A preliminary version of this paper appears in CRYPTO 2019. This is the full version.

Nonces are Noticed: AEAD Revisited

Mihir Bellare[1] Ruth Ng[2] Björn Tackmann[3]

June 1, 2019

Abstract

We draw attention to a gap between theory and usage of nonce-based symmetric encryption, under which the way the former treats nonces can result in violation of privacy in the latter. We bridge the gap with a new treatment of nonce-based symmetric encryption that modifies the syntax (decryption no longer takes a nonce), upgrades the security goal (asking that not just messages, but also nonces, be hidden) and gives simple, efficient schemes conforming to the new definitions. We investigate both basic security (holding when nonces are not reused) and advanced security (misuse resistance, providing best-possible guarantees when nonces are reused).

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: mihir@eng.ucsd.edu. URL: http://cseweb.ucsd.edu/~mihir/. Supported in part by NSF grants CNS-1526801 and CNS-1717640, ERC Project ERCC FP7/615074 and a gift from Microsoft.

[2] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: ring@eng.ucsd.edu. URL: http://cseweb.ucsd.edu/~ring. Supported by DSO National Laboratories

[3] IBM Research – Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland. Email: bta@zurich.ibm.com. URL: https://researcher.watson.ibm.com/researcher/view.php?person=zurich-BTA.

Contents

1 Introduction

This paper revisits nonce-based symmetric encryption, raising some concerns, and then addressing them, via a new syntax, a new framework of security definitions, and schemes that offer both usability and security benefits.

BACKGROUND. As the applications and usage of symmetric encryption have evolved and grown, so has a theory that seeks to support and guide them. A definition of symmetric encryption (as with any other primitive) involves a *syntax* and then, for this syntax, definitions of *security*. In the first modern treatment [10], the syntax asked the encryption algorithm to be randomized or stateful. Security for these syntaxes evolved from asking for various forms of privacy [10] to asking for both privacy and authenticity [13, 11, 32], inaugurating authenticated encryption (AE). The idea that encryption be a deterministic algorithm taking as additional input a non-repeating quantity called a nonce seems to originate in [50] and reached its current form with Rogaway [46, 48].

NBE1 AND AE1-SECURITY. We refer to the syntax of this current form of nonce-based symmetric encryption [46, 48] as NBE1. An NBE1 scheme $\mathsf{SE1}$ specifies a *deterministic* encryption algorithm $\mathsf{SE1.Enc}$ that takes the key K, a nonce N, message M and a header (also called associated data) H to return what we call a core ciphertext C_1. Deterministic decryption algorithm $\mathsf{SE1.Dec}$ takes K, N, C_1, H to return either a message or $\perp$.

Security asks for privacy of M and integrity of both M and H *as long as nonces are unique*, meaning not re-used. Rogaway's formalization [46] asks that an adversary given oracles for encryption (taking nonce, message and header) and decryption (taking nonce, core ciphertext and header) be unable to distinguish between the case where they perform their prescribed tasks under a hidden key, and the case where the former returns random strings and the latter returns $\perp$, as long as the adversary does not repeat a nonce across its encryption queries. We will refer to this as basic AE1-security.

NBE1 providing basic AE1-security has been the goal of recent schemes, standards and proposed standards, as witnessed by GCM [40, 21] (used in TLS), OCB [50, 47, 34], CAESAR candidates [16] and RFC 5116 [39]. The security of NBE1, which we revisit, is thus of some applied interest.

THE GAP. Our concern is a gap between theory and usage that can result in privacy vulnerabilities in the latter. Recall that the decryption algorithm $\mathsf{SE1.Dec}$, to be run by the receiver, takes as input not just the key K, core ciphertext C_1 and header H, but *also the nonce* N. The theory says that how the receiver gets the nonce is "outside of the model" [46] or that it is assumed to be communicated "out-of-band" [48]. Usage cannot so dismiss it, and must find a way to convey the nonce to the receiver. The prevailing understanding, reflected in the following quote from RBBK [50], is that this is a simple matter— if the receiver does not already have the nonce N, just send it in the clear along with the core ciphertext C_1:

> The nonce N is needed both to encrypt and to decrypt. Typically it would be communicated, in the clear, along with the (core) ciphertext.

RFC 5116 is a draft standard for an interface for authenticated encryption [39]. It also considers it fine to send the nonce in the clear:

> ... there is no need to coordinate the details of the nonce format between the encrypter and the decrypter, as long *the entire nonce is sent* or stored with the ciphertext and is thus available to the decrypter ... the nonce MAY be stored *or transported* with the ciphertext ...

To repeat and summarize, the literature and proposed standards suggest transmitting what we call the "full" ciphertext, consisting of the nonce and the core ciphertext. Yet, as we now explain, this can be wrong.

NONCES CAN COMPROMISE PRIVACY. We point out that communicating a nonce in the clear with the ciphertext can damage, or even destroy, message privacy. One simple example is a nonce $N = F(M)$ that is a hash —under some public, collision-resistant hash function F— of a low-entropy message M, meaning one, like a password, which the attacker knows is likely to fall in some small set or dictionary D. Given a (full) ciphertext $C_2 = (N, C_1)$ consisting of the core ciphertext $C_1 = \mathsf{SE1.Enc}(K, N, M, H)$ together with the nonce $N = F(M)$, the attacker can recover M via "For $M' \in D$ do: If $F(M') = N$ then return M'." To take a more extreme case, consider that the nonce is some part of the message, or even the entire message, in which case the full ciphertext clearly reveals information about the message.

The concern that (adversary-visible) nonces compromise privacy, once identified, goes much further. Nonces are effectively meta-data. Even recommended and innocuous-seeming choices like counters, device identities, disk-sector numbers or packet headers reveal information about the system and identity of the sender. For example, the claim that basic-AE1-secure NBE1 provides anonymity —according to [49, Slide 19/40], this is a dividend of the requirement that core ciphertexts be indistinguishable from random strings— is moot when the nonce includes sender identity. Yet the latter is not only possible but explicitly recommended in RFC 5116 [39], which says: "When there are multiple devices performing encryption ... use a nonce format that contains a field that is distinct for each one of the devices." As another concrete example, counters are *not* a good choice of nonce from a user privacy perspective, as indicated in the ECRYPT-CSA *Challenges in Authenticated Encryption* report [5].

The above issues apply to all NBE1 schemes and do not contradict their (often, proven) AE1-security. They are not excluded by the unique nonce requirement or by asking for misuse resistance [51], arising in particular for the encryption of a single message with a single corresponding nonce.

A natural critique is that the privacy losses we have illustrated occur only for "pathological" choices of nonces, and choices made in practice, such as random numbers or counters, are "fine." This fails, first, to recognize the definitional gap that allows the "pathological" choices. With regard to usage, part of the selling point of NBE1 was exactly that *any* (non-repeating, unique) nonce is fine, and neither existing formalisms [46] nor existing standards [39] preclude nonce choices of the "pathological" type. Also, application designers and users cannot, and should not, carry the burden of deciding which nonces are "pathological" and which are "fine," a decision that may not be easy. (And as discussed above, for example, counters may *not* be fine.) Finally, Section 8 indicates that poor choices can in fact arise in practice.

Our perspective is that the above issues reflect a gap between the NBE1 formalism and the privacy provided by NBE1 in usage. Having pointed out this gap, we will also bridge it.

CONTRIBUTIONS IN BRIEF. The first contribution of this paper is to suggest that the way NBE1 treats nonces can result (as explained above) in compromise of privacy of messages or users. The second contribution is to address these concerns. We give a modified syntax for nonce-based encryption, called NBE2, in which decryption does not get the nonce, a corresponding framework of security definitions called AE2 that guarantee nonce privacy in addition to authenticity and message privacy, and simple ways to turn NBE1 AE1-secure schemes into NBE2 AE2-secure schemes.

AE2-secure NBE2 obviates application designers and users from the need to worry about privacy implications of their nonce choices, simplifying design and usage. With AE2-secure NBE2, one can use any nonce, even a message-dependent one such as a hash of the message, without compromising privacy of the message. And the nonces themselves are hidden just as well as messages, so user-identifying information in nonces doesn't actually identify users.

OUR NBE2 SYNTAX. In an NBE2 scheme $\mathsf{SE2}$, the inputs to the deterministic encryption algorithm $\mathsf{SE2.Enc}$ continue to be key K, nonce N, message M and header H, the output C_2 now called a ciphertext rather than a core ciphertext. The deterministic decryption algorithm $\mathsf{SE2.Dec}$ *no longer gets a nonce*, taking just key K, ciphertext C_2 and header H to return either a message M or $\perp$.

Just as an interface, NBE2 already benefits application designers and users, absolving them of the burden they had, under NBE1, of figuring out and architecting a way to communicate the nonce from sender to receiver. The NBE2 receiver, in fact, is nonce-oblivious, not needing to care, or even know, that something called a nonce was used by the sender. By reducing choice (how to communicate the nonce), NBE2 reduces error and misuse.

We associate to a given NBE1 scheme $\mathsf{SE1}$ the NBE2 scheme $\mathsf{SE2} = \mathbf{TN}[\mathsf{SE1}]$ that sets the ciphertext to the nonce plus the core ciphertext: $\mathsf{SE2.Enc}(K, N, M, H) = (N, \mathsf{SE1.Enc}(K, N, M, H))$ and $\mathsf{SE2.Dec}(K, (N, C_1), H) = \mathsf{SE1.Dec}(K, N, C_1, H)$. We refer to $\mathbf{TN}$ as the Transmit Nonce transform. This is worth defining because it will allow us, in Section 4, to formalize the above-discussed usage weaknesses in NBE1, but $\mathsf{SE2} = \mathbf{TN}[\mathsf{SE1}]$ is certainly not nonce hiding and will fail to meet the definitions we discuss next.

OUR AE2-SECURITY FRAMEWORK. Our AE2 game gives the adversary an encryption oracle ENC (taking nonce N, message M and header H to return a ciphertext C_2) and decryption oracle DEC (as per the NBE2 syntax, taking ciphertext C_2 and header H but no nonce, to return either a message M or $\perp$). When the challenge bit is $b = 1$, these oracles reply as per the encryption algorithm $\mathsf{SE2.Enc}$ and decryption algorithm $\mathsf{SE2.Dec}$ of the scheme, respectively, using a key chosen by the game. When the challenge bit is $b = 0$, oracle ENC returns a ciphertext that is drawn at random from a space $\mathsf{SE2.CS}(|N|, |M|, |H|)$ that is prescribed by the scheme $\mathsf{SE2}$ and that depends only on the lengths of the nonce, message and header, which guarantees privacy of both the nonce and message. (This space may be, but unlike for AE1 need not be, the set of all strings of some length, because NBE2 ciphertexts, unlike NBE1 core ciphertexts, may have some structure.) In the $b = 0$ case, decryption oracle DEC returns $\perp$ on any non-trivial query. The adversary eventually outputs a guess b' as to the value of b, and its advantage is $2\Pr[b = b'] - 1$.

We say that $\mathsf{SE2}$ is AE2[$\mathcal{A}$]-secure if practical adversaries in the class $\mathcal{A}$ have low advantage. Let $\mathcal{A}^{\text{ae2}}_{\text{u-n}}$ be the class of unique-nonce adversaries, meaning ones that do not reuse a nonce across their ENC queries. We refer to AE2[$\mathcal{A}^{\text{ae2}}_{\text{u-n}}$]-security as basic AE2-security. As the nonce-hiding analogue of basic AE1-security, it will be our first and foremost target.

Before moving to schemes, we make two remarks. First that above, for simplicity, we described our definitions in the single-user setting, but the definitions and results in the body of the paper are in the multi-user setting. Second, the framework of a single game with different notions captured via different adversary classes allows us to unify, and compactly present, many variant definitions, including basic, advanced (misuse resistance), privacy-only and random-nonce security, and in Section 3 we give such a framework not just for AE2 but also for AE1.

OUR TRANSFORMS. In the presence of a portfolio of efficient AE1-secure NBE1 schemes supported by proofs of security with good concrete bounds [50, 40, 16, 34, 30, 54, 44, 25, 24, 17, 29], designing AE2-secure NBE2 schemes from scratch seems a step backwards. Instead we give simple, cheap ways to transform AE1-secure NBE1 schemes into AE2-secure NBE2 schemes, obtaining a corresponding portfolio of AE2-secure NBE2 schemes and also allowing implementors to more easily upgrade deployed AE1-secure NBE1 to AE2-secure NBE2.

Since NBE2 schemes effectively take care of nonce communication, we expect ciphertext length to grow by at least $\mathsf{SE1.nl}$, the nonce length of the base NBE1 scheme. The *ciphertext overhead* is defined as the difference between the ciphertext length and the sum of plaintext length and $\mathsf{SE1.nl}$.

NBE2 scheme	AE2-security provided	
	Basic	Advanced
HN1[SE1, F]	Yes	Yes
HN2[SE1, ℓ, E, Spl]	Yes	Yes if $\ell \geq 128$
HN3[SE1, F]	Yes	No
HN4[SE1, ℓ, F]		Yes
HN5[TE, ℓ, ℓ_t]		Yes

Figure 1: Security attributes of the NBE2 schemes defined by our Hide-Nonce (HN) transforms. In the table SE1 denotes an NBE1 scheme, F a PRF, E a block cipher, and TE a variable-length tweakable block cipher. Spl is a splitting function, and ℓ, ℓ_t are non-negative integer parameters. A blank entry in the Basic column means the transform is not for that purpose. Note that **HN1**'s advanced security only holds when ciphertexts have sufficiently large (e.g. 128 bits) minimum length, and **HN2**'s depends on the length of the stolen ciphertext.

All our transforms have zero ciphertext overhead. One challenge in achieving this is that nonce lengths like SE1.nl = 96 are widely-used but short of the block length 128 of many blockciphers, precluding inclusion of an extra blockcipher output in the ciphertext. With regard to computational overhead, the challenge is that it should be constant, meaning independent of the lengths of the message and header for encryption, and of the ciphertext and header for decryption. *All our transforms have constant computational overhead.* Note that all overhead is in comparison to transmitting the nonce in the clear (i.e. the **TN** transform).

The following discussion first considers achieving basic security and then advanced security. Security attributes of our corresponding "Hide-Nonce (HN)" transforms are summarized in Figure 1.

Basic HN transforms. We prove that all the following transforms turn a basic-AE1-secure NBE1 scheme SE1 into a basic-AE2-secure NBE2 scheme SE2. (Recall basic means nonces are unique, never reused across encryption queries.) Pseudocode and pictures for the transforms are in Figure 4.

Having first produced a core ciphertext C_1 under SE1, the idea of scheme SE2 = **HN1**[SE1, F] is to use C_1 itself as a nonce to encrypt the actual nonce in counter mode under PRF F. A drawback is that this requires the minimal core-ciphertext length SE1.mccl to be non-trivial, like at least 128, which is not true for all SE1. Scheme SE2 = **HN2**[SE1, ℓ, E, Spl] turns to the perhaps more obvious idea of enciphering the nonce with a PRF-secure blockcipher E. The difficulty is the typicality of 96-bit nonces and 128-bit blockciphers, under which naïve enciphering would add a 32-bit ciphertext overhead, which we resolve by ciphertext stealing, ℓ representing the number of stolen bits (32 in our example) and Spl an ability to choose how the splitting is done. Scheme SE2 = **HN3**[SE1, F] uses the result of PRF F on the actual nonce as a derived nonce under which to run SE1. This is similar to SIV [51, 44]; the difference is to achieve AE2 rather than AE1 and to apply the PRF only to the nonce (rather than nonce, message and header) to have constant computational overhead.

Advanced HN transforms. Unique nonces are easier to mandate in theory than assure in practice, where nonces may repeat due to errors, system resets, or replication. In that case (returning here to NBE1), not only does basic AE1-security give no security guarantees, but also damaging attacks are possible for schemes including CCM and GCM [31, 53]. Rogaway and Shrimpton's misuse resistant NBE1, which we refer to as advanced-AE1-secure NBE1, minimizes the damage

from reused nonces, retaining AE1-security as long as no nonce, message, header triple is re-encrypted [51]. This still being for the NBE1 syntax, however, the concerns with adversary-visible nonces compromising message and user privacy are unchanged. We seek the NBE2 analogue, correspondingly defining and achieving advanced-AE2-secure NBE2 to provide protection against reused nonces while also hiding them.

With our framework, the definition is easy, calling for no new games; the goal is simply AE2[$\mathcal{A}^{\text{ae2}}_{\text{u-nmh}}$]-security where $\mathcal{A}^{\text{ae2}}_{\text{u-nmh}}$ is the class of unique-nonce, message, header adversaries, meaning ones that do not repeat a query to their ENC oracle. The presence of well-analyzed advanced-AE1-secure NBE1 schemes [51, 27, 25, 24, 17] again motivates transforms rather than from-scratch designs.

We start by revisiting our basic-security preserving transforms, asking whether they also preserve advanced security, meaning, if the starting NBE1 scheme is advanced-AE1-secure, is the transformed NBE2 scheme advanced-AE2-secure? We show that for **HN1**, the answer is YES. We then show that it is YES also for **HN2** as long as the amount ℓ of stolen ciphertext is large enough. (In practical terms, at least 128.) For **HN3**, the answer is NO.

That **HN1** and **HN2** have these properties is good, but we would like to do better. (Limitations of the above are that **HN1** puts a lower bound on $\mathsf{SE1.mccl}$ that is not always met, and setting $\ell = 128$ in **HN2** with typical 96-bit nonces will call for a 224-bit blockcipher.) We offer **HN4** and **HN5**, showing they provide advanced AE2-security. Pseudocode and pictures are in Figure 5.

Scheme $\mathsf{SE2} = \mathbf{HN4}[\mathsf{SE1}, \ell, \mathsf{F}]$ uses the result of PRF F on the actual nonce, message and header as a derived nonce for $\mathsf{SE1}$. The difference with SIV [51, 44] is that what is encrypted under $\mathsf{SE1}$ includes the actual nonce in order to hide it. The computational overhead stays constant because $\mathsf{SE1}$ need provide only privacy, which it can do in one pass. Scheme $\mathsf{SE2} = \mathbf{HN5}[\mathsf{TE}, \ell, \ell_t]$ is different, using the encode-then-encipher paradigm [13] to set the ciphertext to an enciphering, under an arbitrary-input-length, tweakable cipher TE, of the nonce, message and ℓ_t-bits of redundancy, with the header as tweak. Instantiating TE via the very fast AEZ tweakable block cipher [27] yields correspondingly fast, advanced-AE2-secure NBE2.

DEDICATED TRANSFORMS. While our generic transforms are already able, with low overhead, to immunize GCM [40, 21] —by this we mean turn this basic-AE1-secure NBE1 scheme into a basic-AE2-secure NBE2 scheme— we ask if a dedicated transform —ones that exploit the structure of GCM— can do even better. The goal is not just even lower overhead, but minimization of software changes. We show that simply pre-pending a block of 0s to the message and then GCM-encrypting provides basic-AE2-security, so neither the key nor the encryption software need be changed. Decryption software however does need a change, and, unlike with our generic transforms, we incur 32 bits of ciphertext overhead.

RELATED WORK. As a technical step in achieving security against release of unverified plaintext (RUP), Ashur, Dunkelman and Luykx (ADL) [4] use a syntax identical to NBE2, and their techniques bear some similarities with ours that we discuss further in Section 7.

The CAESAR competition's call for authenticated encryption schemes describes a syntax where encryption receives, in place of a nonce, a public message number (PMN) and a secret message number (SMN), decryption taking only the former [18]. The formalization of Namprempre, Rogaway and Shrimpton (NRS) [43] dubs this "AE5." In this light, an NBE1 scheme is a AE5 scheme without a SMN and an NBE2 scheme is an AE5 scheme without a PMN.

POSSIBLE FUTURE WORK. The concerns we have raised with regard to a gap between theory and usage, and privacy vulnerabilities created by adversary-visible nonces in the latter, arise fundamentally from the choice of *syntax* represented by NBE1, and as such hold also in other contexts where

an NBE1-style syntax is used. This includes AE secure under release of unverified plaintext [3], robust AE [27], online AE [22, 28], committing AE [23, 20], indifferentiable AE [6], leakage-resilient AE [7] and MiniAE [42]. A direction for future work is to treat these with an NBE2-style syntax (decryption does not get the nonce) to provide nonce hiding.

While our transforms can be applied to promote the advanced-AE1-secure AES-GCM-SIV NBE1 scheme [24] to an advanced-AE2-secure NBE2 scheme, the bounds we get are inferior to those of [17]. Bridging this gap to get advanced-AE2-secure NBE2 with security bounds like [17] is a direction for future work. Similarly, while we have many ways to turn GCM into a basic-AE2-secure NBE2 scheme with little overhead, one that matches the bounds of [38, 29] would be desirable.

2 Preliminaries

Notation and terminology. By ε we denote the empty string. By $|Z|$ we denote the length of a string Z. If Z is a string then $Z[i..j]$ is bits i through j of Z if $1 \le i \le j \le |Z|$, and otherwise is ε. By $x\|y$ we denote the concatenation of strings x, y. If x, y are equal-length strings then $x \oplus y$ denotes their bitwise xor. If i is an integer in the range $0 \le i < 2^n$ then $\langle i \rangle_n \in \{0,1\}^n$ denotes the representation of i as a string of (exactly) n bits. (For example, $\langle 3 \rangle_4 = 0011$.) If S is a finite set, then $|S|$ denotes it size. We say that a set S is *length-closed* if, for any $x \in S$ it is the case that $\{0,1\}^{|x|} \subseteq S$. (This will be a requirement for message, header and nonce spaces.) If D, R are sets and $f : D \to R$ is a function then its image is $\mathrm{Im}(f) = \{\, f(x) \,:\, x \in D \,\} \subseteq R$.

If X is a finite set, we let $x \xleftarrow{\$} X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. If A is an algorithm, we let $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots; \omega)$ denote running A on inputs $x_1, \ldots$ and coins ω, with oracle access to $\mathrm{O}_1, \ldots$, and assigning the output to y. By $y \xleftarrow{\$} A^{\mathrm{O}_1,\cdots}(x_1, \ldots)$ we denote picking ω at random and letting $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots; \omega)$. We let $[A^{\mathrm{O}_1,\cdots}(x_1, \ldots)]$ denote the set of all possible outputs of A when run on inputs $x_1, \ldots$ and with oracle access to $\mathrm{O}_1, \ldots$. An adversary is an algorithm. Running time is worst case, which for an algorithm with access to oracles means across all possible replies from the oracles. We use $\perp$ (bot) as a special symbol to denote rejection, and it is assumed to not be in $\{0,1\}^*$.

Games. We use the code-based game-playing framework of BR [14]. A game G (see Fig. 2 for an example) starts with an optional Initialize procedure, followed by a non-negative number of additional procedures called oracles, and ends with a Finalize procedure. If Finalize is omitted, it is understood to be the trivial procedure that simply returns (outputs) its input. Execution of adversary A with game G consists of running A with oracle access to the game procedures, with the restrictions that A's first call must be to Initialize (if present), its last call must be to Finalize, and it can call these procedures at most once. The output of the execution is the output of Finalize. By $\Pr[\mathrm{G}(A)]$ we denote the probability that the execution of game G with adversary A results in this output being the boolean true. In games, integer variables, set variables boolean variables and string variables are assumed initialized, respectively, to 0, the empty set $\emptyset$, the boolean false and $\perp$.

Multi-user security. There is growing recognition that security should be considered in the multi-user (mu) setting [8] rather than the traditional single-user (su) one. Our main definitions are in the mu setting. The games provide the adversary a New oracle, calling which results in a new user being initialized, with a fresh key. Other oracles are enhanced (relative to the su setting) to take an additional argument i indicating the user (key). We assume that adversaries do not

Game $\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}$	Game $\mathbf{G}^{\mathrm{stprp}}_{\mathsf{TE}}$
procedure INITIALIZE $b \leftarrow\!\!\$\ \{0,1\}$	procedure INITIALIZE $b \leftarrow\!\!\$\ \{0,1\}$
procedure NEW $v \leftarrow v+1$ $K_v \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$	procedure NEW $v \leftarrow v+1$; $K_v \leftarrow\!\!\$\ \{0,1\}^{\mathsf{TE.kl}}$
procedure FN(i, X) If $(\mathbf{Y}[i,X] = \bot)$ then $\quad Y_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.ol}}$ $\quad Y_1 \leftarrow \mathsf{F.Ev}(K_i, X)$ $\quad \mathbf{Y}[i,X] \leftarrow Y_b$ Return $\mathbf{Y}[i,X]$	procedure FN(i, T, X) If $(\mathbf{Y}[i,T,X] = \bot)$ then $\quad Y_0 \leftarrow\!\!\$\ \{0,1\}^{\lvert X\rvert} \setminus \mathcal{Y}_{i,T}$; $Y_1 \leftarrow \mathsf{TE.Ev}(K_i, T, X)$ $\quad \mathbf{Y}[i,T,X] \leftarrow Y_b$; $\mathbf{X}[i,T,Y_b] \leftarrow X$ $\quad \mathcal{X}_{i,T} \leftarrow \mathcal{X}_{i,T} \cup \{X\}$; $\mathcal{Y}_{i,T} \leftarrow \mathcal{Y}_{i,T} \cup \{Y_b\}$ Return $\mathbf{Y}[i,T,X]$
procedure FINALIZE(b') Return $(b = b')$	procedure FNINV(i, T, Y) If $(\mathbf{X}[i,T,Y] = \bot)$ then $\quad X_0 \leftarrow\!\!\$\ \{0,1\}^{\lvert Y\rvert} \setminus \mathcal{X}_{i,T}$; $X_1 \leftarrow \mathsf{TE.In}(K_i, T, Y)$ $\quad \mathbf{X}[i,T,Y] \leftarrow X_b$; $\mathbf{Y}[i,T,X_b] \leftarrow Y$ $\quad \mathcal{X}_{i,T} \leftarrow \mathcal{X}_{i,T} \cup \{X_b\}$; $\mathcal{Y}_{i,T} \leftarrow \mathcal{Y}_{i,T} \cup \{Y\}$ Return $\mathbf{X}[i,T,Y]$ procedure FINALIZE(b') Return $(b = b')$

Figure 2: Left: Games defining multi-user PRF security for function family F. Right: Game defining multi-user stPRP security for tweakable cipher TE.

---

make oracle queries to users (also called sessions) they have not initialized.

FUNCTION FAMILIES. A function family F specifies a deterministic evaluation algorithm $\mathsf{F.Ev} : \{0,1\}^{\mathsf{F.kl}} \times \mathsf{F.D} \to \{0,1\}^{\mathsf{F.ol}}$ that takes a key K and input x to return output $\mathsf{F.Ev}(K,x)$, where $\mathsf{F.kl}$ is the key length, $\mathsf{F.D}$ is the domain and $\mathsf{F.ol}$ is the output length. We say that F is invertible if there is an inversion algorithm $\mathsf{F.In} : \{0,1\}^{\mathsf{F.kl}} \times \{0,1\}^{\mathsf{F.ol}} \to \mathsf{F.D} \cup \{\bot\}$ such that for all $K \in \{0,1\}^{\mathsf{F.kl}}$ we have (1) $\mathsf{F.In}(K, \mathsf{F.Ev}(K,x)) = x$ for all $x \in \mathsf{F.D}$, and (2) $\mathsf{F.In}(K,y) = \bot$ for all $y \notin \mathrm{Im}(\mathsf{F.Ev}(K,\cdot))$. We say that F is a permutation family if it is invertible and $\mathsf{F.D} = \{0,1\}^{\mathsf{F.ol}}$. In that case, we also refer to F as a block cipher and to $\mathsf{F.ol}$ as the block length of F, which we may denote $\mathsf{F.bl}$.

PRF SECURITY. We define multi-user PRF security [9] for a function family F and adversary A via the game $\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}(A)$ in Fig. 2. Here b is the challenge bit and $\mathbf{Y}[\cdot,\cdot]$ is a table, all of whose entries are assumed to initially be $\bot$. It is required that any FN(i,X) query of A satisfies $i \le v$ and $X \in \mathsf{F.D}$. The multi-user PRF advantage of adversary A is $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(A) = 2\Pr[\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}(A)] - 1$.

TWEAKABLE CIPHERS. A tweakable cipher TE [36, 27] specifies a deterministic evaluation algorithm $\mathsf{TE.Ev} : \{0,1\}^{\mathsf{TE.kl}} \times \mathsf{TE.TS} \times \{0,1\}^* \to \{0,1\}^*$ and a deterministic inversion algorithm $\mathsf{TE.In} : \{0,1\}^{\mathsf{TE.kl}} \times \mathsf{TE.TS} \times \{0,1\}^* \to \{0,1\}^*$. Here, $\mathsf{TE.kl}$ is the key length and $\mathsf{TE.TS}$ is the tweak space. We require that for all $K \in \{0,1\}^{\mathsf{TE.kl}}$, $T \in \mathsf{TE.TS}$ and $X \in \{0,1\}^*$ we have $|\mathsf{TE.Ev}(K,T,X)| = |X|$ and $\mathsf{TE.In}(K,T,\mathsf{TE.Ev}(K,T,X)) = X$.

STPRP SECURITY. We define multi-user stPRP (strong tweakable PRP) security [37] for tweakable cipher TE and adversary A via the game $\mathbf{G}^{\mathrm{stprp}}_{\mathsf{TE}}(A)$ in Fig. 2. In the game, b is the challenge bit and

$\mathbf{X}[\cdot,\cdot,\cdot]$, $\mathbf{Y}[\cdot,\cdot,\cdot]$ are tables whose entries are assumed initialized to $\perp$. In this game, the adversary has access to an evaluation oracle FN and an inversion oracle FNINV. When $b = 0$, they sample without replacement (within each session) from the set of strings of the same length as the input. If $b = 1$ they evaluate $\mathsf{TE.Ev}$ and $\mathsf{TE.In}$ under game-chosen keys. It is required that any $\text{FN}(i, T, X)$ or $\text{FNINV}(i, T, Y)$ query of A satisfies $i \le v$, $T \in \mathsf{TE.TS}$ and $X, Y \in \{0,1\}^*$. The multi-user stPRP advantage of adversary A is $\mathbf{Adv}^{\text{stprp}}_{\mathsf{TE}}(A) = 2\Pr[\mathbf{G}^{\text{stprp}}_{\mathsf{TE}}(A)] - 1$.

3 Two frameworks for nonce-based encryption

We give definitions for both AE1-secure NBE1—current nonce-based encryption [50, 46, 48]— and AE2-secure NBE2—our new nonce-based encryption. In each case there is a single security game, different variant definitions then being captured by different adversary classes. This allows a unified and compact treatment.

