Restricting attention to adversaries which never make trivial attack queries is, indeed, a fourth way one could define security.

[5] The standard reduction *would* be memory tight for $w = 3$.

```
Algorithm $\mathsf{CAU}[\mathrm{E},\mathrm{H}].\mathsf{E}(K,N,M)$
$Y \leftarrow \mathrm{pad}(N)$
$M_1 \ldots M_\ell \leftarrow_n M$
For $i = 1, ..., \ell$ do
    $C_i \leftarrow M_i \oplus E_K(Y+i)$
$C \leftarrow C_1 \ldots C_\ell$
$L \leftarrow E_K(0^n)$
$T \leftarrow H_L(A,C) \oplus E_K(Y)$
Return $T \,\|\, C$
```

```
Algorithm $\mathsf{CAU}[\mathrm{E},\mathrm{H}].\mathsf{D}(K,N,T \,\|\, C)$
$L \leftarrow E_K(0^n)$; $Y \leftarrow \mathrm{pad}(N)$
$C_1 \ldots C_\ell \leftarrow_n C$
$T' \leftarrow H_L(A,C) \oplus E_K(Y)$
If $T \neq T'$ then return $\perp$
For $i = 1, ..., \ell$ do
    $M_i \leftarrow C_i \oplus E_K(Y+i)$
$M \leftarrow M_1 \ldots M_\ell$
Return $M$
```

Fig. 5. Encryption scheme CAU parameterized by function family E (typically a block cipher) and hash function H. In the code, $\mathrm{pad}(N) = N \,\|\, 0^m \,\|\, 1$ for the appropriate choice of m and $M_1 \ldots M_\ell \leftarrow_n M$ splits M into n-bit blocks.

Construction. We recall the CAU construction of an encryption scheme. Fix a key length $\mathsf{CAU.kl} \in \mathbb{N}$, a block length $n = \mathsf{CAU.bl} \in \mathbb{N}$, and a nonce length $\mathsf{CAU.nl} < \mathsf{CAU.bl}$. Then let E be a function family with $\mathrm{E} : \{0,1\}^{\mathsf{CAU.kl}} \times \{0,1\}^{\mathsf{CAU.bl}} \to \{0,1\}^{\mathsf{CAU.bl}}$ and H be a function family with $\mathrm{H} : \{0,1\}^{\mathsf{CAU.bl}} \times (\{0,1\}^* \times \{0,1\}^*) \to \{0,1\}^{\mathsf{CAU.bl}}$. The scheme constructed from E and H is denoted $\mathsf{CAU}[\mathrm{E},\mathrm{H}]$. Its message space $\mathsf{CAU}[\mathrm{E},\mathrm{H}].\mathsf{M}$ is the set of all strings of length at most $n \cdot (2^{n-\mathsf{CAU.nl}} - 1)$ and its nonce space $\mathsf{CAU}[\mathrm{E},\mathrm{H}].\mathsf{N}$ is the set $\{0,1\}^{\mathsf{CAU.nl}}$.

The algorithms of $\mathsf{CAU}[\mathrm{E},\mathrm{H}]$ are shown in Fig. 5. The code uses $\mathrm{pad}(\cdot)$ to denote the padding function which on input N outputs $N \,\|\, 0^{n-\mathsf{CAU}[\mathrm{E},\mathrm{H}].\mathsf{nl}-1} \,\|\, 1$. Since our simplified notation does not use associated data we instead assume there is a fixed associated data string A used with every message.

The encryption algorithm parses the input message into ℓ blocks of length n (except for the last, which may be shorter) and pads the nonce to a string Y of length n. It encrypts the message using counter-mode encryption with $Y+1$ as the first counter. This gives it a partial ciphertext C. The authentication is inspired by a Carter-Wegman MAC. A key L for the hash function is obtained as $L \leftarrow E_K(0^n)$. This key is used to compute the tag T as $T \leftarrow H_L(A,C) \oplus E_K(Y)$ and then $T \,\|\, C$ is the full ciphertext output by encryption.

The decryption algorithm parses the input ciphertext as $T \,\|\, C$. It computes the correct tag T' for C by setting $L \leftarrow E_K(0^n)$ and $T \leftarrow H_L(A,C) \oplus E_K(Y)$ (as was done in encryption). If $T \neq T'$ the ciphertext is rejected by returning $M = \perp$. Otherwise the message M is obtained by counter-mode decrypting C.

INDR security of CAU. The following theorem formalizes that CAU is nonce-respecting INDR secure assuming E is a secure PRF.

Theorem 1. *Let $\mathcal{A}$ be an adversary against the nonce-respecting* INDR *security of* $\mathsf{CAU}[\mathrm{E},\mathrm{H}]$ *that makes at most q oracle queries, each at most $p \cdot \mathsf{CAU.bl}$ bits long. Then we can construct a $\mathcal{A}_{\mathsf{prf}}$ such that*

$$\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{CAU}[\mathrm{E},\mathrm{H}]}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{E}}(\mathcal{A}_{\mathsf{prf}}) .$$

Adversary $\mathcal{A}_{\mathsf{prf}}$ has runtime essentially that of $\mathcal{A}$, makes at most $q(p+1)+1$ queries to its oracle, has memory/time complexity essentially that of $\mathcal{A}$ and never repeats queries to its oracle.

It is important that $\mathcal{A}_{\mathsf{prf}}$ never repeats queries because it allows us to apply the time-memory switching lemma from Section 2. This gives us roughly,

$$\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{CAU}[\mathrm{E},\mathrm{H}]}(\mathcal{A}) \in \mathsf{Adv}^{\mathsf{prp}}_{\mathrm{E}}(\mathcal{A}_{\mathsf{prf}}) + O(S \cdot pq \cdot \log(pq)/2^n)$$

where S is a bound on the memory complexity of $\mathcal{A}$. For variants other than nonce-respecting INDR it would not be clear how to prevent $\mathcal{A}_{\mathsf{prf}}$ from repeating queries without storing the prior queries of $\mathcal{A}$.

Proof (Sketch). One constructs $\mathcal{A}_{\mathsf{prf}}$ to first set $L \leftarrow \mathrm{E{\small VAL}}(0^n)$. Then it runs $\mathcal{A}$ and simulates encryption queries by running $\mathsf{CAU.E}$ while using its $\mathrm{E{\small VAL}}$ oracle in place of E_K. It does not recompute L each time because it has already computed it. Its final output is whatever $\mathcal{A}$ outputs. One can verify that the view of $\mathcal{A}$ when simulated by $\mathcal{A}_{\mathsf{prf}}$ is "real" encryptions when $b = 1$ and random strings when $b = 0$, so the claimed advantage bound follows. □

```
CH Syntax
$(\sigma^s, \sigma^r) \leftarrow\$ \mathsf{CH.Sg}$
$(\sigma^s, C) \leftarrow\$ \mathsf{CH.S}(\sigma^s, M)$
$(\sigma^r, M) \leftarrow \mathsf{CH.R}(\sigma^r, C)$
```

```
Game $\mathsf{G}^{\mathsf{ch\text{-}corr}}_{\mathsf{CH},b}(\mathcal{A})$
$(\sigma^s, \sigma^r) \leftarrow\$ \mathsf{CH.Sg}$
$b' \leftarrow\$ \mathcal{A}^{\text{EncDec}}$
Return $b' = 1$
```

```
Oracle $\text{EncDec}_b(M_0)$
$(\sigma^s, C) \leftarrow\$ \mathsf{CH.S}(\sigma^s, M_0)$
$(\sigma^r, M_1) \leftarrow \mathsf{CH.R}(\sigma^r, C)$
Return $M_b$
```

Fig. 6. Left: Syntax of channel algorithms. **Right:** Channel correctness game.

CTXT/AE SECURITY OF CAU. It does not appear to be possible to give a similar time-memory trade-off for the CTXT or AE security of CAU. The standard analysis of either of these first uses PRF security to replace the output of E with random. It then argues that the adversary's view is independent of the $\mathrm{H}_L(A, C)$ values produced in encryption so that it can apply the security of H. For $x = \mathsf{ae}$ or $x = \mathsf{ctxt}$ this would give a bound of the form,

$$\mathsf{Adv}^{x\text{-}1}_{\mathsf{CAU}[\mathrm{E},\mathrm{H}]}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{E}}(\mathcal{A}_{\mathsf{prf}}) + \mathsf{Adv}^{\mathsf{axu}}_{\mathrm{H}}(\mathcal{X}) \; .$$

However, this PRF adversary $\mathcal{A}_{\mathsf{prf}}$ needs to to simulate a decryption oracle to $\mathcal{A}$. The natural ways of doing this (remembering all prior encryption queries or using EVAL to run decryption) either require significant use of memory or repeating queries to EVAL. This prevents us from applying the switching lemmas of [11,7] to get appealing time-memory tradeoffs when E is a PRP.

In Section 4.3, we will use a new technique for memory-tight reductions to prove that using CAU in a channel can provide (the channel equivalent of) CTXT security (and thus AE security from Section 4.2).

4 Memory-tight Reductions for Cryptographic Channels

In this section we show that memory-tight reductions can be given for the combined confidentiality and integrity security of cryptographic channels. These are a form of stateful encryption which provide the guarantee that messages cannot be duplicated or reordered, in addition to the typical confidentiality and integrity goals of encryption.

4.1 Syntax and Security Notions

SYNTAX AND CORRECTNESS. A (cryptographic) channel CH specifies algorithms $\mathsf{CH.Sg}$, $\mathsf{CH.S}$, and $\mathsf{CH.R}$ along with message space $\mathsf{CH.M} \subseteq \{0,1\}^*$. The syntax of these algorithms is shown in Fig. 6. The state generation algorithm $\mathsf{CH.Sg}$ takes no input. It returns sender state σ^s and receiver state σ^r. The sending algorithm $\mathsf{CH.S}$ takes a sender state σ^s and message $M \in \mathsf{CH.M}$. It returns updated sender state σ^s and a ciphertext C. The receiving algorithm $\mathsf{CH.R}$ takes a receiver state σ^r and a ciphertext C. It returns updated receiver state σ^r and a message $M \in \mathsf{CH.M} \cup \{\bot\}$. When $M = \bot$, this represents the receiver rejecting the message as invalid.

A channel is expected to never again return $M \neq \bot$ after if it has rejected a message. This models the behavior of protocols such as TLS which are assumed to be run over a reliable transport layer and has been the standard notion for channels since the work of Bellare, Kohno, and Namprempre [2]. When a protocol (e.g. QUIC or DTLS) is run over an unreliable transport layer, then a *robust* channel is used instead [8]. We leave memory-tight proofs of security for robust channels as an interesting direction for future work.

We typically assume there is a ciphertext-length function $\mathsf{CH.cl} : \mathbb{N} \to \mathbb{N}$ such that for any $M \in \mathsf{CH.M}$ and state σ^s, we have $\Pr[|C| = \mathsf{CH.cl}(|M|) : (\sigma^s, C) \leftarrow\$ \mathsf{CH.S}(\sigma^s, M)] = 1$.

Correctness requires that if the receiver is given the ciphertexts sent by the sender in order and without modification then the receiver will output the same sequence of messages that were sent. One way to formalize this is via the game $\mathsf{G}^{\mathsf{ch\text{-}corr}}_{\mathsf{CH},b}$ shown in Fig. 6. We define $\mathsf{Adv}^{\mathsf{ch\text{-}corr}}_{\mathsf{CH}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{ch\text{-}corr}}_{\mathsf{CH},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{ch\text{-}corr}}_{\mathsf{CH},0}(\mathcal{A})]$. Perfect correctness requires that $\mathsf{Adv}^{\mathsf{ch\text{-}corr}}_{\mathsf{CH}}(\mathcal{A}) = 0$ for all (even unbounded) $\mathcal{A}$. This implies that the M_1 output by $\mathsf{CH.R}$ always equals M_0.

```
Game $\mathsf{G}^{\text{ch-ae}}_{\mathsf{CH},b}(\mathcal{A})$
    $\mathsf{sync} \leftarrow \mathtt{true}$
    $(\sigma^s, \sigma^r) \leftarrow\$ \mathsf{CH.Sg}$
    $b' \leftarrow\$ \mathcal{A}^{\text{Enc}_b,\text{Dec}_b}$
    Return $b' = 1$

Game $\mathsf{G}^{\text{ch-indr}}_{\mathsf{CH},b}(\mathcal{A})$
    $\mathsf{sync} \leftarrow \mathtt{true}$
    $(\sigma^s, \sigma^r) \leftarrow\$ \mathsf{CH.Sg}$
    $b' \leftarrow\$ \mathcal{A}^{\text{Enc}_b}$
    Return $b' = 1$

Game $\mathsf{G}^{\text{ch-ctxt}}_{\mathsf{CH},b}(\mathcal{A})$
    $\mathsf{sync} \leftarrow \mathtt{true}$
    $(\sigma^s, \sigma^r) \leftarrow\$ \mathsf{CH.Sg}$
    $b' \leftarrow\$ \mathcal{A}^{\text{Enc}_1,\text{Dec}_b}$
    Return $b' = 1$

Oracle $\text{Enc}_b(M)$
    $(\sigma^s, C_1) \leftarrow\$ \mathsf{CH.S}(\sigma^s, M)$
    $C_0 \leftarrow\$ \{0,1\}^{\mathsf{CH.cl}(|M|)}$
    $\mathbf{M}.\mathsf{add}(M)$; $\mathbf{C}.\mathsf{add}(C_b)$
    Return $C_b$

Oracle $\text{Dec}_b(C)$
    $(\sigma^r, M_1) \leftarrow \mathsf{CH.R}(\sigma^r, C)$
    $M_0 \leftarrow \perp$
    $M' \leftarrow \mathbf{M}.\mathsf{dq}()$
    $C' \leftarrow \mathbf{C}.\mathsf{dq}()$
    If $\mathsf{sync}$ then
        If $C = C'$ then
            Return $M'$
        $\mathsf{sync} \leftarrow \mathtt{false}$
    Return $M_b$
```

Fig. 7. Games defining the INDR, CTXT, and AE security of a channel.

```
Game $\mathsf{G}^{\mathsf{it}}_{L,\delta}(\mathcal{A}_1, \mathcal{A}_2)$
    $R \leftarrow\$ \{0,1\}^L$
    $(i, \sigma) \leftarrow\$ \mathcal{A}_1(R)$
    $r \leftarrow\$ \mathcal{A}_2(i, \sigma, R[: i-1])$
    Return $r = R[i : i+\delta]$
```

Fig. 8. Information theoretic game in which $\mathcal{A}$ tries to remember a δ bit sequence in an L-bit random string.

Security definitions. We consider indistinguishability from random, integrity of ciphertext, and authenticated encryption security for channels just like we did for nonce based encryption.

Authenticated encryption security of a channel CH is defined by game $\mathsf{G}^{\text{ch-ae}}_{\mathsf{CH},b}$ defined in Fig. 7. In it the adversary is given access to an encryption oracle and a decryption oracle. The adversary's goal is to distinguish between a "real" and "ideal" world. In the real world ($b = 1$) the oracles use CH to encrypt messages and decrypt ciphertexts. In the ideal world ($b = 0$) encryption returns random messages of the appropriate length and decryption returns $\perp$. In both worlds, as long as the adversary's queries to decryption have consisted of the outputs of encryption in the correct order, the oracles are considered in sync and decryption just returns the appropriate message that was queried to encryption.[6] After the first time the adversary queries something else, the oracles are out of sync and will never be in sync again (so Dec will always return M_b).

Authenticated encryption security is a combined confidentiality and integrity notion. We can also define separate notions. INDR security is defined by the game $\mathsf{G}^{\text{ch-indr}}_{\mathsf{CH},b}$ which is the same as $\mathsf{G}^{\text{ch-ae}}_{\mathsf{CH},b}$ except the adversary is only given oracle access to Enc_b. CTXT security is defined by the game $\mathsf{G}^{\text{ch-ctxt}}_{\mathsf{CH},b}$ which is the same as $\mathsf{G}^{\text{ch-ae}}_{\mathsf{CH},b}$ except the adversary is given oracle access to Enc_1 and Dec_b. These games are given explicitly in Fig. 7. We define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^x_{\mathsf{CH}}(\mathcal{A}) = \Pr[\mathsf{G}^x_{\mathsf{CH},1}(\mathcal{A})] - \Pr[\mathsf{G}^x_{\mathsf{CH},0}(\mathcal{A})]$ for $x \in \{\text{ch-ae}, \text{ch-indr}, \text{ch-ctxt}\}$.

4.2 Confidentiality and Integrity Imply Authenticated Encryption

We will show that INDR security plus CTXT security imply AE security using a memory-tight reduction. While the normal proof that INDR and CTXT security suffice to imply AE security is not particularly difficult, it uses a non-memory tight reduction to INDR security. Making the proof memory tight will require more involved analysis.

Information theoretic lemma. Before proceeding to the proof, we first will provide a simple information theoretic lemma that will be a useful subcomponent of that proof. Consider the game $\mathsf{G}^{\mathsf{it}}_{L,\delta}$ shown in Fig. 8. In it, an adversary is given a length L string R and tries to choose an index i for which it is able to remember the next δ-bits of the string using state σ. We say that an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ is S-bounded if $|\sigma| = S$ always. We define $\mathsf{Adv}^{\mathsf{it}}_{L,\delta}(\mathcal{A}_1, \mathcal{A}_2) = \Pr[\mathsf{G}^{\mathsf{it}}_{L,\delta}(\mathcal{A}_1, \mathcal{A}_2)]$.

Lemma 1. *Let $L, \delta, S \in \mathbb{N}$. Let $(\mathcal{A}_1, \mathcal{A}_2)$ be an S-bounded adversary. Then*

$$\mathsf{Adv}^{\mathsf{it}}_{L,\delta}(\mathcal{A}_1, \mathcal{A}_2) \leqslant L \cdot 2^S / 2^\delta \; .$$

[6] This matches the convention of CTXT-1 and AE-1 for encryption schemes. We believe it to be "correct" for the same reasons discussed for those definitions.