NBE1. An NBE1 scheme $\mathsf{SE1}$ specifies several algorithms and related quantities, as follows. Deterministic encryption algorithm $\mathsf{SE1.Enc} : \mathsf{SE1.KS} \times \mathsf{SE1.NS} \times \mathsf{SE1.MS} \times \mathsf{SE1.HS} \to \{0,1\}^*$ takes a key K in the (finite) key-space $\mathsf{SE1.KS}$, a nonce N in the nonce-space $\mathsf{SE1.NS}$, a message M in the message space $\mathsf{SE1.MS}$ and a header H in the header space $\mathsf{SE1.HS}$ to return what we call a core ciphertext C_1. This is a string of length $\mathsf{SE1.ccl}(|N|, |M|, |H|)$, where $\mathsf{SE1.ccl}$ is the core-ciphertext length function. $\mathsf{SE1}$ also specifies a deterministic decryption algorithm $\mathsf{SE1.Dec} : \mathsf{SE1.KS} \times \mathsf{SE1.NS} \times \{0,1\}^* \times \mathsf{SE1.HS} \to \mathsf{SE1.MS} \cup \{\perp\}$ that takes key K, nonce N, core ciphertext C_1 and header H to return an output that is either a message $M \in \mathsf{SE1.MS}$, or $\perp$. It is required that $\mathsf{SE1.NS}, \mathsf{SE1.MS}, \mathsf{SE1.HS}$ are length-closed sets as defined in Section 2. Most often nonces are of a fixed length denoted $\mathsf{SE1.nl}$, meaning $\mathsf{SE1.NS} = \{0,1\}^{\mathsf{SE1.nl}}$. Decryption correctness requires that $\mathsf{SE1.Dec}(K, N, \mathsf{SE1.Enc}(K, N, M, H), H) = M$ for all $K \in \mathsf{SE1.KS}$, $N \in \mathsf{SE1.NS}$, $M \in \mathsf{SE1.MS}$ and $H \in \mathsf{SE1.HS}$.

AE1 GAME AND ADVANTAGE. Let $\mathsf{SE1}$ be an NBE1 scheme and A an adversary. We associate to them the game $\mathbf{G}^{\text{ae1}}_{\mathsf{SE1}}(A)$ shown on the top left of Fig. 3. (We use the name "AE1" to associate the game with the NBE1 syntax). The AE1-advantage of adversary A is $\mathbf{Adv}^{\text{ae1}}_{\mathsf{SE1}}(A) = 2\Pr[\mathbf{G}^{\text{ae1}}_{\mathsf{SE1}}(A)] - 1$. The game is in the multi-user setting, oracle NEW allowing the adversary to initialize a new user with a fresh key. It is required that any $\text{ENC}(i, N, M, H)$ query of A satisfy $1 \le i \le v$, $N \in \mathsf{SE1.NS}$, $M \in \mathsf{SE1.MS}$ and $H \in \mathsf{SE1.HS}$. When the challenge bit b is 1, the encryption oracle will return a core ciphertext as stipulated by $\mathsf{SE1.Enc}$, using the key for the indicated user i. In the $b = 0$ case, ENC will return a random string of length $\mathsf{SE1.ccl}(|N|, |M|, |H|)$. The array $\mathbf{M}$ is assumed to initially be $\perp$ everywhere, and holds core ciphertexts returned by ENC. It is required that any $\text{DEC}(i, N, C_1, H)$ query of A satisfy $1 \le i \le v$, $N \in \mathsf{SE1.NS}$ and $H \in \mathsf{SE1.HS}$. When the challenge bit b is 1, the decryption oracle will perform decryption as stipulated by $\mathsf{SE1.Dec}$, using the key for the indicated user i. In the $b = 0$ case, DEC will return $\perp$ on any core ciphertext not previously returned by the encryption oracle.

AE1 SECURITY METRICS. AE1-security is clearly not achievable without restrictions on the adversary. For example, if A repeats a query i, N, M, H to ENC, then, when $b = 1$ it gets back the same reply both times, while if $b = 0$ it likely does not, allowing it to determine b with high probability. We define different classes of adversaries, summarized by the table at the bottom of Figure 3, with the superscript "x" here being ae1. We say that NBE1 scheme $\mathsf{SE1}$ is AE1[$\mathcal{A}$]-secure if adversaries in $\mathcal{A}$ have low AE1-advantage. The definition is in the multi-user setting, but restricting attention to adversaries in the class $\mathcal{A}^{\text{ae1}}_1$ allows us to recover the single-user setting. Different security no-

Game $\mathbf{G}^{\text{ae1}}_{\mathsf{SE1}}$	Game $\mathbf{G}^{\text{ae2}}_{\mathsf{SE2}}$
procedure INITIALIZE $b \leftarrow\!\!\$\ \{0,1\}$	procedure INITIALIZE $b \leftarrow\!\!\$\ \{0,1\}$
procedure NEW $v \leftarrow v+1$; $K_v \leftarrow\!\!\$\ \mathsf{SE1.KS}$	procedure NEW $v \leftarrow v+1$; $K_v \leftarrow\!\!\$\ \mathsf{SE2.KS}$

procedure ENC(i,N,M,H)
If $(b=1)$ then
  $C_1 \leftarrow \mathsf{SE1.Enc}(K_i,N,M,H)$
  Else $C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.ccl}(\lvert N\rvert,\lvert M\rvert,\lvert H\rvert)}$
$\mathbf{M}[i,N,C_1,H] \leftarrow M$; Return C_1
	procedure ENC(i,N,M,H)
	If $(b=1)$ then
	  $C_2 \leftarrow \mathsf{SE2.Enc}(K_i,N,M,H)$
	  Else $C_2 \leftarrow\!\!\$\ \mathsf{SE2.CS}(\lvert N\rvert,\lvert M\rvert,\lvert H\rvert)$
	$\mathbf{M}[i,C_2,H] \leftarrow M$; Return C_2

procedure DEC(i,N,C_1,H)
If $(\mathbf{M}[i,N,C_1,H] \neq \perp)$ then
  Return $\mathbf{M}[i,N,C_1,H]$
If $(b=0)$ then $M \leftarrow \perp$
  Else $M \leftarrow \mathsf{SE1.Dec}(K_i,N,C_1,H)$
Return M
	procedure DEC(i,C_2,H)
	If $(\mathbf{M}[i,C_2,H] \neq \perp)$ then
	  Return $\mathbf{M}[i,C_2,H]$
	If $(b=0)$ then $M \leftarrow \perp$
	  Else $M \leftarrow \mathsf{SE2.Dec}(K_i,C_2,H)$
	Return M

procedure FINALIZE(b') Return $(b=b')$	procedure FINALIZE(b') Return $(b=b')$

$\mathcal{A}^{\text{x}}_{\text{u-n}}$	Unique nonce adversaries — $A \in \mathcal{A}^{\text{x}}_{\text{u-n}}$ does not repeat a user-nonce pair i,N across its ENC queries
$\mathcal{A}^{\text{x}}_{\text{u-nmh}}$	Unique nonce-message-header adversaries — $A \in \mathcal{A}^{\text{x}}_{\text{u-nmh}}$ does not repeat a query to ENC
$\mathcal{A}^{\text{x}}_{\text{priv}}$	Privacy adversaries — $A \in \mathcal{A}^{\text{x}}_{\text{priv}}$ makes no DEC queries
$\mathcal{A}^{\text{x}}_{1}$	Single-user adversaries — $A \in \mathcal{A}^{\text{x}}_{1}$ makes only one NEW query
$\mathcal{A}^{\text{x}}_{\text{r-n}}$	Random-nonce adversaries — The nonces in the ENC queries of $A \in \mathcal{A}^{\text{x}}_{\text{r-n}}$ are distributed uniformly and independently at random

Figure 3: **Top Left:** Game defining AE1-security of NBE1 scheme SE1. **Top Right:** Game defining AE2-security of NBE2 scheme SE2. **Bottom:** Some classes of adversaries, leading to different security notions, where $\text{x} \in \{\text{ae1}, \text{ae2}\}$.

---

tions in the literature are then captured as AE1[$\mathcal{A}$]-security for different classes of adversaries $\mathcal{A}$, as we illustrate below:

- $\mathcal{A}^{\text{ae1}}_{\text{u-n}}$ is the class of adversaries whose ENC queries never repeat a user-nonce pair. AE1[$\mathcal{A}^{\text{ae1}}_{\text{u-n}} \cap \mathcal{A}^{\text{ae1}}_{1}$]-security is thus AEAD as defined in [46, 48].
- AE1[$\mathcal{A}^{\text{ae1}}_{\text{u-n}}$]-security is the extension of this to the multi-user setting as defined in [15], which we have referred to as basic AE1-security in Section 1.
- Adversaries in $\mathcal{A}^{\text{ae1}}_{\text{u-nmh}} \supseteq \mathcal{A}^{\text{ae1}}_{\text{u-n}}$ are allowed to re-use a user-nonce pair across ENC queries as long as they never repeat an entire query. AE1[$\mathcal{A}^{\text{ae1}}_{\text{u-nmh}} \cap \mathcal{A}^{\text{ae1}}_{1}$]-security is misuse resistant AE [51].
- AE1[$\mathcal{A}^{\text{ae1}}_{\text{u-nmh}}$]-security is the extension of this to the multi-user setting [17], which we have

referred to as advanced-AE1-security in Section 1.

- Adversaries in $\mathcal{A}^{\text{ae1}}_{\text{r-n}}$ pick the nonces in their ENC queries uniformly and independently at random from $\mathsf{SE1.NS}$. (While the intent here is likely understandable, what precisely it means for an adversary to be in this class does actually need a careful definition, which is given in Appendix A.) No restriction is placed on how the adversary picks nonces in DEC queries. AE1$[\mathcal{A}^{\text{ae1}}_{\text{r-n}} \cap \mathcal{A}^{\text{ae1}}_{1}]$-security is thus classical randomized AE [11] for schemes which make encryption randomness public, which is the norm.
- Sometimes, in the unique-nonce setting, we consider schemes that provide only privacy, not authenticity, and, rather than giving a separate game, can capture this as AE1$[\mathcal{A}^{\text{ae1}}_{\text{priv}} \cap \mathcal{A}^{\text{ae1}}_{\text{u-n}}]$-security. AE1$[\mathcal{A}^{\text{ae1}}_{\text{priv}} \cap \mathcal{A}^{\text{ae1}}_{\text{u-n}} \cap \mathcal{A}^{\text{ae1}}_{1}]$-security is $\mathsf{IND\$\text{-}CPA}$ security, as defined in [46].

Further adversary classes can be defined to capture limited nonce reuse [17] or other resource restrictions.

The following says that AE1$[\mathcal{A}^{\text{ae1}}_{\text{u-n}}]$-security implies AE1$[\mathcal{A}^{\text{ae1}}_{\text{r-n}}]$-security with a degradation in advantage corresponding to the probability that a nonce repeats for some user. We will refer to this later. We omit the (obvious) proof.

Proposition 3.1 *Let* $\mathsf{SE1}$ *be an NBE1 scheme. Given adversary* $A_{\text{rn}} \in \mathcal{A}^{\text{ae1}}_{\text{r-n}}$ *making at most* u NEW *queries and at most* q ENC *queries per user, we construct adversary* $A_{\text{un}} \in \mathcal{A}^{\text{ae1}}_{\text{u-n}}$ *such that*

$$\mathbf{Adv}^{\text{ae1}}_{\mathsf{SE1}}(A_{\text{rn}}) \leq \mathbf{Adv}^{\text{ae1}}_{\mathsf{SE1}}(A_{\text{un}}) + \frac{uq(q-1)}{2^{\mathsf{SE1.nl}}} \ .$$

Adversary A_{un} *preserves the resources of* A_{rn}.

Saying A_{un} preserves the resources of A_{rn} means that the number of queries to all oracles are the same for both.

We believe our (above) AE1 framework (single game, many adversary classes) is of independent interest, as a way to unify, better understand and compactly present existing and new notions of security for NBE1 schemes. We give a similar framework for AE2 next.

NBE2 SYNTAX. An NBE2 scheme $\mathsf{SE2}$ specifies several algorithms and related quantities, as follows. Deterministic encryption algorithm $\mathsf{SE2.Enc} : \mathsf{SE2.KS} \times \mathsf{SE2.NS} \times \mathsf{SE2.MS} \times \mathsf{SE2.HS} \to \{0,1\}^*$, just like for NBE1, takes a key K in the (finite) key-space $\mathsf{SE2.KS}$, a nonce N in the nonce-space $\mathsf{SE2.NS}$, a message M in the message space $\mathsf{SE2.MS}$ and a header H in the header space $\mathsf{SE2.HS}$ to return a ciphertext C_2 that is in the ciphertext space $\mathsf{SE2.CS}(|N|,|M|,|H|)$. $\mathsf{SE2}$ also specifies a deterministic decryption algorithm $\mathsf{SE2.Dec} : \mathsf{SE2.KS} \times \{0,1\}^* \times \mathsf{SE2.HS} \to \mathsf{SE2.MS} \cup \{\perp\}$ that takes key K, ciphertext C_2 and header H to return an output that is either a message $M \in \mathsf{SE2.MS}$, or $\perp$. (Unlike in NBE1, it does *not* take a nonce input.) It is required that $\mathsf{SE2.NS}, \mathsf{SE2.MS}, \mathsf{SE2.HS}$ are length-closed sets as defined in Section 2. Most often nonces are of a fixed length denoted $\mathsf{SE2.nl}$, meaning $\mathsf{SE2.NS} = \{0,1\}^{\mathsf{SE2.nl}}$. Decryption correctness requires that $\mathsf{SE2.Dec}(K, \mathsf{SE2.Enc}(K,N,M,H),H) = M$ for all $K \in \mathsf{SE2.KS}, N \in \mathsf{SE2.NS}, M \in \mathsf{SE2.MS}$ and $H \in \mathsf{SE2.HS}$.

AE2 GAME AND ADVANTAGE. Let $\mathsf{SE2}$ be an NBE2 scheme and A an adversary. We associate to them the game $\mathbf{G}^{\text{ae2}}_{\mathsf{SE2}}(A)$ shown on the top right of Fig. 3. (We use the name "AE2" to associate the game with the NBE2 syntax). The AE2-advantage of adversary A is $\mathbf{Adv}^{\text{ae2}}_{\mathsf{SE2}}(A) = 2\Pr[\mathbf{G}^{\text{ae2}}_{\mathsf{SE2}}(A)] - 1$. The game is in the multi-user setting, oracle NEW allowing the adversary to initialize a new user with a fresh key. It is required that any ENC(i,N,M,H) query of A satisfy $1 \leq i \leq v$, $N \in \mathsf{SE2.NS}$, $M \in \mathsf{SE2.MS}$ and $H \in \mathsf{SE2.HS}$. When the challenge bit b is 1, the encryption oracle will return a ciphertext as stipulated by $\mathsf{SE2.Enc}$, using the key for the indicated user i. In the $b = 0$

case, ENC will return a random element of the ciphertext space $\mathsf{SE2.CS}(|N|, |M|, |H|)$. The array $\mathbf{M}$ is assumed to initially be $\perp$ everywhere, and holds ciphertexts returned by ENC. It is required that any $\textsc{Dec}(i, C_2, H)$ query of A satisfy $1 \leq i \leq v$ and $H \in \mathsf{SE2.HS}$. When the challenge bit b is 1, the decryption oracle will perform decryption as stipulated by SE2.Dec, using the key for the indicated user i. In the $b = 0$ case, DEC will return $\perp$ on any ciphertext not previously returned by the encryption oracle.

AE2 SECURITY METRICS. As with AE1-security, restrictions must be placed on the adversary to achieve AE2-security, and we use adversary classes to capture restrictions corresponding to different notions of interest. The classes are summarized by the table at the bottom of Figure 3, with the superscript "x" now being ae2. The classes and resulting notions are analogous to those for AE1. Thus, AE2[$\mathcal{A}_1^{\mathrm{ae2}}$]-security recovers the single-user setting. $\mathcal{A}_{\text{u-n}}^{\mathrm{ae2}}$ is the class of adversaries whose ENC queries never repeat a user-nonce pair, so AE2[$\mathcal{A}_{\text{u-n}}^{\mathrm{ae2}}$]-security is what we have referred to as basic AE2-security in Section 1. Adversaries in $\mathcal{A}_{\text{u-nmh}}^{\mathrm{ae2}} \supseteq \mathcal{A}_{\text{u-n}}^{\mathrm{ae2}}$ are allowed to re-use a user-nonce pair across ENC queries as long as they never repeat an entire query, so AE2[$\mathcal{A}_{\text{u-nmh}}^{\mathrm{ae2}}$]-security is what we have referred to as advanced AE2-security in Section 1. Adversaries in $\mathcal{A}_{\text{r-n}}^{\mathrm{ae2}}$ pick the nonces in their ENC queries uniformly and independently at random from SE2.NS. AE2[$\mathcal{A}_{\text{priv}}^{\mathrm{ae2}}$]-security is privacy only.

DISCUSSION. The main (small but important) change in the syntax from NBE1 to NBE2 is that in the latter, the decryption algorithm no longer gets the nonce as input. It is up to encryption to ensure that the ciphertext contains everything (beyond key and header) needed to decrypt. Nonces are thus no longer magically communicated, making the interface, and the task of application designers, simpler and less error-prone, reducing the possibility of loss of privacy from poor choices of nonces and opening the door to nonce-hiding security as captured by AE2. Another change is that, rather than a ciphertext length function, an NBE2 scheme specifies a ciphertext space. The reason is that a ciphertext might have some structure, like being a pair (C, C'). Ciphertexts like this cannot be indistinguishable from random strings, but they can be indistinguishable from pairs of random strings, which is captured by defining the ciphertext space correspondingly. This follows [23], in whose committing AE definition the same issue arose.

NONCE-RECOVERING NBE2. A natural subclass of NBE2 schemes are those which recover the nonce explicitly during decryption. We provide definitions to capture such schemes. We say that an NBE2 scheme SE2 is nonce-recovering if there exists a deterministic nonce-plus-message recovery algorithm SE2.NMR such that for any $(K, C_2, H) \in \mathsf{SE2.KS} \times \{0,1\}^* \times \mathsf{SE2.HS}$, if $\mathsf{SE2.NMR}(K, C_2, H) \neq \perp$ then it parses as a pair $(M, N) \in \mathsf{SE2.MS} \times \mathsf{SE2.NS}$ satisfying $\mathsf{SE2.Dec}(K, C_2, H) = M$ and $\mathsf{SE2.Enc}(K, N, M, H) = C_2$. Most of our transforms from NBE1 scheme to NBE2 schemes yield nonce-recovering NBE2 schemes.

4 Usage of NBE1: The Transmit-Nonce transform

With AE1-secure NBE1, the nonce is needed for decryption. But how does the decryptor get it? This is a question about usage not addressed in the formalism. The understanding, however, is that the nonce can be communicated in the clear, with the core ciphertext. One might argue this is fine because, in the AE1-formalism, the adversary picks the nonce, so seeing the nonce again in the ciphertext cannot give the adversary an advantage.

We have discussed in the introduction why this fails to model cases where the nonce is chosen by the user, and why, at least in general, nonce transmission may violate message privacy. But

the claim, so far, was informal. The reason was that transmitting the nonce represents a *usage* of NBE1 and we had no definitions to capture this. With AE2-secure NBE2, that gap is filled and we are in a position to formalize the claim of usage insecurity.

Some readers may see this is unnecessary, belaboring an obvious point. Indeed, the intuition is clear enough. But formalizing it serves also as an introduction to exercising our framework. We capture the usage in question as an NBE2 scheme $\mathsf{SE}_{\mathrm{TN}} = \mathbf{TN}[\mathsf{SE1}]$ built from a given NBE1 scheme $\mathsf{SE1}$ by what we call the transmit-nonce transform $\mathbf{TN}$. We detail the (rather obvious) claim that $\mathsf{SE}_{\mathrm{TN}}$ fails to meet AE2-security, and discuss how it will also fail to meet other, weaker privacy goals.

The **TN** Transform. Our **TN** (Transmit Nonce) transform takes an NBE1 scheme $\mathsf{SE1}$ and returns the NBE2 scheme $\mathsf{SE}_{\mathrm{TN}} = \mathbf{TN}[\mathsf{SE1}]$, that, as the name suggests, transmits the nonce in the clear, meaning the $\mathsf{SE}_{\mathrm{TN}}$ ciphertext is the nonce together with the $\mathsf{SE1}$ core ciphertext. In more detail, encryption algorithm $\mathsf{SE}_{\mathrm{TN}}.\mathsf{Enc}(K, N, M, H)$ lets $C_1 \leftarrow \mathsf{SE1.Enc}(K, N, M, H)$ and returns ciphertext $C_2 \leftarrow (N, C_1)$. Decryption algorithm $\mathsf{SE}_{\mathrm{TN}}.\mathsf{Dec}(K, C_2, H)$ parses C_2 as a pair (N, C_1) with $N \in \mathsf{SE1.NS}$ —we write this as $(N, C_1) \leftarrow C_2$— returning $\perp$ if the parsing fails, and else returning $M \leftarrow \mathsf{SE1.Dec}(K, N, C_1, H)$. NBE2 scheme $\mathsf{SE}_{\mathrm{TN}}$ has the same key space, message space and header space as $\mathsf{SE1}$, and we define its ciphertext space via $\mathsf{SE}_{\mathrm{TN}}.\mathsf{CS}(\ell_n, \ell_m, \ell_h) = \mathsf{SE1.NS} \times \{0,1\}^{\mathsf{SE1.ccl}(\ell_n, \ell_m, \ell_h)}$ for all $\ell_n, \ell_m, \ell_h \geq 0$. Usage of $\mathsf{SE1}$ in which the nonce is sent in the clear (along with the core ciphertext) can now be formally modeled by asking what formal security notions for NBE2 schemes are met by $\mathsf{SE}_{\mathrm{TN}} = \mathbf{TN}[\mathsf{SE1}]$.

Insecurity of $\mathbf{TN}[\mathsf{SE1}]$. Let $\mathsf{SE1}$ be *any* NBE1 scheme. It might, like GCM, be $\mathrm{AE1}[\mathcal{A}^{\mathrm{ae1}}_{\text{u-n}}]$-secure, or it might even be $\mathrm{AE1}[\mathcal{A}^{\mathrm{ae1}}_{\text{u-nmh}}]$-secure. Regardless, we claim that NBE2 scheme $\mathsf{SE}_{\mathrm{TN}} = \mathbf{TN}[\mathsf{SE1}]$ fails to be $\mathrm{AE2}[\mathcal{A}^{\mathrm{ae2}}_{\mathrm{priv}} \cap \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}]$-secure, meaning fails to provide privacy even for adversaries that do not reuse a nonce. This is quite obvious, since the adversary can test whether the nonce in its Enc query matches the one returned in the ciphertext. In detail:

Adversary A
Initialize
Pick some $(N, M, H) \in \mathsf{SE1.NS} \times \mathsf{SE1.MS} \times \mathsf{SE1.HS}$ with $|N| \geq 1$
New // Initialize one user
$(N^*, C_1) \leftarrow\!\!\$\ \textsc{Enc}(1, N, M, H)$ // Ciphertext returned is a pair
If $(N^* = N)$ then $b' \leftarrow 1$ else $b' \leftarrow 0$
Finalize(b')

This adversary has advantage $\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{TN}}}(A) \geq 1 - 1/2 = 1/2$, so represents a violation of $\mathrm{AE2}[\mathcal{A}^{\mathrm{ae2}}_{\mathrm{priv}} \cap \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}]$-security.

Discussion. The attack above may be difficult to reconcile with $\mathsf{SE1}$ being $\mathrm{AE1}[\mathcal{A}^{\mathrm{ae1}}_{\text{u-n}}]$-secure, the question being that, in the AE1 game, the adversary picks the nonce, and thus already knows it, so why should seeing it again in the ciphertext give the adversary extra information? The answer is that in usage the adversary does not know the nonce *a priori* and seeing may provide additional information. This is not modeled in AE1 but is modeled in AE2. To be clear, the above violation of AE2 security does *not* contradict the assumed AE1-security of $\mathsf{SE1}$.

One might (correctly) argue that AE2 is a strong requirement so failing it does not represent a concerning violation of security, but it is clear that $\mathsf{SE}_{\mathrm{TN}}$ will fail to meet even much weaker notions of privacy for NBE2 schemes that one could formalize in natural ways, such as message recovery security or semantic security. (The nonce could be message dependent, in the extreme equal to the message.) One might also suggest that the losses of privacy occur for pathological choices of nonces,

and nonce transmission is just fine if the nonce is a random number or counter, to which there are two responses. (1) The pitch and promise of AE1[$\mathcal{A}^{\text{ae1}}_{\text{u-n}}$]-secure NBE1 is that *any* (non-repeating) nonce is fine. For example RBBK [50] says "The entity that encrypts chooses a new nonce for every message with the *only* restriction that no nonce is used twice," and RFC 5116 says "Applications SHOULD use the nonce formation method defined in Section 3.2, and MAY use any other method that meets the uniqueness requirement." It is important to know (both to prevent misuse and for our understanding) that in usage of NBE1, security requires more than just uniqueness of nonces; one must be concerned with how they are conveyed to the receiver. (2) A counter nonce can lead to loss of user privacy, for example revealing identity information, that is resolved by moving to AE2[$\mathcal{A}^{\text{ae2}}_{\text{u-n}}$]-secure NBE2, which is nonce hiding.

5 Basic transforms

We have explained that AE2-secure NBE2 offers valuable security and usability benefits over current encryption. So we now turn to achieving it. We follow the development path of NBE1, first, in this section, targeting basic AE2-security —no user reuses a nonce, which in our framework corresponds to adversaries in the class $\mathcal{A}^{\text{ae2}}_{\text{u-n}}$— and then, in Section 6, targeting advanced AE2-security —misuse resistance, where nonce-reuse is allowed, which in our framework corresponds to adversaries in the class $\mathcal{A}^{\text{ae2}}_{\text{u-nmh}}$.

Significant effort has gone into the design and analysis of basic-AE1-secure NBE1 schemes. We want to leverage rather than discard this. Accordingly, rather than from-scratch designs, we seek *transforms* of basic-AE1-secure NBE1 schemes into basic-AE2-secure NBE2 ones. This section gives three transforms that are simple and efficient and minimize quantitative security loss.

Preliminaries. We assume for simplicity that the NBE1 schemes provided as input to our transforms have nonces of a fixed length, meaning that $\mathsf{SE1.NS} = \{0,1\}^{\mathsf{SE1.nl}}$. This holds for most real-world AE1-secure NBE1 schemes. All our transforms can be adapted to allow variable-length nonces.

Core ciphertexts in practical NBE1 schemes tend to be no shorter than a certain minimal value, for example 96 bits for typical usage of GCM with AES [21]. We refer to this value as the minimal core-ciphertext length of the scheme SE1, formally defining $\mathsf{SE1.mccl} = \min_{N,M,H} \{\mathsf{SE1.ccl}(|N|,|M|,|H|)\}$ where the minimum is over all $(N,M,H) \in \mathsf{SE1.NS} \times \mathsf{SE1.MS} \times \mathsf{SE1.HS}$. This is relevant because some of our transforms need SE1.mccl to be non-trivial to provide security.

All transforms here use two keys, meaning the key for the constructed NBE2 scheme SE2 is a pair consisting of a key for a PRF and a key for SE1. An implementation can, starting from a single overlying key, derive these sub-keys and store them, so that neither key size nor computational cost increase. This is well understood and is done as part of OCB, GCM and many other designs.

The ciphertext overhead is the bandwidth cost of the transform. We now discuss how to measure it. In the NBE2 scheme SE2 constructed by any of our transforms from an NBE1 scheme SE1, the ciphertext space is the set of strings of some length, $\mathsf{SE2.CS}(\ell_n,\ell_m,\ell_h) = \{0,1\}^{\mathsf{SE2.cl}(\ell_n,\ell_m,\ell_h)}$. Since NBE1 decryption gets the nonce for free while NBE2 decryption must, effectively, communicate it via the ciphertext, the "fair" definition of the ciphertext overhead of the transform is the maximum, over all possible choices of ℓ_n,ℓ_m,ℓ_h, of

$$\mathsf{SE2.cl}(\ell_n,\ell_m,\ell_h) - \mathsf{SE2.ccl}(\ell_n,\ell_m,\ell_h) - \mathsf{SE1.nl}\ .$$

Another way to put it is that the ciphertext overhead is how much longer ciphertexts are in SE2 than in **TN**[SE1]. All our transforms have ciphertext overhead zero, meaning are optimal in terms of bandwidth usage.

THE **HN1** TRANSFORM. The idea of our first transform is that a piece of the core ciphertext may be used as a nonce under which to encrypt the actual nonce. Let $\mathsf{SE1}$ be an NBE1 scheme and F a function family with $\mathsf{F.ol} = \mathsf{SE1.nl}$, so that outputs of $\mathsf{F.Ev}$ can be used to mask nonces for $\mathsf{SE1}$. Assume $\mathsf{SE1.mccl} \geq \mathsf{F.il}$, so that an $\mathsf{F.il}$-bit prefix of a core ciphertext can be used as an input to $\mathsf{F.Ev}$. Invertibility of F is not required, so it can, but need not, be a blockcipher. Our **HN1** transform defines NBE2 scheme $\mathsf{SE}_{\mathrm{HN1}} = \mathbf{HN1}[\mathsf{SE1}, \mathsf{F}]$ whose encryption and decryption algorithms are shown in Figure 4. A key (K_{F}, K_1) for $\mathsf{SE}_{\mathrm{HN1}}$ is a pair consisting of a key K_{F} for F and a key K_1 for $\mathsf{SE1}$, so that the key space is $\mathsf{SE}_{\mathrm{HN1}}.\mathsf{KS} = \{0,1\}^{\mathsf{F.kl}} \times \mathsf{SE1.KS}$. The message, header and nonce spaces are unchanged. The parsing $Y \| C_1 \leftarrow C_2$ in the second line of the decryption algorithm $\mathsf{SE}_{\mathrm{HN1}}$ is such that $|Y| = \mathsf{SE1.nl}$. The ciphertext overhead is zero. The computational overhead is one call to $\mathsf{F.Ev}$ for each of encryption or decryption. The following says that if the starting NBE1 scheme $\mathsf{SE1}$ is basic-AE1-secure and F is a PRF then the NBE2 scheme $\mathsf{SE}_{\mathrm{HN1}}$ returned by the transform is basic-AE2-secure. The proof is in Appendix B.