Proof. Let $L, \delta, S, \mathcal{A}_1, \mathcal{A}_2$ be defined as in the theorem statement. Without loss of generality we can assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic. Then for any fixed choice of i and σ, the probability that $\mathcal{A}_2(i, \sigma, R[: i-1]) = R[i : i+\delta]$ will be exactly $1/2^\delta$. Then we can calculate as follows.

$$\begin{aligned}\Pr[\mathsf{G}^{\mathsf{it}}_{L,n}(\mathcal{A}_1, \mathcal{A}_2)] &\leqslant \Pr_R[\exists i, \sigma \text{ s.t. } R[i : i+\delta] = \mathcal{A}_2(i, \sigma, R[: i-1])] \\ &\leqslant \sum_{i,\sigma} \Pr[R[i : i+\delta] = \mathcal{A}_2(i, \sigma, R[: i-1])] \\ &= \sum_{i,\sigma} 1/2^\delta \leqslant L \cdot 2^S/2^\delta \ .\end{aligned}$$

The last inequality follows from there being at most $L \cdot 2^\delta$ choices for (i, σ). □

Security result. Now we can proceed to our security result showing that AE security can be implied by INDR and CTXT security in a memory-tight manner. The technical crux of the result is the reduction adversary $\mathcal{A}_\delta$ which simulates the view of an AE adversary $\mathcal{A}$ to attack the INDR security of the channel. In our theorem statement this reduction adversary is parameterized by a variable δ which determines how much local memory it uses. Using Lemma 1, our concrete advantage bound is expressed in terms of δ and establishes that the reduction can be successful with this value not much larger than the local memory of $\mathcal{A}$.

Theorem 2. *Let* CH *be a cryptographic channel. Let* $\mathcal{A}$ *be an adversary with memory complexity* S *and making at most* q *queries to its* Enc *oracle, each of which returns a ciphertext of length at most* x*. Then for any* $\delta \in \mathbb{N}$ *we can build an adversary* $\mathcal{A}_\delta$ *(described in the proof) such that*

$$\mathsf{Adv}^{\mathsf{ch\text{-}ae}}_{\mathsf{CH}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{ch\text{-}ctxt}}_{\mathsf{CH}}(\mathcal{A}) + 2 \cdot \mathsf{Adv}^{\mathsf{ch\text{-}indr}}_{\mathsf{CH}}(\mathcal{A}_\delta) + 2q \cdot x \cdot 2^S/2^\delta \ .$$

Adversary $\mathcal{A}_\delta$ *has running time approximately that of* $\mathcal{A}$ *and uses about* $S + 2\delta$ *bits of state.*

Setting $\delta = S + \log(qx) + \kappa$ makes the last term about $1/2^\kappa$ while limiting the memory usage of $\mathcal{A}_\delta$ to only $2S + 2\log(qx) + 2\kappa$.

The standard way of proving that INDR security and CTXT security imply AE security would first use CTXT security to transition from a world in which $\mathcal{A}$ is given oracle access to $(\text{Enc}_1, \text{Dec}_1)$ to a world in which $\mathcal{A}$ is given oracle access to $(\text{Enc}_1, \text{Dec}_0)$. Then INDR security would be used to transition to $\mathcal{A}$ being given oracle access to $(\text{Enc}_0, \text{Dec}_0)$. The issue in our setting with this proof arises in the second step. The INDR reduction adversary needs to simulate Dec_0 for $\mathcal{A}$. The natural way of doing so requires storing the entirety of the tables $\mathbf{M}$ and $\mathbf{C}$ which means that $\mathcal{A}_\delta$ may use much more memory than $\mathcal{A}$.

Our proof of Thm. 2 follows this same general proof flow, but uses a more involved analysis for the reduction to INDR security. In particular, we make use of the following insight: If $\mathcal{A}$ has memory complexity S but cannot distinguish the ciphertexts it sees from random (because of INDR security), then from Lemma 1 it cannot remember many more than S of the ciphertext bits that it has received from Enc but not yet forwarded to Dec.

If $\mathcal{A}$ ever queries a ciphertext which is not the next ciphertext in $\mathbf{C}$, then Dec_0 oracle will never again return anything other than $\perp$. Because we can assume that $\mathcal{A}$ will be unable to remember too many bits of ciphertext, we can just have our reduction adversary $\mathcal{A}_\delta$ remember a few more bits of ciphertext than $\mathcal{A}$ can. If the total length of ciphertext that $\mathcal{A}$ has received from its encryption oracle, but not forwarded on to its decryption oracle ever exceeds the amount that $\mathcal{A}_\delta$ will store, then $\mathcal{A}_\delta$ assumes $\mathcal{A}$ must have forgotten some intermediate ciphertext before that point, allowing the reduction to cease storing future ciphertexts because sync will be `false` before that point.

Proof. We will construct INDR adversaries $\mathcal{A}'_\delta$, $\mathcal{A}''_\delta$, and S-bounded adversary $(\mathcal{A}_1, \mathcal{A}_2)$ and show that

$$\mathsf{Adv}^{\mathsf{ch\text{-}ae}}_{\mathsf{CH}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{ch\text{-}ctxt}}_{\mathsf{CH}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ch\text{-}indr}}_{\mathsf{CH}}(\mathcal{A}'_\delta) + 2 \cdot \mathsf{Adv}^{\mathsf{it}}_{q \cdot x, \delta}(\mathcal{A}_1, \mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ch\text{-}indr}}_{\mathsf{CH}}(\mathcal{A}''_\delta) \ .$$

The stated theorem then follows by applying Lemma 1 and constructing the adversary $\mathcal{A}_\delta$ which runs either $\mathcal{A}'_\delta$ or $\mathcal{A}''_\delta$ (chosen at random) and outputs whatever that adversary does. The resulting $\mathcal{A}_\delta$ will satisfy the efficiency constraints stated in the theorem statement. We will prove this bound via a sequence of transformations that slowly change $\mathsf{G}^{\mathsf{ch\text{-}ae}}_{\mathsf{CH},1}$ to $\mathsf{G}^{\mathsf{ch\text{-}ae}}_{\mathsf{CH},0}$.

```
Games $\boxed{\mathsf{G}_2}$, $\mathsf{G}_3$, $\mathsf{G}_4$, $\boxed{\mathsf{G}_5}$
$\mathsf{flag} \leftarrow \mathtt{true}$
$\mathsf{sync} \leftarrow \mathtt{true}$
$(\sigma^s, \sigma^r) \leftarrow\$ \mathsf{CH.Sg}$
$b' \leftarrow\$ \mathcal{A}^{\text{Enc},\text{Dec}}$
Return $b' = 1$
```

```
Oracle Enc$(M)$
$(\sigma^s, C) \leftarrow\$ \mathsf{CH.S}(\sigma^s, M)$
$C \leftarrow\$ \{0,1\}^{\mathsf{CH.cl}(|M|)}$
$\mathbf{M}_2.\mathsf{add}(M)$; $\mathbf{C}_2.\mathsf{add}(C)$
If $\mathsf{flag}$ then
  If $||\mathbf{C}|| + |C| < \delta$ then
    $\mathbf{M}.\mathsf{add}(M)$; $\mathbf{C}.\mathsf{add}(C)$
  Else
    $\mathsf{flag} \leftarrow \mathtt{false}$
Return $C$
```

```
Oracle Dec$(C)$
$M' \leftarrow \mathbf{M}.\mathsf{dq}()$; $M_2 \leftarrow \mathbf{M}_2.\mathsf{dq}()$
$C' \leftarrow \mathbf{C}.\mathsf{dq}()$; $C_2 \leftarrow \mathbf{C}_2.\mathsf{dq}()$
If $\mathsf{sync}$ then
  If $C = C'$ then
    Return $M'$
  Elif $C = C_2$ then
    $\mathsf{bad} \leftarrow \mathtt{true}$
    $\boxed{\text{Return } M_2}$
  $\mathsf{sync} \leftarrow \mathtt{false}$
Return $\perp$
```

Fig. 9. Hybrid games for proof of Theorem 2. Highlighted code is only included in highlighted games. Boxed code is only included in boxed games.

CTXT TRANSITION. Let $\mathsf{G}_0 = \mathsf{G}^{\mathsf{ch\text{-}ae}}_{\mathsf{CH},1}(\mathcal{A})$ and $\mathsf{G}_1 = \mathsf{G}^{\mathsf{ch\text{-}ctxt}}_{\mathsf{CH},0}(\mathcal{A})$. Because $\mathsf{G}^{\mathsf{ch\text{-}ae}}_{\mathsf{CH},1}(\mathcal{A})$ and $\mathsf{G}^{\mathsf{ch\text{-}ctxt}}_{\mathsf{CH},1}(\mathcal{A})$ are identical games we have that $\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_1] = \mathsf{Adv}^{\mathsf{ch\text{-}ctxt}}_{\mathsf{CH}}(\mathcal{A})$.

TRANSITION TO LIMITED MEMORY GAME. Next we want to transition to a version of G_1 that stores a bounded amount of local state. Consider the games G_2 and G_3 shown in Fig. 9. The tables $\mathbf{M}_2$ and $\mathbf{C}_2$ track the messages and ciphertexts as in the real game. Because of this $\Pr[\mathsf{G}_1] = \Pr[\mathsf{G}_2]$.

In the transition to G_3 we are going to stop using these tables and instead solely rely on the tables $\mathbf{M}$ and $\mathbf{C}$. With these tables, if the total number of bits of ciphertexts that would be stored in $\mathbf{C}$ exceeds δ then we permanently stop adding elements to these tables – we assume that the adversary will cause sync to be set to $\mathtt{false}$ at some point earlier in the game. Note that up until this point the tables $(\mathbf{M}_2, \mathbf{C}_2)$ and $(\mathbf{M}, \mathbf{C})$ are used identically. The two games only differ in the boxed code in DEC which returns M_2 if the adversary has queried a ciphertext stored in $\mathbf{C}_2$ that was not stored in $\mathbf{C}$. Hence, these games are identical-until-bad so the Fundamental Lemma of Game Playing [3] gives,

$$\Pr[\mathsf{G}_3] - \Pr[\mathsf{G}_2] \leqslant \Pr[\mathsf{G}_3 \text{ sets } \mathsf{bad}].$$

We want to apply Lemma 1 to bound the probability that bad is set. To do so we need to be able to treat the ciphertexts as random strings. Thus we defer the analysis of the probability that it occurs until after applying INDR security.

INDR TRANSITION. Now consider the game G_4. It is identical to G_3 except that the ciphertexts returned by ENC are chosen at random instead of using CH. We can transition to this game using a reduction to INDR security. It is important here that our reduction adversary will not need to use too much memory because of the way that we have limited the memory needed for G_3.

Consider the adversaries $\mathcal{A}_\delta$ and $\mathcal{A}'_\delta$ shown in Fig. 10. Highlighted code is only included in the latter adversary.

Adversary $\mathcal{A}'_\delta$ uses its ENC oracle to present $\mathcal{A}$ with a view identical to G_3 if $b = 1$ and identical to G_4 if $b = 0$. Note here that the tables $(\mathbf{M}_2, \mathbf{C}_2)$ do not effect the view of $\mathcal{A}$ in either of these game, allowing $\mathcal{A}_\delta$ not to have to store them. We have that $\Pr[\mathsf{G}^{\mathsf{ch\text{-}indr}}_{\mathsf{CH},1}(\mathcal{A}'_\delta)] = \Pr[\mathsf{G}_3]$ and $\Pr[\mathsf{G}^{\mathsf{ch\text{-}indr}}_{\mathsf{CH},0}(\mathcal{A}'_\delta)] = \Pr[\mathsf{G}_4]$. In other words, $\mathsf{Adv}^{\mathsf{ch\text{-}indr}}_{\mathsf{CH}}(\mathcal{A}'_\delta) = \Pr[\mathsf{G}_3] - \Pr[\mathsf{G}_4]$.

Adversary $\mathcal{A}''_\delta$ instead uses its INDR oracle to simulate the view of $\mathcal{A}$, but returns 1 if the flag bad would have been set. Because this can only be set by the first ciphertext not stored in $\mathbf{C}$ we only need to be able to simulate the games up until that point. So we store this extra ciphertext and put an $*$ in $\mathbf{C}$ so that in DEC we know when we have reached the relevant point. We have that $\Pr[\mathsf{G}^{\mathsf{ch\text{-}indr}}_{\mathsf{CH},1}(\mathcal{A}''_\delta)] = \Pr[\mathsf{G}_3 \text{ sets } \mathsf{bad}]$ and $\Pr[\mathsf{G}^{\mathsf{ch\text{-}indr}}_{\mathsf{CH},0}(\mathcal{A}''_\delta)] = \Pr[\mathsf{G}_4 \text{ sets } \mathsf{bad}]$. In other words, $\Pr[\mathsf{G}_3 \text{ sets } \mathsf{bad}] \leqslant \mathsf{Adv}^{\mathsf{ch\text{-}indr}}_{\mathsf{CH}}(\mathcal{A}''_\delta) + \Pr[\mathsf{G}_4 \text{ sets } \mathsf{bad}]$.

```
Adversary $\mathcal{A}'^{\textsc{Enc}}_{\delta}$
flag ← true
sync ← true
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{SimEnc},\textsc{SimDec}}$
Return $b' = 1$

Adversary $\mathcal{A}''^{\textsc{Enc}}_{\delta}$
flag ← true
sync ← true
$\mathcal{A}^{\textsc{SimEnc},\textsc{SimDec}}$
Return 0
```

```
Oracle SimEnc($M$)
flag ← true
sync ← true
$C \leftarrow\!\!\$\ \textsc{Enc}(M)$
If flag then
    If $||\mathbf{C}|| + |C| < \delta$ then
        $\mathbf{M}$.add($M$); $\mathbf{C}$.add($C$)
    Else
        $\mathbf{C}$.add($*$); $C^* \leftarrow C$
        flag ← false
Return $C$
```

```
Oracle SimDec($C$)
$M' \leftarrow \mathbf{M}$.dq(); $C' \leftarrow \mathbf{C}$.dq()
If sync then
    If $C = C'$ then
        Return $M'$
    Elif $C' = *$ and $C = C^*$ then
        abort(1)
    sync ← false
Return $\perp$
```

Fig. 10. INDR adversaries for proof of Theorem 2. Highlighting indicates code that is only used by adversary $\mathcal{A}'_\delta$.

Final transition. The final transition is from G_4 to G_5. These two games are identical-until-bad as can be seen in Dec. Because of this we have that

$$\Pr[\mathsf{G}_4] - \Pr[\mathsf{G}_5] \leqslant \Pr[\mathsf{G}_4 \text{ sets } \mathsf{bad}].$$

Using all of $\mathbf{M}_2$ and $\mathbf{C}_2$ instead of just $\mathbf{M}$ and $\mathbf{C}$ makes G_5 identical to $\mathsf{G}^{\text{ch-ae}}_{\mathsf{CH},0}$.

Bounding probability of bad. We conclude by bounding the probability G_4 sets bad via a reduction to our information theoretic analysis. Consider the S-bounded $(\mathcal{A}_1, \mathcal{A}_2)$ that behaves as follows. First, $\mathcal{A}_1$ internally simulates the view of $\mathcal{A}$ in G_4 using the coins for $\mathcal{A}$ which maximize the probability of bad and using the bits of R as the ciphertext bits returned by encryption. If $\mathcal{A}$ causes flag to be set to `false`, $\mathcal{A}_1$ will halt and output the current state of $\mathcal{A}$ as σ with i chosen so the next δ bits of $\mathbf{C}$ and c are the values of R for $\mathcal{A}_2$ to guess.

Then $\mathcal{A}_2$ will resume executing $\mathcal{A}$ using σ. When $\mathcal{A}$ makes encryption queries it will just make up its own responses. When $\mathcal{A}$ makes a decryption query for a ciphertext C then $\mathcal{A}_2$ will concatenate it into its guess r. It just assumes this was the correct next ciphertext that should have been stored in $\mathbf{C}$ (otherwise $\mathcal{A}$ would fail in setting bad). To determine which M to return for this query, $\mathcal{A}_2$ re-runs $\mathcal{A}$ from the beginning using the same coins $\mathcal{A}_1$ used. It uses its given prefix of R and the current value of r to respond to encryption queries until it reaches the encryption query corresponding to the current decryption query. Whatever message $\mathcal{A}$ queried for this encryption query is then returned for the decryption query. Once r is δ bits long, $\mathcal{A}_2$ outputs that as its guess.

We can see that when bad would be set in G_4, the view of $\mathcal{A}$ is perfectly simulated up until that point and $\mathcal{A}_2$ will guess r correctly. This gives us $\Pr[\mathsf{G}_4 \text{ sets } \mathsf{bad}] \leqslant \mathsf{Adv}^{\mathsf{it}}_{q \cdot x, \delta}(\mathcal{A}_1, \mathcal{A}_2)$ as desired.

Combining all the bounds we have shown completes the proof. □

4.3 AE Security of a TLS 1.3-like Channel

We have shown that the AE security of a channel can be reduced to its constituent INDR and CTXT security in a way that preserves memory complexity. This is, of course, only meaningful if we have channels for which we can give provable time-memory tradeoffs for their INDR and CTXT security. Using the ideas of Jaeger and Tessaro [11] it is easy to give such examples for INDR security.

Using the ideas from proof of Thm. 2 we will prove the security of a channel based on GCM (or more generally CAU). The resulting channel can be viewed as a (simplified) version of the channel obtained by using GCM in TLS 1.3.

The construction. The construction we consider is a straightforward construction of a channel from a nonce-based encryption scheme NE by using a counter for the nonce. The INDR security of this channel follows easily from the nonce-respecting INDR security of NE. Proving integrity of the channel from the

```
NCH[NE].Sg
$K \leftarrow_{\$} \mathsf{NE.Kg}$
$N \leftarrow_{\$} \mathsf{NE.N}$
Return $((K,N),(K,N))$
```

```
NCH[NE].S((K,N),M)
$N \leftarrow N+1$
$C \leftarrow \mathsf{NE.E}(K,N,M)$
Return $((K,N),C)$
```

```
NCH[NE].R((K,N),C)
If $N = \bot$ then
    Return $((\bot,\bot),\bot)$
$N \leftarrow N+1$
$M \leftarrow \mathsf{NE.D}(K,N,C)$
If $M = \bot$ then
    Return $((\bot,\bot),\bot)$
Return $((K,N),M)$
```

Fig. 11. Algorithms of channel NCH[NE] constructed from encryption scheme NE.

integrity of NE is possible, but of limited applicability since we do not have examples of encryption schemes with proven time-memory tradeoffs for integrity. We will instead only show integrity for the specific case that NE = CAU.

The channel NCH[NE] is parameterized by an encryption scheme NE. It has NCH[NE].M = NE.M. We assume that NE.N can be interpreted as a cyclic group written using additive notation. Its algorithms are shown in Fig. 11. State generation sets the state of both parties equal to a shared random key and nonce. Encryption increments the nonce and uses NE to encrypt the message with the current nonce. Decryption increments the nonce and uses NE to decrypt the ciphertext with the current nonce. If the ciphertext is rejected ($M = \bot$), the receiver will replace its state with $\bot$'s. Henceforth it will reject all ciphertexts it receives (via the first line which checks if $N = \bot$ already holds.