Theorem 5.1 *Let $\mathsf{SE}_{\mathrm{HN1}} = \mathbf{HN1}[\mathsf{SE1}, \mathsf{F}]$ be obtained as above. Then, given adversary $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}$, making q_n queries to its* NEW *oracle, q_e queries per user to its* ENC *oracle, and q_d queries per user to its* DEC *oracle, we construct adversaries $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{u-n}}$ and B such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN1}}}(A_2) \leq \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B) + \frac{q_n(q_e+q_d)(q_e+q_d-1)}{2^{\mathsf{F.il}+1}} \ .$$

Adversary A_1 preserves the resources of A_2. Adversary B makes q_n queries to its NEW *oracle and $q_e + q_d$ queries per user to its* FN *oracle. Adversary B has about the same running time as A_2.*

SPLITTING. Our next transform employs ciphertext stealing [41] to get zero ciphertext overhead. There are many choices with regard to how to implement stealing, for example whether one steals from the first part of the core ciphertext or the last, and implementations may have different preferences. Accordingly, we do not pin down a choice but instead parameterize the transform by a splitting algorithm responsible for splitting a given string X (the core ciphertext) into segments x (the stolen part, of a prescribed length ℓ) and y (the rest). Formally, splitting scheme Spl specifies a deterministic algorithm $\mathsf{Spl.Ev}$ that takes an integer $\ell \geq 0$ and a string X with $|X| \geq \ell$, and returns a pair of strings $(x, y) \leftarrow \mathsf{Spl.Ev}(\ell, X)$ with $|x| = \ell$. If $(x, y) \in \mathrm{Im}(\mathsf{Spl.Ev}(|x|, \cdot))$ —the image of a function was defined in Section 2— then $X \leftarrow \mathsf{Spl.In}(x, y)$ recovers the unique X such that $\mathsf{Spl.Ev}(|x|, X) = (x, y)$, and otherwise returns $X = \perp$.

This isn't enough because for security we want that if X is random then so are x, y. A simple way to ensure this is to require that the split sets x to some bit positions of X and y to the rest, *with the choice of positions depending only on $|X|$*. Formally, we require that there is a (deterministic) function $\mathsf{Spl.St}$ that given integers ℓ, n with $n \geq \ell \geq 0$ returns a starting index $s = \mathsf{Spl.St}(\ell, n)$ in the range $1 \leq s \leq n - \ell + 1$, and $\mathsf{Spl.Ev}(\ell, X)$ returns $x = X[s..(s+\ell-1)]$ and $y = X[1..(s-1)] \| X[(s+\ell)..|X|]$ for $s = \mathsf{Spl.St}(\ell, |X|)$. The most common choices are that $\mathsf{Spl.St}(\ell, n) = 1$, so that $x = X[1..\ell]$ is the ℓ-bit prefix of X and $y = X[(\ell+1)..|X|]$ is the rest (corresponding to stealing from the first part of X), or $\mathsf{Spl.St}(\ell, n) = n - \ell + 1$, so that $x = X[(|X| - \ell + 1)..|X|]$ is the ℓ-bit suffix of X and $y = X[1..(|X| - \ell)]$ is the rest (corresponding to stealing from the last part of X), but other choices are possible.

THE **HN2** TRANSFORM. The starting idea of this transform is that our NBE2 scheme can encrypt under the given NBE1 scheme and then also include in the ciphertext an enciphering, under a blockcipher E, of the nonce. We enhance this to encipher, along with the nonce, ℓ bits stolen from the core ciphertext. The stealing has two dividends. First, nonces are often shorter than the block length of E —for example $\mathsf{SE1.nl} = 96$ and $\mathsf{E.bl} = 128$ for AES-GCM and OCB [50, 34]— so in the

$\underline{\mathsf{SE}_{\mathrm{HN1}}.\mathsf{Enc}((K_{\mathsf{F}}, K_1), N, M, H)}$	$\underline{\mathsf{SE}_{\mathrm{HN1}}.\mathsf{Dec}((K_{\mathsf{F}}, K_1), C_2, H)}$
$C_1 \leftarrow \mathsf{SE1.Enc}(K_1, N, M, H)$	If $(\lvert C_2\rvert < \mathsf{SE1.nl} + \mathsf{F.il})$ then return $\perp$
$x \leftarrow C_1[1..\mathsf{F.il}]$; $P \leftarrow \mathsf{F}(K_{\mathsf{F}}, x)$	$Y \Vert C_1 \leftarrow C_2$; $x \leftarrow C_1[1..\mathsf{F.il}]$; $P \leftarrow \mathsf{F}(K_{\mathsf{F}}, x)$
$Y \leftarrow P \oplus N$; $C_2 \leftarrow Y \Vert C_1$	$N \leftarrow P \oplus Y$; $M \leftarrow \mathsf{SE1.Dec}(K_1, N, C_1, H)$
Return C_2	Return M
$\underline{\mathsf{SE}_{\mathrm{HN2}}.\mathsf{Enc}((K_{\mathsf{E}}, K_1), N, M, H)}$	$\underline{\mathsf{SE}_{\mathrm{HN2}}.\mathsf{Dec}((K_{\mathsf{E}}, K_1), C_2, H)}$
$C_1 \leftarrow \mathsf{SE1.Enc}(K_1, N, M, H)$	If $(\lvert C_2\rvert < \mathsf{E.bl})$ then return $\perp$
$(x, y) \leftarrow \mathsf{Spl.Ev}(\ell, C_1)$	$N \Vert x \leftarrow \mathsf{E.In}(K_{\mathsf{E}}, C_2[1..\mathsf{E.bl}])$
$C_{2,1} \leftarrow \mathsf{E.Ev}(K_{\mathsf{E}}, N \Vert x)$	$y \leftarrow C_2[\mathsf{E.bl}+1..\lvert C_2\rvert]$; $C_1 \leftarrow \mathsf{Spl.In}(x, y)$
$C_2 \leftarrow C_{2,1} \Vert y$; Return C_2	If $(C_1 = \perp)$ then return $\perp$
	$M \leftarrow \mathsf{SE1.Dec}(K_1, N, C_1, H)$; Return M
$\underline{\mathsf{SE}_{\mathrm{HN3}}.\mathsf{Enc}((K_{\mathsf{F}}, K_1), N, M, H)}$	$\underline{\mathsf{SE}_{\mathrm{HN3}}.\mathsf{Dec}((K_{\mathsf{F}}, K_1), C_2, H)}$
$N_1 \leftarrow \mathsf{F.Ev}(K_{\mathsf{F}}, N)$	If $(\lvert C_2\rvert < \mathsf{F.ol})$ then return $\perp$
$C_1 \leftarrow \mathsf{SE1.Enc}(K_1, N_1, M, H)$	$N_1 \Vert C_1 \leftarrow C_2$; $M \leftarrow \mathsf{SE1.Dec}(K_1, N_1, C_1, H)$
$C_2 \leftarrow N_1 \Vert C_1$; Return C_2	Return M

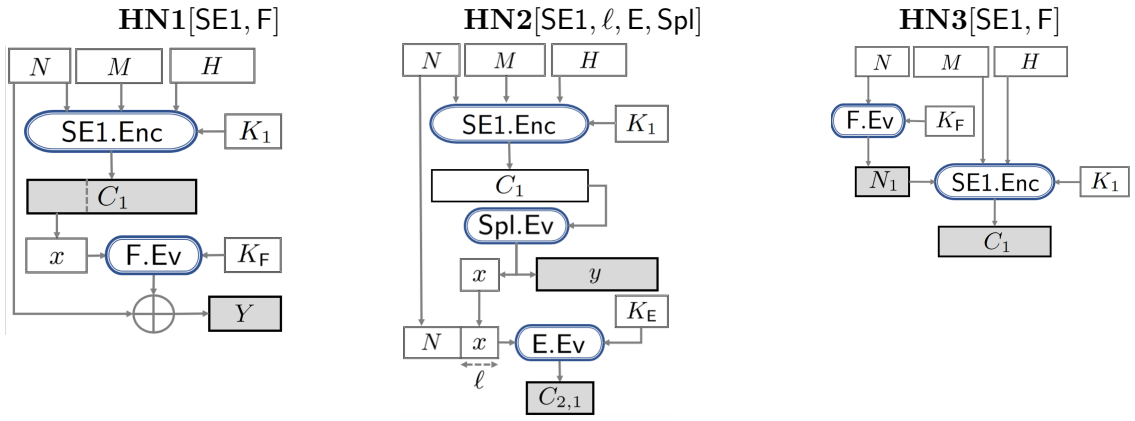


Figure 4: Top: Encryption and decryption algorithms of the NBE2 schemes constructed by our basic transforms. From top to bottom: $\mathsf{SE}_{\mathrm{HN1}} = \mathbf{HN1}[\mathsf{SE1}, \mathsf{F}]$, $\mathsf{SE}_{\mathrm{HN2}} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ and $\mathsf{SE}_{\mathrm{HN3}} = \mathbf{HN3}[\mathsf{SE1}, \mathsf{F}]$. Bottom: Diagrams illustrating the encryption algorithms of the constructed schemes.

absence of stealing, the nonce would be padded before enciphering, leading to ciphertext overhead. Second, while we show here (Theorem 5.2) that the scheme preserves basic security regardless of the amount ℓ stolen, we show later (Theorem 6.2) that it preserves even advanced security if ℓ is non-trivial (128 bits or more). We now proceed to the full description.

Let $\mathsf{SE1}$ be an NBE1 scheme, Spl a splitting scheme and $\ell \geq 0$ the prescribed length of the stolen segment of the core ciphertext. We assume the minimal core-ciphertext length of $\mathsf{SE1}$ satisfies $\mathsf{SE1.mccl} \geq \ell$, which ensures that core ciphertexts are long enough to allow the desired splitting. Let E be a blockcipher with block length $\mathsf{E.bl} = \mathsf{SE1.nl} + \ell$. Our **HN2** transform defines NBE2 scheme $\mathsf{SE}_{\mathrm{HN2}} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ whose encryption and decryption algorithms are shown in Figure 4. The parsing in the second line of the decryption algorithm $\mathsf{SE}_{\mathrm{HN2}}$ is such that $|N| = \mathsf{SE1.nl}$. A key (K_{E}, K_1) for $\mathsf{SE}_{\mathrm{HN2}}$ is a pair consisting of a key K_{E} for E and a key K_1 for $\mathsf{SE1}$, so that the key space is $\mathsf{SE}_{\mathrm{HN2}}.\mathsf{KS} = \{0,1\}^{\mathsf{E.kl}} \times \mathsf{SE1.KS}$. The nonce, message and header spaces are unchanged. The length

of ciphertext C_2 is $\mathsf{E.bl} + |C_1| - \ell = |C_1| + \mathsf{SE1.nl}$, so the ciphertext space is $\mathsf{SE}_{\mathrm{HN2}}.\mathsf{CS}(\ell_n, \ell_m, \ell_h) = \{0,1\}^{\mathsf{SE1.nl}+\mathsf{SE1.ccl}(\ell_n,\ell_m,\ell_h)}$. The ciphertext overhead is zero. The computational overhead is an extra blockcipher call for encryption and a blockcipher inverse for decryption.

A typical instantiation for basic security is $\mathsf{E} = \mathsf{AES}$, so that $\mathsf{E.bl} = 128$. Nonces would have length $\mathsf{SE1.nl} = 96$. We then set $\ell = 32$ and $\mathsf{Spl.St}(\ell, n) = 1$ for all n. This means $\mathsf{SE1.mccl}$ must be at least 32, which is true for all real-world schemes we know. This reduction in the required value of $\mathsf{SE1.mccl}$ for security is the benefit that **HN2** offers over **HN1**. Recall the latter needs $\mathsf{F.il} \geq \mathsf{SE1.mccl}$, and hence by Theorem 5.1 needs $\mathsf{SE1.mccl} \geq 128$, for the same security that **HN2** can offer with $\mathsf{SE1.mccl} \geq 32$.

The following says that if the starting NBE1 scheme $\mathsf{SE1}$ is basic-AE1-secure and E is a PRF, then the NBE2 scheme $\mathsf{SE}_{\mathrm{HN2}}$ returned by the transform is basic-AE2-secure. This holds regardless of the value of ℓ. The proof is in Appendix C.

Theorem 5.2 *Let* $\mathsf{SE}_{\mathrm{HN2}} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ *be obtained as above. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}$*, making* q_n *queries to its* New *oracle,* q_e *queries per user to its* Enc *oracle, and* q_d *queries per user to its* Dec *oracle, we construct adversaries* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{u-n}}$ *and* B *such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_2) \;\leq\; \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{E}}(B)\;. \tag{1}$$

Adversary A_1 *preserves the resources of* A_2*. Adversary* B *makes* q_n *queries to its* New *oracle and* q_e *queries per user to its* Fn *oracle. Adversary* B *has about the same running time as* A_2*.*

The **HN3** Transform. Our third transform uses what we call nonce-based nonce-derivation, in which encryption is performed under $\mathsf{SE1}$ using as nonce the result $N_1 = \mathsf{F}(K_{\mathsf{F}}, N)$ of a PRF F on the actual nonce N. The idea comes from SIV [51] but differences include that: (1) SIV constructs an AE1-secure NBE1 scheme while we construct an AE2-secure NBE2 scheme. (2) SIV decryption needs to have the original nonce. (3) Our synthetic nonce N_1 is a function only of the actual nonce while the one in SIV is also a function of the message and header.

Proceeding to the details, let $\mathsf{SE1}$ be an NBE1 scheme. Let F be a function family with $\mathsf{F.ol} = \mathsf{SE1.nl}$, meaning outputs of $\mathsf{F.Ev}$ can be used as nonces for $\mathsf{SE1}$. Invertibility of F is not required, so it can, but need not, be a blockcipher. Our **HN3** transform defines NBE2 scheme $\mathsf{SE}_{\mathrm{HN3}} = \mathbf{HN3}[\mathsf{SE1}, \mathsf{F}]$ whose encryption and decryption algorithms are shown in Figure 4. A key (K_{F}, K_1) for $\mathsf{SE}_{\mathrm{HN3}}$ is a pair consisting of a key K_{F} for F and a key K_1 for $\mathsf{SE1}$, so that the key space is $\mathsf{SE}_{\mathrm{HN3}}.\mathsf{KS} = \{0,1\}^{\mathsf{F.kl}} \times \mathsf{SE1.KS}$. The message and header spaces are unchanged, and the nonce space is $\mathsf{SE}_{\mathrm{HN3}}.\mathsf{NS} = \{0,1\}^{\mathsf{F.il}}$, meaning inputs to F are nonces for $\mathsf{SE2}$. The parsing in the second line of the decryption algorithm $\mathsf{SE}_{\mathrm{HN3}}$ of Figure 4 is such that $|N_1| = \mathsf{SE1.nl}$. Note that the decryption algorithm does not use F or K_{F}.

As with **HN1** and **HN2**, the **HN3** transform has zero ciphertext overhead. The computational overhead for encryption is one invocation of F. Advantages emerge with decryption, where there is now *no* computational overhead. Indeed decryption in $\mathsf{SE}_{\mathrm{HN3}}$ is effectively the same as in $\mathsf{SE1}$. In particular, in the typical case that F is a blockcipher on which $\mathsf{SE1}$ is itself based, decryption (unlike with **HN2**) no longer needs to implement its inverse, which can be a benefit in hardware and for reducing code size.

It is natural and convenient here to assume $\mathsf{SE1}$ is AE1$[\mathcal{A}^{\mathrm{ae1}}_{\text{r-n}}]$-secure. (Recall this is AE1-security for the class of adversaries that pick the nonce at random.) By Proposition 3.1 this is implied by its being AE1$[\mathcal{A}^{\mathrm{ae1}}_{\text{u-n}}]$-secure (that is, basic-AE1-secure). Assuming additionally that F is a PRF, the following says that $\mathbf{HN3}[\mathsf{SE1}, \mathsf{F}]$ is AE2$[\mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}]$-secure (that is, basic-AE2-secure). The proof is in Appendix D.

$\underline{\mathsf{SE}_{\mathrm{HN4}}.\mathsf{Enc}((K_{\mathsf{F}}, K_1), N, M, H)}$	$\underline{\mathsf{SE}_{\mathrm{HN4}}.\mathsf{Dec}((K_{\mathsf{F}}, K_1), C_2, H)}$
$N_1 \leftarrow \mathsf{F.Ev}(K_{\mathsf{F}}, (N, M, H))$	If $(\lvert C_2\rvert < \mathsf{F.ol})$ then return $\perp$
$C_1 \leftarrow \mathsf{SE1.Enc}(K_1, N_1, N\|M, H)$	$N_1\|C_1 \leftarrow C_2$; $X \leftarrow \mathsf{SE1.Dec}(K_1, N_1, C_1, H)$
$C_2 \leftarrow N_1\|C_1$	If $(X = \perp)$ then return $\perp$
Return C_2	$N\|M \leftarrow X$; $T \leftarrow \mathsf{F.Ev}(K_{\mathsf{F}}, (N, M, H))$
	If $(T = N_1)$ then return M else return $\perp$
$\underline{\mathsf{SE}_{\mathrm{HN5}}.\mathsf{Enc}(K_{\mathsf{TE}}, N, M, H)}$	$\underline{\mathsf{SE}_{\mathrm{HN5}}.\mathsf{Dec}(K_{\mathsf{TE}}, C_2, H)}$
$C_2 \leftarrow \mathsf{TE.Ev}(K_{\mathsf{TE}}, H, 0^{\ell_t}\|N\|M)$	$X \leftarrow \mathsf{TE.In}(K_{\mathsf{TE}}, H, C_2)$
Return C_2	If $X[1..\ell_t] \neq 0^{\ell_t}$ then return $\perp$
	$N\|M \leftarrow X[(\ell_t + 1)..\lvert X\rvert]$; Return M

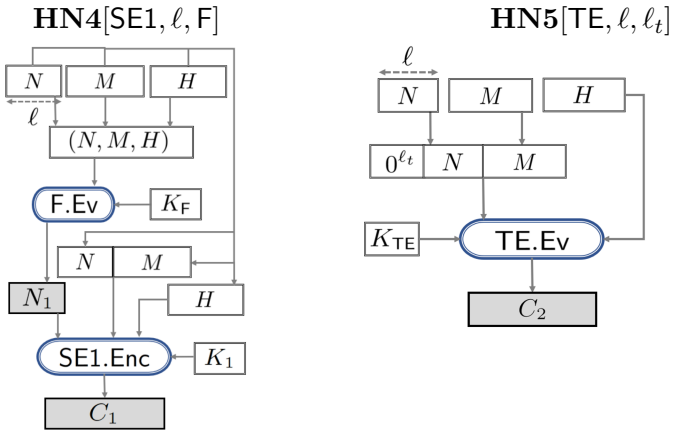


Figure 5: Top: Encryption and decryption algorithms of the NBE2 schemes constructed by our advanced transforms. From top to bottom: $\mathsf{SE}_{\mathrm{HN4}} = \mathbf{HN4}[\mathsf{SE1}, \ell, \mathsf{F}]$ and $\mathsf{SE}_{\mathrm{HN5}} = \mathbf{HN5}[\mathsf{TE}, \ell, \ell_t]$. Bottom: Diagrams illustrating the encryption algorithms of the constructed schemes.

Theorem 5.3 *Let* $\mathsf{SE}_{\mathrm{HN3}} = \mathbf{HN3}[\mathsf{SE1}, \mathsf{F}]$ *be obtained as above. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}$ *that makes* q_n *queries to its* New *oracle,* q_e *queries to its* Enc *oracle, and* q_d *queries to its* Dec *oracle, we construct adversaries* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{r-n}}$ *and* B *such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE2}}(A_2) \;\leq\; \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B)\ .$$

Adversary A_1 *preserves the resources of* A_2*. Adversary* B *makes* q_n *queries to its* New *oracle and* q_e *queries to its* Fn *oracle, respectively. Adversary* B *has about the same running time as* A_2*.*

6 Advanced transforms

We now turn to achieving AE2-security in the nonce-misuse setting, which we formalized as AE2$[\mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}]$-security. We discuss various transforms for this purpose.

Advanced security of **HN1**. We showed in Theorem 5.1 that **HN1** preserves basic security. It turns out that it also preserves advanced security. The following says that if the starting NBE1 scheme SE1 is advanced-AE1-secure and F is a PRF then the NBE2 scheme $\mathsf{SE}_{\mathrm{HN1}}$ returned by the transform is advanced-AE2-secure. The proof is in Appendix B.

Theorem 6.1 *Let* $\mathsf{SE}_{\mathrm{HN1}} = \mathbf{HN1}[\mathsf{SE1}, \mathsf{F}]$ *be obtained as above. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}$*, making* q_n *queries to its* New *oracle,* q_e *queries per user to its* Enc *oracle, and* q_d *queries per user to its* Dec *oracle, we construct adversaries* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{u-nmh}}$ *and* B *such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_2) \;\le\; \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B) + \frac{q_n(q_e+q_d)(q_e+q_d-1)}{2^{\mathsf{F.il}+1}} \;. \tag{2}$$

Adversary A_1 *preserves the resources of* A_2*. Adversary* B *makes* q_n *queries to its* New *oracle and* $q_e + q_d$ *queries per user to its* Fn *oracle. Adversary* B *has about the same running time as* A_2*.*

Advanced security of **HN2**. We showed in Theorem 5.2 that **HN2** preserves basic security regardless of the amount ℓ of stolen core-ciphertext—even if $\ell = 0$. For small ℓ, **HN2** may, however, leak information about the nonce in the advanced (misuse resistance) setting. The transformation does therefore not provide AE2$[\mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}]$-security. This is easy to see when $\ell = 0$, in which case if two different message-header pairs are encrypted with the same nonce, then the first part of the ciphertext is the same, leading to an $\mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}$-adversary with advantage $1 - 2^{-\mathsf{E.bl}}$. The advantage of this attack however decreases (exponentially) as ℓ increases. The following theorem says that once ℓ is non-trivial (say, 128 bits or more), the transform actually preserves advanced security as well. In Appendix E, we prove this theorem and describe the attack alluded to above in detail, showing that the bound in Theorem 6.2 is tight.

Theorem 6.2 *Let* $\mathsf{SE}_{\mathrm{HN2}} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ *be obtained as above. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}$*, making* q_n *queries to its* New *oracle and* q_e *queries per user to its* Enc *oracle, we construct adversaries* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{u-nmh}}$ *and* B *such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_2) \;\le\; \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{E}}(B) + \frac{q_n q_e(q_e-1)}{2^{\ell+1}} \;. \tag{3}$$

Adversary A_1 *preserves the resources of* A_2*. Adversary* B *makes* q_n *queries to its* New *oracle and* q_e *queries per user to its* Fn *oracle. Adversary* B *has about the same running time as* A_2*.*

This however is not ideal because security would need $\ell = 128$, which requires $\mathsf{SE1.mccl} \ge 128$ (not always true) and also, assuming 96-bit nonces, would require that the blockcipher E have block length 128+96=224, which precludes AES. We now give further transforms that do better.

The **HN4** Transform. The **HN3** transform clearly does *not* provide advanced-AE2-security because, if a nonce is repeated, the resulting ciphertexts have the same synthetic nonce, and hence the same first parts, which an adversary can notice. The starting idea for **HN4** is to obtain the synthetic nonce N_1 by applying the PRF F, not just to the actual nonce N as in **HN3**, but, as in SIV [51], to (N, M, H). If we now encrypt with N_1 under an NBE1 scheme $\mathsf{SE1}$, we can indeed show that AE2$[\mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}]$-security is achieved, assuming $\mathsf{SE1}$ is AE1$[\mathcal{A}^{\mathrm{ae1}}_{\text{u-nmh}}]$-secure. The latter assumption, however, is not satisfactory here because AE1$[\mathcal{A}^{\mathrm{ae1}}_{\text{u-nmh}}]$-security (typically achieved via SIV itself) already requires two passes through the entire input, so our computation of N_1 adds another entire pass, resulting in significant (non-constant) computational overhead. To avoid this we ask whether it would be enough for $\mathsf{SE1}$ to provide only privacy, meaning be AE1$[\mathcal{A}^{\mathrm{ae1}}_{\text{r-n}} \cap \mathcal{A}^{\mathrm{ae1}}_{\text{priv}}]$-secure, because this can be achieved in one pass. Indeed, this is what SIV assumes, but the difficulty is that SIV decryption makes crucial use of the original nonce N to provide authenticity, recomputing it and checking that it matches the one in the ciphertext. But to be nonce hiding, we cannot transmit N. We resolve this by including N as part of the message encrypted under $\mathsf{SE1}$.

Proceeding to the details, let $\mathsf{SE1}$ be an NBE1 scheme. Let F be a function family with $\mathsf{F.ol} = \mathsf{SE1.nl}$, meaning outputs of $\mathsf{F.Ev}$ can be used as nonces for $\mathsf{SE1}$, and also with $\mathsf{SE1.NS} \times$

$\mathsf{SE1.MS} \times \mathsf{SE1.HS} \subseteq \mathsf{F.D}$, meaning triples (N, M, H) can be used as inputs to F. Let $\ell \geq 1$ be an integer prescribing the nonce length of the constructed scheme. Our **HN4** transform defines NBE2 scheme $\mathsf{SE}_{\mathrm{HN4}} = \mathbf{HN4}[\mathsf{SE1}, \ell, \mathsf{F}]$ whose encryption and decryption algorithms are shown in Figure 5. A key (K_{F}, K_1) for $\mathsf{SE}_{\mathrm{HN4}}$ is a pair consisting of a key K_{F} for F and a key K_1 for $\mathsf{SE1}$, so that the key space is $\mathsf{SE}_{\mathrm{HN4}}.\mathsf{KS} = \{0,1\}^{\mathsf{F.kl}} \times \mathsf{SE1.KS}$. The message and header spaces are unchanged, and the nonce space is $\mathsf{SE}_{\mathrm{HN4}}.\mathsf{NS} = \{0,1\}^{\ell}$. The parsing in the second line of the decryption algorithm $\mathsf{SE}_{\mathrm{HN4}}$ of Figure 4 is such that $|N_1| = \mathsf{SE1.nl}$. The ciphertext overhead is zero, and if $\mathsf{SE1}$ is a standard one-pass privacy only scheme like counter-mode, then the computational overhead is constant.

Security, as with SIV, requires that $\mathsf{SE1}$ satisfies tidiness [44]. Formally, for all K, N, C_1, H, if $\mathsf{SE1.Dec}(K, N, C_1, H) = M \neq \perp$ then $\mathsf{SE1.Enc}(K, N, M, H) = C_1$. We capture the assumption that $\mathsf{SE1}$ provides only privacy in the nonce respecting setting, and it continues to be convenient for this to be for adversaries that pick the nonce at random, so our assumption for $\mathsf{SE1}$ is $\mathrm{AE1}[\mathcal{A}^{\mathrm{ae1}}_{\text{r-n}} \cap \mathcal{A}^{\mathrm{ae1}}_{\mathrm{priv}}]$-security. By Proposition 3.1 this is implied by its being $\mathrm{AE1}[\mathcal{A}^{\mathrm{ae1}}_{\text{u-n}} \cap \mathcal{A}^{\mathrm{ae1}}_{\mathrm{priv}}]$-secure. Assuming additionally that F is a PRF, the following says that $\mathbf{HN4}[\mathsf{SE1}, \ell, \mathsf{F}]$ is $\mathrm{AE2}[\mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}]$-secure. The proof is in Appendix F.

Theorem 6.3 *Let* $\mathsf{SE}_{\mathrm{HN4}} = \mathbf{HN4}[\mathsf{SE1}, \ell, \mathsf{F}]$ *be obtained as above, and let* $\mathsf{SE1}$ *satisfy tidiness. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}$ *making* q_n *queries to its* NEW *oracle and* q_e, q_d *encryption and decryption queries for each user, respectively, we construct adversaries* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{r-n}} \cap \mathcal{A}^{\mathrm{ae1}}_{\mathrm{priv}}$ *and* B *such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE2}}(A_2) \leq \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B) + \frac{q_n q_d}{2^{\mathsf{SE1.nl}}} \ .$$

Adversary A_1 *makes* q_n *queries to its* NEW *oracle,* q_e *queries to its* ENC *oracle per user, and no queries to its* DEC *oracle.* B *makes* q_n *queries to its* NEW *oracle, and* $q_e + q_d$ *queries to its* FN *oracle per user. Adversaries* A_1 *and* B *both have about the same running time as* A_2*.*

Our final transform **HN5** is different. It does not start from an NBE1 scheme but rather from a (arbitrary-input-length) tweakable cipher, extending the encode-then-encipher paradigm [13] to provide advanced-AE2-security. Instantiation via a fast tweakable cipher like AEZ [27] results in correspondingly fast advanced-AE2-secure NBE2.

THE **HN5** TRANSFORM. We encipher the nonce, message and some redundancy, using the header as the tweak. The change from [27] is to move the nonce from tweak to an input so as to hide it, which we will show is enough to confer AE2-security.

Proceeding to the details, let TE be a tweakable cipher as defined in Section 2. Let $\ell \geq 1$ be an integer prescribing the nonce length of the constructed scheme. Let $\ell_t \geq 0$ be the number of bits of redundancy we introduce to provide authenticity [13]. Our transform defines NBE2 scheme $\mathsf{SE}_{\mathrm{HN5}} = \mathbf{HN5}[\mathsf{TE}, \ell, \ell_t]$ whose encryption and decryption algorithms are shown in Figure 5. The key space of $\mathsf{SE}_{\mathrm{HN5}}$ is the key space of TE. The message space is $\{0,1\}^*$. The header space $\mathsf{SE}_{\mathrm{HN5}}.\mathsf{HS}$ is set to the tweak space $\mathsf{TE.TS}$ of TE. The nonce space is $\mathsf{SE}_{\mathrm{HN5}}.\mathsf{NS} = \{0,1\}^{\ell}$. The length of ciphertext $\mathsf{SE}_{\mathrm{HN5}}.\mathsf{Enc}(K, N, M, H)$ is $\ell_t + |N| + |M|$, so $\mathsf{SE}_{\mathrm{HN5}}.\mathsf{CS}(\ell_n, \ell_m, \ell_h) = \{0,1\}^{\ell_t + \ell + \ell_m}$. Ciphertext overhead, in this case, is not relative to an underlying NBE1 scheme, since there isn't any, but we see that ciphertexts are longer than message plus nonce by just ℓ_t bits, which is effectively optimal [27].

The following theorem shows that $\mathsf{SE}_{\mathrm{HN5}}$ is advanced-AE2-secure if tweakable cipher TE is an stPRP (as defined in Section 2) and ℓ_t is sufficiently large.

Theorem 6.4 *Let* $\mathsf{SE}_{\mathrm{HN5}} = \mathbf{HN5}[\mathsf{TE}, \ell, \ell_t]$ *be obtained as above. Let* $A \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}$ *be an adversary making* q_n *queries to its* New *oracle,* q_e *queries per user to its* Enc *oracle with minimum message length* ℓ_1*, and* q_d *queries with minimum ciphertext length* $\ell_2 \geq \ell_t$ *per user to its* Dec *oracle. We construct adversary* B *such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN5}}}(A) \leq \mathbf{Adv}^{\mathrm{stprp}}_{\mathsf{TE}}(B) + \frac{q_n q_e(q_e+1)}{2^{\ell_t+\ell+\ell_1+1}} + \frac{q_n q_d(q_d+1)}{2^{\ell_2+1}} + \frac{q_n q_d}{2^{\ell_t}} \ .$$

Adversary B *makes* q_n *queries to its* New *oracle,* q_e *queries per user to its* Fn *oracle, and* q_d *queries per user to its* FnInv *oracle.*

7 Dedicated transform for GCM

We have shown that our generic transforms allow us to immunize NBE1 schemes with low overhead. We now present a transform specific to a real-world NBE1 scheme: GCM. Our transform takes advantage of the underlying structure of GCM to further minimize overhead. Crucially, we also minimize changes to the scheme so that existing hardware and software can easily adapt.

Generalizing GCM to CAU1. Following BT [15], we generalize GCM via a transform CAU1. (We add the "1" to indicate that it is an NBE1 scheme.)

Let E be a block cipher, H be a function family and $\ell \geq 1$ be an integer indicating the desired nonce-length. Then $\mathsf{CAU1} = \mathbf{CAU1}[\mathsf{E}, \mathsf{H}, \ell]$ is an NBE1 scheme. $\mathsf{E.bl}(2^{\mathsf{E.bl}-\ell} - 2)$ is the maximum message length for CAU1 so we require that $1 \leq \ell < \mathsf{E.bl}$. We recall the algorithms CAU1.Enc, CAU1.Dec given by BT in Appendix H. Core ciphertexts returned by CAU1.Enc take the form $\tau \| C$, where τ is a tag of length E.bl. CAU1's keys are keys to its underlying block cipher E, meaning that CAU1.kl = E.kl. We use function family H to compute the tag τ. H takes input of the form (C, H) and returns an output of length E.bl. It uses a key which is generated by enciphering $0^{\mathsf{E.bl}}$ using E. This means that we require that $\mathsf{H.D} = \{0,1\}^* \times \mathsf{CAU1.HS}$ and $\mathsf{H.ol} = \mathsf{H.kl} = \mathsf{E.bl}$.

AES-GCM, as proposed by MV [40] and standardized by NIST [21], is obtained by instantiating $\mathsf{E} = \mathsf{AES}$ (so $\mathsf{E.bl} = 128$), $\mathsf{H} = \mathsf{GHASH}$ and $\ell = 96$. It is widely used in practice and achieves basic AE1-security (i.e. AE1$[\mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}]$-security). CAU1 has a fixed-length nonce, reflecting the standardized version of GCM, but a variant with variable-length nonces can be obtained by pre-processing the nonce, as discussed in [40].

AE2-secure CAU2. We exploit a feature of GCM, that the nonce can be derived from the authentication tag τ. In particular, if $\tau \| C \leftarrow \mathsf{CAU1.Enc}(K, N, M, H)$, then $\tau = \mathsf{H.Ev}(\mathsf{E.Ev}(K, 0^{\mathsf{E.bl}}), (C, H)) \oplus \mathsf{E.Ev}(K, N \| \langle 1 \rangle_{\mathsf{E.bl}-\ell})$. (Recall that, as defined in Section 2, $\langle i \rangle_n$ is the n-bit representation of integer i.) Therefore, in constructing our NBE2 variant of GCM, CAU2, we make use of the fact that the sender does not need to communicate the nonce – the receiver uses the tag to recover it. In other words, we exploit the "parsimoniousness" of $\mathbf{TN}$[CAU1] [12].

Unfortunately, the recovery procedure will succeed for any given ciphertext with probability $2^{-\mathsf{E.bl}+\ell}$, since this is the probability that *some* nonce with suffix $\langle 1 \rangle_{\mathsf{E.bl}-\ell}$ is recovered. This would be unacceptable in GCM since an adversary would be able to forge valid tags with probability 2^{-32}.

So in order to make the scheme work, we add redundancy to the scheme by prepending the message with 0^ℓ. CAU2 decryption will check that the message returned by CAU1.Dec indeed starts with such a string; this check works because a decryption with a "wrong" nonce leads to a random ciphertext. (For this reason, the maximum message length of CAU2 is ℓ bits shorter than CAU1.) A similar technique is used by ADL [4] in their scheme, GCM-RUP, but for a slight different variant of GCM.

$\underline{\mathsf{CAU2.Enc}(K, N, M, H)}$	$\underline{\mathsf{CAU2.Dec}(K, C_2, H)}$
$C_2 \leftarrow \mathsf{CAU1.Enc}(K, N, 0^\ell \| M, H)$	$\tau \| C \leftarrow C_2$; $h \leftarrow \mathsf{H.Ev}(\mathsf{E.Ev}(K, 0^{\mathsf{E.bl}}), (C, H))$
Return C_2	$y \leftarrow \mathsf{E.In}(K, \tau \oplus h)$; $N \leftarrow y[1..\ell]$
	If $(y[(\ell+1)..\mathsf{E.bl}] \neq \langle 1 \rangle_{\mathsf{E.bl}-\ell})$ then return $\perp$
	$M^* \leftarrow \mathsf{CAU1.Dec}(K, N, C_2, H)$
	If $(M^* = \perp)$ then return $\perp$
	$x \| M \leftarrow M^*$
	If $(x \neq 0^\ell)$ then return $\perp$ else return M

Figure 6: Encryption and decryption algorithms of NBE2 scheme $\mathsf{CAU2} = \mathbf{CAU2}[\mathsf{E}, \mathsf{H}, \ell]$, a special case of which is an AE2$[\mathcal{A}^{\text{ae2}}_{\text{u-n}}]$-secure variant of GCM.

More formally, the transform **CAU2** defines an NBE2 scheme $\mathsf{CAU2} = \mathbf{CAU2}[\mathsf{E}, \mathsf{H}, \ell]$ whose encryption and decryption algorithms are shown in Fig. 6. The parsing in the first and sixth line of CAU2.Dec is such that $|\tau| = \mathsf{E.bl}$ and $|x| = \ell$. If either parsing fails, CAU2.Dec will return $\perp$.

The theorem below demonstrates that CAU2 achieves basic security assuming that E is an sPRP and H is an (ϵ_1, ϵ_2)-AXU function family (as defined in [35, 15, 33, 2] and others). We recall the these two notions of security in Appendix H.

Theorem 7.1 *Let* $\mathsf{CAU2} = \mathbf{CAU2}[\mathsf{E}, \mathsf{H}, \ell]$ *be the NBE2 scheme defined above where* H *is an* (ϵ_1, ϵ_2)*-AXU function family. Let* $A \in \mathcal{A}^{\text{ae2}}_{\text{u-n}}$ *be an adversary making* q_n *calls to its* New *oracle,* q_e *calls to its* Enc *oracle per session and* q_d *calls to its* Dec *oracle per session. The total number of message blocks passed to the encryption oracle by* A *for any single session does not exceed* Q' *and the lengths of* C_2, H *passed to the decryption oracle by* A *do not exceed* ℓ'_1, ℓ_2*, respectively. Let* $Q = Q' + q_e$ *and* $\ell_1 = \ell'_1 + \mathsf{E.bl}$*. Then we can construct adversary* B *such that:*

$$\begin{aligned}\mathbf{Adv}^{\text{ae2}}_{\mathsf{CAU2}}(A) \leq & 2\mathbf{Adv}^{\text{sprp}}_{\mathsf{E}}(B) + q_n(q_e q_d + q_d^2) \cdot \epsilon_1(\ell_1, \ell_2) + q_n(q_d^2 - q_d) \cdot \epsilon_2(\ell_1, \ell_2) \\ & + q_n\left(\frac{2Q^2 + 2Q + q_d^2 + 4q_dQ + 3q_d + 2}{2^{\mathsf{E.bl}+1}} + \frac{q_d^2 + q_d + 2q_e q_d}{2^{\ell+1}}\right)\end{aligned}$$

B makes q_n queries to its New *oracle, no more than $Q+1$ queries to its* Fn *oracle for each user and no more than q_d queries to its* Dec *oracle for each user.*

The proof of the theorem is in Appendix H. Future work can apply the techniques used in recent work to improve upon this bound [15, 38, 29].

CAU2 has some advantages over the schemes obtained through our basic transforms described in Section 5. CAU2 only makes use of the same keys and (often extensively optimized) primitives already existing in CAU1. This allows for code reuse, making it easy for existing hardware and software to adapt. In contrast to the generic transforms, CAU2 has a $(\mathsf{E.bl} - \ell)$-bit ciphertext overhead (for reference, in AES-GCM this is 32-bits), but lower or comparable computational overhead—a single block cipher call in both encryption and decryption.

8 A real-world perspective

In addition to bridging the gap between theory and usage, our framework allows us to formalize some weaknesses of real-world schemes which communicate nonces in the clear.

First, it allows us to formalize an intuitive fact: pathologically chosen nonces cannot be communicated in the clear. It may seem obvious that message or key-dependent nonces violate security but such pathological nonce choices have occurred in the wild. For instance, CakePHP, a web framework, used the key as the nonce [1] when encrypting data. The use of a hash of a message has also been proposed, and subsequently argued as insecure, in an Internet forum [45].

Second, it disallows metadata leakage through the nonce. Implicit nonces with a device specific field, such as those recommended in RFC 5116 [39] enable an adversary to distinguish between different user sessions. Even the "standard" nonce choices are not safe against these adversaries. A counter will allow an adversary distinguish between sessions with high traffic and low traffic, and a randomly chosen nonce can detect devices with poor entropy (RSA public keys were used to a similar end by HDWH [26]).

Finally, we note that our framework can capture more subtle attacks as well. The recent Juniper DualEC incident showed how small implementation choices coupled with surfaced nonces can render a complete system vulnerable. CMGFCG [19] showed how random nonces in the TLS handshake protocol can be used to reconstruct the internal state of a PRNG, leading to key recovery. While this concrete attack was unrelated to authenticated encryption, one can imagine a similar attack on symmetric encryption where the way a nonce is enciphered or embedded in the ciphertext enables an attack on other parts of the encryption scheme. Since AE2-security encompasses the communication of the nonce in-band, it will flag these schemes as insecure.

9 Acknowledgements.

We thank the anonymous reviewers for their feedback and suggestions.

References

[1] CakePHP: Using the IV as the key. `http://www.cryptofails.com/post/70059594911/cakephp-using-the-iv-as-the-key`. Accessed: 2019-02-12. 23

[2] F. Abed, S. R. Fluhrer, C. Forler, E. List, S. Lucks, D. A. McGrew, and J. Wenzel. Pipelineable on-line encryption. In C. Cid and C. Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 205–223. Springer, Heidelberg, Mar. 2015. 22, 44

[3] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha, and K. Yasuda. How to securely release unverified plaintext in authenticated encryption. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 105–125. Springer, Heidelberg, Dec. 2014. 7

[4] T. Ashur, O. Dunkelman, and A. Luykx. Boosting authenticated encryption robustness with minimal modifications. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 3–33. Springer, Heidelberg, Aug. 2017. 6, 21

[5] J. Aumasson, S. Babbage, D. Bernstein, C. Cid, J. Daemen, O. Dunkelman, K. Gaj, S. Gueron, P. Junod, A. Langley, D. McGrew, K. Paterson, B. Preneel, C. Rechberger, V. Rijmen, M. Robshaw, P. Sarkar, P. Schaumont, A. Shamir, and I. Verbauwhede. CHAE: Challenges in authenticated encryption. ECRYPT-CSA D1.1, Revision 1.05, March 2017. `https://chae.cr.yp.to/whitepaper.html`. 3

[6] M. Barbosa and P. Farshim. Indifferentiable authenticated encryption. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 187–220. Springer, Heidelberg, Aug. 2018. 7

[7] G. Barwell, D. P. Martin, E. Oswald, and M. Stam. Authenticated encryption in the face of protocol and side channel leakage. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 693–723. Springer, Heidelberg, Dec. 2017. 7

[8] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Heidelberg, May 2000. 7

[9] M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *37th FOCS*, pages 514–523. IEEE Computer Society Press, Oct. 1996. 8

[10] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, Oct. 1997. 2

[11] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, Dec. 2000. 2, 11

[12] M. Bellare and P. Rogaway. On the construction of variable-input-length ciphers. In L. R. Knudsen, editor, *FSE'99*, volume 1636 of *LNCS*, pages 231–244. Springer, Heidelberg, Mar. 1999. 21

[13] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 317–330. Springer, Heidelberg, Dec. 2000. 2, 6, 20

[14] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 7, 30, 37

[15] M. Bellare and B. Tackmann. The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 247–276. Springer, Heidelberg, Aug. 2016. 10, 21, 22, 44

[16] D. Bernstein. CAESAR call for submissions, final (2014.01.27), 2014. 2, 4

[17] P. Bose, V. T. Hoang, and S. Tessaro. Revisiting AES-GCM-SIV: Multi-user security, faster key derivation, and better bounds. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 468–499. Springer, Heidelberg, Apr. / May 2018. 4, 6, 7, 10, 11

[18] CAESAR Committee. Cryptographic competitions: Caesar call for submissions, final (2014.01.27). `https://competitions.cr.yp.to/caesar-call.html`. Accessed: 2018-07-23. 6

[19] S. Checkoway, J. Maskiewicz, C. Garman, J. Fried, S. Cohney, M. Green, N. Heninger, R.-P. Weinmann, E. Rescorla, and H. Shacham. A systematic analysis of the juniper dual EC

incident. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 468–479. ACM Press, Oct. 2016. 23

[20] Y. Dodis, P. Grubbs, T. Ristenpart, and J. Woodage. Fast message franking: From invisible salamanders to encryptment. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 155–186. Springer, Heidelberg, Aug. 2018. 7

[21] M. Dworkin. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, November 2007. 2, 6, 14, 21

[22] E. Fleischmann, C. Forler, and S. Lucks. McOE: A family of almost foolproof on-line authenticated encryption schemes. In A. Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 196–215. Springer, Heidelberg, Mar. 2012. 7

[23] P. Grubbs, J. Lu, and T. Ristenpart. Message franking via committing authenticated encryption. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 66–97. Springer, Heidelberg, Aug. 2017. 7, 12

[24] S. Gueron, A. Langley, and Y. Lindell. AES-GCM-SIV: Specification and analysis. Cryptology ePrint Archive, Report 2017/168, 2017. `http://eprint.iacr.org/2017/168`. 4, 6, 7

[25] S. Gueron and Y. Lindell. GCM-SIV: Full nonce misuse-resistant authenticated encryption at under one cycle per byte. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 109–119. ACM Press, Oct. 2015. 4, 6

[26] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In *USENIX Security Symposium*, volume 8, page 1, 2012. 23

[27] V. T. Hoang, T. Krovetz, and P. Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 15–44. Springer, Heidelberg, Apr. 2015. 6, 7, 8, 20

[28] V. T. Hoang, R. Reyhanitabar, P. Rogaway, and D. Vizár. Online authenticated-encryption and its nonce-reuse misuse-resistance. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 493–517. Springer, Heidelberg, Aug. 2015. 7

[29] V. T. Hoang, S. Tessaro, and A. Thiruvengadam. The multi-user security of GCM, revisited: Tight bounds for nonce randomization. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 1429–1440. ACM Press, Oct. 2018. 4, 7, 22

[30] T. Iwata, K. Ohashi, and K. Minematsu. Breaking and repairing GCM security proofs. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 31–49. Springer, Heidelberg, Aug. 2012. 4

[31] A. Joux. Authentication failures in NIST version of GCM, 2006. Comments submitted to NIST modes of operation process, `https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/comments/800-38-series-drafts/gcm/joux_comments.pdf`. 5

[32] J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In B. Schneier, editor, *FSE 2000*, volume 1978 of *LNCS*, pages 284–299. Springer, Heidelberg, Apr. 2001. 2

[33] H. Krawczyk. LFSR-based hashing and authentication. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 129–139. Springer, Heidelberg, Aug. 1994. 22, 44

[34] T. Krovetz and P. Rogaway. The software performance of authenticated-encryption modes. In A. Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 306–327. Springer, Heidelberg, Feb. 2011. 2, 4, 15

[35] K. Kurosawa and T. Iwata. TMAC: Two-key CBC MAC. In M. Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 33–49. Springer, Heidelberg, Apr. 2003. 22, 44

[36] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable block ciphers. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, Heidelberg, Aug. 2002. 8

[37] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable block ciphers. *Journal of Cryptology*, 24(3):588–613, July 2011. 8

[38] A. Luykx, B. Mennink, and K. G. Paterson. Analyzing multi-key security degradation. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 575–605. Springer, Heidelberg, Dec. 2017. 7, 22

[39] D. McGrew. An interface and algorithms for authenticated encryption. IETF Network Working Group, RFC 5116, January 2008. 2, 3, 23

[40] D. A. McGrew and J. Viega. The security and performance of the Galois/counter mode (GCM) of operation. In A. Canteaut and K. Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, Heidelberg, Dec. 2004. 2, 4, 6, 21

[41] C. H. Meyer and S. M. Matyas. *CRYPTOGRAPHY: A new dimension in computer data security: A guide for the design and implementation of secure systems*. Wiley, 1982. 15

[42] K. Minematsu. Authenticated encryption with small stretch (or, how to accelerate AERO). In J. K. Liu and R. Steinfeld, editors, *ACISP 16, Part II*, volume 9723 of *LNCS*, pages 347–362. Springer, Heidelberg, July 2016. 7

[43] C. Namprempre, P. Rogaway, and T. Shrimpton. AE5 security notions: Definitions implicit in the CAESAR call. Cryptology ePrint Archive, Report 2013/242, 2013. `http://eprint.iacr.org/2013/242`. 6

[44] C. Namprempre, P. Rogaway, and T. Shrimpton. Reconsidering generic composition. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 257–274. Springer, Heidelberg, May 2014. 4, 5, 6, 20

[45] Reddit. Hash of message as nonce?, 2015. `https://redd.it/3c504m`. 23

[46] P. Rogaway. Authenticated-encryption with associated-data. In V. Atluri, editor, *ACM CCS 2002*, pages 98–107. ACM Press, Nov. 2002. 2, 3, 9, 10, 11, 46

[47] P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, Heidelberg, Dec. 2004. 2

[48] P. Rogaway. Nonce-based symmetric encryption. In B. K. Roy and W. Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 348–359. Springer, Heidelberg, Feb. 2004. 2, 9, 10

[49] P. Rogaway. The evolution of authenticated encryption. Real World Cryptography Workshop, Stanford, January 2013. `https://crypto.stanford.edu/RealWorldCrypto/slides/phil.pdf`. 3

[50] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In M. K. Reiter and P. Samarati, editors, *ACM CCS 2001*, pages 196–205. ACM Press, Nov. 2001. 2, 4, 9, 14, 15

[51] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006. 3, 5, 6, 10, 17, 19

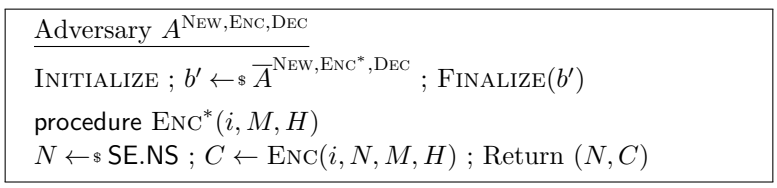
[52] P. Rogaway, M. Wooding, and H. Zhang. The security of ciphertext stealing. In A. Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 180–195. Springer, Heidelberg, Mar. 2012. 35

[53] S. Vaudenay and D. Vizár. Under pressure: Security of caesar candidates beyond their guarantees. Cryptology ePrint Archive, Report 2017/1147, 2017. `https://eprint.iacr.org/2017/1147`. 5

[54] H. Wu and B. Preneel. AEGIS: A fast authenticated encryption algorithm. In T. Lange, K. Lauter, and P. Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 185–201. Springer, Heidelberg, Aug. 2014. 4

A Adversary class $\mathcal{A}^{\mathrm{x}}_{\mathrm{r\text{-}n}}$

Informally, an adversary A is in class $\mathcal{A}^{\mathrm{x}}_{\mathrm{r\text{-}n}}$, where $x \in \{\mathrm{ae1}, \mathrm{ae2}\}$ if the nonces in its queries to Enc are picked uniformly at random, from the (assumed finite) nonce space of the underlying scheme, and independently of each other. (No restriction is placed on the nonces that the adversary submits in Dec queries.) More formally, let A be an adversary attacking scheme SE, where the latter is an NBE1 scheme if $x = \mathrm{ae1}$ and an NBE2 scheme if $x = \mathrm{ae2}$. Then $A \in \mathcal{A}^{\mathrm{x}}_{\mathrm{r\text{-}n}}$ if there is another adversary $\overline{A}$, called the *random-nonce adversary* associated to A, such that A is defined in terms of $\overline{A}$ as follows:



As the above indicates, adversary $\overline{A}$ has oracles New, Enc^*, Dec where New, Dec are as in game $\mathbf{G}^{\mathrm{x}}_{\mathsf{SE}}$, but Enc^*, differently from Enc, takes only i, M, H (no nonce) and returns both a nonce and a ciphertext. (The latter means the random nonces are not "hidden" from $\overline{A}$.)

B Proofs of Theorems 5.1 and 6.1

We prove the following which implies both theorems.

Lemma B.1 *Let* $\mathrm{y} \in \{\text{u-n}, \text{u-nmh}\}$. *Let* $\mathbf{HN1}[\mathsf{SE1}, \mathsf{F}] = \mathsf{SE}_{\mathrm{HN1}}$, *be obtained as in Section 5. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{y}}_{\mathrm{ae2}}$, *making* q_n *queries to its* New *oracle and* q_e *queries per user to its* Enc *oracle, and* q_d *queries per user to its* Dec *oracle, we construct adversaries* $A_1 \in \mathcal{A}^{\mathrm{y}}_{\mathrm{ae1}}$ *and* B *such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN1}}}(A_2) \;\le\; \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B) + \frac{q_n(q_e+q_d)(q_e+q_d-1)}{2^{\mathsf{F.il}+1}} \; . \tag{4}$$

Adversary A_1 *preserves the resources of* A_2. *Adversary* B *makes* q_n *queries to its* New *oracle and* $q_e + q_d$ *queries per user to its* Fn *oracle. Adversary* B *has about the same running time as* A_2.

Proof: We define the games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3, \mathrm{G}_4, \mathrm{G}_5$, and G_6 as specified in Fig. 7 and Fig. 7. Let b_1 be the challenge bit in $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN1}}}(A_2)$, and b'_1 be the bit returned by A_2. Then, by the definition of $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN1}}}(A_2)$ and since $\mathsf{SE}_{\mathrm{HN1}}.\mathsf{CS}(\ell_n, \ell_m, \ell_h) = \{0,1\}^{\mathsf{SE1.nl}+\mathsf{SE1.ccl}(\ell_n,\ell_m,\ell_h)}$, we have that $\Pr[b'_1 = 1 | b_1 = 1] = \Pr[\mathrm{G}_0(A_2)]$ and $\Pr[b'_1 = 1 | b_1 = 0] = \Pr[\mathrm{G}_6(A_2)]$. Therefore, we have $\Pr[\mathrm{G}_0(A_2)] - \Pr[\mathrm{G}_6(A_2)] = \mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN1}}}(A_2)$.

Now we define adversaries A_1 and B as described in Fig. 9, and show below that that the following equations hold:

$$\Pr[\mathrm{G}_0(A_2)] - \Pr[\mathrm{G}_1(A_2)] = \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) \; , \tag{5}$$

$$\Pr[\mathrm{G}_1(A_2)] - \Pr[\mathrm{G}_2(A_2)] = \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B) \; , \tag{6}$$

$$\Pr[\mathrm{G}_2(A_2)] = \Pr[\mathrm{G}_3(A_2)] \; , \tag{7}$$

$$\big|\Pr[\mathrm{G}_3(A_2)] - \Pr[\mathrm{G}_4(A_2)]\big| \le \frac{q_n q_e(q_e-1)}{2^{\mathsf{F.il}+1}} \; , \tag{8}$$

$$\Pr[\mathrm{G}_4(A_2)] = \Pr[\mathrm{G}_5(A_2)] = \Pr[\mathrm{G}_6(A_2)] \; . \tag{9}$$

From here, the theorem statement immediately follows. What remains is to prove equations (5), (6), (7), (8), and (9). We will do this one by one.

The strategy of A_1 is to run A_2 with simulated oracles. These oracles follow the description of the algorithms $\mathsf{SE}_{\mathrm{HN1}}.\mathsf{Enc}$ and $\mathsf{SE}_{\mathrm{HN1}}.\mathsf{Dec}$. Adversary A_1 simulates the enciphering of x, then relies on the oracles Enc_a and Dec_a provided to it to perform the $\mathsf{SE1}$ encryption and decryption. Let b_2 be the challenge bit in $\mathbf{G}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1)$ and b'_2 be the bit returned by A_1. When $b_2 = 1$, encryption and decryption will be performed with $\mathsf{SE1.Enc}$ and $\mathsf{SE1.Dec}$. Therefore, $\Pr[b'_2 = 1 | b_2 = 1] = \Pr[\mathrm{G}_0(A_2)]$. When $b_2 = 0$, encryption will return a random string and decryption will return $\perp$ unless C_1 was returned by some call to Enc_a for the same user and nonce (henceforth, we will refer to this as a "trivial decryption query"). Therefore, $\Pr[b'_2 = 1 | b_2 = 0] = \Pr[\mathrm{G}_1(A_2)]$. This gives us equation (5). Notice that the nonces A_2 tries to encrypt with are also the nonces A_1 will use in its Enc_a calls. Therefore, if $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}$, then $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{u-n}}$.

The strategy of B is also to run A_2 with simulated oracles. It simulates $\mathsf{SE1}$ encryption by selecting random strings and decryption by returning $\perp$ on all non-trivial queries. Adversary B enciphers x using its Fn_b oracle. Let b_* be the challenge bit in $\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}(B)$ and b'_* be the bit returned by B. When $b_* = 1$, enciphering occurs using $\mathsf{F.Ev}$. This means that $\Pr[b'_* = 1 | b_* = 1] = \Pr[\mathrm{G}_1(A_2)]$. When $b_* = 0$, encipherment will behave like a random mapping from $\mathsf{F.D}$ to $\{0,1\}^{\mathsf{F.ol}}$. This means that $\Pr[b'_* = 1 | b_* = 0] = \Pr[\mathrm{G}_2(A_2)]$. This gives us equation (6). We also observe that adversary B makes one Fn_b call during each simulated encryption and each simulated decryption, for a total of $q_e + q_d$ queries to Fn_b.

procedure INITIALIZE // For all games
Return

procedure FINALIZE(b') // For all games
Return ($b' = 1$)

Game G_0

procedure NEW_0
$v \leftarrow v + 1$; $K_v \leftarrow \mathsf{SE}_{\mathrm{HN1}}.\mathsf{KS}$

procedure $\mathrm{ENC}_0(i, N, M, H)$
$C_2 \leftarrow \mathsf{SE}_{\mathrm{HN1}}.\mathsf{Enc}(K_i, N, M, H)$
Return C_2

procedure $\mathrm{DEC}_0(i, C_2, H)$
$M \leftarrow \mathsf{SE}_{\mathrm{HN1}}.\mathsf{Dec}(K_i, C_2, H)$
Return M

Game G_1

procedure NEW_1
$v \leftarrow v + 1$; $K_{\mathsf{E},v} \leftarrow \{0,1\}^{\mathsf{E.kl}}$

procedure $\mathrm{ENC}_1(i, N, M, H)$
$C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.ccl}(|N|,|M|,|H|)}$
$\mathbf{M}[i, N, C_1, H] \leftarrow M$; $x \leftarrow C_1[1..\mathsf{F.il}]$
$P \leftarrow \mathsf{F.Ev}(K_{\mathsf{F}}, x)$; $Y \leftarrow P \oplus N$
$C_2 \leftarrow Y \| C_1$; Return C_2

procedure $\mathrm{DEC}_1(i, C_2, H)$
If ($|C_2| < \mathsf{SE1.nl} + \mathsf{F.il}$) then return $\perp$
$Y \| C_1 \leftarrow C_2$; $x \leftarrow C_1[1..\mathsf{F.il}]$
$P \leftarrow \mathsf{F.Ev}(K_{\mathsf{F}}, x)$; $N \leftarrow P \oplus Y$
$M \leftarrow \mathbf{M}[i, N, C_1, H]$; Return M

Game G_2

procedure NEW_2
Return

procedure $\mathrm{ENC}_2(i, N, M, H)$
$C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.ccl}(|N|,|M|,|H|)}$; $x \leftarrow C_1[1..\mathsf{F.il}]$
$\mathbf{M}[i, N, C_1, H] \leftarrow M$; $P \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.ol}}$
If ($\mathbf{P}[i, x] \neq \perp$) then $P \leftarrow \mathbf{P}[i, x]$
  Else $\mathbf{P}[i, x] \leftarrow P$
$Y \leftarrow P \oplus N$; $C_2 \leftarrow Y \| C_1$; Return C_2

procedure $\mathrm{DEC}_2(i, C_2, H)$
If ($|C_2| < \mathsf{SE1.nl} + \mathsf{F.il}$) then return $\perp$
$Y \| C_1 \leftarrow C_2$; $x \leftarrow C_1[1..\mathsf{F.il}]$; $P \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.ol}}$
If ($\mathbf{P}[i, x] \neq \perp$) then $P \leftarrow \mathbf{P}[i, x]$
  Else $\mathbf{P}[i, x] \leftarrow P$
$N \leftarrow P \oplus Y$; $M \leftarrow \mathbf{M}[i, N, C_1, H]$; Return M

Games $\boxed{\mathrm{G}_3}$,G_4

procedure NEW_3
Return

procedure $\mathrm{ENC}_3(i, N, M, H)$
$C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.ccl}(|N|,|M|,|H|)}$
$x \leftarrow C_1[1..\mathsf{F.il}]$; $P \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.ol}}$
If ($\mathbf{P}[i, x] \neq \perp$) then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{P \leftarrow \mathbf{P}[i, x]}$
  Else $\mathbf{P}[i, x] \leftarrow P$
$Y \leftarrow P \oplus N$; $C_2 \leftarrow Y \| C_1$
$\mathbf{N}[i, C_2, H] \leftarrow M$; Return C_2

procedure $\mathrm{DEC}_3(i, C_2, H)$
Return $\mathbf{N}[i, C_2, H]$

Figure 7: Games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3, \mathrm{G}_4$ used in Lemma B.1. INITIALIZE, FINALIZE are common to all games.

We argue that A_2 cannot distinguish interacting with ENC_2 and DEC_2 from ENC_3 and DEC_3.[1] (Here, ENC_3 includes the boxed code.) Suppose A_2 makes an encryption query. Notice that ENC_2 and ENC_3 differ only in whether it maintains $\mathbf{M}$ or $\mathbf{N}$ (and the setting of the flag bad in ENC_3, which does not affect the game at all). Since neither encryption algorithm reads from their respective table, their outputs are indistinguishable. Suppose the adversary makes a decryption query $\mathrm{DEC}_3(i, C_2, H)$ and receives the output $M \neq \perp$. This means that if we let $Y \| C_1 \leftarrow C_2$, there was some call $\mathrm{ENC}_3(i, N, M, H)$ such that C_1 was the randomly selected string and Y is the output of masking the enciphered $C_1[1..\mathsf{F.il}]$ with N. Recall that the encipherment is a random mapping from $\mathsf{F.D}$ to $\{0,1\}^{\mathsf{F.ol}}$, as discussed above. This means that, if the adversary had instead been interacting

[1] In fact, the games make exactly the same random choices, and for the same randomness they always return the same result.