INDR SECURITY. The INDR security of NCH[NE] follows easily from nonce-respecting INDR security of NE. This is captured by the following theorem.

Theorem 3. *Let $\mathcal{A}$ be an adversary against the* INDR *security of* NCH[NE] *that makes less than* |NE.N| *oracle queries. Then we can construct $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathsf{ch\text{-}indr}}_{\mathsf{NCH[NE]}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ has complexity comparable to that of $\mathcal{A}$ and is nonce-respecting.

Proof (Sketch). Adversary $\mathcal{B}$ picks N at random and then starts executing $\mathcal{A}$. Whenever $\mathcal{A}$ makes a $\textsc{Enc}(M)$ query, $\mathcal{B}$ increments N, queries $C \leftarrow \textsc{Enc}(N,M)$, and returns C to $\mathcal{A}$. Adversary $\mathcal{B}$ outputs whatever $\mathcal{A}$ does. Verifying the claims made about this adversary is straightforward. □

CTXT SECURITY. For CTXT security we need to focus our attention on the particular construction of NCH[NE] obtained when using the encryption scheme NE = CAU[E, H] for some function families E and H.

In our proof, we will take advantage of the fact that the adversary can essentially only make a single forgery attempt. If it fails at this attempt, then the state of the decryption algorithm can be erased and it will henceforth always return $\bot$. Because CAU uses a Carter-Wegman style MAC we have to first use the PRF security of E to hide the values of $H_L(A,C)$ used in encryption queries. To get our desired state-aware results we need to make sure that our PRF reduction does not use much more memory than the original adversary. This creates an issue similar to what we saw in Section 4 where it can be difficult to simulate the values returned by DEC. This issue is resolved by adjusting the proof technique used to establish Thm. 2 where we exploit the fact that ciphertexts look random to assume that $\mathcal{A}$ cannot remember too many ciphertexts.

Theorem 4. *Let* NE = CAU[E, H] *for some* E *and* H. *Let $\mathcal{A}$ be a nonce-respecting adversary against the* CTXT *security of* NCH[NE] *with memory complexity S that makes at most $q \leqslant 2^{\mathsf{CAU.nl}}-1$ encryption queries, each of which returns a ciphertext of length at most x. Then for any $\delta \in \mathbb{N}$ we can construct an adversary $\mathcal{A}_{\mathsf{prf}}$ such that*

$$\mathsf{Adv}^{\mathsf{ch\text{-}ctxt}}_{\mathsf{NCH[NE]}}(\mathcal{A}) \leqslant 2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{E}}(\mathcal{A}_{\mathsf{prf}}) + \mathsf{Adv}^{\mathsf{axu}}_{\mathrm{H}}(\mathcal{X}) + q \cdot x \cdot 2^S/2^\delta \ .$$

Adversary $\mathcal{A}_{\mathsf{prf}}$ has running time approximately that of $\mathcal{A}$ and uses about $S + 2\delta$ bits of state. It makes at most $q(x/n+2)+1$ non-repeating queries to its oracle.

```
Reduction $\mathcal{R}[\mathcal{A}]^{\mathrm{O}}$
    $\sigma \leftarrow\!\!{\scriptstyle\$}\ \mathcal{R}^{\mathrm{O}}.\mathsf{Init}$
    $i \leftarrow 0$
    While $i \leqslant \mathcal{R}.\mathrm{rew}$ do
        $b \leftarrow\!\!{\scriptstyle\$}\ \mathcal{A}^{\mathrm{REnc},\mathrm{RDec}}$
        $\sigma \leftarrow\!\!{\scriptstyle\$}\ \mathcal{R}^{\mathrm{O}}.\mathsf{Upd}(\sigma, b)$
        NEXT: $i \leftarrow i + 1$
    Return $\mathcal{R}^{\mathrm{O}}.\mathsf{Fin}(\sigma)$

Oracle REnc$(N, M)$
    $(\sigma, \mathsf{rf}, C) \leftarrow\!\!{\scriptstyle\$}\ \mathcal{R}^{\mathrm{O}}.\mathsf{SimEnc}(\sigma, N, M)$
    If $\mathsf{rf}$ then goto NEXT
    Return $C$

Oracle RDec$(N, C)$
    $(\sigma, \mathsf{rf}, M) \leftarrow\!\!{\scriptstyle\$}\ \mathcal{R}^{\mathrm{O}}.\mathsf{SimDec}(\sigma, N, C)$
    If $\mathsf{rf}$ then goto NEXT
    Return $M$
```

Fig. 12. Syntax of a black-box reductions $\mathcal{R}$ running AE-1 adversary $\mathcal{A}$. We represent the oracles $\mathcal{R}$ has access to collectively as O.

The proof is given in Appendix F. As with Thm. 1, the PRF adversary we give never repeats queries so we can apply the switching lemma to obtain a bound using PRP security of E. Here it is important that the memory of $\mathcal{A}_{\mathsf{prf}}$ is not much more than that of $\mathcal{A}$. Assuming $|A| < p$, setting $\delta \approx S + n$, and assuming $S > n$ we can combine all of our theorems so far to obtain a bound of

$$\mathsf{Adv}^{\mathsf{ch\text{-}ae}}_{\mathsf{NCH[NE]}}(\mathcal{A}) \leqslant 4 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{E}}(\mathcal{B}) + O\left(\frac{Spq\log(pq) + c(p+q)}{2^n}\right)$$

for a $\mathcal{B}$ with comparable efficiency to $\mathcal{A}$ and assuming H is c-AXU.

5 Negative Results for Memory-tight AE Reductions

In this section we give impossibility results for giving a memory-tight reduction (for a natural restricted class of black-box reductions) from AE-1 security to nonce-respecting INDR and CTXT-1 security. This establishes that our restriction to the channel setting for Section 4 was necessary for our positive results.

Black-box Reductions. A reduction $\mathcal{R}$ maps an adversary $\mathcal{A}$ to an adversary $\mathcal{R}[\mathcal{A}]$. We consider reductions that run an AE-1 adversary $\mathcal{A}$ in a black-box manner as shown in Fig. 12. It starts with initial state σ output by $\mathcal{R}.\mathsf{Init}$. The parameter $\mathcal{R}.\mathrm{rew}$ determines how many times $\mathcal{R}$ will perform a full rewind of $\mathcal{A}$. Then it runs $\mathcal{A}$ while simulating its encryption and decryption oracles. For every encryption query, $\mathcal{R}$ runs $\mathcal{R}.\mathsf{SimEnc}$ with the query and its state as input to produce the updated state, a flag rf, and a ciphertext. If the flag rf is `true`, then $\mathcal{R}$ starts running $\mathcal{A}$ from the beginning again. Otherwise, it answers with the query answer $\mathcal{R}.\mathsf{SimEnc}$ returned. Decryption queries are handled analogously. If $\mathcal{R}$ did not rewind $\mathcal{A}$ before $\mathcal{A}$ finished its execution, then it runs $\mathcal{R}.\mathsf{Upd}$ on $\mathcal{A}$'s output to updates its state and starts running $\mathcal{A}$ from the beginning if has not already rewinded $\mathcal{R}.\mathrm{rew}$ times. Finally, $\mathcal{R}$ outputs whatever $\mathcal{R}.\mathsf{Fin}(\sigma)$ returns. The following definition captures some restrictions we will place on reductions.

Definition 1. *Let $\mathcal{R}$ be a reduction using the syntax from Fig. 12. It is* full-rewinding *if $\mathcal{R}.\mathrm{rew} > 0$ or* straightline *if $\mathcal{R}.\mathrm{rew} = 0$. It is* nonce-respecting *if $\mathcal{R}[\mathcal{A}]$ is nonce-respecting when $\mathcal{A}$ is nonce-respecting. It is* faithful *if $\mathcal{R}[\mathcal{A}]$ answers encryption queries of $\mathcal{A}$ consistent with its own encryption oracle, i.e., $\mathcal{R}$ responds with C on an encryption query made on (N, M), only if it previously queried its own encryption oracle with (N, M) and received C as the answer.*

Additional notation. We fix an understood nonce-based encryption scheme NE for which we assume that $\{0,1\}^{\mathsf{ml}} \subseteq \mathsf{NE.M}$. We also assume $\mathbb{N} \times \mathbb{N} \subseteq \mathsf{NE.N}$ and we use $\mathsf{N} = \mathsf{NE.N}$ as shorthand. We assume that $[\mathsf{NE.Kg}] = \{0,1\}^{\mathsf{kl}}$. We let $\mathsf{C} = \{0,1\}^{\mathsf{NE.cl(ml)}}$. We also introduce some new notation for the complexity of an algorithm $\mathcal{A}$. First, $\mathsf{Mem}(\mathcal{A})$ is defined as the number of bits of memory that $\mathcal{A}$ uses. The total number of queries to its oracles is $\mathsf{Query}(\mathcal{A})$, and the number of computation steps $\mathsf{Time}(\mathcal{A})$. For a reduction $\mathcal{R}$ we use $\mathsf{Mem}(\mathcal{R})$ to denote the number of bits of memory that $\mathcal{R}$ uses in addition any memory of the adversary it runs.

```
Game $\mathsf{G}^{\text{bf-it-chl}}_{u,m,P}(\mathcal{B}_1,\mathcal{B}_2)$
$(I,Q) \leftarrow\!\$\ \mathcal{B}_1$ // $|I| = |Q| = P$
For $j \in [u]$ do
    $M_j \leftarrow\!\$\ \{0,1\}^m$
For $j \in [P]$ do
    $M_{I[j]} \leftarrow Q[j]$
$j^* \leftarrow\!\$\ [u]$
$M \leftarrow\!\$\ \mathcal{B}_2(I,Q,j^*)$
If $M_{j^*} \neq M$ then return false
Return true
```

```
Adversary $\mathcal{D}'_1(\perp, M_1,\cdots,M_u)$
While(true)
    $\mathsf{win} \leftarrow 1; \phi \leftarrow \perp$
    For $k \in [i-1]$ do
        For $j \in [u]$ do
            $M'_j \leftarrow\!\$\ \{0,1\}^m$
        $\phi \leftarrow \mathcal{D}^*_1(\phi, M'_1,\cdots,M'_u)$
        $j^* \leftarrow\!\$\ [u]$
        $(\phi, M') \leftarrow \mathcal{D}^*_2(\phi, j^*)$
        If $M' \neq M'_{j^*}$ then $\mathsf{win} \leftarrow 0$
    If $\mathsf{win} = 1$ then break
$\phi \leftarrow \mathcal{D}^*_1(\phi, M_1,\cdots,M_u)$
Return $\phi$
```

Fig. 14. Game $\mathsf{G}^{\text{bf-it-chl}}_{u,m,P}$ and adversary $\mathcal{D}'_1$ used in the proof of Lemma 2.

INFORMATION THEORETIC LEMMA. We give a lemma that will be a useful sub-component of our proofs. It pertains to game $\mathsf{G}^{\text{it-chl-}r}_{u,m}$ in Fig. 13. It is an r-round game, played by a two-stage adversary $(\mathcal{D}_1,\mathcal{D}_2)$. In each round, $\mathcal{D}_1$ gets state σ from the prior round, along with u random strings $M_1,\ldots,M_u$ each of length m. Adversary $\mathcal{D}_1$ outputs state σ which is input to $\mathcal{D}_2$ along with a randomly sampled index j^* from $[u]$. Then $\mathcal{D}_2$ outputs a string M and state σ that is passed to $\mathcal{D}_1$ in the next round. If $M = M_{j^*}$, we say that $(\mathcal{D}_1,\mathcal{D}_2)$ has answered the challenge of this round correctly. If $(\mathcal{D}_1,\mathcal{D}_2)$ answers all the r challenges correctly, the game returns `true`. Otherwise it returns `false`. We define $\mathsf{Adv}^{\text{it-chl-}r}_{u,m}(\mathcal{D}_1,\mathcal{D}_2) = \Pr[\mathsf{G}^{\text{it-chl-}r}_{u,m}(\mathcal{D}_1,\mathcal{D}_2)]$. Adversary $(\mathcal{D}_1,\mathcal{D}_2)$ is S-bounded if the state output by $\mathcal{D}_1$ is at most S bits long. We can prove the following.

```
Game $\mathsf{G}^{\text{it-chl-}r}_{u,m}(\mathcal{D}_1,\mathcal{D}_2)$
$\sigma \leftarrow \perp$
For $i \in [r]$ do
    For $j \in [u]$ do
        $M_j \leftarrow\!\$\ \{0,1\}^m$
    $\sigma \leftarrow \mathcal{D}_1(\sigma, M_1,\ldots,M_u)$
    $j^* \leftarrow\!\$\ [u]$
    $(\sigma, M) \leftarrow \mathcal{D}_2(\sigma, j^*)$
    If $M_{j^*} \neq M$ then return false
Return true
```

Fig. 13. Information theoretic game played by adversary $(\mathcal{D}_1,\mathcal{D}_2)$.

Lemma 2. *If $(\mathcal{D}_1,\mathcal{D}_2)$ is S-bounded, then*

$$\mathsf{Adv}^{\text{it-chl-}r}_{u,m}(\mathcal{D}_1,\mathcal{D}_2) \leqslant \left(\frac{2(S+m)}{u} + \frac{3}{2^m}\right)^r .$$

The proof reduces to the $r = 1$ case and analyzes this case using techniques from the AI-ROM setting [5].

Proof. We split this lemma into the following two claims. The first claim bounds advantage of any S-bounded adversary playing the one round version of the game ($r = 1$). The second step bounds the advantage of an adversary playing the full r rounds in using an adversary playing only one round.

Claim. If $(\mathcal{D}_1,\mathcal{D}_2)$ is S-bounded, then $\mathsf{Adv}^{\text{it-chl-1}}_{u,m}(\mathcal{D}_1,\mathcal{D}_2) \leqslant \frac{2S+2m}{u} + \frac{3}{2^m}$

Claim. If $\mathsf{Adv}^{\text{it-chl-1}}_{u,m}(\mathcal{D}_1,\mathcal{D}_2) \leqslant \varepsilon$ for all S-bounded $(\mathcal{D}_1,\mathcal{D}_2)$, then $\mathsf{Adv}^{\text{it-chl-}r}_{u,m}(\mathcal{D}^*_1,\mathcal{D}^*_2) \leqslant \varepsilon^r$ for all r and all S-bounded $(\mathcal{D}^*_1,\mathcal{D}^*_2)$.

FIRST CLAIM. In order to show the upper bound on $\mathsf{Adv}^{\text{it-chl-1}}_{u,m}(\mathcal{D}_1,\mathcal{D}_2)$, we first define the game $\mathsf{G}^{\text{bf-it-chl}}_{u,m,P}$ in Fig. 14. It is the "Bit-Fixed Random Oracle" [5] version of $\mathsf{G}^{\text{it-chl-1}}_{u,m}$. (The sequence of u random strings that are input to the adversary can be viewed as a random oracle $[u] \to \{0,1\}^m$.) This game is played by a two-stage adversary $(\mathcal{B}_1,\mathcal{B}_2)$. The first stage $\mathcal{B}_1$ outputs two lists I and Q, each of length P. All the entries of I are in $[u]$ and all the entries of Q are in $\{0,1\}^m$. Then u bit-strings $M_1, \cdots, M_u$ are sampled uniformly at random from $\{0,1\}^m$. For each $j \in [P]$, the bit-string M_i is overwritten by $Q[j]$, where $i = I[j]$. The second stage adversary $\mathcal{B}_2$ is given the lists and a randomly sampled j^* as input. It returns $M \in \{0,1\}^m$. The game returns `true` if $M = M_{j*}$ and `false` otherwise. We define $\mathsf{Adv}^{\mathsf{bf\text{-}it\text{-}chl}}_{u,m,P}(\mathcal{B}_1, \mathcal{B}_2) = \Pr[\mathsf{G}^{\mathsf{bf\text{-}it\text{-}chl}}_{u,m,P}(\mathcal{B}_1, \mathcal{B}_2)]$.

We claim that $\mathsf{Adv}^{\mathsf{bf\text{-}it\text{-}chl}}_{u,m,P}(\mathcal{B}_1, \mathcal{B}_2) \leqslant \frac{P}{u} + \frac{1}{2^m}$. The first term captures the probability that j^* is in I. The second term captures the probability that $\mathcal{B}_2$ guesses the random string $M_{j*} \in \{0,1\}^m$ when j^* is not in I. Since $(\mathcal{D}_1, \mathcal{D}_2)$ is S-bounded, Theorem 6 of [5] gives that for any $P \in \mathbb{N}$ and every $\gamma > 0$, if $P \geqslant S + \log \gamma^{-1}$ then there exists an adversary $(\mathcal{B}_1, \mathcal{B}_2)$ such that

$$\mathsf{Adv}^{\mathsf{it\text{-}chl\text{-}1}}_{u,m}(\mathcal{D}_1, \mathcal{D}_2) \leqslant 2 \cdot \mathsf{Adv}^{\mathsf{bf\text{-}it\text{-}chl}}_{u,m,P}(\mathcal{B}_1, \mathcal{B}_2) + \gamma \; .$$

Setting $\gamma = \frac{1}{2^m}$ and $P = S + m$, gives the claimed bound.