Game G_5	Game G_6
procedure $\textsc{New}_5$ Return	**procedure** $\textsc{New}_6$ Return
procedure $\textsc{Enc}_5(i, N, M, H)$ $C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.ccl}(\lvert N\rvert,\lvert M\rvert,\lvert H\rvert)}$ $x \leftarrow C_1[1..\mathsf{F.il}]$; $Y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.ol}}$ $C_2 \leftarrow Y \Vert C_1$; $\mathbf{N}[i, C_2, H] \leftarrow M$; Return C_2	**procedure** $\textsc{Enc}_6(i, N, M, H)$ $C_2 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.nl}+\mathsf{SE1.ccl}(\lvert N\rvert,\lvert M\rvert,\lvert H\rvert)}$ $\mathbf{N}[i, C_2, H] \leftarrow M$; Return C_2
procedure $\textsc{Dec}_5(i, C_2, H)$ Return $\mathbf{N}[i, C_2, H]$	**procedure** $\textsc{Dec}_6(i, C_2, H)$ Return $\mathbf{N}[i, C_2, H]$

Figure 8: Games $\mathrm{G}_5, \mathrm{G}_6$ used in Lemma B.1. Initialize, Finalize are as in Fig. 7.

Adversary $A_1^{\textsc{New}_a, \textsc{Enc}_a, \textsc{Dec}_a}$	Adversary $B^{\textsc{New}_b, \textsc{Fn}_b}$
Initialize ; $b' \leftarrow\!\!\$\ A_2^{\textsc{New}_a^*, \textsc{Enc}_a^*, \textsc{Dec}_a^*}$ Finalize(b')	Initialize ; $b' \leftarrow\!\!\$\ A_2^{\textsc{New}_b, \textsc{Enc}_b^*, \textsc{Dec}_b^*}$ Finalize(b')
procedure $\textsc{New}_a^*$ $v \leftarrow v + 1$; $k_v \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$; $\textsc{New}_b$	**procedure** $\textsc{Enc}_b^*(i, N, M, H)$ $C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.ccl}(\lvert N\rvert,\lvert M\rvert,\lvert H\rvert)}$ $\mathbf{M}[i, N, C_1, H] \leftarrow M$; $x \leftarrow C_1[1..\mathsf{F.il}]$ $P \leftarrow\!\!\$\ \textsc{Fn}_b(i, x)$; $Y \leftarrow P \oplus N$ $C_2 \leftarrow Y \Vert C_1$; Return C_2
procedure $\textsc{Enc}_a^*(i, N, M, H)$ $C_1 \leftarrow\!\!\$\ \textsc{Enc}_a(i, N, M, H)$ $x \leftarrow C_1[1..\mathsf{F.il}]$; $Y \leftarrow N \oplus \mathsf{F.Ev}(k_i, x)$ $C_2 \leftarrow Y \Vert C_1$; Return C_2	**procedure** $\textsc{Dec}_b^*(i, C_2, H)$ If $(\lvert C_2\rvert < \mathsf{SE1.nl} + \mathsf{F.il})$ then return $\perp$ $Y \Vert C_1 \leftarrow C_2$; $x \leftarrow C_1[1..\mathsf{F.il}]$ $P \leftarrow\!\!\$\ \textsc{Fn}_b(i, x)$; $N \leftarrow P \oplus Y$ Return $\mathbf{M}[i, N, C_1, H]$
procedure $\textsc{Dec}_a^*(i, C_2, H)$ If $(\lvert C_2\rvert < \mathsf{SE1.nl} + \mathsf{F.il})$ then return $\perp$ $Y \Vert C_1 \leftarrow C_2$; $x \leftarrow C_1[1..\mathsf{F.il}]$ $N \leftarrow Y \oplus \mathsf{F.Ev}(k_i, x)$ $M \leftarrow \textsc{Dec}_a(i, N, C_1, H)$; Return M	

Figure 9: Adversaries A_1, B used in proving Lemma B.1.

with $\textsc{Enc}_2$, $\mathbf{M}[i, N, C_1, H]$ was initialized with M and $M \leftarrow \textsc{Dec}_2(i, C_2, H)$. Now consider that there was a call $\textsc{Dec}_3(i, C_2, H)$ which returned $\perp$. This means that no call to $\textsc{Enc}_3$ resulted in $\mathbf{N}[i, C_2, H]$ being initialized. This means that if we unmasked Y using the encipherment of $C_1[1..\mathsf{F.il}]$ to get N, there would be no equivalent call to $\textsc{Enc}_2(i, N, M, H)$ resulting in $\mathbf{M}[i, N, C_1, H]$ being initialized. Therefore, $\textsc{Dec}_2(i, C_2, H)$ would also have returned $\perp$. From this, we obtain (7).

Notice that G_3 and G_4 are identical-until-bad. Using the fundamental lemma of game playing [14], we have $\left|\Pr[\mathrm{G}_3(A_2)] - \Pr[\mathrm{G}_4(A_2)]\right| \leq \Pr[\mathrm{G}_3(A_2) \text{ sets } \mathsf{bad}]$. Since x is a substring of C_1, which is a uniformly random string, x is also uniformly random and we use the collision probability and the union bound to arrive at (8).

Now we look at $\textsc{Enc}_3$ in G_4 (which does not include the boxed code) and note that P is always a randomly selected element of $\{0,1\}^{\mathsf{F.ol}}$. This value of P is used to mask Y. This is therefore indistinguishable from randomly selecting Y from $\{0,1\}^{\mathsf{F.ol}}$ directly. This means that

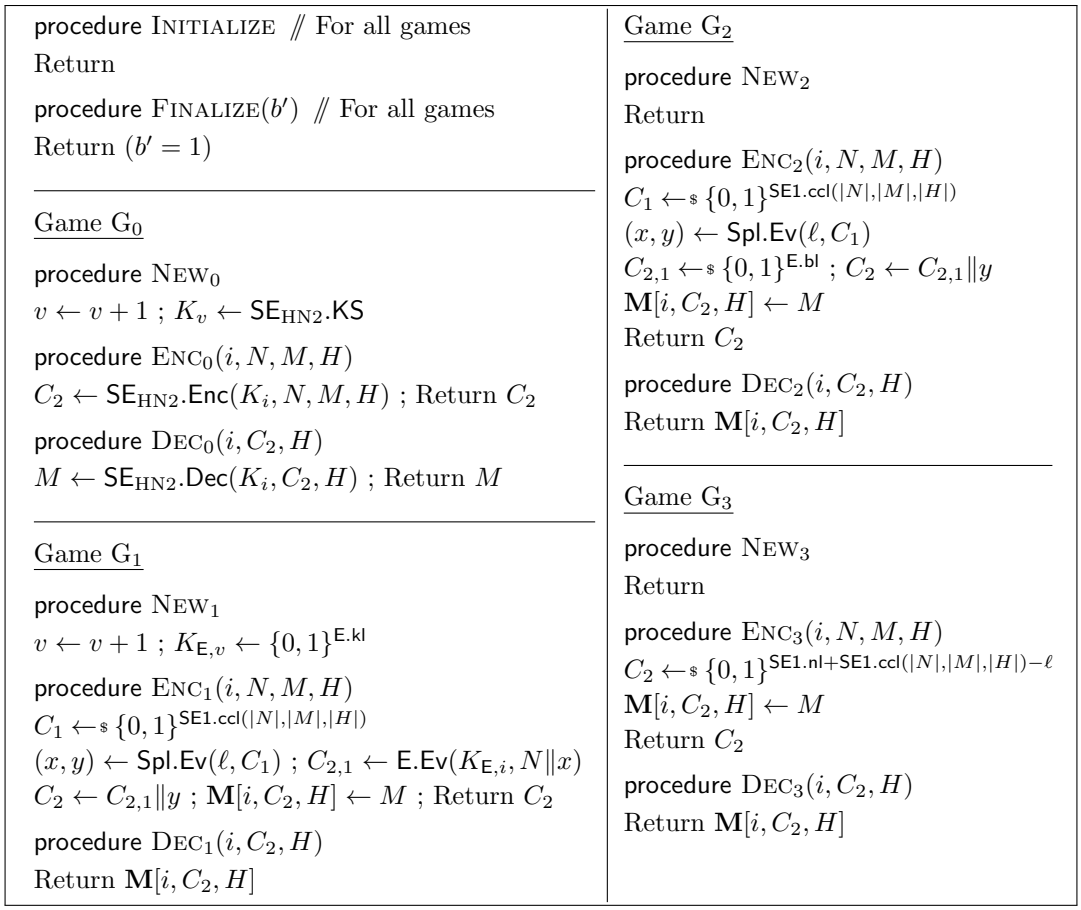


Figure 10: Games G_0, G_1, G_2, and G_3 used in proving Theorem 5.2. INITIALIZE and FINALIZE are common to all games.

$\Pr[G_4(A_2)] = \Pr[G_5(A_2)]$. Finally, since ENC$_5$ sets Y, C_1 to be random strings of lengths $\mathsf{SE1.nl}$ and $\mathsf{SE1.ccl}(|N|, |M|, |H|)$, respectively, this is equivalent to picking C_2 to be a random string of their combined length. This is what happens in ENC$_6$. Therefore, $\Pr[G_5(A)] = \Pr[G_6(A)]$. This gives (9) and concludes the proof. ▌

C Proof of Theorem 5.2

Theorem 5.2. *Let* $\mathsf{SE}_{\mathrm{HN2}} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ *be obtained as in Section 5. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}$*, making* q_n *queries to its* NEW *oracle,* q_e *queries per user to its* ENC *oracle, and* q_d *queries per user to its* DEC *oracle, we construct adversaries* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\mathrm{u\text{-}n}}$ *and* B *such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_2) \leq \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{E}}(B) \ . \tag{10}$$

Adversary A_1 *preserves the resources of* A_2*. Adversary* B *makes* q_n *queries to its* NEW *oracle and* q_e *queries per user to its* FN *oracle. Adversary* B *has about the same running time as* A_2*.*

Proof:

We define games G_0, G_1, G_2, and G_3 as in Fig. 10. Let b_1 be the challenge bit in the game

Adversary $A_1^{\text{New}_a,\text{Enc}_a,\text{Dec}_a}$	Adversary $B^{\text{New}_b,\text{Fn}_b}$
Initialize ; $b' \leftarrow\!\!\$\ A_2^{\text{New}_a^*,\text{Enc}_a^*,\text{Dec}_a^*}$; Finalize(b')	Initialize
procedure New_a^*	$b' \leftarrow\!\!\$\ A_2^{\text{New}_b,\text{Enc}_b^*,\text{Dec}_b^*}$
$v \leftarrow v+1$; $k_v \leftarrow\!\!\$\ \{0,1\}^{\mathsf{E.kl}}$; New_a	Finalize(b')
procedure $\text{Enc}_a^*(i,N,M,H)$	procedure $\text{Enc}_b^*(i,N,M,H)$
$C_1^* \leftarrow\!\!\$\ \text{Enc}_a(i,N,M,H)$; $(x,y) \leftarrow \mathsf{Spl.Ev}(\ell, C_1)$	$C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.ccl}(\lvert N\rvert,\lvert M\rvert,\lvert H\rvert)}$
$C_{2,1} \leftarrow \mathsf{E.Ev}(k_i, N\|x)$; $C_2 \leftarrow C_{2,1}\|y$; Return C_2	$(x,y) \leftarrow \mathsf{Spl.Ev}(\ell, C_1)$
procedure $\text{Dec}_a^*(i,C_2,H)$	$C_{2,1} \leftarrow \text{Fn}_b(i, N\|x)$
If ($\lvert C_2\rvert < \mathsf{E.bl}$) then return $\perp$	$C_2 \leftarrow C_{2,1}\|y$
$N\|x \leftarrow \mathsf{E.In}(k_i, C_2[1..\mathsf{E.bl}])$; $y \leftarrow C_2[(\mathsf{E.bl}+1)..\lvert C_2\rvert]$	$\mathbf{M}[i,C_2,H] \leftarrow M$; Return C_2
$C_1 \leftarrow \mathsf{Spl.In}(x,y)$; If ($C_1 = \perp$) then return $\perp$	procedure $\text{Dec}_b^*(i,C_2,H)$
$M \leftarrow \text{Dec}_a(i,N,C_1,H)$; Return M	Return $\mathbf{M}[i,C_2,H]$

Figure 11: Adversaries A_1, B used in proving Theorem 5.2.

$\mathbf{G}^{\text{ae2}}_{\mathsf{SE}_{\text{HN2}}}(A_2)$ and b'_1 be the bit returned by A_2. Then, by the definition of $\mathbf{G}^{\text{ae2}}_{\mathsf{SE}_{\text{HN2}}}(A_2)$ and since $\mathsf{SE}_{\text{HN2}}.\mathsf{CS}(\ell_n,\ell_m,\ell_h) = \{0,1\}^{\mathsf{SE1.nl}+\mathsf{SE1.ccl}(\ell_n,\ell_m,\ell_h)}$, we have that $\mathbf{Adv}^{\text{ae2}}_{\mathsf{SE}_{\text{HN2}}}(A_2) = \Pr[b'_1 = 1 | b_1 = 1] - \Pr[b'_1 = 1 | b_1 = 0] = \Pr[\text{G}_0(A_2)] - \Pr[\text{G}_3(A_2)]$.

Now we define adversaries A_1, B as specified in Fig. 11. This is such that the following equations hold:

$$\Pr[\text{G}_0(A_2)] - \Pr[\text{G}_1(A_2)] = \mathbf{Adv}^{\text{ae1}}_{\mathsf{SE1}}(A_1)\ , \tag{11}$$

$$\Pr[\text{G}_1(A_2)] - \Pr[\text{G}_2(A_2)] = \mathbf{Adv}^{\text{prf}}_{\mathsf{E}}(B)\ , \tag{12}$$

$$\Pr[\text{G}_2(A_2)] = \Pr[\text{G}_3(A_2)]\ . \tag{13}$$

From here, we can immediately derive the theorem statement. What remains to be shown is to derive equations (11), (12), and (13). We will do this in the following, one by one.

The general strategy of A_1 is to run A_2 with simulated oracles. Intuitively, the adversary performs all the computation associated to splitting, ciphertext stealing and nonce enciphering (using E), but uses its Enc_a and Dec_a oracles to simulate $\mathsf{SE1}$ encryption and decryption. Let b_2 be the challenge bit in $\mathbf{G}^{\text{ae1}}_{\mathsf{SE1}}(A_1)$ and b'_2 be the bit returned by A_1. When $b_2 = 0$, the encryption oracle will return random strings of the appropriate length and the decryption oracle will return $\perp$ except on trivial queries (i.e. when the adversary decrypts a ciphertext returned by the encryption oracle, using the same header and nonce, if any). This means that the outputs of $\text{Enc}_0, \text{Enc}_1$ returned to A_2 are the same in games G_0, G_1 and in the cases where $b_2 = 0, 1$ in $\mathbf{G}^{\text{ae1}}_{\mathsf{SE1}}(A_1)$. Notice that if $\mathbf{M}[i,C_2,H]$ in $\text{G}_1(A_2)$ is initialized with some value M, then $\mathbf{M}[i,N,C_1,H]$ in $\mathbf{G}^{\text{ae1}}_{\mathsf{SE1}}$ will be set to M. Therefore, Dec_1 is equivalent to Dec_a^* when $b_2 = 0$ and $\Pr[b'_2 = 1 | b_2 = 0] = \Pr[\text{G}_1(A_2)]$. By the definition of $\mathsf{SE}_{\text{HN2}}.\mathsf{Enc}$ and $\mathsf{SE}_{\text{HN2}}.\mathsf{Dec}$, we have that $\Pr[b'_2 = 1 | b_2 = 1] = \Pr[\text{G}_0(A_2)]$. Subtracting yields (11).

Note that if A_2 does not repeat a nonce within a session, then A_1 will not repeat a nonce either. Therefore, we know that $A_1 \in \mathcal{A}^{\text{ae1}}_{\text{u-n}}$. In addition, since each call to a simulated oracle (e.g. $\text{Enc}_a^*, \text{Dec}_a^*$) requires a single call to the respective oracle provided to A_1 (e.g. $\text{Enc}_a, \text{Dec}_a$), we know that A_1 preserves the resources of A_2.

B also runs A_2, but simulates oracles via a different strategy: It performs all computation associated

to splitting and ciphertext stealing as before, using the FN_b oracle provided to encipher nonces, and picks $\mathsf{SE1}$ core ciphertexts at random. Decryption always returns $\perp$ unless the ciphertext provided was returned by the ENC_b^* oracle. Let b_* be the challenge bit in $\mathbf{G}_{\mathsf{E}}^{\text{prf}}(B)$ and b'_* be the bit returned by B. Then, when $b_* = 1$, the enciphering is done using E and $\Pr[b'_* = 1 | b_* = 1] = \Pr[\text{G}_1(A_2)]$. When $b_* = 0$, the FN_b oracle will return random elements of $\{0,1\}^{\mathsf{E.bl}}$ and $\Pr[b'_* = 1 | b_* = 0] = \Pr[\text{G}_2(A_2)]$. Subtracting yields (12). Note that the NEW_b oracle is provided to A_2, and that each call to ENC_b^* requires a single call to FN_b. In this way, B makes q_n queries to its NEW oracle and q_e queries per user to its FN oracle.

Finally, we want to show that G_2 and G_3 return true with equal probability. In ENC_2, we set $C_{2,1}$ to be a random string of length $\mathsf{E.bl}$. In addition, by the definition of Spl, we have that $y = C_2[1..s-1] \| C_1[(s+\ell)..|C_1|]$, meaning that y is a random string of length $|C_1| - \ell$. Therefore, in ENC_2, C_2 is a random string of length $|C_1| + \mathsf{SE1.nl}$. This is exactly how C_2 is selected in ENC_3. From this, we have (13). ∎

D Proof of Theorem 5.3

Theorem 5.3. *Let* $\mathsf{SE}_{\text{HN3}} = \mathbf{HN3}[\mathsf{SE1}, \mathsf{F}]$ *be obtained as in Section 5. Then, given adversary* $A_2 \in \mathcal{A}^{\text{ae2}}_{\text{u-n}}$ *that makes* q_n *queries to its* NEW *oracle,* q_e *queries to its* ENC *oracle, and* q_d *queries to its* DEC *oracle, we construct adversaries* $A_1 \in \mathcal{A}^{\text{ae1}}_{\text{r-n}}$ *and* B *such that*

$$\mathbf{Adv}^{\text{ae2}}_{\mathsf{SE}_{\text{HN3}}}(A_2) \;\le\; \mathbf{Adv}^{\text{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\text{prf}}_{\mathsf{F}}(B)\ .$$

Adversary A_1 *preserves the resources of* A_2*. Adversary* B *makes* q_n *queries to its* NEW *oracle and* q_e *queries to its* FN *oracle, respectively. Adversary* B *has about the same running time as* A_2*.*

Proof: We define the games G_0, G_1, and G_2 as in Fig. 12. All games share the same INITIALIZE and FINALIZE procedures, listed at the top right of Fig. 12. Let b_1 be the challenge bit in $\mathbf{G}^{\text{ae2}}_{\mathsf{SE}_{\text{HN3}}}(A_2)$ and b'_1 be the bit returned by A_2. Then, by the definition of the game $\mathbf{G}^{\text{ae2}}_{\mathsf{SE}_{\text{HN3}}}(A_2)$ and since $\mathsf{SE}_{\text{HN3}}.\mathsf{CS}(\ell_n, \ell_m, \ell_h) = \{0,1\}^{\mathsf{SE1.nl}+\mathsf{SE1.ccl}(\ell_n,\ell_m,\ell_h)}$, we have that $\mathbf{Adv}^{\text{ae2}}_{\mathsf{SE}_{\text{HN3}}}(A_2) = \Pr[b'_1 = 1 | b_1 = 1] - \Pr[b'_1 = 1 | b_1 = 0] = \Pr[\text{G}_0(A_2)] - \Pr[\text{G}_2(A_2)]$.

We define adversaries $\overline{A}_2$ and B as specified in Fig. 12. Recall that $A_1 \in \mathcal{A}^{\text{ae1}}_{\text{r-n}}$ can be expressed as in Fig. **??**, discussed in Appendix A, with random nonce adversary $\overline{A}_2$ and NBE1 scheme $\mathsf{SE1}$. This is such that the following equations hold:

$$\Pr[\text{G}_0(A_2)] - \Pr[\text{G}_1(A_2)] = \mathbf{Adv}^{\text{ae1}}_{\mathsf{SE1}}(A_1)\ , \tag{14}$$

$$\Pr[\text{G}_1(A_2)] - \Pr[\text{G}_2(A_2)] = \mathbf{Adv}^{\text{prf}}_{\mathsf{F}}(B)\ . \tag{15}$$

From here, we can immediately derive the theorem statement. What remains to be done is to derive equations (14) and (15). We will do this one by one.

The general strategy of B is to run A_2 with simulated oracles. Intuitively, the adversary uses the FN_b oracle to get a "derived nonce" then encrypts and decrypts as stipulated in $\mathsf{SE1}$'s algorithms. Let b_* be the challenge bit in $\mathbf{G}^{\text{prf}}_{\mathsf{F}}(B)$ and b'_* be the bit returned by B. When $b_* = 1$, encryption will proceed with a nonce derived using $\mathsf{F.Ev}$, and when $b_* = 0$, it will proceed with a randomly chosen nonce. Therefore, we have that $\Pr[b'_* = 1 | b_* = 1] = \Pr[\text{G}_0(A_2)]$ and $\Pr[b'_* = 1 | b_* = 0] = \Pr[\text{G}_1(A_2)]$. Subtracting yields (14).

$\overline{A}_2$ is a random nonce adversary since it does not provide a nonce to the encryption oracle. This means that $A_1 \in \mathcal{A}^{\text{ae1}}_{\text{r-n}}$. It will also run A_2 with simulated oracles. For encryption, it ignores the

procedure INITIALIZE // For all games Return procedure FINALIZE(b') // For all games Return $(b' = 1)$	Game G_1 procedure NEW_1 $v \leftarrow v + 1$; $K_v \leftarrow\!\!\$\ \mathsf{SE1.KS}$ procedure $\text{ENC}_1(i, N, M, H)$ $N_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.nl}}$ $C_1 \leftarrow \mathsf{SE1.Enc}(K_i, N_1, M, H)$; Return $N_1 \| C_1$ procedure $\text{DEC}_1(i, C_2, H)$ $N_1 \| C_1 \leftarrow C_2$; $M \leftarrow \mathsf{SE1.Dec}(K_i, N_1, C_1, H)$ Return M
Game G_0 procedure NEW_0 $v \leftarrow v + 1$ $K_v \leftarrow\!\!\$\ \mathsf{SE}_{\text{HN3}}.\mathsf{KS}$ procedure $\text{ENC}_0(i, N, M, H)$ $C_2 \leftarrow \mathsf{SE}_{\text{HN3}}.\mathsf{Enc}(K_i, N, M, H)$ Return C_2 procedure $\text{DEC}_0(i, C_2, H)$ $M \leftarrow \mathsf{SE}_{\text{HN3}}.\mathsf{Dec}(K_i, C_2, H)$ Return M	Game G_2 procedure NEW_2 Return procedure $\text{ENC}_2(i, N, M, H)$ $C_2 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.nl}+\mathsf{SE1.ccl}(\ell_n,\ell_m,\ell_h)}$ $\mathbf{M}[i, C_2, H] \leftarrow M$; Return C_2 procedure $\text{DEC}_2(i, C_2, H)$ Return $\mathbf{M}[i, C_2, H]$

Adversary $\overline{A}_2^{\text{NEW}_a, \text{ENC}_a, \text{DEC}_a}$ INITIALIZE ; $b' \leftarrow\!\!\$\ A_2^{\text{NEW}_a, \text{ENC}_a^*, \text{DEC}_a^*}$ FINALIZE(b') procedure $\text{ENC}_a^*(i, N, M, H)$ $(N_1, C_1) \leftarrow\!\!\$\ \text{ENC}_a(i, M, H)$ $C_2 \leftarrow N_1 \| C_1$; Return C_2 procedure $\text{DEC}_a^*(i, C_2, H)$ $N_1 \| C_1 \leftarrow C_2$ $M \leftarrow \text{DEC}_b(i, N_1, C_1, H)$ Return M	Adversary $B^{\text{NEW}_b, \text{FN}_b}$ INITIALIZE ; $b' \leftarrow\!\!\$\ A_2^{\text{NEW}_b^*, \text{ENC}_b^*, \text{DEC}_b^*}$; FINALIZE(b') procedure NEW_b^* $v \leftarrow v + 1$; NEW ; $K_v \leftarrow\!\!\$\ \mathsf{SE1.KS}$ procedure $\text{ENC}_b^*(i, N, M, H)$ $N_1 \leftarrow \text{FN}_b(i, N)$; $C_1 \leftarrow \mathsf{SE1.Enc}(K_i, N_1, M, H)$ $C_2 \leftarrow N_1 \| C_1$; Return C_2 procedure $\text{DEC}_b^*(i, C_2, H)$ $N_1 \| C_1 \leftarrow C_2$; $M \leftarrow \mathsf{SE1.Dec}(K_i, N_1, C_1, H)$ Return M

Figure 12: Top: Games used in proving Theorem 5.3. INITIALIZE, FINALIZE are common to all games. Bottom: Adversaries used in proving Theorem 5.3.

nonce provided by A_2 and passes the remaining arguments to ENC_a, the encryption oracle provided to the random nonce adversary. For decryption, it parses the ciphertext into core ciphertext C_1 and nonce N_1, and passes them to DEC_a (along with i and H). Let b_2 be the challenge bit in $\mathbf{G}^{\text{ae1}}_{\mathsf{SE1}}(A_1)$ and b'_2 be the bit returned by A_1. When $b_2 = 1$, encryption will use the encryption algorithm $\mathsf{SE1.Enc}$ and randomly chosen nonces. When $b_2 = 0$, both N_1 and the core ciphertext C_1 will be random selected from $\{0,1\}^{\mathsf{SE1.nl}}$ and $\{0,1\}^{\mathsf{SE1.ccl}(|N|,|M|,|H|)}$, respectively. This means that $\Pr[b'_2 = 1 | b_2 = 1] = \Pr[\text{G}_1(A_2)]$ and $\Pr[b'_2 = 1 | b_2 = 0] = \Pr[\text{G}_2(A_2)]$. Subtracting yields (15). ▌

E Theorem 6.2 is tight

Let $\mathsf{SE1}$ be an NBE1 scheme, Spl be a splitting scheme and $\ell \in \mathbb{N}$ be such that $\ell \leq \mathsf{SE1.mccl}$. Let ℓ_m, ℓ_h be such that $\{0,1\}^{\ell_m} \subset \mathsf{SE1.MS}$ and $\{0,1\}^{\ell_h} \subset \mathsf{SE1.HS}$. (Note that we know that some ℓ_m, ℓ_h must exist since we assumed $\mathsf{SE1.MS}, \mathsf{SE1.HS}$ were length-closed.) We define collision probability $\mathbf{CP}_{\mathsf{SE1}}(\ell_m, \ell_h, q)$, for all k, q where $1 < k \leq |\mathsf{SE1.KS}|$ and $0 < q < 2^{\ell_m}$, as the probability that the following returns true.

For $i = 1, 2 \ldots k$ do
  $K \leftarrow\!\!\$\ \mathsf{SE1.KS}$
  For $j = 1, 2 \ldots q$ do
    $C_1 \leftarrow \mathsf{SE1.Enc}(K, 0^{\mathsf{SE1.nl}}, \langle j \rangle_{\ell_m}, 0^{\ell_h})$; $(x, y) \leftarrow \mathsf{Spl.Ev}(\ell, C_1)$
    If $(x \in S_i)$ then return true else $S_i \leftarrow S_i \cup \{x\}$
Return false

We discussed briefly in Section 6 that the **HN2** transform does not, in general, provide advanced security. We formalize this statement by providing a nonce-misusing adversary, then show that it achieves a high advantage when ℓ is small. This attack generalizes the attack provided by RWZ on CTR-mode with Meyer-Matyas ciphertext stealing [52].

Proposition E.1 *Let $\mathsf{SE}_{\mathrm{HN2}} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ be obtained as in Section 5. Let ℓ_m, ℓ_h be as described above. Then, we can construct an adversary $A_{\ell_m,\ell_h} \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}$ such that:*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_{\ell_m,\ell_h}) \geq \mathbf{CP}_{\mathsf{SE1}}(\ell_m, \ell_h, q_n, q_e) - \frac{q_n q_e (q_e - 1)}{2^{\mathsf{E.bl}+1}}$$

A_{ℓ_m,ℓ_h} is a very-low-resource adversary making q_n queries to its New *oracle, q_e queries per user to its* Enc *oracle and no* Dec *queries.*

Adversary $A^{\text{New},\text{Enc},\text{Dec}}_{\ell_m,\ell_h}$

Initialize
For $i = 1, 2 \ldots q_n$ do
  New
  For $j = 1, 2 \ldots q$ do
    $C_{i,j} \leftarrow \text{Enc}(i, 0^{\mathsf{SE1.nl}}, \langle j \rangle_{\ell_m}, 0^h)$; $z_{i,j} \leftarrow C_{i,j}[1..\mathsf{E.bl}]$
    If $(z_{i,j} \in S_i)$ then $b' \leftarrow 1$ else $S_i \leftarrow S_i \cup \{z_{i,j}\}$
Finalize(b')

Proof: A_{ℓ_m,ℓ_h} will return 1 when $x_{i,j} = x_{i,j'}$ for some i, j, j'. Let b be the challenge bit and b' be the bit returned in $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_{\ell_m,\ell_h})$. When $b = 1$, this means that the output of the block cipher call in $\mathsf{SE}_{\mathrm{HN2}}.\mathsf{Enc}$ during the encryption of $M_{i,j}$ and $M_{i,j'}$ were the same. This, in turn, means that the first elements of the outputs of the splitting function was the same. Therefore, $\Pr[b' = 1 | b = 1] = \mathbf{CP}_{\mathsf{SE1}}(\ell_m, \ell_h, q_n, q_e)$. When $b = 0$, all the $C_{i,j}$ will be random strings. Using the birthday collision probability and the union bound, this means that $\Pr[b' = 1 | b = 0] \leq \frac{q_n q_e (q_e - 1)}{2^{\mathsf{E.bl}+1}}$. Therefore, $\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_{\ell_m,\ell_h}) \geq \mathbf{CP}_{\mathsf{SE1}}(\ell_m, \ell_h, q_n, q_e) - \frac{q_n q_e (q_e - 1)}{2^{\mathsf{E.bl}+1}}$ ∎

Next, we show that if $\mathsf{SE1}$ is $\mathrm{AE1}[\mathcal{A}^{\mathrm{ae1}}_{\text{u-nmh}}]$-secure, then the collision probability can only be high if ℓ is small.