Second claim. Let us assume, towards contradiction, that there exists S-bounded $(\mathcal{D}_1^*, \mathcal{D}_2^*)$ such that $\mathsf{Adv}^{\mathsf{it\text{-}chl\text{-}}r}_{u,m}(\mathcal{D}_1^*, \mathcal{D}_2^*) > \varepsilon^r \neq 0$. Then we can construct an adversary $(\mathcal{D}_1', \mathcal{D}_2')$ such that $\mathsf{Adv}^{\mathsf{it\text{-}chl\text{-}1}}_{u,m}(\mathcal{D}_1', \mathcal{D}_2') > \varepsilon$. We give $\mathcal{D}_1'$ in Fig. 14 and let $\mathcal{D}_2' = \mathcal{D}_2^*$. We say $(\mathcal{D}_1^*, \mathcal{D}_2^*)$ wins round j if $\mathsf{G}^{\mathsf{it\text{-}chl\text{-}}r}_{u,m}$ did not return `false` before the $(j+1)^{\text{st}}$ iteration. It follows that,

$$\mathsf{Adv}^{\mathsf{it\text{-}chl\text{-}}r}_{u,m}(\mathcal{D}_1^*, \mathcal{D}_2^*) = \prod_{j=1}^{r} \Pr\left[(\mathcal{D}_1^*, \mathcal{D}_2^*) \text{ wins round } j \,\middle|\, (\mathcal{D}_1^*, \mathcal{D}_2^*) \text{ wins round } j-1\right].$$

Since $\mathsf{Adv}^{\mathsf{it\text{-}chl\text{-}}r}_{u,m}(\mathcal{D}_1^*, \mathcal{D}_2^*) > \varepsilon^r$, there must exist some $i \in [r]$, such that,

$$\Pr\left[(\mathcal{D}_1^*, \mathcal{D}_2^*) \text{ wins round } i \,\middle|\, (\mathcal{D}_1^*, \mathcal{D}_2^*) \text{ wins round } i-1\right] > \varepsilon \; .$$

Adversary $\mathcal{D}_1'$ keeps running $(\mathcal{D}_1^*, \mathcal{D}_2^*)$ until it wins round $i-1$ on randomly generated $M_{i,j}$'s. Once this occurs $\mathcal{D}_1'$ runs $\mathcal{D}_1^*$ on its own inputs and returns the state returned by $\mathcal{D}_1^*$. Adversary $\mathcal{D}_2'$ runs $\mathcal{D}_2^*$ on its own inputs and returns whatever $\mathcal{D}_2^*$ returns. Note that $(\mathcal{D}_1', \mathcal{D}_2')$ is S-bounded because $(\mathcal{D}_1^*, \mathcal{D}_2^*)$ is. Moreover,

$$\Pr\left[\mathsf{G}^{\mathsf{it\text{-}chl\text{-}1}}_{u,m}(\mathcal{D}_1', \mathcal{D}_2')\right] = \Pr\left[(\mathcal{D}_1^*, \mathcal{D}_2^*) \text{ win round } i \,\middle|\, (\mathcal{D}_1^*, \mathcal{D}_2^*) \text{ win round } i-1\right] \; .$$

Thus $\mathsf{Adv}^{\mathsf{it\text{-}chl\text{-}}r}_{u,m}(\mathcal{D}_1', \mathcal{D}_2') > \varepsilon$, which contradicts our assumption. □

5.1 Memory Lower Bound for Straightline Reductions

Our first theorem shows that it is not possible to give memory-tight, straightline reductions proving the AE-1 security of an encryption scheme from its INDR and CTXT-1 security. (As the theorem statement is somewhat complicated, we will describe how to interpret it below.)

Theorem 5 (Impossibility for straightline reductions). *Let* NE *be a nonce-based encryption scheme. Fix* $u, r \in \mathbb{N}$ *and define the nonce-respecting adversary* $\mathcal{A}$ *as shown in Fig. 16. Let* $\mathcal{R}$ *be a straightline, nonce-respecting, faithful black-box reduction from* AE-1 *to nonce-respecting* INDR *with* $\mathsf{Mem}(\mathcal{R}) = S$*. Let* $\mathcal{R}'$ *be a straightline, nonce-respecting, faithful reduction from* AE-1 *to* CTXT-1*. Then, we can construct adversaries* $\mathcal{C}$ *and* $\mathcal{W}$ *such that,*

(i) $\mathsf{Adv}^{\mathsf{ae\text{-}1}}_{\mathsf{NE}}(\mathcal{A}) \geqslant 1 - \dfrac{2^{\mathsf{kl}}}{2^{2\mathsf{NE.cl(ml)}}}$,

(ii) $\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{A}]) \leqslant 2 \cdot \left(\dfrac{2(S + \log r + \mathsf{kl}) + 2\mathsf{ml}}{u} + \dfrac{3}{2^{\mathsf{ml}}}\right)^r + 4 \cdot \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C})$,

(iii) $\mathsf{Adv}^{\mathsf{ctxt\text{-}1}}_{\mathsf{NE}}(\mathcal{R}'[\mathcal{A}]) \leqslant \mathsf{Adv}^{\mathsf{ctxt\text{-}1}}_{\mathsf{NE}}(\mathcal{R}'[\mathcal{W}])$.

Moreover, $\mathcal{A}$ *satisfies* $\mathsf{Query}(\mathcal{A}) = (u+1) \cdot r + 2$ *and* $\mathsf{Mem}(\mathcal{A}) \leqslant 2\mathsf{kl} + 2\mathsf{ml} + 2\mathsf{NE.cl(ml)} + 2\log|\mathsf{N}| + \log u \cdot r$*. Also* $\mathcal{C}$ *and* $\mathcal{W}$ *satisfy* $\mathsf{Query}(\mathcal{C}) < \mathsf{Query}(\mathcal{R}) + \mathsf{Query}(\mathcal{A})$*,* $\mathsf{Time}(\mathcal{C}) \in O(\mathsf{Query}(\mathcal{A}) + \mathsf{Time}(\mathcal{R}))$*, and* $\mathsf{Query}(\mathcal{W}) = \mathsf{Query}(\mathcal{A})$ *and* $\mathsf{Time}(\mathcal{W}) \in O(\mathsf{Query}(\mathcal{A}))$*.*

```
Games $\mathsf{G}_b^0$, $\mathsf{G}^1$
$b \leftarrow 1$
$K^* \leftarrow\$ \mathsf{NE.Kg}()$; $\sigma \leftarrow\$ \mathcal{R}^{\textsc{Enc}_b}.\text{Init}$
For $i \in [r]$ do
    $j^* \leftarrow\$ [u]$
    For $j \in [u]$ do
        $M_j \leftarrow\$ \{0,1\}^{\mathsf{ml}}$; $N_j \leftarrow (i, j)$
        $(\sigma, \cdot, C_j) \leftarrow\$ \mathcal{R}^{\textsc{Enc}_b}.\textsc{SimEnc}(\sigma, N_j, M_j)$
    $(\sigma, \cdot, M) \leftarrow\$ \mathcal{R}^{\textsc{Enc}_b}.\textsc{SimDec}(\sigma, N_{j*}, C_{j*})$
    If $M \neq M_{j*}$ then return false
Return true

Oracle $\textsc{Enc}_b(N, M)$
$C_0 \leftarrow\$ \mathsf{C}$
$C_1 \leftarrow \mathsf{NE.E}(K^*, N, M)$
$C_1 \leftarrow \mathsf{NE.E}(K^*, N, 0^{\mathsf{ml}})$
Return $C_b$
```

Fig. 15. Games G^1 and G_b^0 for $b \in \{0,1\}$. Highlighted code is only included in G^1.

```
Adversaries $\mathcal{A}^{\textsc{Enc},\textsc{Dec}}$, $\boxed{\mathcal{S}^{\textsc{Enc},\textsc{Dec}}}$, $\mathcal{W}^{\textsc{Enc},\textsc{Dec}}$
For $i \in [r]$ do
    $j^* \leftarrow\$ [u]$
    For $j \in [u]$ do
        $M_j \leftarrow\$ \{0,1\}^{\mathsf{ml}}$
        $N_j \leftarrow (i, j)$
        $C_j \leftarrow \textsc{Enc}(N_j, M_j)$
    $M \leftarrow \textsc{Dec}(N_{j*}, C_{j*})$
    If $M \neq M_{j*}$ then return 1
bad $\leftarrow$ true
Return $\mathcal{B}^{\textsc{Enc}}$; $\boxed{\text{Return } 0;}$ Return $\mathcal{E}^{\textsc{Enc}}$

Adversaries $\mathcal{B}^{\textsc{Enc}}$, $\mathcal{E}^{\textsc{Enc}}$
$M_1 \leftarrow\$ \{0,1\}^{\mathsf{ml}}$
$M_2 \leftarrow\$ \{0,1\}^{\mathsf{ml}}$
$N_1 \leftarrow (0, 1)$; $N_2 \leftarrow (0, 2)$
$C_1 \leftarrow \textsc{Enc}(N_1, M_1)$
$C_2 \leftarrow \textsc{Enc}(N_2, M_2)$
For $K \in \{0,1\}^{\mathsf{kl}}$ do
    $\mathsf{eq}_1 \leftarrow (\mathsf{NE.E}(K, N_1, M_1) = C_1)$
    $\mathsf{eq}_2 \leftarrow (\mathsf{NE.E}(K, N_2, M_2) = C_2)$
    If $\mathsf{eq}_1$ and $\mathsf{eq}_2$ then return 1
Return 0; Return 1
```

Fig. 16. Adversaries against the AE-1 security of NE. Boxed code is only included in $\mathcal{S}$. Highlighted code is only included in $\mathcal{A}$ and $\mathcal{B}$.

To interpret this theorem, assume that the parameters of NE are such that the advantage of $\mathcal{A}$ is essentially one. Hence, a successful pair of reductions $\mathcal{R}$ and $\mathcal{R}'$ would need at least one of $\mathcal{R}[\mathcal{A}]$ or $\mathcal{R}'[\mathcal{A}]$ to have high advantage. For memory-tight $\mathcal{R}$ and $\mathcal{R}'$ we expect there to be linear functions f_1 and f_2 such that their local computation time and memory usage when interacting with an adversary $\mathcal{A}$ would be bounded by $f_1(q_{\mathcal{A}})$ and $f_2(s_{\mathcal{A}})$ where $q_{\mathcal{A}} = \mathsf{Query}(\mathcal{A})$ and $s_{\mathcal{A}} = \mathsf{Mem}(\mathcal{A})$.

Suppose this was the case. Then we can fix upper bounds for $\log(u)$ and $\log(r)$, determining the memory usage of $\mathcal{A}$ and hence $f_1(s_{\mathcal{A}}) = S$. Now we can pick reasonable u and r such that, $2 \cdot \left(\frac{2(S+\log r+\mathsf{kl})+2\mathsf{ml}}{u} + \frac{3}{2^{\mathsf{ml}}}\right)^r$ is very small (by say making the inside of the parenthesis less than 1/2 and setting $r = 128$). Then, for one of the reductions to have high advantage, one of $\mathcal{C}$ or $\mathcal{R}'[\mathcal{W}]$ would have to have high advantage. But the efficiencies of these are bounded as small functions of the query complexity of $\mathcal{A}$ (rather than its local runtime) so cannot be too large. But then assuming security of NE prevents any of them from having high advantage.

Proof. Consider the adversary $\mathcal{A}$ in Fig. 16 against AE-1 security of NE. Note that it is nonce-respecting. It has a challenge phase followed by an invocation of $\mathcal{B}$. Each iteration of the challenge phase consists of $\mathcal{A}$ making u encryption queries with unique nonces and making one decryption query on one of the u ciphertexts it received as answers chosen uniformly at random with its corresponding nonce. If the answer of the decryption query is not consistent with the prior encryption query, $\mathcal{A}$ returns 1. There are r iterations of the challenge phase. If these are all passed, $\mathcal{A}$ runs adversary $\mathcal{B}$ (shown on the right) with its $\textsc{Enc}$ oracle and outputs whatever $\mathcal{B}$ outputs. From the code of $\mathcal{A}$ we can see that it makes $r \cdot u + 2$ encryption queries,

r decryption queries, and satisfies

$$\mathsf{Mem}(\mathcal{A}) \leqslant 2\mathsf{kl} + 2\mathsf{ml} + 2\mathsf{NE.cl}(\mathsf{ml}) + 2\log|\mathsf{N}| + \log u \cdot r \; .$$

To prove the theorem we need to separately establish the three advantage claims (and corresponding statements about the efficiency of various algorithms). For the first claim, note that $\mathsf{Adv}^{\mathsf{ae\text{-}1}}_{\mathsf{NE}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{ae\text{-}1}}_{\mathsf{NE}}(\mathcal{B})$ because M will always equal M_{j*} when $\mathcal{A}$ is playing $\mathsf{G}^{\mathsf{ae\text{-}1}}_{\mathsf{NE},b}$. The simple analysis giving the needed bound on $\mathsf{Adv}^{\mathsf{ae\text{-}1}}_{\mathsf{NE}}(\mathcal{B})$ is deferred to Appendix B.

For the third claim, consider adversary $\mathcal{W}$ defined as shown in Fig. 16. It is identical to $\mathcal{A}$, except that it calls $\mathcal{E}$, which is similar to $\mathcal{B}$ but always returns 1. Because $\mathcal{R}'$ is faithful, $\mathcal{B}$ would never return 0 when run by $\mathcal{R}'[\mathcal{A}]$ playing $\mathsf{G}^{\mathsf{ctxt\text{-}1}}_{\mathsf{NE},b}$ so $\mathsf{Adv}^{\mathsf{ctxt\text{-}1}}_{\mathsf{NE}}(\mathcal{R}'[\mathcal{A}]) = \mathsf{Adv}^{\mathsf{ctxt\text{-}1}}_{\mathsf{NE}}(\mathcal{R}'[\mathcal{W}])$ holds trivially.

We spend the rest of the proof establishing the second claim. Consider the adversary $\mathcal{S}$ in Fig. 16. It behaves identically to $\mathcal{A}$ until the flag bad is set. Using the Fundamental Lemma of Game Playing [3], we can obtain for $b \in \{0,1\}$ that

$$\left|\mathsf{Pr}\left[\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}(\mathcal{R}[\mathcal{A}])\right] - \mathsf{Pr}\left[\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}(\mathcal{R}[\mathcal{S}])\right]\right| \leqslant \mathsf{Pr}\left[\mathcal{R}[\mathcal{A}] \text{ sets } \mathsf{bad} \text{ in } \mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}\right] \; .$$

Consider the games G^0_b for $b \in \{0,1\}$ in Fig. 15. In it, we assume that $\mathcal{R}$ always outputs $\mathsf{rf} = \mathtt{false}$ since it is straightline. Note that G^0_b simulates the challenge phase of $\mathcal{A}$ and the game $\mathsf{G}^{\mathsf{indr}}_b$ to $\mathcal{R}$ perfectly, so it returns $\mathtt{true}$ whenever $\mathcal{R}[\mathcal{A}]$ would set bad is set in $\mathsf{G}^{\mathsf{indr}}_b$. From this we can show

$$\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{A}]) \leqslant \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{S}]) + \mathsf{Pr}\left[\mathsf{G}^0_0\right] + \mathsf{Pr}\left[\mathsf{G}^0_1\right] \; . \tag{3}$$

Now consider the game G^1 defined in the same figure. It is identical to either G^0_b except that it answers all encryption queries with the encryption of the message 0^{ml}. We now state two lemmas which give bounds on both $\mathsf{Pr}\left[\mathsf{G}^0_b\right]$'s via G^1. First, in Lemma 3, we use that the INDR security of NE implies G^1's encryption oracle is indistinguishable from those in either G^0_b to transition to G^1. Next, in Lemma 4 we give a bound on $\mathsf{Pr}\left[\mathsf{G}^1\right]$ which was obtained by using $\mathcal{R}$ to construct an adversary for $\mathsf{G}^{\mathsf{it\text{-}chl\text{-}}r}_{u,\mathsf{ml}}$ and bounding its advantage with Lemma 2. The proofs of these lemmas are deferred to Appendix C.1 and Appendix D, respectively.

Lemma 3. *There exist adversaries $\mathcal{C}_1$ and $\mathcal{C}_2$ such that*

$$\mathsf{Pr}\left[\mathsf{G}^0_1\right] \leqslant \mathsf{Pr}\left[\mathsf{G}^0_0\right] + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_1),$$
$$\mathsf{Pr}\left[\mathsf{G}^0_0\right] \leqslant \mathsf{Pr}\left[\mathsf{G}^1\right] + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_2)$$

where G^0_b and G^1 are defined as in Fig. 15. Moreover $\mathsf{Query}(\mathcal{C}_1) < \mathsf{Query}(\mathcal{R}) + \mathsf{Query}(\mathcal{A})$ and $\mathsf{Time}(\mathcal{C}_1) \in O(\mathsf{Query}(\mathcal{A}) + \mathsf{Time}(\mathcal{R}))$. Adversary $\mathcal{C}_2$'s complexity is the same.

Lemma 4. *If $\mathcal{R}$ is a straightline, nonce-respecting, faithful black-box reduction from* AE-1 *to nonce-respecting* INDR *with $\mathsf{Mem}(\mathcal{R}) = S$. Then,*

$$\mathsf{Pr}\left[\mathsf{G}^1\right] \leqslant \left(\frac{2(S + \log r + \mathsf{kl}) + 2\mathsf{ml}}{u} + \frac{3}{2^{\mathsf{ml}}}\right)^r$$

where G^1 is defined as in Fig. 15.

Applying these lemmas to equation (3) gives

$$\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{A}]) \leqslant 2 \cdot \left(\frac{2(S + \log r + \mathsf{kl}) + 2\mathsf{ml}}{u} + \frac{3}{2^{\mathsf{ml}}}\right)^r + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{S}]) + 2 \cdot \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_1) + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_2) \; .$$

To complete the proof, we combine the three INDR adversaries $\mathcal{R}[\mathcal{S}]$, $\mathcal{C}_1$, and $\mathcal{C}_2$. Let $\mathcal{C}$ be the INDR adversary that randomly chooses one of $\mathcal{R}[\mathcal{S}]$, $\mathcal{C}_1$, or $\mathcal{C}_2$ (with probabilities 1/4, 1/2, and 1/4, respectively) then runs the adversary it chose, outputting whatever that adversary does. Simple calculations give

$$4 \cdot \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}) = \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{S}]) + 2 \cdot \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_1) + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_2) \; .$$

The claimed complexity of $\mathcal{C}$ follows from that of $\mathcal{R}[\mathcal{S}]$, $\mathcal{C}_1$, and $\mathcal{C}_2$. □

5.2 Memory Lower Bound for Full-Rewinding Reductions

We can extend our result to cover full-rewinding reductions as captured by the following theorem. Its interpretation works similarly to that of Thm. 5.

Theorem 6 (Impossibility for full-rewinding reductions). *Let* NE *be a nonce-based encryption scheme. Fix* $u, r, c \in \mathbb{N}$. *We can construct a nonce-respecting adversary* $\mathcal{A}$ *such that for all full-rewinding, nonce-respecting, restricted reductions* $\mathcal{R}$ *from* AE-1 *to nonce-respecting* INDR *with* $\mathsf{Mem}(\mathcal{R}) = S$ *and all full-rewinding, nonce-respecting, restricted reductions* $\mathcal{R}'$ *from* AE-1 *to* CTXT-1 *there exist adversaries* $\mathcal{C}$ *and* $\mathcal{W}$ *such that,*

(i) $\mathsf{Adv}^{\mathsf{ae}\text{-}1}_{\mathsf{NE}}(\mathcal{A}) \geqslant 1 - \dfrac{2^{\mathsf{kl}}}{2^{2\mathsf{NE.cl(ml)}}}$,

(ii) $\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{A}]) \leqslant 2 \cdot \left(\dfrac{2(S + \log r + \mathsf{kl}) + 2\mathsf{ml}}{u} + \dfrac{3}{2^{\mathsf{ml}}} \right)^{r} + \dfrac{2^{S+1}}{2^{c \cdot \mathsf{NE.cl(ml)}}} + 6 \cdot \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C})$,

(iii) $\mathsf{Adv}^{\mathsf{ctxt}\text{-}1}_{\mathsf{NE}}(\mathcal{R}'[\mathcal{A}]) \leqslant \mathsf{Adv}^{\mathsf{ctxt}\text{-}1}_{\mathsf{NE}}(\mathcal{R}'[\mathcal{W}])$.

Moreover, $\mathcal{A}$ *satisfies* $\mathsf{Query}(\mathcal{A}) = c + (u+1) \cdot r + 2$ *and* $\mathsf{Mem}(\mathcal{A}) \leqslant 2\mathsf{kl} + 2\mathsf{ml} + 2\mathsf{NE.cl(ml)} + 2 \log |\mathsf{N}| + \log u \cdot r$. *Also* $\mathcal{C}$ *and* $\mathcal{W}$ *satisfy* $\mathsf{Query}(\mathcal{C}) < \mathsf{Query}(\mathcal{R}) + \mathsf{Query}(\mathcal{A})$, $\mathsf{Time}(\mathcal{C}) \in O(\mathsf{Query}(\mathcal{A}) + \mathsf{Time}(\mathcal{R}))$, $\mathsf{Query}(\mathcal{W}) = \mathsf{Query}(\mathcal{A})$, *and* $\mathsf{Time}(\mathcal{W}) \in O(\mathsf{Query}(\mathcal{A}))$.

The proof of this result has been deferred to Appendix A. We give a very brief intuition about how this impossibility proof proceeds. We define a new adversary that is similar to $\mathcal{A}$ used for the proof of Thm. 5, but has an additional "buffer" phase before the challenge phase. In the buffer phase, it makes c encryption queries on a fixed message 0^{ml} using different nonces. The key idea is that if the reduction rewinds the adversary after going past the buffer phase and still manages to pass the challenge phase, it must have remembered the c ciphertexts. Because these c ciphertexts look random (from the INDR security of NE), the memory of the reduction has to grow with c. This rules out low memory reductions that pass the challenge phase after rewinding the adversary after going past the buffer phase. As in the previous section, we can show that if a reduction cannot pass the challenge phase, it cannot have a high advantage of breaking INDR security. If the reduction does not rewind after going past the buffer phase, we can bound its advantage analogously to the straightline reduction case.

6 Conclusions

Our work gives memory-sensitive bounds for the security of a particular construction of a channel and shows the difficulty of providing such bounds for encryption schemes. It leaves open a number of interesting questions including: (i) whether memory-sensitive bounds can be given for other practical examples of channels, (ii) whether analogous results can be shown for any robust channels [8], and (iii) whether memory-sensitive bounds can be extended to the multi-user setting.

Acknowledgements. This work was partially supported by NSF grants CNS-1719146, CNS-1553758 (CAREER), and by a Sloan Research Fellowship.

References

1. Benedikt Auerbach, David Cash, Manuel Fersch, and Eike Kiltz. Memory-tight reductions. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 101–132. Springer, Heidelberg, August 2017.
2. Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Breaking and provably repairing the ssh authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):206–241, 2004.

3. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.
4. Mihir Bellare and Björn Tackmann. The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 247–276. Springer, Heidelberg, August 2016.
5. Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random oracles and non-uniformity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 227–258. Springer, Heidelberg, April / May 2018.
6. Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 649–665. Springer, Heidelberg, August 2010.
7. Itai Dinur. On the streaming indistinguishability of a random permutation and a random function. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 433–460. Springer, Heidelberg, May 2020.
8. Marc Fischlin, Felix Günther, and Christian Janson. Robust channels: Handling unreliable networks in the record layers of quic and dtls 1.3. Cryptology ePrint Archive, Report 2020/718, 2020. https://eprint.iacr.org/2020/718.
9. Ashrujit Ghoshal and Stefano Tessaro. On the memory-tightness of hashed ElGamal. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 33–62. Springer, Heidelberg, May 2020.
10. Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and repairing GCM security proofs. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 31–49. Springer, Heidelberg, August 2012.
11. Joseph Jaeger and Stefano Tessaro. Tight time-memory trade-offs for symmetric encryption. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 467–497. Springer, Heidelberg, May 2019.
12. David A. McGrew and John Viega. The security and performance of the Galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, Heidelberg, December 2004.
13. J. M. Pollard. A monte carlo method for factorization. *BIT Numerical Mathematics*, 15(3):331–334, Sep 1975.
14. Jean-Jacques Quisquater and Jean-Paul Delescaille. How easy is collision search. New results and applications to DES. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 408–413. Springer, Heidelberg, August 1990.
15. Phillip Rogaway. Nonce-based symmetric encryption. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 348–359. Springer, Heidelberg, February 2004.
16. Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006.
17. Stefano Tessaro and Aishwarya Thiruvengadam. Provable time-memory trade-offs: Symmetric cryptography against memory-bounded adversaries. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 3–32. Springer, Heidelberg, November 2018.
18. Serge Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS... In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 534–546. Springer, Heidelberg, April / May 2002.
19. Yuyu Wang, Takahiro Matsuda, Goichiro Hanaoka, and Keisuke Tanaka. Memory lower bounds of reductions revisited. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 61–90. Springer, Heidelberg, April / May 2018.

A Proof of Theorem 6 (Impossibility for Full-Rewinding Reductions)

In this section we prove Thm. 6 which was given in Section 5.2.

Consider the adversary $\mathcal{A}$ shown in Fig. 17 playing against AE-1 security of NE. It has a buffer phase, followed by a challenge phase, followed by an invocation of adversary $\mathcal{B}$. In the buffer phase, $\mathcal{A}$ makes c

```
Adversaries $\mathcal{A}^{\text{Enc,Dec}}$, $\boxed{\mathcal{S}^{\text{Enc,Dec}}}$, $\mathcal{W}^{\text{Enc,Dec}}$
For $i \in [c]$ do // **Buffer Phase**
    $N_i \leftarrow (r+1, i)$
    $C_i \leftarrow \text{Enc}(N_i, 0^{\mathsf{ml}})$
For $i \in [r]$ do // **Challenge Phase**
    $j^* \leftarrow\$ [u]$
    For $j \in [u]$ do
        $M_j \leftarrow\$ \{0,1\}^{\mathsf{ml}}$
        $N_j \leftarrow (i, j)$
        $C_j \leftarrow \text{Enc}(N_j, M_j)$
    $M \leftarrow \text{Dec}(N_{j*}, C_{j*})$
    If $M \neq M_{j*}$ then return 1
$\mathsf{bad} \leftarrow \mathtt{true}$
Return $\mathcal{B}^{\text{Enc}}$; $\boxed{\text{Return } 0;}$ Return $\mathcal{E}^{\text{Enc}}$
```

```
Adversaries $\mathcal{B}^{\text{Enc}}$, $\mathcal{E}^{\text{Enc}}$
$M_1 \leftarrow\$ \{0,1\}^{\mathsf{ml}}$
$M_2 \leftarrow\$ \{0,1\}^{\mathsf{ml}}$
$N_1 \leftarrow (0,1)$; $N_2 \leftarrow (0,2)$
$C_1 \leftarrow \text{Enc}(N_1, M_1)$
$C_2 \leftarrow \text{Enc}(N_2, M_2)$
For $K \in \{0,1\}^{\mathsf{kl}}$ do
    $\mathsf{eq}_1 \leftarrow (\mathsf{NE.E}(K, N_1, M_1) = C_1)$
    $\mathsf{eq}_2 \leftarrow (\mathsf{NE.E}(K, N_2, M_2) = C_2)$
    If $\mathsf{eq}_1$ and $\mathsf{eq}_2$ then return 1
Return 0; Return 1
```

Fig. 17. Adversaries against the AE-1 security of NE used for proof of Thm. 6. Highlighted code is only included in $\mathcal{A}$. Boxed code is only included in $\mathcal{S}$. Adversaries $\mathcal{B}$ (used by $\mathcal{A}$) and $\mathcal{E}$ (used by $\mathcal{W}$) are repeated from Fig. 16.

encryption queries on a fixed message 0^{ml} using different nonces. The next phase is the challenge phase. Each iteration of the challenge phase consists of $\mathcal{A}$ making u encryption queries to and asking for the decryption of one randomly chosen ciphertext from the ones it received as the answer of the encryption queries. If the answer of the decryption query is not consistent with the corresponding encryption query, then $\mathcal{A}$ halts and returns 1. There are r iterations of the challenge phase. If all of these phases are passed, then $\mathcal{A}$ runs adversary $\mathcal{B}$ which is defined in the same figure (identically to the $\mathcal{B}$ that was defined in Fig. 16). From the code of $\mathcal{A}$ we can see that it makes $r \cdot u + c + 2$ encryption queries, r decryption queries, and satisfies $\mathsf{Mem}(\mathcal{A}_2) \leqslant 2\mathsf{kl} + 2\mathsf{ml} + 2\mathsf{NE.cl}(\mathsf{ml}) + 2\log|\mathsf{N}| + \log u \cdot r$.

To prove the theorem we need to separately establish the three advantage claims (and corresponding statements about the efficiency of various algorithms).

First and third claims. For the first claim, note that $\mathsf{Adv}^{\mathsf{ae}\text{-}1}_{\mathsf{NE}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{ae}\text{-}1}_{\mathsf{NE}}(\mathcal{B})$ because M will always equal M_{j*} when $\mathcal{A}$ is playing $\mathsf{G}^{\mathsf{ae}\text{-}1}_{\mathsf{NE},b}$. The simple analysis showingn that $\mathsf{Adv}^{\mathsf{ae}\text{-}1}_{\mathsf{NE}}(\mathcal{B}) \geqslant 1 - 2^{\mathsf{kl}}/2^{2\mathsf{NE.cl}(\mathsf{ml})}$ is given in Appendix B.

For the third claim, consider adversary $\mathcal{W}$ defined as shown in Fig. 17. It is identical to $\mathcal{A}$, except that it calls $\mathcal{E}$ which is defined in the same figure (identically to the $\mathcal{E}$ that was defined Fig. 16), which is similar to $\mathcal{B}$ but always returns 1. Because $\mathcal{R}'$ is faithful, $\mathcal{B}$ would never return 0 when run by $\mathcal{R}'[\mathcal{A}]$ playing $\mathsf{G}^{\mathsf{ctxt}\text{-}1}_{\mathsf{NE},b}$ so $\mathsf{Adv}^{\mathsf{ctxt}\text{-}1}_{\mathsf{NE}}(\mathcal{R}'[\mathcal{A}]) = \mathsf{Adv}^{\mathsf{ctxt}\text{-}1}_{\mathsf{NE}}(\mathcal{R}'[\mathcal{W}])$ holds trivially. The claims on its complexity follow from its similarity to $\mathcal{A}$.

Second claim. We spend the rest of the proof establishing the second claim. Consider the adversary $\mathcal{S}$ in Fig. 17. It behaves identically to $\mathcal{A}$ until the flag bad is set. Using the Fundamental Lemma of Game Playing [3], we obtain for $b \in \{0,1\}$ that

$$\left|\Pr\left[\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}(\mathcal{R}[\mathcal{A}])\right] - \Pr\left[\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}(\mathcal{R}[\mathcal{S}])\right]\right| \leqslant \Pr\left[\mathcal{R}[\mathcal{A}] \text{ sets } \mathsf{bad} \text{ in } \mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}\right] .$$

We will use the sequence of games shown in Fig. 18 to bound these probabilities that $\mathcal{R}[\mathcal{A}]$ sets bad. Start by considering the games G^0_b for $b \in \{0,1\}$. They were obtained by plugging the code of $\mathcal{R}[\mathcal{A}]$ into $\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}$, then changing the code to output $\mathtt{true}$ if $\mathcal{R}$ passes all of $\mathcal{A}$'s challenges and output $\mathtt{false}$ otherwise (and adding a flag crw that we will use later). In other words, G^0_b outputs $\mathtt{true}$ whenever $\mathcal{R}[\mathcal{A}]$ would set bad in $\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}$. This gives us that

$$\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{A}]) \leqslant \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{S}]) + \Pr\left[\mathsf{G}^0_0\right] + \Pr\left[\mathsf{G}^0_1\right] . \quad (4)$$

```
Games $\mathsf{G}_b^0$, $\boxed{\mathsf{G}^1}$, $\boxed{\mathsf{H}^2}$
$\boxed{b \leftarrow 1}$
$K^* \leftarrow_\$ \mathsf{NE.Kg}()$; $k \leftarrow 0$; $\sigma \leftarrow_\$ \mathcal{R}^{\text{Enc}_b}.\mathsf{Init}$
While $k \leqslant \mathcal{R}.\text{rew}$ do
  For $i \in [c]$ do  // **Buffer Phase**
    $N_i \leftarrow (r+1, i)$
    $(\sigma, \mathsf{rf}, C_i) \leftarrow_\$ \mathcal{R}^{\text{Enc}_b}.\mathsf{SimEnc}(\sigma, N_i, 0^{\mathsf{ml}})$
    If $\mathsf{rf}$ then goto NEXT2
  For $i \in [r]$ do  // **Challenge Phase**
    $j^* \leftarrow_\$ [u]$
    For $j \in [u]$ do
      $M_j \leftarrow_\$ \{0,1\}^{\mathsf{ml}}$
      $N_j \leftarrow (i, j)$
      $(\sigma, \mathsf{rf}, C_j) \leftarrow_\$ \mathcal{R}^{\text{Enc}_b}.\mathsf{SimEnc}(\sigma, N_j, M_j)$
      If $\mathsf{rf}$ then goto NEXT
    $(\sigma, \mathsf{rf}, M) \leftarrow_\$ \mathcal{R}^{\text{Enc}_b}.\mathsf{SimDec}(\sigma, N_{j^*}, C_{j^*})$
    If $\mathsf{rf}$ then goto NEXT
    If $M \neq M_{j^*}$ then
      $\sigma \leftarrow_\$ \mathcal{R}^{\text{Enc}_b}.\mathsf{Upd}(\sigma, 1)$
      Goto NEXT
  Return true
  NEXT: $\mathsf{crw} \leftarrow$ true; $k \leftarrow \mathcal{R}.\text{rew} + 1$
  NEXT2: $k \leftarrow k + 1$
Return false

Oracle $\text{Enc}_b(N, M)$
$C_0 \leftarrow_\$ \mathsf{C}$
$C_1 \leftarrow \mathsf{NE.E}(K^*, N, M)$
$\boxed{C_1 \leftarrow \mathsf{NE.E}(K^*, N, 0^{\mathsf{ml}})}$
Return $C_b$
```

```
Games $\mathsf{G}^2$, $\boxed{\mathsf{G}^3}$
$b \leftarrow 1$
$K^* \leftarrow_\$ \mathsf{NE.Kg}()$; $k \leftarrow 0$; $\sigma \leftarrow_\$ \mathcal{R}^{\text{Enc}_b}.\mathsf{Init}$
While $k \leqslant \mathcal{R}.\text{rew}$ do
  For $i \in [c]$ do  // **Buffer Phase**
    $N_i \leftarrow (r+1, i)$
    $(\sigma, \mathsf{rf}, C_i) \leftarrow_\$ \mathcal{R}^{\text{Enc}_b}.\mathsf{SimEnc}(\sigma, N_i, 0^{\mathsf{ml}})$
    If $\mathsf{rf}$ then goto NEXT2
  For $i \in [r]$ do  // **Challenge Phase**
    $j^* \leftarrow_\$ [u]$
    For $j \in [u]$ do
      $M_j \leftarrow_\$ \{0,1\}^{\mathsf{ml}}$
      $N_j \leftarrow (i, j)$
      $(\sigma, \mathsf{rf}, C_j) \leftarrow_\$ \mathcal{R}^{\text{Enc}_b}.\mathsf{SimEnc}(\sigma, N_j, M_j)$
      If $\mathsf{rf}$ then goto NEXT
    $(\sigma, \mathsf{rf}, M) \leftarrow_\$ \mathcal{R}^{\text{Enc}_b}.\mathsf{SimDec}(\sigma, N_{j^*}, C_{j^*})$
    If $\mathsf{rf}$ then goto NEXT
    If $M \neq M_{j^*}$ then
      $\sigma \leftarrow_\$ \mathcal{R}^{\text{Enc}_b}.\mathsf{Upd}(\sigma, 1)$
      Goto NEXT
  Return true $\wedge$ $\mathsf{crw}$
  NEXT: $\mathsf{crw} \leftarrow$ true
  NEXT2: $k \leftarrow k + 1$
Return false

Oracle $\text{Enc}_b(N, M)$
$C_1 \leftarrow \mathsf{NE.E}(K^*, N, 0^{\mathsf{ml}})$
$\boxed{C_1 \leftarrow_\$ \mathsf{C}}$
Return $C_b$
```

Fig. 18. Sequence of games used in proof Theorem 6. On the left, boxed code is *not* included in G^1 and highlighted code is only included in H^2. On the right, highlighted code is included in both games (indicating where they differ from G^1) and boxed code is only included in G^3.

Now consider the game G^1 defined in the same figure. It is identical to either G_b^0 except that it answers all encryption queries with the encryption of the message 0^{ml}. The following lemma allows us to bound the $\Pr\left[\mathsf{G}_b^0\right]$'s using $\Pr[\mathsf{G}^1]$ and the advantage of two adversaries attacking the INDR security of NE. Its proof is given in Section C.2.

Lemma 5. *There exist adversaries $\mathcal{C}_1$ and $\mathcal{C}_2$ such that*

$$\Pr\left[\mathsf{G}_1^0\right] \leqslant \Pr\left[\mathsf{G}_0^0\right] + \mathsf{Adv}_{\mathsf{NE}}^{\mathsf{indr}}(\mathcal{C}_1) \text{ and } \Pr\left[\mathsf{G}_0^0\right] \leqslant \Pr\left[\mathsf{G}^1\right] + \mathsf{Adv}_{\mathsf{NE}}^{\mathsf{indr}}(\mathcal{C}_2)$$

where G_b^0 and G^1 are defined as in Fig. 18. Moreover $\mathsf{Query}(\mathcal{C}_1) < \mathsf{Query}(\mathcal{R}) + \mathsf{Query}(\mathcal{A})$ and $\mathsf{Time}(\mathcal{C}_1) \in O(\mathsf{Query}(\mathcal{A}) + \mathsf{Time}(\mathcal{R}))$. Adversary $\mathcal{C}_2$'s complexity is the same.