Proposition E.2 *Let* $\mathsf{SE1}, \ell_m, \ell_h, \ell$ *be as defined above. We can construct an adversary* A_1 *such that:*

$$\mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) \geq \mathbf{CP}_{\mathsf{SE1}}(\ell_m, \ell_h, q_n, q_e) - \frac{q_n q_e(q_e-1)}{2^{\ell+1}}$$

A is a very-low-resource adversary making q_n *queries to its* NEW *oracle and* q_e *queries per user to its* ENC *oracle, . .* A_1 *makes no* DEC *queries.*

Adversary $A_1^{\text{NEW,ENC,DEC}}$

INITIALIZE
For $i = 1, 2 \ldots q_n$ do
  NEW
  For $j = 1, 2 \ldots q$ do
    $C_{i,j} \leftarrow \text{ENC}(i, 0^{\mathsf{SE1.nl}}, \langle j \rangle_{\ell_m}, 0^h)$; $(x_{i,j}, y) \leftarrow \mathsf{Spl}(\ell, C_{i,j})$
    If $(x_{i,j} \in S_i)$ then $\_ \leftarrow 1$ else $S_i \leftarrow S_i \cup \{x_{i,j}\}$
FINALIZE(b')

Proof: As with A_{ℓ_m,ℓ_h}, adversary A_1 will return 1 only if $x_{i,j} = x_{i,j'}$ for some i, j, j'. Let b be the challenge bit and b' be the bit returned in $\mathbf{G}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1)$. When $b = 1$, the probability that $b' = 1$ is exactly the collision probability. This is because A_{ℓ_m,ℓ_h} keeps the nonce constant and $z_{i,j}$ is derived from the nonce and $x_{i,j}$. So a collision in the $x_{i,j}$ terms occurs if and only if there is a collision in the $z_{i,j}$ terms. Therefore, $\Pr[b' = 1 | b = 1] = \mathbf{CP}_{\mathsf{SE1}}(\ell_m, \ell_h, q_n, q_e)$. Recall that $x_{i,j}$ is a substring of length ℓ of $C_{i,j}$. Therefore, when $b = 0$, this is a random string. Therefore, using a birthday collision probability and the union bound, $\Pr[b' = 1 | b = 0] \leq \frac{q_n q_e(q_e-1)}{2^{\ell+1}}$. From this, we have $\mathbf{CP}_{\mathsf{SE1}}(\ell_m, \ell_h, q_n, q_e) - \frac{q_n q_e(q_e-1)}{2^{\ell+1}} \leq \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1)$. ∎

Collectively, these two propositions show us that if $\mathsf{SE1}$ is AE1$[\mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}]$-secure and $\mathsf{E.bl}$ is non-trivial, then $\mathbf{CP}_{\mathsf{SE1}}(\ell_m, \ell_h, q_n, q_e) \approx \frac{q_n q_e(q_e-1)}{2^{\ell+1}}$ will be the dominant term in $\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_{\ell_m,\ell_h})$, meaning that $\mathsf{SE}_{\mathrm{HN2}}$ does not achieve AE2-security in the nonce-misuse setting if ℓ is small.

Now we turn to proving Theorem 6.2. Notice that the last term in the bound in Theorem 6.2 is equal to the dominant term in $\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_{\ell_m,\ell_h})$ (under the assumptions discussed above). This means that the bound below is tight.

Theorem 6.2. *Let* $\mathsf{SE}_{\mathrm{HN2}} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ *be obtained as above. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}$*, making* q_n *queries to its* NEW *oracle and* q_e *queries per user to its* ENC *oracle, we construct adversaries* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{u-nmh}}$ *and* B *such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_2) \leq \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B) + \frac{q_n q_e(q_e-1)}{2^{\ell+1}} \ . \tag{16}$$

Adversary A_1 *preserves the resources of* A_2*. Adversary* B *makes* q_n *queries to its* NEW *oracle and* q_e *queries per user to its* FN *oracle. Adversary* B *has about the same running time as* A_2*.*

Proof: Our proof is uses some of the same logic as the proof of Theorem 5.2. We define the games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3$, and G_4 as in Fig. 13. Games G_0 and G_4 are identical, respectively, to games G_0 and G_3 in Fig. 10 used in the proof of Theorem 5.2. As such, we have that $\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_2) = \Pr[\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_2) | b_1 = 1] - \Pr[\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN2}}}(A_2) | b_1 = 0] = \Pr[\mathrm{G}_0(A_2)] - \Pr[\mathrm{G}_4(A_2)]$.

Additionally, the adversaries A_1, B are exactly the same as those used in the proof of Theorem 5.2 (Fig. 11) and $\mathrm{G}_1, \mathrm{G}_3, \mathrm{G}_4$ from Fig. 13 are equivalent, respectively, to $\mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3$ used in the proof

of Theorem 5.2 (see Fig. 10). Therefore, by similar logic as was presented in Theorem 5.2,

$$\Pr[\mathrm{G}_0(A_2)] - \Pr[\mathrm{G}_1(A_2)] = \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1)$$

and

$$\Pr[\mathrm{G}_3(A_2)] = \Pr[\mathrm{G}_4(A_2)].$$

Notice that if A_2 never repeats a nonce-message-header triple to any one user, then neither does A_1. Therefore, since $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}$ we know $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{u-nmh}}$.

Now we reach the point where the proofs differ. We will show the following statements:

$$\Pr[\mathrm{G}_1(A_2)] - \Pr[\mathrm{G}_2(A_2)] = \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{E}}(B)\ , \tag{17}$$

$$\left|\Pr[\mathrm{G}_2(A_2)] - \Pr[\mathrm{G}_3(A_2)]\right| \leq \frac{q_n q_e(q_e-1)}{2^{\ell+1}}\ , \tag{18}$$

which together with the above equations complete the proof.

Let b_* be the challenge bit in the game $\mathbf{G}^{\mathrm{prf}}_{\mathsf{E}}(B)$ and b'_* be the bit returned by B. Now that we allow the adversary to repeat nonces, we are no longer guaranteed that, within each user, the calls to $\textsc{Fn}_b$ will all have unique values of $N\|x$. Therefore, in G_2, we maintain a table $\mathbf{Y}$ which detects when a single user makes two calls to $\textsc{Fn}_b$ with the same $N\|x$, and returns the appropriate value if so. From this, we have $\Pr[b'_* = 1 | b_* = 0] = \Pr[\mathrm{G}_2(A_2)]$. When $b_* = 1$, as in the proof of Theorem 5.2, we have that $\Pr[b'_* = 1 | b_* = 1] = \Pr[\mathrm{G}_1(A_2)]$. Subtracting, we have (17).

Then, notice that $\mathrm{G}_2, \mathrm{G}_3$ are identical-until-bad. Therefore, we can use the fundamental lemma of game playing [14]. This yields $\left|\Pr[\mathrm{G}_2(A_2)] - \Pr[\mathrm{G}_3(A_2)]\right| \leq \Pr[\mathrm{G}_2(A_2) \text{ sets } \mathsf{bad}]$. Then, we can upper bound the probability that $\mathrm{G}_2(A_2)$ sets bad by the probability that there is a collision on x for any two queries to $\textsc{Enc}_2$ by the same user. Since x in $\textsc{Enc}_2$ is a substring of C_1, by a birthday collision argument, this yields (18). ∎

F Proof of Theorem 6.3

Theorem 6.3. *Let* $\mathsf{SE}_{\mathrm{HN4}} = \mathbf{HN4}[\mathsf{SE1}, \ell, \mathsf{F}]$ *be obtained as in Section 6, and let* $\mathsf{SE1}$ *satisfy tidiness. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}$ *making* q_n *queries to its* New *oracle and* q_e, q_d *encryption and decryption queries for each user, respectively, we construct adversaries* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{r-n}} \cap \mathcal{A}^{\mathrm{ae1}}_{\mathrm{priv}}$ *and* B *such that*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE2}}(A_2) \leq \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B) + \frac{q_n q_d}{2^{\mathsf{SE1.nl}}}\ .$$

Adversary A_1 *makes* q_n *queries to its* New *oracle,* q_e *queries to its* Enc *oracle per user, and no queries to its* Dec *oracle.* B *makes* q_n *queries to its* New *oracle, and* $q_e + q_d$ *queries to its* Fn *oracle per user. Adversaries* A_1 *and* B *both have about the same running time as* A_2*.*

Proof: We define the games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3$, and G_4 as specified in Fig. 15. Let b_1 be the challenge bit in $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN4}}}(A_2)$ and b'_1 be the bit returned by A_2. Then, by the definition of game $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN4}}}(A_2)$ and since $\mathsf{SE}_{\mathrm{HN4}}.\mathsf{CS}(\ell_n, \ell_m, \ell_h) = \{0,1\}^{\mathsf{SE1.nl}+\mathsf{SE1.ccl}(\mathsf{SE1.nl}, \ell+\ell_m, \ell_h)}$, we have that $\Pr[b'_1 = 1 | b_1 = 1] = \Pr[\mathrm{G}_0(A_2)]$ and $\Pr[b'_1 = 1 | b_1 = 0] = \Pr[\mathrm{G}_4(A_2)]$. Therefore, $\Pr[\mathrm{G}_0(A_2)] - \Pr[\mathrm{G}_4(A_2)] = \mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN4}}}(A_2)$.

We define adversaries $\overline{A}_2$ and B as specified in Fig. 14. We define $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\text{r-n}}$ to be as expressed in Fig. **??**, with random nonce adversary $\overline{A}_2$ and NBE1 scheme $\mathsf{SE1}$. This is such that the following

procedure INITIALIZE // For all games
Return

procedure FINALIZE(b') // For all games
Return ($b' = 1$)

Game G_0

procedure NEW_0
$v \leftarrow v + 1$; $K_v \leftarrow \mathsf{SE}_{\mathrm{HN2}}.\mathsf{KS}$

procedure $\mathrm{ENC}_0(i, N, M, H)$
$C_2 \leftarrow \mathsf{SE}_{\mathrm{HN2}}.\mathsf{Enc}(K_i, N, M, H)$; Return C_2

procedure $\mathrm{DEC}_0(i, C_2, H)$
$M \leftarrow \mathsf{SE}_{\mathrm{HN2}}.\mathsf{Dec}(K_i, C_2, H)$; Return M

Game G_1

procedure NEW_1
$v \leftarrow v + 1$; $K_{\mathsf{E},v} \leftarrow \{0,1\}^{\mathsf{E.kl}}$

procedure $\mathrm{ENC}_1(i, N, M, H)$
$C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.ccl}(|N|,|M|,|H|)}$
$(x, y) \leftarrow \mathsf{Spl.Ev}(\ell, C_1)$; $C_{1,1} \leftarrow \mathsf{E.Ev}(K_{\mathsf{E},i}, N \| x)$
$C_2 \leftarrow C_{1,1} \| y$; $\mathbf{M}[i, C_2, H] \leftarrow M$; Return C_2

procedure $\mathrm{DEC}_1(i, C_2, H)$
Return $\mathbf{M}[i, C_2, H]$

Games $\boxed{\mathrm{G}_2}$,G_3

procedure NEW_2
Return

procedure $\mathrm{ENC}_2(i, N, M, H)$
$C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.ccl}(|N|,|M|,|H|)}$
$(x, y) \leftarrow \mathsf{Spl.Ev}(\ell, C_1)$
$C_{1,1} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{E.bl}}$
If ($\mathbf{Y}[i, N \| x] \neq \perp$) then
  $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{C_{1,1} \leftarrow \mathbf{Y}[i, N \| x]}$
  Else $\mathbf{Y}[i, N \| x] \leftarrow C_{1,1}$
$C_2 \leftarrow C_{1,1} \| y$; $\mathbf{M}[i, C_2, H] \leftarrow M$
Return C_2

procedure $\mathrm{DEC}_2(i, C_1, H)$
Return $\mathbf{M}[i, C_1, H]$

Game G_4

procedure NEW_4
Return

procedure $\mathrm{ENC}_4(i, N, M, H)$
$C_2 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.nl}+\mathsf{SE1.ccl}(|N|,|M|,|H|)-\ell}$
$\mathbf{M}[i, C_1, H] \leftarrow M$; Return C_2

procedure $\mathrm{DEC}_4(i, C_1, H)$
Return $\mathbf{M}[i, C_1, H]$

Figure 13: Games used in the proof of Theorem 6.2. INITIALIZE, FINALIZE are common to all games.

---

equations hold:

$$\Pr[\mathrm{G}_0(A_2)] - \Pr[\mathrm{G}_1(A_2)] = \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B) \;, \tag{19}$$

$$\left|\Pr[\mathrm{G}_1(A_2)] - \Pr[\mathrm{G}_2(A_2)]\right| \leq \frac{q_n q_d}{2^{\mathsf{SE1.nl}}} \;, \tag{20}$$

$$\Pr[\mathrm{G}_2(A_2)] - \Pr[\mathrm{G}_3(A_2)] = \mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A_1) \;, \tag{21}$$

$$\Pr[\mathrm{G}_3(A_2)] = \Pr[\mathrm{G}_4(A_2)] \;. \tag{22}$$

Together with the above arguments, these equations immediately imply the theorem statement. What remains is to derive the equations (19), (20), (21), and (22). We will do this one by one.

The general strategy of B is to run A_2 with simulated oracles. These oracles follow the description of the scheme algorithms $\mathsf{SE}_{\mathrm{HN4}}.\mathsf{Enc}$ and $\mathsf{SE}_{\mathrm{HN4}}.\mathsf{Dec}$. They simulate the $\mathsf{SE1}$ encryption and decryption in their entirety, and adversary B uses its FN_b oracle to generate the synthetic nonces N_1 (in encryption) and tags T (in decryption). Let b_* be the challenge bit in $\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}(B)$ and b'_* be the bit returned by B. When $b_* = 1$, then N_1 and T are generated using $\mathsf{F.Ev}$. This means that $\Pr[b'_* = 1 | b_* = 1] = \Pr[\mathrm{G}_0(A_2)]$. When $b_* = 0$, N_1 and T will be generated using a random mapping

Adversary $\overline{A}_2^{\text{NEW}_a,\text{ENC}_a,\text{DEC}_a}$	Adversary $B^{\text{NEW}_b,\text{FN}_b}$
INITIALIZE ; $b' \leftarrow\!\!\$\ A_2^{\text{NEW}_a,\text{ENC}_a^*,\text{DEC}_a^*}$ FINALIZE(b')	INITIALIZE ; $b' \leftarrow\!\!\$\ A_2^{\text{NEW}_b^*,\text{ENC}_b^*,\text{DEC}_b^*}$ FINALIZE(b')
procedure $\text{ENC}_a^*(i, N, M, H)$ $(N_1, C_1) \leftarrow\!\!\$\ \text{ENC}_a(i, N\|M, H)$ $\mathbf{M}[i, N_1, C_1, H] \leftarrow N\|M$ $\mathbf{Y}[i, (N, M, H)] \leftarrow N_1$ $C_2 \leftarrow N_1\|C_1$; Return C_2	procedure NEW_b^* $v \leftarrow v + 1$; NEW ; $K_v \leftarrow\!\!\$\ \mathsf{SE1.KS}$
procedure $\text{DEC}_a^*(i, C_2, H)$ $N_1\|C_1 \leftarrow C_2$; $X \leftarrow \mathbf{M}[i, N_1, C_1, H]$ If $(X = \perp)$ then return $\perp$ $N\|M \leftarrow X$; $T \leftarrow \mathbf{Y}[i, (N, M, H)]$ If $(N_1 = T)$ then return M Else return $\perp$	procedure $\text{ENC}_b^*(i, N, M, H)$ $N_1 \leftarrow \text{FN}_b(i, (N, M, H))$ $C_1 \leftarrow \mathsf{SE1.Enc}(K_i, N_1, N\|M, H)$ $C_2 \leftarrow N_1\|C_1$; Return C_2 procedure $\text{DEC}_b^*(i, C_2, H)$ $N_1\|C_1 \leftarrow C_2$; $X \leftarrow \mathsf{SE1.Dec}(K_i, N_1, C_1, H)$ If $(X = \perp)$ then return $\perp$ $N\|M \leftarrow X$; $T \leftarrow \text{FN}_b(i, (N, M, H))$ If $(N_1 = T)$ then return M else return $\perp$

Figure 14: Adversaries $\overline{A}_2, B$ used in proving Theorem 6.3.

from $\mathsf{SE1.NS} \times \mathsf{SE1.MS} \times \mathsf{SE1.HS}$ to $\{0,1\}^{\mathsf{F.ol}}$. This means that $\Pr[b'_* = 1 | b_* = 0] = \Pr[\text{G}_1(A_2)]$, which gives us (19).

Games G_1 and G_2 differ in how they derive X in their decryption oracles. DEC_1 uses $\mathsf{SE1.Dec}$ while DEC_2 will return $\perp$ except on trivial queries. (Recall that a trivial query is when the adversary decrypts a ciphertext returned by the encryption oracle, using the same header and nonce.) They also differ in how they derive synthetic nonce N_1 and tag T. In G_1, this is a random mapping as described above, but in G_2, it generates synthetic nonce N_1 by picking random elements of $\{0,1\}^{\mathsf{F.ol}}$ and sets tag T to be $\perp$ unless a synthetic nonce (for the same N, M, H) has been generated by ENC_2. Notice that this is equivalent to using a random mapping because the adversary never repeats a message-nonce-header triple for the same user. Therefore, to bound A_2's advantage in distinguishing G_1 from G_2, we need only look at when the output of DEC_1 is different from DEC_2. To do this, we are only concerned with the case when DEC_1 does not return $\perp$, but DEC_2 does, since they behave the same in every other case by the correctness of the encryption scheme. If DEC_1 does not return $\perp$, then $(X \neq \perp) \wedge (N_1 = T)$. Now we distinguish two cases: either (1) $\mathbf{Y}[i, y]$ is already initialized, or (2) it is not. Assuming (1), the tidiness of the encryption scheme and the fact that the same $y = (N, M, H)$ have occurred before implies that the same core ciphertext also occurred before. As the query is non-trivial, this means that the nonce N_1 is incorrect and the DEC returns $\perp$, in contradiction to our previous assumption. So we are in case (2), where $\mathbf{Y}[i, y]$ is not initialized. Therefore, $\Pr[N_1 = T] \leq \frac{1}{|\{0,1\}^{\mathsf{SE1.nl}}|}$. If A_2 makes q_d decryption queries for each user, by the union bound, we obtain (20).

A_1 also runs A_2 with simulated oracles. It simulates $\mathsf{SE1}$ encryption using ENC_a and $\mathsf{SE1}$ decryption by returning $\perp$ on all non-trivial queries. It selects N_1 randomly and sets T to be $\perp$ unless a synthetic nonce (for the same N, M, H) has been generated by ENC_a. Let b_2 be the challenge bit in $\mathbf{G}^{\text{ae1}}_{\mathsf{SE1}}(A_1)$ and b'_2 be the bit returned by A_1. When $b_2 = 1$, ENC_a performs encryption using $\mathsf{SE1.Enc}$ and when $b_2 = 0$, it performs encryption by selecting random strings. Therefore, $\Pr[b'_2 = 1 | b_2 = 1] = \Pr[\text{G}_2(A_2)]$ and $\Pr[b'_2 = 1 | b_2 = 0] = \Pr[\text{G}_3(A_2)]$. Subtracting yields (21).

procedure INITIALIZE // For all games
Return
procedure FINALIZE(b') // For all games
Return $(b' = 1)$

Game G_0

procedure NEW_0
$v \leftarrow v + 1$; $K_v \leftarrow\!\!\$\ \mathsf{SE}_{\mathrm{HN4}}.\mathsf{KS}$
procedure $\mathrm{ENC}_0(i, N, M, H)$
$C_2 \leftarrow \mathsf{SE}_{\mathrm{HN4}}.\mathsf{Enc}(K_i, N, M, H)$
Return C_2
procedure $\mathrm{DEC}_0(i, C_2, H)$
$M \leftarrow \mathsf{SE}_{\mathrm{HN4}}.\mathsf{Dec}(K_i, C_2, H)$
Return M

Game G_1

procedure NEW_1
$v \leftarrow v + 1$; $K_v \leftarrow\!\!\$\ \mathsf{SE1.KS}$
procedure $\mathrm{ENC}_1(i, N, M, H)$
$N_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.nl}}$; $y \leftarrow (N, M, H)$
If $(\mathbf{Y}[i, y] \neq \perp)$ then $N_1 \leftarrow \mathbf{Y}[i, y]$
  Else $\mathbf{Y}[i, y] \leftarrow N_1$
$C_1 \leftarrow \mathsf{SE1.Enc}(K_i, N_1, N\|M, H)$
Return $N_1\|C_1$
procedure $\mathrm{DEC}_1(i, C_2, H)$
$N_1\|C_1 \leftarrow C_2$
$X \leftarrow \mathsf{SE1.Dec}(K_i, N_1, C_1, H)$
If $(X = \perp)$ then return $\perp$
$N\|M \leftarrow X$; $T \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.nl}}$
$y \leftarrow (N, M, H)$
If $(\mathbf{Y}[i, y] \neq \perp)$ then $T \leftarrow \mathbf{Y}[i, y]$
  Else $\mathbf{Y}[i, y] \leftarrow T$
If $(N_1 = T)$ then return M else return $\perp$

Game G_2

procedure NEW_2
$v \leftarrow v + 1$; $K_v \leftarrow\!\!\$\ \mathsf{SE1.KS}$
procedure $\mathrm{ENC}_2(i, N, M, H)$
$N_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.nl}}$; $\mathbf{Y}[i, (N, M, H)] \leftarrow N_1$
$C_1 \leftarrow \mathsf{SE1.Enc}(K_i, N_1, N\|M, H)$
$\mathbf{M}[i, N_1, C_1, H] \leftarrow N\|M$; Return $N_1\|C_1$
procedure $\mathrm{DEC}_2(i, C_2, H)$
$N_1\|C_1 \leftarrow C_2$; $X \leftarrow \mathbf{M}[i, N_1, C_1, H]$
If $(X = \perp)$ then return $\perp$
$N\|M \leftarrow X$; $T \leftarrow \mathbf{Y}[i, (N, M, H)]$
If $(N_1 = T)$ then return M else return $\perp$

Game G_3

procedure NEW_3
$v \leftarrow v + 1$; $K_v \leftarrow\!\!\$\ \mathsf{SE1.KS}$
procedure $\mathrm{ENC}_3(i, N, M, H)$
$N_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.nl}}$; $\mathbf{Y}[i, (N, M, H)] \leftarrow N_1$
$C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE1.ccl}(|N_1|, |N|+|M|, |H|)}$
$\mathbf{M}[i, N_1, C_1, H] \leftarrow N\|M$; Return $N_1\|C_1$
procedure $\mathrm{DEC}_3(i, C_2, H)$
$N_1\|C_1 \leftarrow C_2$; $X \leftarrow \mathbf{M}[i, N_1, C_1, H]$
If $(X = \perp)$ then return $\perp$
$N\|M \leftarrow X$; $T \leftarrow \mathbf{Y}[i, (N, M, H)]$
If $(N_1 = T)$ then return M else return $\perp$

Game G_4

procedure NEW_4 // This oracle does nothing
procedure $\mathrm{ENC}_4(i, N, M, H)$
$C_2 \leftarrow\!\!\$\ \mathsf{SE}_{\mathrm{HN4}}.\mathsf{CS}(|N|, |M|, |H|)$
$\mathbf{M}[i, C_2, H] \leftarrow M$; Return C_2
procedure $\mathrm{DEC}_4(i, C_2, H)$
Return $\mathbf{M}[i, C_2, H]$

Figure 15: Games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3, \mathrm{G}_4$ used in proving Theorem 6.3. INITIALIZE and FINALIZE are common to all games.

We note that A_1 does not make use of its decryption oracle, and, as discussed before, has random-nonce-adversary $\overline{A}_2$. Therefore, $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\mathrm{r\text{-}n}} \cap \mathcal{A}^{\mathrm{ae1}}_{\mathrm{priv}}$.

Finally, notice that in ENC_3, the synthetic nonce and SE1 ciphertext are both randomly selected. This is equivalent to ENC_4 since $\mathsf{SE}_{\mathrm{HN4}}.\mathsf{CS}(\ell_n, \ell_m, \ell_h) = \{0,1\}^{\mathsf{SE1.nl}+\mathsf{SE1.ccl}(\mathsf{SE1.nl}, \ell+\ell_m, \ell_h)}$. Also, since $C_2 = N_1\|C_1$, $\mathbf{M}$ in G_3 and G_4 are functionally equivalent. As discussed earlier $\mathbf{Y}[i, (N, M, H)] \neq \perp$ if and only if $\mathbf{M}[i, N_1, C_1, H] \neq \perp$ since they would have been defined in the same call to ENC_3. Therefore, we can simplify the decryption oracle to that which is in DEC_4. From this, we have (8),

Adversary $B^{\text{New}_a,\text{Fn}_a,\text{FnInv}_a}$

Initialize ; $b' \leftarrow\!\!\$\ A^{\text{New}_a,\text{Enc}^*_a,\text{Dec}_4}$; Finalize(b')

procedure $\text{Enc}^*_a(i, N, M, H)$
$C_2 \leftarrow \text{Fn}(i, H, 0^{\ell_t}\|N\|M)$; Return C_2

procedure $\text{Dec}_4(i, C_2, H)$
$X \leftarrow \text{FnInv}(i, H, C_2)$; If $X[1..\ell_t] \neq 0^{\ell_t}$ then return $\perp$
$N\|M \leftarrow X[(\ell_t + 1)..|X|]$; Return M

Figure 16: Adversary B used in the proof of Theorem 6.4.

which concludes the proof. ∎

G Proof of Theorem 6.4

Theorem 6.4. *Let* $\mathsf{SE}_{\text{HN5}} = \mathbf{HN5}[\mathsf{TE}, \ell, \ell_t]$ *be obtained as in Section 6. Let* $A \in \mathcal{A}^{\text{ae2}}_{\text{u-nmh}}$ *be an adversary making* q_n *queries to its* New *oracle,* q_e *queries per user to its* Enc *oracle with minimum message length* ℓ_1*, and* q_d *queries with minimum ciphertext length* $\ell_2 \geq \ell_t$ *per user to its* Dec *oracle. We construct adversary* B *such that*

$$\mathbf{Adv}^{\text{ae2}}_{\mathsf{SE}_{\text{HN5}}}(A) \leq \mathbf{Adv}^{\text{stprp}}_{\mathsf{TE}}(B) + \frac{q_n q_e(q_e+1)}{2^{\ell_t+\ell+\ell_1+1}} + \frac{q_n q_d(q_d+1)}{2^{\ell_2+1}} + \frac{q_n q_d}{2^{\ell_t}} \ .$$

Adversary B *makes* q_n *queries to its* New *oracle,* q_e *queries per user to its* Fn *oracle, and* q_d *queries per user to its* FnInv *oracle.*

Proof: We define the games $\text{G}_0, \text{G}_1, \text{G}_2$, and G_3 as specified in Fig. 17, and games $\text{G}_4, \text{G}_5, \text{G}_6, \text{G}_7$, and G_8 as specified in Fig. 18. Let b_1 be the challenge bit in $\mathbf{G}^{\text{ae2}}_{\mathsf{SE}_{\text{HN5}}}(A)$ and b'_1 be the bit returned by A. Then, by the definition of the game $\mathbf{G}^{\text{ae2}}_{\mathsf{SE}_{\text{HN5}}}(A)$ and since $\mathsf{SE}_{\text{HN5}}.\mathsf{CS}(\ell_n, \ell_m, \ell_h) = \{0,1\}^{\ell_t+\ell+\ell_m}$, we have that $\Pr[b'_1 = 1 | b_1 = 1] = \Pr[\text{G}_0(A)]$ and $\Pr[b'_1 = 1 | b_1 = 0] = \Pr[\text{G}_8(A)]$. Therefore, $\Pr[\text{G}_0(A)] - \Pr[\text{G}_8(A)] = \mathbf{Adv}^{\text{ae2}}_{\mathsf{SE}_{\text{HN5}}}(A)$.

We define adversaries $\overline{A}_2$ and B as specified in Fig. 14. This is such that the following equations hold:

$$\Pr[\text{G}_0(A)] - \Pr[\text{G}_1(A)] = \mathbf{Adv}^{\text{stprp}}_{\mathsf{TE}}(B) \ , \tag{23}$$

$$\Pr[\text{G}_1(A)] = \Pr[\text{G}_2(A)] \ , \tag{24}$$

$$|\Pr[\text{G}_2(A)] - \Pr[\text{G}_3(A)]| \leq \frac{q_n q_e(q_e-1)}{2^{\ell_t+\ell+\ell_1+1}} + \frac{q_n q_d}{2^{\ell_2}} \ , \tag{25}$$

$$\Pr[\text{G}_3(A)] = \Pr[\text{G}_4(A)] \ , \tag{26}$$

$$|\Pr[\text{G}_4(A)] - \Pr[\text{G}_5(A)]| \leq \frac{q_n q_e}{2^{\ell_t+\ell+\ell_1}} + \frac{q_n q_d(q_d-1)}{2^{\ell_2+1}} \ , \tag{27}$$

$$\Pr[\text{G}_5(A)] = \Pr[\text{G}_6(A)] \ , \tag{28}$$

$$|\Pr[\text{G}_6(A)] - \Pr[\text{G}_7(A)]| \leq \frac{q_n q_d}{2^{\ell_t}} \ , \tag{29}$$

$$\Pr[\text{G}_7(A)] = \Pr[\text{G}_8(A)] \ . \tag{30}$$

Together with the above arguments, these equations can be combined to derive the theorem statement:

$$\begin{aligned}
&\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN5}}}(A) \\
&= \Pr[\mathrm{G}_0(A)] - \Pr[\mathrm{G}_8(A)] \\
&\leq \mathbf{Adv}^{\mathrm{stprp}}_{\mathsf{TE}}(B) + \frac{q_n q_e(q_e-1)}{2^{\ell_t+\ell+\ell_1+1}} + \frac{q_n q_d}{2^{\ell_2}} + \frac{q_n q_e}{2^{\ell_t+\ell+\ell_1}} + \frac{q_n q_d(q_d-1)}{2^{\ell_2+1}} + \frac{q_n q_d}{2^{\ell_t}} \\
&= \mathbf{Adv}^{\mathrm{stprp}}_{\mathsf{TE}}(B) + \frac{q_n q_e(q_e-1)+2q_n q_e}{2^{\ell_t+\ell+\ell_1+1}} + \frac{q_n q_d(q_d-1)+2q_n q_d}{2^{\ell_2+1}} + \frac{q_n q_d}{2^{\ell_t}} \\
&= \mathbf{Adv}^{\mathrm{stprp}}_{\mathsf{TE}}(B) + \frac{q_n q_e(q_e+1)}{2^{\ell_t+\ell+\ell_1+1}} + \frac{q_n q_d(q_d+1)}{2^{\ell_2+1}} + \frac{q_n q_d}{2^{\ell_t}} \ .
\end{aligned}$$

What remains is to derive the equations (23), (24), (25), (26), (27), (28), (29), and (30). We will do this one by one.

The general strategy of B is simply to run A with simulated oracles. It does this by using the Fn and FnInv oracles to perform the tweakable cipher evaluation and inversion. Let b_* be the challenge bit in the game $\mathbf{G}^{\mathrm{stprp}}_{\mathsf{TE}}(A)$ and b'_* be the bit returned by B. When $b_* = 0$, evaluation and inversion occurs according to a length-preserving random permutation from $\{0,1\}^* \to \{0,1\}^*$. This is captured in G_1, where we sample from the domain and range at random. The sets $\mathcal{X}_{i,H}$,$\mathcal{Y}_{i,H}$ ensure that we achieve a bijection (i.e. we sample without replacement) and the tables $\mathbf{X}, \mathbf{Y}$ will ensure that repeated queries or trivial encryption/ decryption queries (recall that this is when we take the output of the encryption oracle and give it to the decryption oracle, or vice versa) will give consistent results. Hence, $\Pr[b'_* = 1 | b_* = 0] = \Pr[\mathrm{G}_1(A)]$. When $b^* = 1$, the Fn and FnInv oracles will use TE for the evaluation and inversion. This is captured in G_0 so $\Pr[b'_* = 1 | b_* = 1] = \Pr[\mathrm{G}_0(A)]$. Subtracting yields (23).

G_1 and G_2 differ only in their encryption functions. In G_1, we select C_2 from $\left(\{0,1\}^{|M^*|}\right) \setminus \mathcal{Y}_{i,H}$, the set of all strings of the correct length which have not selected as a ciphertext yet (within that session). In G_2, we select C_2 from $\{0,1\}^{|M^*|}$ and only if there is a collision between C_2 and a previously selected ciphertext within the same session do we reselect C_2 from the former set. From here, we see that the outputs of $\textsc{Enc}_1$ and $\textsc{Enc}_2$ (which includes the boxed code) are complete indistinguishable. This gives us (24).

Now notice that $\mathrm{G}_2, \mathrm{G}_3$ are identical-until-bad. Note that G_2 sets bad if some C_2 selected within the encryption oracle collides with an element in $\mathcal{Y}_{i,H}$. We begin by computing the probability of this within a single session. Notice that a string C'_2 ends up in $\mathcal{Y}_{i,H}$ one of two ways. The first way is that some call was made to $\textsc{Dec}_2$ with second argument C'_2. At most q_d different values of C'_2 are passed to $\textsc{Dec}_2$ so to bound the probability that an encryption query leads to a collision with one of these, we conservatively assume all the ciphertexts are of the same minimal length of ℓ_2. This probability bound is $\frac{q_d}{2^{\ell_2}}$. The second way is that some prior call to $\textsc{Enc}_2$ randomly selected the string C'_2. To bound the probability of this, we conservatively assume that all the M^* selected are of the same minimal length (i.e. $|M^*| = \ell_t + \ell + \ell_1$) and use a birthday collision bound. This bound is $\frac{q_e(q_e-1)}{2^{\ell_t+\ell+\ell_1+1}}$. Using the union bound to accommodate all sessions, we get (25).

Next, notice that G_4 simplifies the encryption function of G_3. As above, encryption will always pick random strings (now with replacement). We remove the set $\mathcal{Y}_{i,H}$ since it is no longer used. For decryption, $\textsc{Dec}_2$ and $\textsc{Dec}_3$ (with the boxed code) are completely indistinguishable in an analogous way to how $\textsc{Enc}_1$ and $\textsc{Enc}_2$ (with the boxed code) are. We omit formally showing this for brevity. From here, we get (26)

Game G_0

procedure $\textsc{New}_0$
$v \leftarrow v+1$; $K_v \leftarrow\!\!\$\ \{0,1\}^{\mathsf{TE.kl}}$

procedure $\textsc{Enc}_0(i,N,M,H)$
$C_2 \leftarrow \mathsf{TE.Ev}(K_{\mathsf{TE}}, H, 0^{\ell_t}\|N\|M)$
Return C_2

procedure $\textsc{Dec}_0(i,C_2,H)$
$T\|N\|M \leftarrow \mathsf{TE.In}(i,H,C_2)$
If $(T \neq 0^{\ell_t})$ then return $\perp$ else return M

Game G_1

procedure $\textsc{New}_1$
Return

procedure $\textsc{Enc}_1(i,N,M,H)$
$M^* \leftarrow 0^{\ell_t}\|N\|M$
If $(\mathbf{Y}[i,H,M^*] = \perp)$ then
  $C_2 \leftarrow\!\!\$\ \left(\{0,1\}^{|M^*|}\right) \setminus \mathcal{Y}_{i,H}$
  $\mathbf{X}[i,H,C_2] \leftarrow M^*$; $\mathbf{Y}[i,H,M^*] \leftarrow C_2$
  $\mathcal{X}_{i,H} \leftarrow \mathcal{X}_{i,H} \cup \{M^*\}$; $\mathcal{Y}_{i,H} \leftarrow \mathcal{Y}_{i,H} \cup \{C_2\}$
Return $\mathbf{Y}[i,H,M^*]$

procedure $\textsc{Dec}_1(i,C_2,H)$
If $(\mathbf{X}[i,H,C_2] = \perp)$ then
  $M^* \leftarrow\!\!\$\ \left(\{0,1\}^{|C_2|}\right) \setminus \mathcal{X}_{i,H}$
  $\mathbf{X}[i,H,C_2] \leftarrow M^*$; $\mathbf{Y}[i,H,M^*] \leftarrow C_2$
  $\mathcal{X}_{i,H} \leftarrow \mathcal{X}_{i,H} \cup \{M^*\}$; $\mathcal{Y}_{i,H} \leftarrow \mathcal{Y}_{i,H} \cup \{C_2\}$
$T\|N\|M \leftarrow \mathbf{X}[i,H,C_2]$
If $(T \neq 0^{\ell_t})$ then return $\perp$ return M

procedure $\textsc{Initialize}$ // For all games
Return

procedure $\textsc{Finalize}(b')$ // For all games
Return $(b'=1)$

Games $\boxed{\mathrm{G}_2}$,G_3

procedure $\textsc{New}_2$
Return

procedure $\textsc{Enc}_2(i,N,M,H)$
$M^* \leftarrow 0^{\ell_t}\|iv\|M$
If $(\mathbf{Y}[i,H,M^*] = \perp)$ then
  $C_2 \leftarrow\!\!\$\ \{0,1\}^{|M^*|}$
  If $C_2 \in \mathcal{Y}_{i,H}$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$
    $\boxed{C_2 \leftarrow\!\!\$\ \left(\{0,1\}^{|M^*|}\right) \setminus \mathcal{Y}_{i,H}}$
  $\mathbf{X}[i,H,C_2] \leftarrow M^*$; $\mathbf{Y}[i,H,M^*] \leftarrow C_2$
  $\mathcal{X}_{i,H} \leftarrow \mathcal{X}_{i,H} \cup \{M^*\}$
  $\mathcal{Y}_{i,H} \leftarrow \mathcal{Y}_{i,H} \cup \{C_2\}$
Return $\mathbf{Y}[i,H,M^*]$

procedure $\textsc{Dec}_2(i,C_2,H)$
If $(\mathbf{X}[i,H,C_2] = \perp)$ then
  $M^* \leftarrow\!\!\$\ \left(\{0,1\}^{|C_2|}\right) \setminus \mathcal{X}_{i,H}$
  $\mathbf{X}[i,H,C_2] \leftarrow M^*$; $\mathbf{Y}[i,H,M^*] \leftarrow C_2$
  $\mathcal{X}_{i,H} \leftarrow \mathcal{X}_{i,H} \cup \{M^*\}$
  $\mathcal{Y}_{i,H} \leftarrow \mathcal{Y}_{i,H} \cup \{C_2\}$
$T\|N\|M \leftarrow \mathbf{X}[i,H,C_2]$
If $(T \neq 0^{\ell_t})$ then return $\perp$
Return M

Figure 17: Games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3$ used in proving Theorem 6.4. $\textsc{Initialize}, \textsc{Finalize}$ are common to all games.

$\mathrm{G}_4, \mathrm{G}_5$ are also identical-until-bad. Computing $\Pr[\mathrm{G}_4(A) \text{ sets } \mathsf{bad}]$ follows similar logic to computing $\Pr[\mathrm{G}_2(A) \text{ sets } \mathsf{bad}]$. G_4 sets bad if some M^* selected within $\textsc{Dec}_4$ collides with an element in $\mathcal{X}_{i,H}$. This element is either equal to $0^{\ell_t}\|N\|M$ for some prior query $\textsc{Enc}_4(i,N,M,H)$ or a random string returned in another query to $\textsc{Dec}_4$. Via an analogous argument to the one used above, we get that $\Pr[\mathrm{G}_4(A) \text{ sets } \mathsf{bad}] \leq \frac{q_n q_e}{2^{\ell_t+\ell+\ell_1}} + \frac{q_n q_d(q_d-1)}{2^{\ell_2+1}}$, which is (27)

Going from G_5 to G_6, we remove the set $\mathcal{X}_{i,H}$, which is never used, and set a bad flag, which has no consequence to the outcome of the game since G_6 does not include the boxed code. This is (28).

$\mathrm{G}_6, \mathrm{G}_7$ are our final pair of identical-until-bad games. Notice that G_6 sets bad if $T = 0^{\ell_t}$ in some decryption query. Since T is a substring of the random string M^*, and a user generates at most q_d values of M^*, we can use the union bound to get $\Pr[\mathrm{G}_6(A) \text{ sets } \mathsf{bad}] \leq \frac{q_n q_d}{2^{\ell_t}}$, giving us (29).

Games G_4,$\boxed{\mathrm{G}_5}$

procedure $\textsc{New}_4$
Return

procedure $\textsc{Enc}_4(i, N, M, H)$
$M^* \leftarrow 0^{\ell_t} \| N \| M$
If $(\mathbf{Y}[i, H, M^*] = \bot)$ then
  $C_2 \leftarrow\!\!\$\ \{0,1\}^{|M^*|}$
  $\mathbf{X}[i, H, C_2] \leftarrow M^*$
  $\mathbf{Y}[i, H, M^*] \leftarrow C_2$
  $\mathcal{X}_{i,H} \leftarrow \mathcal{X}_{i,H} \cup \{M^*\}$
Return $\mathbf{Y}[i, H, M^*]$

procedure $\textsc{Dec}_4(i, C_2, H)$
If $(\mathbf{X}[i, H, C_2] = \bot)$ then
  $M^* \leftarrow\!\!\$\ \{0,1\}^{|C_2|}$
  If $M^* \in \mathcal{X}_{i,H}$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$
    $\boxed{M^* \leftarrow\!\!\$\ \left(\{0,1\}^{|C_2|}\right) \setminus \mathcal{X}_{i,H}}$
  $\mathbf{X}[i, H, C_2] \leftarrow M^*$
  $\mathbf{Y}[i, H, M^*] \leftarrow C_2$
  $\mathcal{X}_{i,H} \leftarrow \mathcal{X}_{i,H} \cup \{M^*\}$
$T \| N \| M \leftarrow \mathbf{X}[i, H, C_2]$
If $(T \neq 0^{\ell_t})$ then return $\bot$
Return M

Games G_6,$\boxed{\mathrm{G}_7}$

procedure $\textsc{New}_6$
Return

procedure $\textsc{Enc}_6(i, N, M, H)$
$M^* \leftarrow 0^{\ell_t} \| N \| M$
If $(\mathbf{Y}[i, H, M^*] = \bot)$ then
  $C_2 \leftarrow\!\!\$\ \{0,1\}^{|M^*|}$
  $\mathbf{X}[i, H, C_2] \leftarrow M^*$; $\mathbf{Y}[i, H, M^*] \leftarrow C_2$
Return $\mathbf{Y}[i, H, M^*]$

procedure $\textsc{Dec}_6(i, C_2, H)$
If $(\mathbf{X}[i, H, C_2 \neq \bot)$ then return $\mathbf{X}[i, H, C_2]$
$M^* \leftarrow\!\!\$\ \{0,1\}^{|C_2|}$; $\mathbf{X}[i, H, C_2] \leftarrow M^*$
$\mathbf{Y}[i, H, M^*] \leftarrow C_2$; $T \| N \| M \leftarrow \mathbf{X}[i, H, C_2]$
If $(T \neq 0^{\ell_t})$ then return $\bot$ else $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\text{Return } \bot}$
Return M

Game G_8

procedure $\textsc{New}_8$
Return

procedure $\textsc{Enc}_8(i, N, M, H)$
$C_2 \leftarrow\!\!\$\ \{0,1\}^{\ell_t + |N| + |M|}$; $\mathbf{M}[i, C_2, H] \leftarrow M$; Return C_2

procedure $\textsc{Dec}_0(i, C_2, H)$
If $(\mathbf{M}[i, C_2, H] \neq \bot)$ then return $\mathbf{M}[i, C_2, H]$; Return $\bot$

Figure 18: Games $\mathrm{G}_4, \mathrm{G}_5, \mathrm{G}_6, \mathrm{G}_7, \mathrm{G}_8$ used in proving Theorem 6.4. $\textsc{Initialize}, \textsc{Finalize}$ are as in Fig. 17.

Finally, notice that in G_7, if the decryption query made is non-trivial we will always return $\bot$. We have no need for $\mathbf{Y}$ anymore. Also, $\mathbf{X}$ in G_6 serves the same purpose as $\mathbf{M}$ in the game $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}_{\mathrm{HN5}}}$ when $b_1 = 0$. Therefore, G_7 can be simplified to G_8, giving us (30). ∎

H Details of CAU1 and Proof of Theorem 7.1

Let $\mathsf{CAU1} = \mathbf{CAU1}[\mathsf{E}, \mathsf{H}, \ell]$ be the NBE1 scheme discussed in Section 7. We begin by recalling the definition of CAU1.Enc, CAU1.Dec, as they were given in [15]. Note that when M, C_1 are being parsed into blocks of the form $\tau, M_i, C_{1,i}$, this is such that all blocks except the last one are of length E.bl. (i.e. $|\tau| = |M_i| = |C_{1,i}| = \mathsf{E.bl}$ for $i < m$ and $|M_m| = |C_{1,m}| \leq \mathsf{E.bl}$.) As before, parsing in CAU2.Dec is such that $|x| = \ell$. If any parsing fails, or if M, C_1 are too long (meaning $\langle m+1 \rangle_{\mathsf{E.bl}-\ell}$ is not defined), we return $\bot$. Recall that, as defined in Section 2, $\langle i \rangle_n$ is the n-bit representation of integer i.

Our proof requires that F is an sPRP and H is an (ϵ_1, ϵ_2)-almost-XOR-universal (AXU) function family (as defined in [35, 15, 33, 2] and others). We begin by recalling these definitions.

$\underline{\mathsf{CAU1.Enc}(K,N,M,H)}$
$M_1\|M_2\|...\|M_m \leftarrow M$
For $i = 1,2,..(m-1)$
  $C_{1,i} \leftarrow M_i \oplus \mathsf{E.Ev}(K, N\|\langle i+1\rangle_{\mathsf{E.bl}-\ell})$
$P \leftarrow \mathsf{E.Ev}(K, N\|\langle m+1\rangle_{\mathsf{E.bl}-\ell})$
$C_{1,m} \leftarrow M_m \oplus P[1..|M_m|]$
$h \leftarrow \mathsf{H.Ev}(\mathsf{E.Ev}(K, 0^{\mathsf{E.bl}}), (C_{1,1}\|..\|C_{1,m}, H))$
$\tau \leftarrow h \oplus \mathsf{E.Ev}(K, N\|\langle 1\rangle_{\mathsf{E.bl}-\ell})$
$C_1 \leftarrow \tau\|C_{1,1}\|..\|C_{1,m}$; Return C_1

$\underline{\mathsf{CAU1.Dec}(K,N,C_1,H)}$
$\tau\|C_{1,1}\|C_{1,2}\|...\|C_{1,m} \leftarrow C_1$
$h \leftarrow \mathsf{H.Ev}(\mathsf{E.Ev}(K, 0^{\mathsf{E.bl}}), (C_{1,1}\|..\|C_{1,m}, H))$
$\tau' \leftarrow h \oplus \mathsf{E.Ev}(K, N\|\langle 1\rangle_{\mathsf{E.bl}-\ell})$
If $(\tau \neq \tau')$ then return $\perp$
For $i = 1,2,..(m-1)$
  $M_i \leftarrow C_{1,i} \oplus \mathsf{E.Ev}(K, N\|\langle i+1\rangle_{\mathsf{E.bl}-\ell})$
$P \leftarrow \mathsf{E.Ev}(K, N\|\langle m+1\rangle_{\mathsf{E.bl}-\ell})$
$M_m \leftarrow C_{1,m} \oplus P[1..|C_{1,m}|]$
Return $M_1\|M_2\|...\|M_m$

$\underline{\mathsf{CAU2.Enc}(K,N,M,H)}$
$M_1\|M_2\|...\|M_m \leftarrow 0^\ell\|M$
For $i = 1,2,..(m-1)$
  $C_{1,i} \leftarrow M_i \oplus \mathsf{E.Ev}(K, N\|\langle i+1\rangle_{\mathsf{E.bl}-\ell})$
$P \leftarrow \mathsf{E.Ev}(K, N\|\langle m+1\rangle_{\mathsf{E.bl}-\ell})$
$C_{1,m} \leftarrow M_m \oplus P[1..|M_m|]$
$h \leftarrow \mathsf{H.Ev}(\mathsf{E.Ev}(K, 0^{\mathsf{E.bl}}), (C_{1,1}\|..\|C_{1,m}, H))$
$\tau \leftarrow h \oplus \mathsf{E.Ev}(K, N\|\langle 1\rangle_{\mathsf{E.bl}-\ell})$
Return $\tau\|C_{1,1}\|..\|C_{1,m}$

$\underline{\mathsf{CAU2.Dec}(K,C_2,H)}$
$\tau\|C_{1,1}\|C_{1,2}\|...\|C_{1,m} \leftarrow C_2$
$h \leftarrow \mathsf{H.Ev}(\mathsf{E.Ev}(K, 0^{\mathsf{E.bl}}), (C_{1,1}\|...\|C_{1,m}, H))$
$y \leftarrow \mathsf{E.In}(K, \tau \oplus h)$; $N \leftarrow y[1..\ell]$
If $(y[(\ell+1)..\mathsf{E.bl}] \neq \langle 1\rangle_{\mathsf{E.bl}-\ell})$ then
  Return $\perp$
For $i = 1,2,..(m-1)$ do
  $M_i \leftarrow C_{1,i} \oplus \mathsf{E.Ev}(K, N\|\langle i+1\rangle_{\mathsf{E.bl}-\ell})$
$P \leftarrow \mathsf{E.Ev}(K, N\|\langle m+1\rangle_{\mathsf{E.bl}-\ell})$
$M_m \leftarrow C_{1,m} \oplus P[1..|C_{1,m}|]$
$x\|M_1' \leftarrow M_1$; If $(x \neq 0^\ell)$ then return $\perp$
Return $M_1'\|M_2\|...\|M_m$

Figure 19: Top: Encryption and decryption algorithms of NBE1 scheme $\mathsf{CAU1} = \mathbf{CAU1}[\mathsf{E},\mathsf{H},\ell]$, a special case of which is GCM. Bottom: Encryption and decryption algorithms of NBE2 scheme $\mathsf{CAU2} = \mathbf{CAU2}[\mathsf{E},\mathsf{H},\ell]$.

Definition H.1 *Let* H *be a function family with* $\mathsf{H.D} = \mathcal{C}\times\mathcal{H}$ *for some length-closed* $\mathcal{C},\mathcal{H} \subset \{0,1\}^*$. *Let* $\epsilon_i : \mathbb{N}\times\mathbb{N} \to [0,1]$ *be functions such that* $\epsilon_i(C,\cdot)$ *and* $\epsilon_i(\cdot,H)$ *are monotonically increasing for all* $C \in \mathcal{C}, H \in \mathcal{H}$ *for* $i = \{1,2\}$. *We say that* H *is an* (ϵ_1,ϵ_2)*-AXU function family if*

(1) For all $(M,H) \in \mathsf{H.D}$ *and all* $y \in \{0,1\}^{\mathsf{H.ol}}$, *we have*

$$\Pr[\mathsf{H}(K,(M,H)) = y : K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.kl}}] \le \epsilon_1(|M|,|H|)$$

(2) For all distinct $(M,H),(M',H') \in \mathsf{H.D}$ *and all* $y \in \{0,1\}^{\mathsf{H.ol}}$, *we have*

$$\Pr[\mathsf{H}(K,(M,H)) \oplus \mathsf{H}(K,(M',H')) = y : K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.kl}}] \le \epsilon_2(\max(|M|,|M'|),\max(|H|,|H'|)).$$

We define a multi-user variant of sPRP (strong PRP) security for invertible function family F and adversary A via the $\mathbf{G}^{\mathrm{sprp}}_{\mathsf{F}}(A)$ in Fig. 20. In the game, b is the challenge bit and $\mathbf{X}[\cdot,\cdot]$, $\mathbf{Y}[\cdot,\cdot]$ are tables whose entries are assumed initialized to $\perp$. In the $b = 0$ case, the Fn, FnInv oracles samples the range and domain respectively, without replacement. This is assured by maintaining the sets $\mathcal{X}_i, \mathcal{Y}_i$. It is required that any $\textsc{Fn}(i,X)$ or $\textsc{FnInv}(i,Y)$ query of A satisfies $i \le v$, $X \in \mathsf{F.D}$ and $Y \in \{0,1\}^{\mathsf{F.ol}}$. The multi-user sPRP advantage of adversary A is $\mathbf{Adv}^{\mathrm{sprp}}_{\mathsf{F}}(A) = 2\Pr[\mathbf{G}^{\mathrm{sprp}}_{\mathsf{F}}(A)] - 1$.

We now wish to prove Theorem 7.1 from Section 7, which confirms the security of CAU2 against

Game $\mathbf{G}^{\text{sprp}}_{\mathsf{E}}$

procedure INITIALIZE
$b \leftarrow\!\!\$\ \{0,1\}$

procedure NEW
$v \leftarrow v+1$; $K_v \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$

procedure FN(i, X)
If $(\mathbf{Y}[i,X] = \bot)$ then
  $Y_0 \leftarrow\!\!\$\ \left(\{0,1\}^{\mathsf{F.ol}}\right) \setminus \mathcal{Y}_i$; $Y_1 \leftarrow \mathsf{F.Ev}(K_i, X)$
  $\mathbf{Y}[i,X] \leftarrow Y_b$; $\mathbf{X}[i,Y_b] \leftarrow X$
  $\mathcal{X}_i \leftarrow \mathcal{X}_i \cup \{X\}$; $\mathcal{Y}_i \leftarrow \mathcal{Y}_i \cup \{Y_b\}$
Return $\mathbf{Y}[i,X]$

procedure FNINV(i, Y)
If $(\mathbf{X}[i,Y] = \bot)$ then
  $X_0 \leftarrow\!\!\$\ \mathsf{F.D} \setminus \mathcal{X}_i$; $X_1 \leftarrow \mathsf{F.In}(K_i, Y)$
  $\mathbf{Y}[i,X] \leftarrow Y_b$; $\mathbf{X}[i,Y_b] \leftarrow X$
  $\mathcal{X}_i \leftarrow \mathcal{X}_i \cup \{X_b\}$; $\mathcal{Y}_i \leftarrow \mathcal{Y}_i \cup \{Y\}$
Return $\mathbf{X}[i,Y]$

procedure FINALIZE(b')
Return $(b = b')$

Game $\mathbf{G}^{\text{auth1}}_{\mathsf{SE2}}$

procedure INITIALIZE
Return

procedure NEW
$v \leftarrow v+1$; $K_v \leftarrow\!\!\$\ \mathsf{SE2.KS}$

procedure ENC(i, N, M, H)
$C_2 \leftarrow \mathsf{SE2.Enc}(K_i, N, M, H)$
Return C_2

procedure VF(i, C_2, H)
$M \leftarrow \mathsf{SE2.Dec}(K_i, C_2, H)$
If $(M \neq \bot)$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $(M = \bot)$

procedure FINALIZE
Return win

Figure 20: Left: Game defining multi-user sPRP security for function family F. Right: Game defining authenticity of NBE2 scheme SE2.

nonce-respecting adversaries. We restate the encryption and decryption algorithms of CAU2 in full in Fig. 19. For clarity, we removed the step from CAU1.Dec which recomputes and checks the tag. Since we recovered the nonce from the tag, the check will always return true (i.e. $\tau = \tau'$) since this is a tautology.

The proof of Theorem 7 can be greatly simplified by considering the privacy and authenticity of the scheme separately. To this end, recall that AE2$[\mathcal{A}^{\text{ae2}}_{\text{priv}} \cap \mathcal{A}^{\text{ae2}}_{\text{u-n}}]$-security (that is, NBE2 schemes secure against adversaries who do not make a decryption query nor repeat a nonce) captures a security notion that is privacy only. We define an authenticity-only game, $\mathbf{G}^{\text{auth1}}_{\mathsf{SE2}}$ for NBE2 scheme SE2, as depicted in Fig. 20. In this game, there is no challenge bit. The adversary gets three oracles: (1) NEW, to generate new sessions, (2) ENC, an encryption oracle which performs encryption using SE2.Enc, and (3) VF, a verification oracle which returns a boolean value indicating if decryption of the provided ciphertext C_2 returns $\bot$. The adversary wins if it can get VF to return true for any non-trivial query. We make the same assumptions about the adversary's queries to ENC, VF that we made of the adversary's queries to ENC, DEC in the AE2-security game. We say that this authenticity advantage of an adversary A is $\mathbf{Adv}^{\text{auth1}}_{\mathsf{SE2}}(A) = \Pr[\mathbf{G}^{\text{auth1}}_{\mathsf{SE2}}]$. This game adapts the notion of AUTH-security as defined by Rogaway in [46], but for NBE2 and in the multi-user setting.

We now prove that a scheme that achieves both privacy and authenticity as defined above will also be AE2$[\mathcal{A}^{\text{ae2}}_{\text{u-n}}]$-secure. We do this in Lemma H.2.

Lemma H.2 *Let* SE2 *be an NBE2 scheme. Let* $A \in \mathcal{A}^{\text{ae2}}_{\text{u-n}}$ *be an adversary making* q_n *calls to its* NEW *oracle,* q_e, q_d *calls to its* ENC, DEC *oracles, respectively, per user. Then, we can construct*

Adversary $B^{\text{New}_b, \text{Enc}_b, \text{Dec}_b}$	Adversary $C^{\text{New}_c, \text{Enc}_c, \text{Vf}_c}$
Initialize ; $b' \leftarrow\!\!\$\ A^{\text{New}_b, \text{Enc}_b, \text{Dec}_b^*}$ Finalize(b')	Initialize ; $b' \leftarrow\!\!\$\ A^{\text{New}_c, \text{Enc}_c^*, \text{Dec}_c^*}$ Finalize
procedure $\text{Dec}_b^*(i, C_2, H)$ Return $\perp$	procedure $\text{Dec}_c^*(i, C_2, H)$ $\text{Vf}_c(i, C_2, H)$; Return $\perp$

Game G_0	Game G_1	Game G_2
procedure Initialize Return	procedure Initialize Return	procedure Initialize Return
procedure New_0 $v \leftarrow v+1$; $K_v \leftarrow\!\!\$\ \mathsf{SE2.KS}$	procedure New_1 $v \leftarrow v+1$; $K_v \leftarrow\!\!\$\ \mathsf{SE2.KS}$	procedure New_2 Return
procedure $\text{Enc}_0(i, N, M, H)$ $C_2 \leftarrow\!\!\$\ \mathsf{SE2.Enc}(K_i, N, M, H)$ Return C_2	procedure $\text{Enc}_1(i, N, M, H)$ $C_2 \leftarrow\!\!\$\ \mathsf{SE2.Enc}(K_i, N, M, H)$ Return C_2	procedure $\text{Enc}_2(i, N, M, H)$ $C_2 \leftarrow\!\!\$\ \mathsf{SE2.CS}(\lvert N\rvert, \lvert M\rvert, \lvert H\rvert)$ Return C_2
procedure $\text{Dec}_0(i, C_2, H)$ $M \leftarrow\!\!\$\ \mathsf{SE2.Dec}(K_i, N, M, H)$ Return M	procedure $\text{Dec}_1(i, C_2, H)$ Return $\perp$	procedure $\text{Dec}_2(i, C_2, H)$ Return $\perp$
procedure Finalize(b') Return $(b' = 1)$	procedure Finalize(b') Return $(b' = 1)$	procedure Finalize(b') Return $(b' = 1)$

Figure 21: Top: Adversaries B, C used in the proof of Lemma H.2. Bottom: Games $\text{G}_0, \text{G}_1, \text{G}_2$ used in proving Lemma H.2.

---

adversaries $B \in \mathcal{A}^{\text{ae2}}_{\text{priv}} \cap \mathcal{A}^{\text{ae2}}_{\text{u-n}}$ *and* $C \in \mathcal{A}^{\text{ae2}}_{\text{u-n}}$ *such that:*

$$\mathbf{Adv}^{\text{ae2}}_{\mathsf{SE2}}(A) \leq \mathbf{Adv}^{1}_{\mathsf{SE2}}(B) + \mathbf{Adv}^{\text{auth1}}_{\mathsf{SE2}}(C)$$

Here, B *makes* q_n *queries to its* New *oracle and* q_e *queries to its* Enc *oracle per user.* C *makes* q_n *queries to its* New *oracle and* q_e, q_d *queries to its* Enc, Vf *oracles, respectively, per user.*

Proof: We define adversaries B, C and games $\text{G}_0, \text{G}_1, \text{G}_2$ as in Fig. 21. As before, we assume that A makes no repeat queries or trivial decryption queries.