Next we are going to bound $\Pr\left[\mathsf{G}^1\right] \leqslant \Pr[\mathsf{H}^2] + \Pr[\mathsf{G}^2]$. Game H^2 does not let $\mathcal{R}$ rewind again if it has already passed the buffer, so it captures the possibility that $\mathcal{R}$ passes the challenges on its first attempt past the buffer. Game G^2 only outputs true if $\mathcal{R}$ has passed the buffer, then rewinded, and then passed the buffer again before successfully completing the challenge. To make this formal, let crw denote the event that crw = true at the end of the execution of a game and $\overline{\mathsf{crw}}$ to denote the opposite event in the following probability statements. We claim that,

$$\Pr\left[\mathsf{G}^1 \wedge \overline{\mathsf{crw}}\right] \leqslant \Pr\left[\mathsf{H}^2\right] \text{ and } \Pr\left[\mathsf{G}^1 \wedge \mathsf{crw}\right] \leqslant \Pr\left[\mathsf{G}^2\right]. \quad (5)$$

```
Adversary $\mathcal{D}_1(\phi, M_1, \cdots, M_u)$
If $\phi = \bot$ then
    $K^* \leftarrow \mathsf{NE.Kg}$; $k \leftarrow 0$; $\sigma \leftarrow\$ \mathcal{R}^{\text{Enc}}.\mathsf{Init}$
    While $k \leqslant \mathcal{R}.\text{rew}$ do
        For $i \in [c]$ do // **Buffer Phase**
            $N_i \leftarrow (r+1, i)$
            $(\sigma, \mathsf{rf}, C_i) \leftarrow\$ \mathcal{R}^{\text{Enc}}.\mathsf{SimEnc}(\sigma, N_i, M_0)$
            If $\mathsf{rf}$ then goto NEXT2
        Goto BUF-END
        NEXT2: $k \leftarrow k+1$
    BUF-END: $\phi \leftarrow (1, K^*, \sigma)$
$(i, K^*, \sigma) \leftarrow \phi_1$
For $j \in [u]$ do
    $N_j \leftarrow (i, j)$
    $(\sigma, \cdot, C_j) \leftarrow\$ \mathcal{R}^{\text{Enc}}.\mathsf{SimEnc}(\sigma, N_j, M_j)$
$\phi \leftarrow (i, K^*, \sigma)$
Return $\phi$

Adversary $\mathcal{D}_2(\phi, j^*)$
$(i, K^*, \sigma) \leftarrow \phi_2$
$N_{j*} \leftarrow (i, j^*)$
$C_{j*} \leftarrow \mathsf{NE.E}(K^*, N_{j*}, 0^{\mathsf{ml}})$
$(\sigma, \cdot, M) \leftarrow\$ \mathcal{R}^{\text{Enc}}.\mathsf{SimDec}(N_{j*}, C_{j*})$
$\phi \leftarrow (i+1, K^*, \sigma)$
Return $(\phi, M)$

Oracle $\text{Enc}(N, M)$
Return $\mathsf{NE.E}(K^*, N, 0^{\mathsf{ml}})$
```

Fig. 19. Adversary $(\mathcal{D}_1, \mathcal{D}_2)$ used to prove Lemma 7.

The former of these holds because G^1 and H^2 are identical until crw has been set (and hence $\Pr\left[\mathsf{G}^1 \wedge \overline{\mathsf{crw}}\right] = \Pr\left[\mathsf{H}^1 \wedge \overline{\mathsf{crw}}\right]$). The latter holds because G^2 is identical to G^1 except it only outputs $\mathtt{true}$ if G^1 would *and* $\mathsf{crw} = \mathtt{true}$.

Because $\mathcal{R}$ only has one attempt to pass the challenge phase in H^2, we can directly bound the probability that it succeeds in this by using $\mathcal{R}$ to construct an adversary for $\mathsf{G}^{\text{it-chl-}r}_{u,\mathsf{ml}}$ and bounding its advantage with Lemma 2. This gives the following lemma.

Lemma 6. *Let $\mathcal{R}$ be a full-rewinding, nonce-respecting, faithful black-box reduction from* AE-1 *to nonce-respecting* INDR *with $\mathsf{Mem}(\mathcal{R}) = S$. Then,*

$$\Pr\left[\mathsf{H}^2\right] \leqslant \left(\frac{2(S + \log r + \mathsf{kl}) + 2\mathsf{ml}}{u} + \frac{3}{2^{\mathsf{ml}}}\right)^r$$

where H^2 is defined as in Fig. 18.

Proof. Consider the adversary $(\mathcal{D}_1, \mathcal{D}_2)$ for $\mathsf{G}^{\text{it-chl-}r}_{u,\mathsf{ml}}$ defined in Fig. 19. It assumes that $\mathcal{R}$ will succeed in H^2 and tries to use this to succeed in its own game. The first time $\mathcal{D}_1$ is executed (i.e. when $\phi = \bot$) it simulates the buffer phase, allowing $\mathcal{R}$ to rewind up to $\mathcal{R}$.rew times until $\mathcal{R}$ completes the buffer phase (if $\mathcal{R}$ never completes the buffer phase, reaching the code "Goto BUF-END", then it couldn't have won in H^2 so we do not need to worry about the behavior of $\mathcal{D}_1$ in this case).

After this, and in all subsequent executions, $\mathcal{D}_1$ uses its input as the messages given to $\mathcal{R}$ in one iteration of the challenge. We let $\mathcal{D}_1$ ignore $\mathcal{R}$'s output rf here because if $\mathcal{R}$ ever output $\mathsf{rf} = \mathtt{true}$ then it would be unable to win that execution of H^2. Then $\mathcal{D}_1$ halts, passing the current state of $\mathcal{R}$ to $\mathcal{D}_2$ which uses $\mathcal{R}$ to attempt to answer its challenge.

The view of $\mathcal{R}$ is perfectly simulated for any execution of H^2 in which it would have succeeded, so $\Pr[\mathsf{H}^2] \leqslant \mathsf{Adv}^{\text{it-chl-}r}_{u,\mathsf{ml}}(\mathcal{D}_1, \mathcal{D}_2)$. The state used by $(\mathcal{D}_1, \mathcal{D}_2)$ is a tuple $\phi = (i, K^*, \sigma)$ where $|i| = \log r$, $|K^*| = \mathsf{kl}$, and $|\sigma| = S$ so $(\mathcal{D}_1, \mathcal{D}_2)$ is $S + \log r + \mathsf{kl}$-bounded. Applying Lemma 2 gives the claimed bound. □

We now state two lemmas which give us a bound on $\Pr\left[\mathsf{G}^2\right]$ via G^3. First, in Lemma 7 we use the INDR security of NE to transition from G^2 to G^3 which is identical, except the encryption oracle returns random ciphertexts to $\mathcal{R}$. Its proof is given in Appendix C.3. Now consider G^3. The reduction $\mathcal{R}$ must pass the buffer phase, then rewind, then pass the buffer phase again to win this game. Because it is faithful, passing the buffer phase the second time requires $\mathcal{R}$ to remember c random ciphertexts. In Lemma 8, we use a compression argument (via a lemma from [6]) to give an information theoretic bound on $\Pr\left[\mathsf{G}^3\right]$. Its proof is given in Appendix E.

Lemma 7. *There exists an adversary $\mathcal{C}_3$ such that*

$$\Pr\left[\mathsf{G}^2\right] \leqslant \Pr\left[\mathsf{G}^3\right] + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_3)$$

where G^2 and G^3 are defined as in Fig. 18. Moreover, $\mathsf{Query}(\mathcal{C}_3) < \mathsf{Query}(\mathcal{R}) + \mathsf{Query}(\mathcal{A})$ and $\mathsf{Time}(\mathcal{C}_3) \in O(\mathsf{Query}(\mathcal{A}) + \mathsf{Time}(\mathcal{R}))$.

Lemma 8. *Let $\mathcal{R}$ be a full-rewinding, nonce-respecting, faithful black-box reduction from* AE-1 *to nonce-respecting* INDR *with $\mathsf{Mem}(\mathcal{R}) = S$. Then,*

$$\Pr\left[\mathsf{G}^3\right] \leqslant \frac{2^S}{|\mathsf{C}|^c} = \frac{2^S}{2^{c\cdot \mathsf{NE.cl(ml)}}}$$

where G^3 is defined as in Fig. 18.

We combine (in order) Equation 4, Lemma 5, Equation 5, and Lemma 7 to obtain

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{A}]) &\leqslant \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{S}]) + \Pr\left[\mathsf{G}^0_0\right] + \Pr\left[\mathsf{G}^0_1\right] \\
&\leqslant \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{S}]) + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_1) + 2\cdot \Pr\left[\mathsf{G}^0_0\right] \\
&\leqslant \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{S}]) + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_1) + 2\cdot (\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_2) + \Pr\left[\mathsf{G}^1\right]) \\
&\leqslant \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{S}]) + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_1) + 2\cdot (\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_2) + \Pr[\mathsf{H}^2] + \Pr[\mathsf{G}^2]) \\
&\leqslant \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{S}]) + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_1) + 2\cdot (\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_2) + \Pr[\mathsf{H}^2] + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_3) + \Pr\left[\mathsf{G}^3\right]).
\end{aligned}$$

To complete the proof, we apply our information theoretic bounds from Lemma 7 and Lemma 8, then combine the four INDR adversaries $\mathcal{R}[\mathcal{S}]$, $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$. Let $\mathcal{C}$ be the INDR adversary that randomly chooses one of $\mathcal{R}[\mathcal{S}]$, $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ (with probabilities 1/6, 1/6, 2/6, and 2/6, respectively), then runs the adversary it chose, outputting whatever that adversary does. Simple calculations give

$$6\cdot \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}) = \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{R}[\mathcal{S}]) + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_1) + 2\cdot \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_2) + 2\cdot \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_3)\ .$$

The claimed complexity of $\mathcal{C}$ follows from that of $\mathcal{R}[\mathcal{S}]$, $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$. □

B Analysis of Brute-Force Adversary's Advantage

We prove a bound on the advantage of the adversary $\mathcal{B}$ we use in the proofs of Thm. 5 and Thm. 6.

Lemma 9. *Fix $\mathsf{kl}, \mathsf{ml} \in \mathbb{N}$. Let* NE *be a nonce-based encryption scheme for which $\{0,1\}^{\mathsf{ml}} \subseteq \mathsf{NE.M}$, $[\mathsf{NE.Kg}] = \{0,1\}^{\mathsf{kl}}$, and $\{(0,1),(0,2)\} \subseteq \mathsf{NE.N}$. Then,*

$$\mathsf{Adv}^{\mathsf{ae\text{-}1}}_{\mathsf{NE}}(\mathcal{B}) \geqslant 1 - 2^{\mathsf{kl}}/2^{2\mathsf{NE.cl(ml)}}.$$

for $\mathcal{B}$ as defined in Fig. 16.

Proof. Adversary $\mathcal{B}$ makes two encryption queries then performs a brute-force to check if there is an encryption key consistent with those queries, returning 1 if so, and 0 otherwise. In $\mathsf{G}^{\mathsf{ae\text{-}1}}_1$ there will always be a consistent key so $\mathcal{B}$ will always return 1. For fixed M_1, M_2, and $N_1 \neq N_2$ there are at most 2^{kl} values the ordered pair $(\mathsf{NE.E}(K, N_1, M_1), \mathsf{NE.E}(K, N_2, M_2))$ can take (one for each $K \in \{0,1\}^{\mathsf{kl}}$). Moreover, there are $2^{2\mathsf{NE.cl(ml)}}$ different values of (C_1, C_2) that might be chosen at random by the encryption oracle, so $\mathcal{B}$ will return 1 with probability at most $2^{\mathsf{kl}}/2^{2\mathsf{NE.cl(ml)}}$ when interacting with $\mathsf{G}^{\mathsf{ae\text{-}1}}_{\mathsf{NE},0}$. Combining these observations gives the desired bound. □

```
Adversary $\mathcal{C}_1^{\textsc{Enc}}$, $\boxed{\mathcal{C}_2^{\textsc{Enc}}}$
$\sigma \leftarrow\!\!\$\ \mathcal{R}^{\textsc{SimEnc}}.\mathsf{Init}$
For $i \in [r]$ do
   $j^* \leftarrow\!\!\$\ [u]$
   For $j \in [u]$ do
      $M_j \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ml}}$
      $N_j \leftarrow (i, j)$
      $(\sigma, ., C_j) \leftarrow\!\!\$\ \mathcal{R}^{\textsc{SimEnc}}.\mathsf{SimEnc}(\sigma, N_j, M_j)$
   $(\sigma, ., M) \leftarrow\!\!\$\ \mathcal{R}^{\textsc{SimEnc}}.\mathsf{SimDec}(\sigma, N_{j*}, C_{j*})$
   If $M \neq M_{j*}$ then return 0; $\boxed{\text{return } 1}$
Return 1; $\boxed{\text{Return } 0}$

Oracle $\textsc{SimEnc}(N, M)$
Return $\textsc{Enc}(N, M)$
$\boxed{\text{Return } \textsc{Enc}(N, 0^{\mathsf{ml}})}$
```

Fig. 20. Adversaries $\mathcal{C}_1$ and $\mathcal{C}_2$ used for Lemma 3. The highlighted code is present only in $\mathcal{C}_1$ and the boxed code is present only in $\mathcal{C}_2$.

C Proofs of Lemma 3, Lemma 5, and Lemma 7 (INDR Reductions)

C.1 Proof of Lemma 3

Consider the adversary $\mathcal{C}_1$ shown in Fig. 20. It was obtained by copying the code of G_0^0 and G_1^0 (from Fig. 15), then modifying the code to use its own $\textsc{Enc}$ oracle to respond to encryption queries from $\mathcal{R}$ and, to return 1 whenever the games would output `true` and 0 otherwise. Consequently, $\mathcal{C}_1$ playing $\mathsf{G}_{\mathsf{NE},0}^{\mathsf{indr}}$ (resp. $\mathsf{G}_{\mathsf{NE},1}^{\mathsf{indr}}$) perfectly simulates G_0^0 (resp. G_1^0) to $\mathcal{R}$ and we have

$$\Pr\left[\mathsf{G}_0^0\right] = \Pr\left[\mathsf{G}_{\mathsf{NE},0}^{\mathsf{indr}}(\mathcal{C}_1)\right] \text{ and } \Pr\left[\mathsf{G}_1^0\right] = \Pr\left[\mathsf{G}_{\mathsf{NE},1}^{\mathsf{indr}}(\mathcal{C}_1)\right] .$$

It follows that,

$$\Pr\left[\mathsf{G}_1^0\right] \leqslant \Pr\left[\mathsf{G}_0^0\right] + \mathsf{Adv}_{\mathsf{NE}}^{\mathsf{indr}}(\mathcal{C}_1) .$$

The claims on $\mathcal{C}_1$'s complexity can be verified from its code.

Now consider the adversary $\mathcal{C}_2$ shown in Fig. 20. It was obtained by copying the code of G_0^0 and G^1 (from Fig. 15), then modifying the code to use its own $\textsc{Enc}$ oracle to respond to encryption queries from $\mathcal{R}$, and to return 0 whenever the games would outputs `true` and 1 otherwise. Consequently, $\mathcal{C}_2$ playing $\mathsf{G}_{\mathsf{NE},0}^{\mathsf{indr}}$ (resp. $\mathsf{G}_{\mathsf{NE},1}^{\mathsf{indr}}$) perfectly simulates G_0^0 (resp. G^1) to $\mathcal{R}$ and we have

$$1 - \Pr\left[\mathsf{G}_0^0\right] = \Pr\left[\mathsf{G}_{\mathsf{NE},0}^{\mathsf{indr}}(\mathcal{C}_2)\right] \text{ and } 1 - \Pr\left[\mathsf{G}^1\right] = \Pr\left[\mathsf{G}_{\mathsf{NE},1}^{\mathsf{indr}}(\mathcal{C}_2)\right] .$$

It follows that,

$$\Pr\left[\mathsf{G}_0^0\right] \leqslant \Pr\left[\mathsf{G}^1\right] + \mathsf{Adv}_{\mathsf{NE}}^{\mathsf{indr}}(\mathcal{C}_2) .$$

The claims on $\mathcal{C}_2$'s complexity can be verified from its code. □

C.2 Proof of Lemma 5

Consider the adversary $\mathcal{C}_1$ shown in Fig. 21. It was obtained by copying the code of G_0^0 and G_1^0 (from Fig. 18), then modifying the code to use its own $\textsc{Enc}$ oracle to respond to encryption queries from $\mathcal{R}$, and to return 1 whenever the games would output `true` and 0 otherwise. Consequently, $\mathcal{C}_1$ playing $\mathsf{G}_{\mathsf{NE},0}^{\mathsf{indr}}$ (resp. $\mathsf{G}_{\mathsf{NE},1}^{\mathsf{indr}}$) perfectly simulates G_0^0 (resp. G_1^0) to $\mathcal{R}$ and we have

$$\Pr\left[\mathsf{G}_0^0\right] = \Pr\left[\mathsf{G}_{\mathsf{NE},0}^{\mathsf{indr}}(\mathcal{C}_1)\right] \text{ and } \Pr\left[\mathsf{G}_1^0\right] = \Pr\left[\mathsf{G}_{\mathsf{NE},1}^{\mathsf{indr}}(\mathcal{C}_1)\right] .$$

```
Adversaries $\mathcal{C}_1^{\text{Enc}}$, $\boxed{\mathcal{C}_2^{\text{Enc}}}$
$k \leftarrow 0$; $\sigma \leftarrow\!\!\$\ \mathcal{R}^{\text{SimEnc}}.\mathsf{Init}$
While $k \leqslant \mathcal{R}.\text{rew}$ do
  For $i \in [c]$ do // Buffer Phase
    $N_i \leftarrow (r+1, i)$
    $(\sigma, \mathsf{rf}, C_i) \leftarrow\!\!\$\ \mathcal{R}^{\text{SimEnc}}.\mathsf{SimEnc}(\sigma, N_i, 0^{\mathsf{ml}})$
    If $\mathsf{rf}$ then goto NEXT2
  For $i \in [r]$ do // Challenge Phase
    $j^* \leftarrow\!\!\$\ [u]$
    For $j \in [u]$ do
      $M_j \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ml}}$
      $N_j \leftarrow (i, j)$
      $(\sigma, \mathsf{rf}, C_j) \leftarrow\!\!\$\ \mathcal{R}^{\text{SimEnc}}.\mathsf{SimEnc}(\sigma, N_j, M_j)$
      If $\mathsf{rf}$ then goto NEXT
    $(\sigma, \mathsf{rf}, M) \leftarrow\!\!\$\ \mathcal{R}^{\text{SimEnc}}.\mathsf{SimDec}(\sigma, N_{j^*}, C_{j^*})$
    If $\mathsf{rf}$ then goto NEXT
    If $M \neq M_{j^*}$ then
      $\sigma \leftarrow\!\!\$\ \mathcal{R}^{\text{Enc}}.\mathsf{Upd}(\sigma, 1)$
      Goto NEXT
  Return 1; $\boxed{\text{Return } 0}$
  NEXT: $\mathsf{crw} \leftarrow \mathtt{true}$
  NEXT2: $k \leftarrow k+1$
Return 0; $\boxed{\text{Return } 1}$

Oracle $\text{SimEnc}(N, M)$
$C \leftarrow \text{Enc}(N, M)$
$\boxed{C \leftarrow \text{Enc}(N, 0^{\mathsf{ml}})}$
Return $C$
```

```
Adversary $\mathcal{C}_3^{\text{Enc}}$
$k \leftarrow 0$; $\sigma \leftarrow\!\!\$\ \mathcal{R}^{\text{SimEnc}}.\mathsf{Init}$
While $k \leqslant \mathcal{R}.\text{rew}$ do
  For $i \in [c]$ do // Buffer Phase
    $N_i \leftarrow (r+1, i)$
    $(\sigma, \mathsf{rf}, C_i) \leftarrow\!\!\$\ \mathcal{R}^{\text{SimEnc}}.\mathsf{SimEnc}(\sigma, N_i, 0^{\mathsf{ml}})$
    If $\mathsf{rf}$ then goto NEXT2
  For $i \in [r]$ do // Challenge Phase
    $j^* \leftarrow\!\!\$\ [u]$
    For $j \in [u]$ do
      $M_j \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ml}}$
      $N_j \leftarrow (i, j)$
      $(\sigma, \mathsf{rf}, C_j) \leftarrow\!\!\$\ \mathcal{R}^{\text{SimEnc}}.\mathsf{SimEnc}(\sigma, N_j, M_j)$
      If $\mathsf{rf}$ then goto NEXT
    $(\sigma, \mathsf{rf}, M) \leftarrow\!\!\$\ \mathcal{R}^{\text{SimEnc}}.\mathsf{SimDec}(\sigma, N_{j^*}, C_{j^*})$
    If $\mathsf{rf}$ then goto NEXT
    If $M \neq M_{j^*}$ then
      $\sigma \leftarrow\!\!\$\ \mathcal{R}^{\text{SimEnc}}.\mathsf{Upd}(\sigma, 1)$
      Goto NEXT
  If $\mathsf{crw}$ then return 1 else return 0
  NEXT: $\mathsf{crw} \leftarrow \mathtt{true}$
  NEXT2: $k \leftarrow k+1$
Return 0

Oracle $\text{SimEnc}(N, M)$
$C \leftarrow \text{Enc}(N, 0^{\mathsf{ml}})$
Return $C$
```

Fig. 21. Adversaries $\mathcal{C}_1$ and $\mathcal{C}_2$ for Lemma 5 and adversary $\mathcal{C}_3$ for Lemma 7. Highlighted code is only included in $\mathcal{C}_1$ and boxed code is only included in $\mathcal{C}_2$.

It follows that,

$$\Pr\left[\mathsf{G}_1^0\right] \leqslant \Pr\left[\mathsf{G}_0^0\right] + \mathsf{Adv}_{\mathsf{NE}}^{\mathsf{indr}}(\mathcal{C}_1) .$$

The claims on $\mathcal{C}_1$'s complexity can be verified from its code.

Now consider the adversary $\mathcal{C}_2$ shown in Fig. 21. It was obtained by copying the code of G_0^0 and G^1 (from Fig. 18), then modifying the code to use its own Enc oracle to respond to encryption queries from $\mathcal{R}$, and to return 0 whenever the games would output `true` and 1 otherwise. Consequently, $\mathcal{C}_2$ playing $\mathsf{G}_{\mathsf{NE},0}^{\mathsf{indr}}$ (resp. $\mathsf{G}_{\mathsf{NE},1}^{\mathsf{indr}}$) perfectly simulates G_0^0 (resp. G^1) to $\mathcal{R}$ and we have

$$1 - \Pr\left[\mathsf{G}_0^0\right] = \Pr\left[\mathsf{G}_{\mathsf{NE},0}^{\mathsf{indr}}(\mathcal{C}_2)\right] \text{ and } 1 - \Pr\left[\mathsf{G}^1\right] = \Pr\left[\mathsf{G}_{\mathsf{NE},1}^{\mathsf{indr}}(\mathcal{C}_2)\right] .$$

It follows that,

$$\Pr\left[\mathsf{G}_0^0\right] \leqslant \Pr\left[\mathsf{G}^1\right] + \mathsf{Adv}_{\mathsf{NE}}^{\mathsf{indr}}(\mathcal{C}_2) .$$

The claims on $\mathcal{C}_2$'s complexity can be verified from its code. □

C.3 Proof of Lemma 7

Proof. Consider the adversary $\mathcal{C}_3$ shown in Fig. 21. It was obtained by copying the code of G^2 and G^3 (from Fig. 18), then modifying the code to use its own Enc oracle to respond to encryption queries from $\mathcal{R}$, and

```
Adversary $\mathcal{D}_1(\phi, M_1, \cdots, M_u)$
If $\phi = \perp$ then
    $i \leftarrow 1$; $K^* \leftarrow \mathsf{NE.Kg}$; $\sigma \leftarrow\!\!\$\ \mathcal{R}^{\text{Enc}}.\mathsf{Init}$
Else
    $(i, K^*, \sigma) \leftarrow \phi$
Foreach $j \in [u]$ do
    $N_j \leftarrow (i, j)$
    $(\sigma, ., C_j) \leftarrow\!\!\$\ \mathcal{R}^{\text{Enc}}.\mathsf{SimEnc}(\sigma, N_{i,j}, M_j)$
$\phi \leftarrow (i, K^*, \sigma)$
Return $\phi$

Adversary $\mathcal{D}_2(\phi, j^*)$
$(i, K^*, \sigma) \leftarrow \phi$
$N_{j*} \leftarrow (i, j^*)$
$C_{j*} \leftarrow \mathsf{NE.E}(K^*, N_{j*}, 0^{\mathsf{ml}})$
$(\sigma, ., M) \leftarrow\!\!\$\ \mathcal{R}^{\text{Enc}}.\mathsf{SimDec}(N_{j*}, M_{j*})$
$\phi \leftarrow (i+1, K^*, \sigma)$
Return $(\phi, M)$

Oracle $\text{Enc}(N, M)$
Return $\mathsf{NE.E}(K^*, N, 0^{\mathsf{ml}})$
```

Fig. 22. Adversary $(\mathcal{D}_1, \mathcal{D}_2)$ used in the proof of Lemma 4.

to return 1 whenever the games would output `true` and 0 otherwise. Consequently, $\mathcal{C}_3$ playing $\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},0}$ (resp. $\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},1}$) perfectly simulates G^3 (resp. G^2) to $\mathcal{R}$ and we have

$$\Pr\left[\mathsf{G}^3\right] = \Pr\left[\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},0}(\mathcal{C}_3)\right] \text{ and } \Pr\left[\mathsf{G}^2\right] = \Pr\left[\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},1}(\mathcal{C}_3)\right] .$$

It follows that,

$$\Pr\left[\mathsf{G}^2\right] \leqslant \Pr\left[\mathsf{G}^3\right] + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{C}_3) .$$

The claims on $\mathcal{C}_3$'s complexity can be verified from its code. □

D Proof of Lemma 4

Consider the adversary $(\mathcal{D}_1, \mathcal{D}_2)$ shown in Fig. 22. Adversary $\mathcal{D}_1$ samples a key for NE and starts running $\mathcal{R}$ from the beginning in the first round of $\mathsf{G}^{\mathsf{it\text{-}chl\text{-}}r}_{u,\mathsf{ml}}$ (when $\phi = \perp$) and from the state at which $\mathcal{R}$ was last stopped in later rounds. It responds to the encryption queries of $\mathcal{R}$ as in G^1 (Fig. 15). Adversary $\mathcal{D}_1$ makes u encryption queries which are distributed identically to the queries made to $\mathcal{R}$ in one iteration of G^1 since the inputs to $\mathcal{D}_1$ are sampled uniformly at random in $\mathsf{G}^{\mathsf{it\text{-}chl\text{-}}r}_{u,\mathsf{ml}}$. It then stops running $\mathcal{R}$ and outputs its state ϕ which consists of the state σ of $\mathcal{R}$ along with the iteration number i and the key K^*. Adversary $\mathcal{D}_2$ resumes running $\mathcal{R}$ from the state it was last stopped and makes a decryption query which is identically distributed to the decryption query made in G^1 because j^* is chosen at random and independently of the M_i's in $\mathsf{G}^{\mathsf{it\text{-}chl\text{-}}r}_{u,\mathsf{ml}}$. It outputs the message M it received as answer to the decryption query along with a state ϕ which consists of the current state of $\mathcal{R}$, the next iteration number $i+1$, and the key K^*.

It follows from their that during the r iterations of $\mathsf{G}^{\mathsf{it\text{-}chl\text{-}}r}_{u,\mathsf{ml}}$, the view of $\mathcal{R}$ provided by $(\mathcal{D}_1, \mathcal{D}_2)$ perfectly matches its view in G^1. Note that $\mathsf{G}^{\mathsf{it\text{-}chl\text{-}}r}_{u,\mathsf{ml}}(\mathcal{D}_1, \mathcal{D}_2)$ will output `true` whenever G^1 would output `true`, so $\Pr[\mathsf{G}^1] \leqslant \mathsf{Adv}^{\mathsf{it\text{-}chl\text{-}}r}_{u,\mathsf{ml}}(\mathcal{D}_1, \mathcal{D}_2)$.

The state of $(\mathcal{D}_1, \mathcal{D}_2)$ is always a tuple (i, K^*, σ) for which $|i| = \log r$, $|K^*| = \mathsf{kl}$, and $|\sigma| = S$ all hold. Hence $(\mathcal{D}_1, \mathcal{D}_2)$ is $(S + \log r + \mathsf{kl})$-bounded and the desired bound follows by applying Lemma 2. □

E Proof of Lemma 8 (Compression Argument)

For this proof we use the compression lemma given in [6], which we state here as a proposition. It quantitatively formalizes that it is impossible to compress a random element in set $\mathcal{X}$ to a string that than $\log|\mathcal{X}|$ bits long, even relative to a random string.

Proposition 1. *Let* Encode *be a randomized map from* $\mathcal{X}$ *to* $\mathcal{Y}$ *and let* Decode *be a randomized map from* $\mathcal{Y}$ *to* $\mathcal{X}$ *such that*

$$\Pr\left[x \leftarrow\!\!\$\ \mathcal{X} : \mathsf{Decode}(\mathsf{Encode}(x)) = x\right] \geqslant \varepsilon ,$$

then $\log|\mathcal{Y}| \geqslant \log|\mathcal{X}| - \log(1/\varepsilon)$.

```
Algorithm $\mathsf{Encode}((T_1, \cdots, T_c))$
$k \leftarrow 0$; $\sigma \leftarrow\$ \mathcal{R}^{\text{Enc}}.\mathsf{Init}$
While $k \leqslant \mathcal{R}.\text{rew}$ do
   For $i \in [c]$ do  // **Buffer Phase**
      $N_i \leftarrow (r+1, i)$
      $(\sigma, \mathsf{rf}, C_i) \leftarrow\$ \mathcal{R}^{\text{Enc}}.\mathsf{SimEnc}(\sigma, N_i, 0^{\mathsf{ml}})$
      If $\mathsf{rf}$ then goto NEXT2
   For $i \in [r]$ do  // **Challenge Phase**
      $j^* \leftarrow\$ [u]$
      For $j \in [u]$ do
         $M_j \leftarrow\$ \{0,1\}^{\mathsf{ml}}$
         $N_j \leftarrow (i, j)$
         $(\sigma, \mathsf{rf}, C_j) \leftarrow\$ \mathcal{R}^{\text{Enc}}.\mathsf{SimEnc}(\sigma, N_j, M_j)$
         If $\mathsf{rf}$ then return $\sigma$
      $(\sigma, \mathsf{rf}, M) \leftarrow\$ \mathcal{R}^{\text{Enc}}.\mathsf{SimDec}(\sigma, N_{j*}, C_{j*})$
      If $\mathsf{rf}$ then return $\sigma$
      If $M \neq M_{j*}$ then
         $\sigma \leftarrow\$ \mathcal{R}^{\text{Enc}}.\mathsf{Upd}(\sigma, 1)$
         Return $\sigma$
   Return $\sigma$
   NEXT: $\mathsf{crw} \leftarrow \mathtt{true}$
   NEXT2: $k \leftarrow k + 1$
Return $\sigma$

Oracle $\text{Enc}(N, M)$
For $i \in [c]$ do
   If $(N, M) = ((r+1, i), 0^{\mathsf{ml}})$ then return $T_i$
$C \leftarrow\$ \mathsf{C}$
Return $C$
```

```
Algorithm $\mathsf{Decode}(\sigma)$
$k \leftarrow 0$
While $k \leqslant \mathcal{R}.\text{rew}$ do
   For $i \in [c]$ do  // **Buffer Phase**
      $N_i \leftarrow (r+1, i)$
      $(\sigma, \mathsf{rf}, C_i^*) \leftarrow\$ \mathcal{R}^{\text{Enc}}.\mathsf{SimEnc}(\sigma, N_i, 0^{\mathsf{ml}})$
      If $\mathsf{rf}$ then goto NEXT2
   NEXT2: $k \leftarrow k + 1$
Return $(C_1^*, \cdots, C_c^*)$

Oracle $\text{Enc}(N, M)$
$C \leftarrow\$ \mathsf{C}$
Return $C$
```

Fig. 23. Procedures Encode and Decode. Highlighting indicates interesting code changes from G^2.

To apply this proposition we use the reduction $\mathcal{R}$ to construct procedures Encode and Decode which attempt to encode an element of $\mathcal{X} = \mathsf{C}^c$ with a state $\sigma \in \{0,1\}^S = \mathcal{Y}$ and succeed whenever G^3 (Fig. 18) would output $\mathtt{true}$. They are shown in Fig. 23.

The procedure Encode was obtained by copying the code of G^3, then making two modifications. First, whenever $\mathcal{R}$ makes a Enc oracle query corresponding to the ith buffer query, Encode responds using its input string $T_i \in \mathsf{C}$ (the other queries are answered at random as in G^3). Second, Encode returns the state σ of $\mathcal{R}$ whenever crw would be set in G^3 (i.e. whenever there was a goto NEXT) or whenever G^3 would halt and return an output.

The procedure Decode uses σ to run $\mathcal{R}$ and see if it would ever again pass the buffer phase. If $\mathcal{R}$ does so, Decode does not need to continue simulating its view correctly because it already has the required values. Because Decode does not know how many times $\mathcal{R}$ has already rewinded, it simply allows $\mathcal{R}$ to rewind $\mathcal{R}$.rew times. The final output of Decode consists of the strings $C_1^*, \cdots, C_c^*$ that $\mathcal{R}$ responded to buffer queries with.

We argue that Decode answers correctly if G^3 would return $\mathtt{true}$. Since G^3 returns $\mathtt{true}$ only if crw is set, we only look at executions of Encode and Decode where crw would have been set in G^3. Observe that Encode simulates G^3 to $\mathcal{R}$ perfectly right up until crw is set in G^3 (the simulation of Enc is perfect because $T_1, \cdots, T_c$ are distributed uniformly at random in C). Then Decode perfectly simulates G^3 to $\mathcal{R}$ from the point crw is set until $\mathcal{R}$ passes the buffer phase (or until G^3 would have halted without $\mathcal{R}$ passing the buffer again). While Encode and Decode may end up running $\mathcal{R}$ for more than $\mathcal{R}$.rew iterations, G^3 would have returned $\mathtt{false}$ if it has completed $\mathcal{R}$.rew iterations and $\mathcal{R}$ had never passed the buffer phase a second time, so we need not worry about incorrectness simulation after the first $\mathcal{R}$.rew iterations.

```
Games $\mathsf{G}_0$, $\mathsf{G}_1$
$\mathsf{sync} \leftarrow \mathtt{true}$
$N_e \leftarrow\!\!\$\ \{0,1\}^{\mathsf{CAU.nl}}$
$N_d \leftarrow N_e$
$K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{CAU.kl}}$
$L \leftarrow \mathrm{E}_K(0^{\mathsf{CAU.bl}})$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Enc},\textsc{Dec}}$
Return $b' = 1$

Oracle $\textsc{Enc}(M)$
$C \leftarrow \mathrm{E}(M)$
$\mathbf{M}.\mathsf{add}(M)$; $\mathbf{C}.\mathsf{add}(C)$
Return $C$
```

```
Oracle $\textsc{Dec}(C)$
If $N_d = \perp$ then return $\perp$
$N_d \leftarrow N_d + 1$
$M' \leftarrow \mathbf{M}.\mathsf{dq}()$
$C' \leftarrow \mathbf{C}.\mathsf{dq}()$
If $\mathsf{sync}$ then
    If $C = C'$ then
        Return $M'$
    $\mathsf{sync} \leftarrow \mathtt{false}$
$M_1 \leftarrow \mathrm{D}(C)$
$M_0 \leftarrow \perp$
Return $M_b$
```

```
Algorithm $\mathrm{E}(M)$
$N_e \leftarrow N_e + 1$
$Y \leftarrow \mathrm{pad}(N_e)$
$M_1 \dots M_\ell \leftarrow_n M$
For $i = 1, \dots, \ell$ do
    $C_i \leftarrow M_i \oplus E_K(Y + i)$
$C \leftarrow C_1 \dots C_\ell$
$T \leftarrow H_L(A, C) \oplus E_K(Y)$
Return $((K, N), T \,\|\, C)$
```

```
Algorithm $\mathrm{D}(T \,\|\, C)$
$Y \leftarrow \mathrm{pad}(N_d)$
$C_1 \dots C_\ell \leftarrow_n C$
$T' \leftarrow H_L(A, C) \oplus E_K(Y)$
If $T = T'$ then
    $\mathsf{bad}_{\mathsf{forge}} \leftarrow \mathtt{true}$
    $N_d \leftarrow \perp$
    Return $\perp$
    For $i = 1, \dots, \ell$ do
        $M_i \leftarrow C_i \oplus E_K(Y + i)$
    $M \leftarrow M_1 \dots M_\ell$
    Return $M$
$N_d \leftarrow \perp$
Return $\perp$
```

Fig. 24. Games G_0 and G_1 introducing $\mathsf{bad}_{\mathsf{forge}}$ for proof of Thm. 4. Highlighted code is only included in G_1.

Now, because $\mathcal{R}$ is faithful, it must have queried the buffer queries to $\textsc{Enc}$ before passing the buffer phase in Encode and on every rewind, it answers with the same answers it had given previously. Hence, if $\mathcal{R}$ passes the buffer phase in both Encode and Decode, then it must have responded with $T_1, \cdots, T_c$ in Decode. Consequently, if G^3 would return $\mathtt{true}$, then $T_1, \cdots, T_c$ will be output by Decode. So, we have that

$$\Pr\left[(T_1, \cdots, T_c) \leftarrow\!\!\$\ \mathsf{C}^c : \mathsf{Decode}(\mathsf{Encode}((T_1, \cdots, T_c))) = (T_1, \cdots, T_c)\right] \geqslant \Pr\left[\mathsf{G}^3\right] .$$

Letting $\Pr[\mathsf{G}^3] = \varepsilon$ and applying Proposition 1 gives $\log 2^S \geqslant \log |\mathsf{C}|^c - \log 1/\varepsilon$. Then rearranging the terms gives $\varepsilon \leqslant 2^S/|\mathsf{C}|^c$, which concludes the proof.

F Proof of Thm. 4 ($\mathsf{NCH}[\mathsf{CAU}[\mathrm{E}, \mathrm{H}]]$ is CTXT secure)

In this section we provide a proof of Thm. 4 which gave a concrete bound on the CTXT security of $\mathsf{NCH}[\mathsf{NE}]$ for $\mathsf{NE} = \mathsf{CAU}[\mathrm{E}, \mathrm{H}]$.

Proof. Consider the game G_0 shown in Fig. 24. It was obtained by plugging the code of $\mathsf{NCH}[\mathsf{NE}]$ into $\mathsf{G}^{\mathsf{ae}}_{\mathsf{NCH[NE]},1}$ and making some minor organizational changes (e.g. L is computed once for all time and encryptions/decryptions are performed in the separate algorithms E and D). The game G_1 is the same except that when a forgery would occur (i.e. when $\mathsf{bad}_{\mathsf{forge}}$ is set) the oracle D sets the nonce to $\perp$ and returns $\perp$ anyway. Equivalently, this game always returns $\perp$ once sync is false. Because of this we have that G_1 is identical to $\mathsf{G}^{\mathsf{ae}}_{\mathsf{NCH[NE]},0}$. Thus we have that

$$\mathsf{Adv}^{\mathsf{ch\text{-}ctxt}}_{\mathsf{NCH[NE]}}(\mathcal{A}) = \Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_1] \leqslant \Pr[\mathsf{G}_1 \text{ sets } \mathsf{bad}_{\mathsf{forge}}].$$

The rest of the proof is dedicated to bounding this probability.

Transitioning to random. Recall that CAU's authentication is inspired by a Carter-Wegman MAC. Thus we ultimately expect to reduce security to the AXU security of H. However, to do so we first will need to replace the output of E with random so that the key L looks random and the encryption queries before the forgery attempt have not revealed anything about L. Naturally, this consists of a reduction to the PRF security of E, but this is somewhat difficult in our setting. We'd like the reduction to be memory tight and not repeat PRF queries. This makes it difficult to simulate $\textsc{Dec}$ while $\mathsf{sync} = \mathtt{true}$.

Ultimately, the approach we will take here closely mirrors that used in our proof of Thm. 2. We will assume that the adversary $\mathcal{A}$ cannot remember too many bits of ciphertext at a time because the ciphertexts should look random to it. If there are ever more than that many bits of ciphertext that $\mathcal{A}$ has received from

```
Games $\boxed{\mathsf{H}_0}$, $\mathsf{H}_1$, $\mathsf{H}_2$, $\mathsf{H}_3$
flag ← true
sync ← true
$N_e \leftarrow\$ \{0,1\}^{\mathsf{CAU.nl}}$
$N_d \leftarrow N_e$
$K \leftarrow\$ \{0,1\}^{\mathsf{CAU.kl}}$
$L \leftarrow \mathrm{RF}(0^{\mathsf{CAU.bl}})$
Run $\mathcal{A}^{\mathrm{Enc},\mathrm{Dec}}$
Return false

Oracle $\mathrm{Enc}(M)$
$C \leftarrow \mathrm{E}(M)$
$\mathbf{M}_2$.add($M$); $\mathbf{C}_2$.add($C$)
If flag then
    If $||\mathbf{C}|| + |C| < \delta$ then
        $\mathbf{M}$.add($M$); $\mathbf{C}$.add($C$)
    Else
        flag ← false
Return $C$
```

```
Oracle $\mathrm{Dec}(C)$
$N_d \leftarrow N_d + 1$
$M' \leftarrow \mathbf{M}$.dq()
$C' \leftarrow \mathbf{C}$.dq()
$M_2 \leftarrow \mathbf{M}_2$.dq()
$C_2 \leftarrow \mathbf{C}_2$.dq()
If $C = C'$ then
    Return $M'$
Elif $C = C_2$ then
    bad ← false
    $\boxed{\text{Return } M_2}$
    abort(false)
$T \,\|\, C \leftarrow C$
$Y \leftarrow \mathrm{pad}(N_d)$
$X \leftarrow H_L(A, C)$
If $T = X \oplus \mathrm{RF}(Y)$ then
    $\mathsf{bad}_{\mathsf{forge}}$ ← true
    abort(true)
abort(false)
```

```
Algorithm $\mathrm{E}(M)$
$N_e \leftarrow N_e + 1$
$Y \leftarrow \mathrm{pad}(N_e)$
$M_1 \ldots M_\ell \leftarrow_n M$
$C_1 \ldots C_\ell \leftarrow\$ \{0,1\}^{|M|}$
For $i = 1, \ldots, \ell - 1$ do
    $F[Y + i] \leftarrow C_i \oplus M_i$
    $C_i \leftarrow M_i \oplus \mathrm{RF}(Y + i)$
$C \leftarrow C_1 \ldots C_\ell$
$T \leftarrow\$ \{0,1\}^{\mathsf{CAU.bl}}$
$F[Y] \leftarrow T \oplus H_L(A, C)$
$T \leftarrow H_L(A, C) \oplus \mathrm{RF}(Y)$
Return $T \,\|\, C$

Oracle $\mathrm{RF}(x)$
If $F[x] = \perp$ then
    $F[x] \leftarrow\$ \{0,1\}^{\mathsf{CAU.bl}}$
$y \leftarrow F[x]$
$y \leftarrow \mathrm{E}_K(x)$
Return $y$
```

Fig. 25. Games used to analyze $\mathsf{bad}_{\mathsf{forge}}$ for proof of Thm. 4. Boxed or highlighted code is only included in correspondingly indicated game(s).

Enc, but not forwarded to Dec then we can stop remembering additional ciphertext bits in our table and assume the sender and receiver will be out of sync before $\mathcal{A}$ has forwarded on all the ciphertext bits.

To start this analysis consider the game H_0 shown in Fig. 25 for which we claim $\Pr[\mathsf{G}_1 \text{ sets } \mathsf{bad}_{\mathsf{forge}}] = \Pr[\mathsf{H}_0]$. Note that H_0 returns $\mathsf{bad}_{\mathsf{forge}}$ rather than something depending on the bit guessed by $\mathcal{A}$. To obtain H_0 from G_1 we have made a few syntactic changes that do not effect the probability $\mathsf{bad}_{\mathsf{forge}}$ is set. First, the use of tables surrounding sync has been re-organized to use two copies of the table as was done in proving Thm. 2. The flag sync has been removed because we will simply abort whenever it would be changed. Next, all executions of E_K has been forwarded to an oracle Rf which simply runs E. Finally, we have added code to Dec to make the game abort and output $\mathsf{bad}_{\mathsf{forge}}$ rather than return $\perp$ and continue execution (after a failed attempt to set $\mathsf{bad}_{\mathsf{forge}}$ in G_1 the nonce N_d would be set to $\perp$, preventing $\mathsf{bad}_{\mathsf{forge}}$ from ever being set again in the future).

The game H_1 is the analog of G_3 in the proof of Thm. 2. It differs from H_0 by never using $\mathbf{M}_2$ to determine the output of Dec, thereby limiting how much memory is needed to simulate it. Games H_0 and H_1 are identical-until-bad so we have that

$$\Pr[\mathsf{H}_0] - \Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_1 \text{ sets } \mathsf{bad}].$$

Prf Transition. Having moved to a game that can be simulated with low memory we can now apply PRF security to replace the output of E with random. Consider the game H_2. It differs from H_1 in that the highlighted code has been removed from Rf which now implements a (lazily sampled) random function. In Fig. 26, we give PRF adversaries $\mathcal{A}'_{\mathsf{prf}}$ and $\mathcal{A}''_{\mathsf{prf}}$ satisfying

$$\Pr[\mathsf{H}_1] - \Pr[\mathsf{H}_2] \leqslant \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{E}}(\mathcal{A}'_{\mathsf{prf}}) \text{ and } \Pr[\mathsf{H}_1 \text{ sets } \mathsf{bad}] - \Pr[\mathsf{H}_2 \text{ sets } \mathsf{bad}] \leqslant \mathsf{Adv}^{\mathsf{prf}}_{\mathrm{E}}(\mathcal{A}''_{\mathsf{prf}}).$$

These adversaries simulate the view of $\mathcal{A}$ in H_1 or H_2 by using its Eval oracle whenever E would be executed, the first outputting 1 whenever $\mathsf{bad}_{\mathsf{forge}}$ would be set and the second outputting 1 whenever bad would be set. They store one extra ciphertext C^* beyond what $\mathbf{C}$ stores to know when bad occurs. We want to be careful that neither adversary repeats queries to Eval. This is immediate for $\mathcal{A}''$. When $\mathcal{A}'$ checks if the tag of a

```
Adversaries $\mathcal{A}'^{\textsc{Eval}}_{\mathsf{prf}}$, $\mathcal{A}''^{\textsc{Eval}}_{\mathsf{prf}}$
$\mathsf{flag} \leftarrow \mathtt{true}$
$N_e \leftarrow_\$ \{0,1\}^{\mathsf{CAU.nl}}$
$N_d \leftarrow N_e$
$L \leftarrow \textsc{Eval}(0^{\mathsf{CAU.bl}})$
Run $\mathcal{A}^{\textsc{SimEnc},\textsc{SimDec}}$
Return 0

Algorithm $\mathrm{E}(M)$
$N_e \leftarrow N_e + 1$
$Y \leftarrow \mathrm{pad}(N_e)$
$M_1 \ldots M_\ell \leftarrow_n M$
For $i = 1, ..., \ell - 1$ do
    $Z_i \leftarrow \textsc{Eval}(Y + i)$
    $C_i \leftarrow M_i \oplus Z_i$
$C \leftarrow C_1 \ldots C_\ell$
$X \leftarrow H_L(A, C)$
$T \leftarrow X \oplus \textsc{Eval}(Y)$
Return $T \,\|\, C$
```

```
Oracle $\textsc{SimEnc}(M)$
$C \leftarrow \mathrm{E}(M)$
If $\mathsf{flag}$ then
    If $||\mathbf{C}|| + |C| < \delta$ then
        $\mathbf{M}.\mathsf{add}(M)$
        $\mathbf{C}.\mathsf{add}(C)$
    Else
        $\mathbf{C}.\mathsf{add}(*)$
        $C^* \leftarrow C$
        $\mathsf{flag} \leftarrow \mathtt{false}$
Return $C$
```

```
Oracle $\textsc{SimDec}(C)$
$N_d \leftarrow N_d + 1$
$M' \leftarrow \mathbf{M}.\mathsf{dq}()$; $C' \leftarrow \mathbf{C}.\mathsf{dq}()$
If $C = C'$ then
    Return $M'$
Elif $C' = *$ and $C = C^*$ then
    abort(1)
    abort(0)
abort(0)
$Y \leftarrow \mathrm{pad}(N_d)$
$T \,\|\, C \leftarrow C$
$X \leftarrow H_L(A, C)$
If $C' = *$ then $C' \leftarrow C^*$
If $C' \neq \perp$ then
    $T' \,\|\, C' \leftarrow C'$
    $X' \leftarrow H_L(A, C')$
    $Z \leftarrow T' \oplus X'$
Else
    $Z \leftarrow \textsc{Eval}(Y)$
If $T = X \oplus Z$ then
    abort(1)
abort(0)
```

Fig. 26. PRF adversaries for proof of Thm. 4. Highlighted code is only used by the highlighted adversary.

ciphertext is correct in SimDec we have to be careful not to create a repeat query. This would be possible if $N_e \geqslant N_d$, but in that case we have the corresponding ciphertext stored from which we can re-derive the appropriate output of Eval (referred to as Z).

Rewriting transition. Our next transition (to H_3) is merely a conceptual change which does not effect the behavior at the game. This gives

$$\Pr[\mathsf{H}_2] - \Pr[\mathsf{H}_3] = 0 \text{ and } \Pr[\mathsf{H}_2 \text{ sets } \mathsf{bad}] - \Pr[\mathsf{H}_3 \text{ sets } \mathsf{bad}] = 0.$$

The change occurs solely in E where, rather than creating the output ciphertext by xor-ing things with entries of the table F (via Rf), we instead pick these outputs at random and use them to define F to match. Note that this results in a uniform distribution for F either way as long as F had not already been defined at that point. This is necessarily the case; E never repeats queries to Rf because N_e is always always increasing and if Dec makes a query to Rf, then the game will abort immediately afterwards.

Bounding probability of bad. At this point we are in a game where the adversary is given ciphertexts which are picked uniformly at random. To set bad is must at some point remember more than δ bits of ciphertext that it delayed sending to Dec. In the exact same way we saw in the proof of Thm. 2 we can create an $(\mathcal{A}_1, \mathcal{A}_2)$ that succeeds in $\mathsf{G}^{\mathsf{it}}_{q \cdot x, \delta}$ with the same probability that bad is set, giving

$$\Pr[\mathsf{H}_3 \text{ sets } \mathsf{bad}] \leqslant \mathsf{Adv}^{\mathsf{it}}_{q \cdot x, \delta}(\mathcal{A}_1, \mathcal{A}_2).$$

This advantage is then bounded by $q \cdot x \cdot 2^S / 2^\delta$ using Lemma 1.

Final bound. Finally we can conclude the proof by analyzing $\Pr[\mathsf{H}_3]$. This reduces naturally to the AXU security of the hash function H. Note that Enc returns random bits, so when $\mathcal{A}$ makes its one attempt to set $\mathsf{bad}_{\mathsf{forge}}$ its view is independent of L. Assuming $N_e \geqslant N_d$ (the $N_e < N_d$ case is strictly harder) at the time of this attempt, the adversary has seen a random tag T and random C from which $F[\mathsf{pad}(N_d)]$ was defined to equal $T \oplus H_L(A, C)$. Letting $T^* \,\|\, C$ represent its final query, it will win if $T^* = H_L(A, C^*) \oplus F[\mathrm{pad}(N_d)]$.

```
Adversary $\mathcal{X}$
$\mathsf{flag} \leftarrow \mathtt{true}$
$N_e \leftarrow_\$ \{0,1\}^{\mathsf{CAU.nl}}$
$N_d \leftarrow N_e$
Run $\mathcal{A}^{\textsc{SimEnc},\textsc{SimDec}}$
Return $\bot$

Oracle $\textsc{SimEnc}(M)$
$C \leftarrow \mathrm{E}(M)$
$\mathbf{M}_2.\mathsf{add}(M)$; $\mathbf{C}_2.\mathsf{add}(C)$
If $\mathsf{flag}$ then
    If $||\mathbf{C}|| + |C| < \delta$ then
        $\mathbf{M}.\mathsf{add}(M)$; $\mathbf{C}.\mathsf{add}(C)$
    Else
        $\mathsf{flag} \leftarrow \mathtt{false}$
Return $C$
```

```
Oracle $\textsc{SimDec}(C)$
$N_d \leftarrow N_d + 1$
$M' \leftarrow \mathbf{M}.\mathsf{dq}()$; $C' \leftarrow \mathbf{C}.\mathsf{dq}()$
$M_2 \leftarrow \mathbf{M}_2.\mathsf{dq}()$; $C_2 \leftarrow \mathbf{C}_2.\mathsf{dq}()$
If $C = C'$ then return $M'$
Elif $C = C_2$ then $\mathbf{abort}(\bot)$
$T \,\|\, C \leftarrow C$
If $C_2 \neq \bot$ then
    $T_2 \,\|\, C_2 \leftarrow C_2$
Else
    $T_2 \leftarrow_\$ \{0,1\}^{\mathsf{CAU.bl}}$
    $C_2 \leftarrow C \oplus 1^{|C|}$
$x \leftarrow (A, C)$
$x_2 \leftarrow (A, C_2)$
$\mathbf{abort}(x_1, x_2, T \oplus T_2)$
```

```
Algorithm $\mathrm{E}(M)$
$N_e \leftarrow N_e + 1$
$C \leftarrow_\$ \{0,1\}^{|M|}$
$T \leftarrow_\$ \{0,1\}^{\mathsf{CAU.bl}}$
Return $T \,\|\, C$
```

Fig. 27. Adversary against the AXU security of H.

Plugging in for F and re-arranging gives $T^* \oplus T = H_L(A, C^*) \oplus H_L(A, C)$. The adversary $\mathcal{X}$ shown in Fig. 27 captures this idea. It simulates the view of $\mathcal{A}$ with random strings until a query is made to $\textsc{Dec}$ that has the potential of setting $\mathsf{bad}_{\mathsf{forge}}$, in which case it provides (In the case that $N_d > N_e$ it just picks T_2 at random and sets C_2 to anything other than C. This gives the correct probability distribution because $\mathcal{A}$'s view would have been independent of $F[\mathsf{pad}(N_d)]$ at that point and it would have needed to guess $H_L(A, C^*) \oplus F[\mathrm{pad}(N_d)]$ to set the flag.)

The above reasoning tells us that

$$\Pr[\mathsf{H}_3] = \mathsf{Adv}^{\mathsf{axu}}_{\mathrm{H}}(\mathcal{X}).$$

Completing the proof is then a simply matter of combining all of the equations we have established. The PRF adversary $\mathcal{A}$ is defined to randomly choose either $\mathcal{A}'_{\mathsf{prf}}$ and $\mathcal{A}''_{\mathsf{prf}}$ (each with probability 1/2) and then run the adversary it chose, outputting whatever that adversary does. □