Let b_1 be the challenge bit in $\mathbf{G}^{\text{ae2}}_{\mathsf{SE2}}(A)$ and b_1' be the challenge bit returned by A. When $b_1 = 1$, encryption and decryption are done using $\mathsf{SE2.Enc}, \mathsf{SE2.Dec}$. This is exactly $\text{G}_0(A)$. When $b_1 = 0$, encryption returns a random string and decryption will always return $\perp$ (since we assumed that no trivial queries will be made). This is what happens in $\text{G}_2(A)$. From this, we have that

$$\mathbf{Adv}^{\text{ae2}}_{\mathsf{SE2}}(A) = \Pr[b_1' = 1 | b_1 = 1] - \Pr[b_1' = 1 | b_1 = 0] = \Pr[\text{G}_0(A)] - \Pr[\text{G}_2(A)].$$

The general strategy of C is to run A and simulate a decryption oracle. The decryption oracle first calls the verification oracle on the inputs it was passed, then returns $\perp$. Let E denote the event that, in $\text{G}_0(A)$, some query to Dec_0 returns $M \neq \perp$. This means that the corresponding query in $\mathbf{G}^{\text{auth1}}_{\mathsf{SE2}}(C)$ will set win to true. Therefore, $\Pr[\text{G}_0(A) | E] \leq \Pr[\mathbf{G}^{\text{auth1}}_{\mathsf{SE2}}(C)]$. If this event does not occur, Dec_0 will return $\perp$ on all decryption queries. This is no different from interacting with

decryption oracle $\textsc{Dec}_1$. Therefore, $\Pr[\mathrm{G}_0(A)|\neg E] = \Pr[\mathrm{G}_1(A)]$. Combining these statements, we have

$$\begin{aligned}&\Pr[\mathrm{G}_0(A)] - \Pr[\mathrm{G}_1(A)]\\ &\qquad \leq \Pr[\mathbf{G}^{\mathrm{auth1}}_{\mathsf{SE2}}(C)] + \Pr[\mathrm{G}_0(A)|\neg E] - \Pr[\mathrm{G}_1(A)] \leq \mathbf{Adv}^{\mathrm{auth1}}_{\mathsf{SE2}}(C).\end{aligned}$$

B also runs A with a simulated decryption oracle which just returns $\perp$ on all inputs. Let b_2 be the challenge bit in $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE2}}(B)$ and b_2' the bit returned by B. Note that since A does not repeat a nonce, neither does B. Since simulated decryption does not call any decryption oracle, $B \in \mathcal{A}^{\mathrm{ae2}}_{\mathrm{priv}} \cap \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}$. When $b_2 = 1$, the encryption oracle uses $\mathsf{SE2.Enc}$ and when $b_2 = 0$, the encryption oracle will just pick random strings. This corresponds to what happens in $\mathrm{G}_1(A), \mathrm{G}_2(A)$ respectively. Therefore,

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE2}}(A) = \Pr[b_2' = 1|b_2 = 1] - \Pr[b_2' = 1|b_2 = 0] = \Pr[\mathrm{G}_1(A)] - \Pr[\mathrm{G}_2(A)].$$

Combining the above equations will give us the desired result. ▌

We now show that $\mathsf{CAU2} = \mathbf{CAU2}[\mathsf{E}, \mathsf{H}, \ell]$ achieves AE2$[\mathcal{A}^{\mathrm{ae2}}_{\mathrm{priv}} \cap \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}]$-security and authenticity as defined above. Privacy is relatively straightforward. The desired theorem is presented in Theorem H.3 along with a proof sketch. Authenticity is a bit more involved, and we tackle this in Theorem H.4.

Theorem H.3 *Let* $\mathsf{CAU2} = \mathbf{CAU2}[\mathsf{E}, \mathsf{H}, \ell]$ *be the NBE2 scheme defined above. Then, let* $A \in \mathcal{A}^{\mathrm{ae2}}_{\mathrm{priv}} \cap \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}$ *be an adversary making* q_n *queries to its* NEW *oracle and* q_e *queries to its* ENC *per session. The total number of message blocks passed to the encryption oracle by* A *for any single session does not exceed* Q'*. Then we can construct adversary* B *such that:*

$$\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{CAU2}}(A) \leq \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{E}}(B).$$

B makes q_n queries to its NEW *oracle, and makes no more than $q_e + Q' + 1$ queries to its* FN *oracle for each user.*

Proof: (Sketch) We define B as in Fig. 22.B runs A with a simulated encryption oracle (there is no decryption oracle because A is privacy-only). This encryption oracle will run $\mathsf{CAU2}$, but substitute all block cipher evaluations $\mathsf{E.Ev}$ with calls to the FN oracle instead. Let b be the challenge bit in $\mathbf{G}^{\mathrm{prf}}_{\mathsf{E}}(B)$. When $b = 1$, encryption proceeds as in $\mathsf{CAU2}$. Since $A \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}$, for each i, no two calls to $\textsc{Fn}(i, \cdot)$ have the same input. This means that when $b = 0$ every call to FN returns a random element of $\{0,1\}^{\mathsf{E.bl}}$. So all the P_i and z are random strings masking the message blocks and tag respectively. This makes the output indistinguishable from a random string of length $|M| + \mathsf{E.bl} + \ell$. From this, we get the main statement in the theorem. ▌

Theorem H.4 *Let* $\mathsf{CAU2} = \mathbf{CAU2}[\mathsf{E}, \mathsf{H}, \ell]$ *be the NBE2 scheme defined above where* H *is an* (ϵ_1, ϵ_2)*-AXU function family. Let* $A \in \mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}$ *be an adversary making* q_n *calls to its* NEW *oracle,* q_e *calls to its* ENC *oracle per session and* q_d *calls to its* DEC *oracle per session. The total number of message blocks passed to the encryption oracle by* A *for any single session does not exceed* Q' *and the lengths of* C_2, H *passed to the decryption oracle by* A *do not exceed* ℓ_1', ℓ_2*, respectively. Then let* $Q = Q' + q_e + 1$ *and* $\ell_1 = \ell_1' + \mathsf{E.bl}$*. Then we can construct adversary* C *such that:*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{auth1}}_{\mathsf{CAU2}}(A) \leq \mathbf{Adv}^{\mathrm{sprp}}_{\mathsf{E}}(C) + q_n\left(\frac{Q^2 + Q + q_d^2 + 4q_dQ + 3q_d + Q2^{-\ell}}{2^{\mathsf{E.bl}+1}}\right)\\ + q_n(q_eq_d + q_d^2 + q_d2^{-\ell}) \cdot \epsilon_1(\ell_1, \ell_2) + q_n(q_d^2 - q_d) \cdot \epsilon_2(\ell_1, \ell_2)\end{aligned}$$

Adversary $B^{\text{New}_b, \text{Fn}_b}$

Initialize
$b' \leftarrow\$ A^{\text{New}_b^*, \text{Enc}_b^*}$
Finalize(b')

procedure New_b^*
New_b
$v \leftarrow v+1$
$k_v \leftarrow\$ \text{Fn}_b(v, 0^{\mathsf{E.bl}})$

procedure $\text{Enc}_b^*(i, N, M, H)$
$M_1 \| M_2 \| ... \| M_m \leftarrow 0^\ell \| M$
For $i = 1, 2, ..(m-1)$
  $P_i \leftarrow\$ \text{Fn}_b(i, N \| \langle i+1 \rangle_{\mathsf{E.bl}-\ell})$
  $C_{1,i} \leftarrow M_i \oplus P_i$
$P_m \leftarrow\$ \text{Fn}_b(i, N \| \langle m+1 \rangle_{\mathsf{E.bl}-\ell})$
$C_{1,m} \leftarrow M_m \oplus P_m[1..|M_m|]$
$z \leftarrow\$ \text{Fn}_b(i, N \| \langle 1 \rangle_{\mathsf{E.bl}-\ell})$
$\tau \leftarrow \mathsf{H.Ev}(k_i, (C_{1,1} \| .. \| C_{1,m}, H)) \oplus z$
Return $\tau \| C_{1,1} \| .. \| C_{1,m}$

Adversary $C^{\text{New}_c, \text{Fn}_c, \text{FnInv}_c}$

Initialize ; $A^{\text{New}_c^*, \text{Enc}_c^*, \text{Vf}_c^*}$
If win then $b' \leftarrow 1$ else $b' \leftarrow 0$; Finalize(b')

procedure New_c^*
New_c ; $v \leftarrow v+1$; $k_v \leftarrow\$ \text{Fn}_c(v, 0^{\mathsf{E.bl}})$

procedure $\text{Enc}_c^*(i, N, M, H)$
$M_1 \| M_2 \| ... \| M_m \leftarrow 0^\ell \| M$
For $i = 1, 2, ..(m-1)$
  $P_i \leftarrow\$ \text{Fn}_c(i, N \| \langle i+1 \rangle_{\mathsf{E.bl}-\ell})$; $C_{1,i} \leftarrow M_i \oplus P_i$
$P_m \leftarrow\$ \text{Fn}_c(i, N \| \langle m+1 \rangle_{\mathsf{E.bl}-\ell})$
$C_{1,m} \leftarrow M_m \oplus P_m[1..|M_m|]$
$z \leftarrow\$ \text{Fn}_c(i, N \| \langle 1 \rangle_{\mathsf{E.bl}-\ell})$
$\tau \leftarrow \mathsf{H.Ev}(k_i, (C_{1,1} \| .. \| C_{1,m}, H)) \oplus z$
Return $\tau \| C_{1,1} \| .. \| C_{1,m}$

procedure $\text{Vf}_c^*(i, C_2, H)$
$\tau \| C_{1,1} \| C_{1,2} \| ... \| C_{1,m} \leftarrow C_2$
$h \leftarrow \mathsf{H.Ev}(k_i, (C_{1,1} \| C_{1,2} \| ... \| C_{1,m}, H))$
$y \leftarrow\$ \text{FnInv}_c(K, \tau \oplus h)$; $N \leftarrow y[1..\ell]$
$P \leftarrow\$ \text{Fn}_c(i, N \| \langle 2 \rangle_{\mathsf{E.bl}-\ell})$; $M_1 \leftarrow P[1..|C_{1,1}|] \oplus C_{1,1}$
If $(M_1[1..\ell] \neq 0^\ell) \vee (y[(\ell+1)..\mathsf{E.bl}] \neq \langle 1 \rangle_{\mathsf{E.bl}-\ell})$ then
  Return false
win $\leftarrow$ true ; Return true

Figure 22: Left: Adversary B used in the proof of Theorem H.3. Right: Adversary C used in the proof of Theorem H.4.

C makes q_n queries to its New *oracle, no more than Q queries to its* Fn *oracle for each user and no more than q_d queries to its* Dec *oracle for each user.*

Proof: We define games $\text{G}_0, \text{G}_1, \text{G}_2, \text{G}_3$ as in Fig. 23, games G_3, G_4 as in Fig. 24 and games G_5, G_6 as in Fig. 25. Note that Initialize, Finalize are the same for all these games, and they are as depicted in Fig. 23. As before, we will assume that A makes no trivial verification queries. This means that A will not query $\text{Vf}(i, C_2, H)$ when C_2 was returned in a prior call to $\text{Enc}(i, N, M, H)$ for some N, M. We will also assume that all decryption queries feature ciphertexts of length at least $\mathsf{E.bl}$.

In G_0, we partially decrypt C_2 using $\mathsf{CAU2.Dec}$, so that we can check if the deciphered nonce has the correct suffix and the message has the correct prefix. This is exactly the two conditions which will decide if $\mathsf{CAU2.Dec}$ will return $\perp$ or not. By the definition of $\mathbf{G}^{\text{auth1}}_{\mathsf{CAU2}}$ above, we have that $\mathbf{Adv}^{\text{auth1}}_{\mathsf{CAU2}}(A) = \Pr[\text{G}_0(A)]$.

From here, we define the adversary C as in Fig. 22. This is such that the following equations hold:

$$\Pr[\text{G}_0(A)] - \Pr[\text{G}_1(A)] = \mathbf{Adv}^{\text{sprp}}_{\mathsf{E}}(C) \ , \tag{31}$$

$$\big|\Pr[\text{G}_1(A)] - \Pr[\text{G}_2(A)]\big| \leq \frac{q_n Q(Q+1)}{2^{\mathsf{E.bl}+1}} + \frac{q_n q_d(q_d+1)}{2^{\mathsf{E.bl}+1}} + \frac{2 q_n Q q_d}{2^{\mathsf{E.bl}}} \ , \tag{32}$$

procedure INITIALIZE // For all games
Return

procedure FINALIZE // For all games
Return win

Game G_0

procedure $\textsc{New}_0$
$v \leftarrow v+1$; $K_v \leftarrow\!\!\$\ \mathsf{SE2.KS}$

procedure $\textsc{Enc}_0(i, N, M, H)$
$C_2 \leftarrow\!\!\$\ \mathsf{SE2.Enc}(K_i, N, M, H)$
Return C_2

procedure $\textsc{Vf}_0(i, C_2, H)$
$M \leftarrow\!\!\$\ \mathsf{SE2.Dec}(K_i, C_2, H)$
If $(M \neq \bot)$ then win $\leftarrow$ true
Return $(M \neq \bot)$

Games $\boxed{\mathrm{G}_1}$,G_2

procedure $\textsc{New}_1$
$v \leftarrow v+1$; $k_v \leftarrow\!\!\$\ \textsc{Ev}_1^*(v, 0^{\mathsf{E.bl}})$

procedure $\textsc{Enc}_1(i, N, M, H)$
$M_1 \| M_1 \| ... \| M_m \leftarrow 0^\ell \| M$
For $i = 1, 2, ..(m-1)$ do
  $P_i \leftarrow\!\!\$\ \textsc{Ev}_1^*(i, N \| \langle i+1 \rangle_{\mathsf{E.bl}-\ell})$; $C_{1,i} \leftarrow M_i \oplus P_i$
$P_m \leftarrow\!\!\$\ \textsc{Ev}_1^*(i, N \| \langle m+1 \rangle_{\mathsf{E.bl}-\ell})$
$C_{1,m} \leftarrow M_m \oplus P_m[1..|M_m|]$; $z \leftarrow\!\!\$\ \textsc{Ev}_1^*(i, N \| \langle 1 \rangle_{\mathsf{E.bl}-\ell})$
$\tau \leftarrow \mathsf{H.Ev}(k_i, (C_{1,1} \| .. \| C_{1,m}, H)) \oplus z$; Return $\tau \| C_{1,1} \| .. \| C_{1,m}$

procedure $\textsc{Vf}_1(i, C_2, H)$
$\tau \| C_{1,1} \| C_{1,2} \| ... \| C_{1,m} \leftarrow C_2$
$h \leftarrow \mathsf{H.Ev}(k_i, (C_{1,1} \| C_{1,2} \| ... \| C_{1,m}, H))$
$y \leftarrow\!\!\$\ \textsc{In}_1^*(K, \tau \oplus h)$; $N \leftarrow y[1..\ell]$
$P \leftarrow\!\!\$\ \textsc{Ev}_1^*(i, N \| \langle 2 \rangle_{\mathsf{E.bl}-\ell})$; $M_1 \leftarrow P[1..|C_{1,1}|] \oplus C_{1,1}$
If $(M_1[1..\ell] \neq 0^\ell) \vee (y[(\ell+1)..\mathsf{E.bl}] \neq \langle 1 \rangle_{\mathsf{E.bl}-\ell})$ then
  Return false
win $\leftarrow$ true ; Return true

procedure $\textsc{Ev}_1^*(i, X)$
If $(\mathbf{Y}[i, X] = \bot)$ then
  $Y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{E.bl}}$
  If $Y \in \mathcal{Y}_i$ then bad $\leftarrow$ true ; $\boxed{Y \leftarrow\!\!\$\ \left(\{0,1\}^{\mathsf{E.bl}}\right) \setminus \mathcal{Y}_i}$
  $\mathbf{Y}[i, X] \leftarrow Y$; $\mathbf{X}[i, Y] \leftarrow X$
Return $\mathbf{Y}[i, X]$

procedure $\textsc{In}_1^*(i, Y)$
If $(\mathbf{X}[i, Y] = \bot)$ then
  $X \leftarrow\!\!\$\ \{0,1\}^{\mathsf{E.bl}}$
  If $X \in \mathcal{X}_i$ then bad $\leftarrow$ true ; $\boxed{X \leftarrow\!\!\$\ \left(\{0,1\}^{\mathsf{E.bl}}\right) \setminus \mathcal{X}_i}$
  $\mathbf{X}[i, Y] \leftarrow X$; $\mathbf{Y}[i, X] \leftarrow Y$
Return $\mathbf{X}[i, Y]$

Figure 23: Games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3, \mathrm{G}_4$ used in the proof of Theorem H.4. INITIALIZE, FINALIZE are common to all games in this proof.

$$\Pr[\mathrm{G}_2(A)] = \Pr[\mathrm{G}_3(A)] \,, \tag{33}$$

$$\left|\Pr[\mathrm{G}_3(A)] - \Pr[\mathrm{G}_4(A)]\right| \leq q_n q_e q_d \cdot \epsilon_1(\ell_1, \ell_2) + q_n q_d (q_d - 1) \cdot \epsilon_2(\ell_1, \ell_2) \,, \tag{34}$$

$$\Pr[\mathrm{G}_4(A)] = \Pr[\mathrm{G}_5(A)] \,, \tag{35}$$

$$\left|\Pr[\mathrm{G}_5(A)] - \Pr[\mathrm{G}_6(A)]\right| \leq \frac{q_n Q}{2^{\mathsf{E.bl}+\ell}} + \frac{q_n q_d}{2^\ell} \epsilon_1(\ell_1, \ell_2) \,, \tag{36}$$

$$\Pr[\mathrm{G}_6(A)] \leq \frac{q_n q_d}{2^{\mathsf{E.bl}}} \,, \tag{37}$$

We can combine these equations to derive the theorem statement. Now we derive equations

Games $\boxed{\mathrm{G}_3}$,G_4

procedure $\textsc{New}_3$
$v \leftarrow v+1$; $k_v \leftarrow\!\!\$\ \mathrm{Ev}_3^*(v, 0^{\mathsf{E.bl}})$

procedure $\textsc{Enc}_3(i, N, M, H)$
$M_1 \| M_2 \| ... \| M_m \leftarrow 0^\ell \| M$
For $i = 1, 2, ..(m-1)$
  $P_i \leftarrow\!\!\$\ \mathrm{Ev}_3^*(i, N \| \langle i+1 \rangle_{\mathsf{E.bl}-\ell})$; $C_{1,i} \leftarrow M_i \oplus P_i$
$P_m \leftarrow\!\!\$\ \mathrm{Ev}_3^*(i, N \| \langle m+1 \rangle_{\mathsf{E.bl}-\ell})$; $C_{1,m} \leftarrow M_m \oplus P_m[1..|M_m|]$
$z \leftarrow\!\!\$\ \mathrm{Ev}_3^*(i, N \| \langle 1 \rangle_{\mathsf{E.bl}-\ell})$; $\tau \leftarrow \mathsf{H.Ev}(k_i, (C_{1,1} \| .. \| C_{1,m}, H)) \oplus z$
Return $\tau \| C_{1,1} \| .. \| C_{1,m}$

procedure $\textsc{Vf}_3(i, C_2, H)$
$\tau \| C_{1,1} \| C_{1,2} \| ... \| C_{1,m} \leftarrow C_2$
$h \leftarrow \mathsf{H.Ev}(k_i, (C_{1,1} \| C_{1,2} \| ... \| C_{1,m}, H))$; $y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{E.bl}}$
If $(\mathbf{X}[i, \tau \oplus h] \neq \perp)$ then
  $X \leftarrow \mathbf{X}[i, \tau \oplus h]$
  If $(X[(\ell+1)..\mathsf{E.bl}] \neq \langle 1 \rangle_{\mathsf{E.bl}-\ell})$ then return $\perp$ else $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{y \leftarrow X}$
$\mathbf{X}[i, \tau \oplus h] \leftarrow y$; $\mathbf{Y}[i, y] \leftarrow \tau \oplus h$; $N \leftarrow y[1..\ell]$
$P \leftarrow\!\!\$\ \mathrm{Ev}_3^*(i, N \| \langle 2 \rangle_{\mathsf{E.bl}-\ell})$; $M_1 \leftarrow P[1..|C_{1,1}|] \oplus C_{1,1}$
If $(M_1[1..\ell] \neq 0^\ell) \vee (y[(\ell+1)..\mathsf{E.bl}] \neq \langle 1 \rangle_{\mathsf{E.bl}-\ell})$ then return false
$\mathsf{win} \leftarrow \mathsf{true}$; Return true

procedure $\mathrm{Ev}_3^*(i, X)$
If $(\mathbf{Y}[i, X] = \perp)$ then $Y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{E.bl}}$; $\mathbf{Y}[i, X] \leftarrow Y$; $\mathbf{X}[i, Y] \leftarrow X$
Return $\mathbf{Y}[i, X]$

Figure 24: Games $\mathrm{G}_3, \mathrm{G}_4$ used in the proof of Theorem H.4. Initialize, Finalize are as in Fig. 23.

(31),(32),(33),(34),(35),(36),(37) one by one.

Adversary C will run A with simulated encryption and verification oracles. These oracles perform encryption as in CAU1.Enc and the partial decryption discussed above, respectively, but use the oracles provided to C to perform block cipher enciphering and deciphering. The advantage of C is exactly the difference between the success of A when the block cipher operations are done using E and when the block cipher operations are a random bijection. Note that the random bijection is captured in $\mathrm{Ev}_1^*, \mathrm{In}_1^*$ in G_1 (with the boxed code). Here, we sample elements of $\{0,1\}^{\mathsf{E.bl}}$ without replacement. We do this by first selecting an element randomly from $\{0,1\}^{\mathsf{E.bl}}$. If this collides with a previously chosen element of the domain or range, we will select another value such that this won't happen. This gives us (31).

Next, $\mathrm{G}_1, \mathrm{G}_2$ are identical-until-bad. Our goal here is to model the block cipher using a random mapping instead of a random permutation. The bad flag is set in Ev_1^* when, within the same session, there is a collision between values of Y. Values of Y are added to $\mathcal{Y}_i$ in two locations, they are randomly selected in the Q calls to Ev_1^* and they are given as input in the q_d calls to In_1^*. There are three ways a collision between values of Y could have occurred: (1) between two calls to Ev_1^*, (2) between two calls to In_1^* and (3) between one call to Ev_1^* and a call to In_1^*. We bound the probability of each of case separately. Note that at most Q distinct calls to $\mathrm{Ev}_1^*(i, \cdot)$ were made, and at most q_d distinct calls to $\mathrm{In}_1^*(i, \cdot)$ were made. The probability of the first two events can be bounded using a birthday collision bound, giving us the first two terms of (32). Then, suppose $\mathrm{Ev}_1^*(i, X)$ sets bad in case (3). This means that the value of Y selected at random collided with

Games $\boxed{\mathrm{G}_5}$,G_6

procedure $\textsc{New}_5$
$v \leftarrow v+1$; $k_v \leftarrow\!\!\$\ \mathrm{Ev}_5^*(v, 0^{\mathsf{E.bl}})$

procedure $\textsc{Enc}_5(i, N, M, H)$
$M_1 \| M_2 \| ... \| M_m \leftarrow 0^\ell \| M$
For $i = 1, 2, ..(m-1)$ do $P_i \leftarrow\!\!\$\ \mathrm{Ev}_5^*(i, N \| \langle i+1 \rangle_{\mathsf{E.bl}-\ell})$; $C_{1,i} \leftarrow M_i \oplus P_i$
$P_m \leftarrow\!\!\$\ \mathrm{Ev}_5^*(i, N \| \langle m+1 \rangle_{\mathsf{E.bl}-\ell})$; $C_{1,m} \leftarrow M_m \oplus P_m[1..|M_m|]$
$z \leftarrow\!\!\$\ \mathrm{Ev}_5^*(i, N \| \langle 1 \rangle_{\mathsf{E.bl}-\ell})$; $\tau \leftarrow \mathsf{H.Ev}(k_i, (C_{1,1} \| .. \| C_{1,m}, H)) \oplus z$
Return $\tau \| C_{1,1} \| .. \| C_{1,m}$

procedure $\textsc{Vf}_5(i, C_2, H)$
$\tau \| C_{1,1} \| C_{1,2} \| ... \| C_{1,m} \leftarrow C_2$; $h \leftarrow \mathsf{H.Ev}(k_i, (C_{1,1} \| C_{1,2} \| ... \| C_{1,m}, H))$
$y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{E.bl}}$; $\mathbf{Y}[i, y] \leftarrow \tau \oplus h$; $N \leftarrow y[1..\ell]$; $N_1 \leftarrow N \| \langle 1 \rangle_{\mathsf{E.bl}-\ell}$; $P \leftarrow\!\!\$\ \{0,1\}^{\mathsf{E.bl}}$
If $(\mathbf{Y}[i, N_1] \neq \perp)$ then
  $Y \leftarrow \mathbf{Y}[i, N_1]$; $M' \leftarrow Y[1..|C_{1,1}|] \oplus C_{1,1}$
  If $(M'[1..\ell] \neq 0^\ell)$ then return $\perp$ else $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{P \leftarrow\!\!\$\ \mathbf{Y}[i, N_1]}$
$\mathbf{Y}[i, N_1] \leftarrow P$; $M_1 \leftarrow P[1..|C_{1,1}|] \oplus C_{1,1}$
If $(M_1[1..\ell] = 0^\ell) \wedge (y[(\ell+1)..\mathsf{E.bl}] = \langle 1 \rangle_{\mathsf{E.bl}-\ell})$ then $\mathsf{win} \leftarrow \mathsf{true}$; Return true
Return false

procedure $\mathrm{Ev}_5^*(i, X)$
If $(\mathbf{Y}[i, X] = \perp)$ then $\mathbf{Y}[i, X] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{E.bl}}$
Return $\mathbf{Y}[i, X]$

Figure 25: Games $\mathrm{G}_5, \mathrm{G}_6$ used in the proof of Theorem H.4. Initialize, Finalize are as in Fig. 23.

---

the input to a prior call to $\mathrm{In}_1^*(i, \cdot)$. Since at most q_d such calls could have occured, the probability of any one call to Ev^* setting bad in case (3) is bounded by $\frac{q_d}{2^{\mathsf{E.bl}}}$. Via a similar argument, the probability of any one call to $\mathrm{In}_1^*(i, \cdot)$ sets bad can be bounded by $\frac{Q}{2^{\mathsf{E.bl}}}$. Using the union bound, we can combine these to get the last term in (32)

In G_3, we replace the call to In_1^* (without the boxed code) in G_2 with pseudocode that first selects y at random from $\{0,1\}^{\mathsf{E.bl}}$ before checking if $\mathbf{X}[i, \tau \oplus h]$ has been initialized before. If it has, we check if the suffix of this entry is $\langle 1 \rangle_{\mathsf{E.bl}-\ell}$ and set y to this value if so. We return $\perp$ otherwise. Since the same check is also performed in G_2, the games stay equivalent so we have (33).

Games $\mathrm{G}_3, \mathrm{G}_4$ are identical-until-bad. There are two ways that the game G_3 can set bad. Let $\tau, h, C_{1,i}$ be as defined in the pseudocode of $\mathrm{Vf}_3(i, C_2, H)$. Then, the first way to set bad is if $\tau \oplus h$ was returned during some call to Ev_3^*. Since we only set bad to true if the suffix of $\mathbf{X}[i, \tau \oplus h]$ is $\langle 1 \rangle_{\mathsf{E.bl}-\ell}$, we need only consider the last call to Ev_3^* from each prior call to $\textsc{Enc}_3$. There are at most q_e of these calls. Let z' be the output of this call to Ev_3^*. Therefore, $z' = \tau \oplus h$ meaning that $z' \oplus \tau = \mathsf{H.Ev}(k_i, (C_{1,1} \| C_{1,2} \| ... \| C_{1,m}, H))$. Because we assumed the adversary does not make trivial verification queries, we are assured that at least one of τ or $C_{1,1} \| ... \| C_{1,m}$ returned by the encryption query will be different from the ones passed into the decryption query. Therefore, the probability that $z' = \tau \oplus h$ for each pair of encryption and decryption queries can be bounded by $\epsilon_1(\ell_1, \ell_2)$. The other way G_3 sets bad is if two calls to the verification oracle have colliding values of $\tau \oplus h$. Let $\tau, h, C_{1,i}, H$ be the variables defined in G_3 during the first of these calls. Let $\tau', h', C'_{1,i}, H'$ be the analogous values for the second call. This means that $\tau \oplus h = \tau' \oplus h$. So $\tau \oplus \tau' = \mathsf{H.Ev}(k_i, (C_{1,1} \| C_{1,2} \| ... \| C_{1,m}, H)) \oplus \mathsf{H.Ev}(k_i, (C'_{1,1} \| C'_{1,2} \| ... \| C'_{1,m}, H'))$. Since we assumed an

adversary that does not make repeated decryption queries, these queries are distinct and we can bound the probability of this is bounded by $\epsilon_2(\ell_1, \ell_2)$. Using the union bound over the sessions and over all choices of pairs of queries, we get (34).

To get G_5, we remove all references to $\mathbf{X}$ since it is no longer used. Then we do a similar thing to P as we did to y when transitioning from G_2 to G_3: we first pick P at random from $\{0,1\}^{\mathsf{E.bl}}$, then, if $\mathbf{Y}[i, N_1]$ has been initialized before, we set P to that value. Also like in G_2, G_3, we add an additional check that the message will be prefixed with 0^ℓ in the case where $\mathbf{Y}[i, N_1]$ (which is the block which will mask the value of M_1) has already been initialized. Finally, we invert the conditional at the end of VF_5 to return true when the conditional is met and false otherwise. All these changes maintain the equivalence of G_4 and G_5, giving us (35).

Games G_5, G_6 are identical-until-bad. We set bad to true when $\mathbf{Y}[i, N_1]$ has been initialized before. We can split the ways in which it can be initialized into two groups. First, suppose it was initialized in one of the Q calls to EV_5^*. At most Q entries in table $\mathbf{Y}$ were initialized in such a way. For each entry $\mathbf{Y}[i, Y]$, the probability that $N_1 = Y$ is $2^{-\mathsf{E.bl}}$. Additionally, the probability that $\mathbf{Y}[i, X] = C_{1,1}$ (so that $M'[1..\ell] = 0^\ell$) is $2^{-\ell}$. So we can bound the probability that bad is set in a call to EV_5^* by $\frac{q_n Q}{2^{\mathsf{E.bl}+\ell}}$. The other way we set bad is if, during some call to $\text{VF}_5(i, \tau \| C_{1,1} \| .. \| C_{1,m}, H)$ where we initialize $\mathbf{Y}[i, y]$ to $Y = \tau \oplus \mathsf{H.Ev}(k_i, C_{1,1} \| .. \| C_{1,m}, H)$, we have that $y = N_1$. To set bad, we need $Y[1..\ell] = C_{1,1}[1..\ell]$. This describes $2^{-\ell}$ of the possible choices of such a Y. For each of these, the probability that $Y = \tau \oplus \mathsf{H.Ev}(k_i, C_{1,1} \| .. \| C_{1,m}, H)$ can be bound using the fact that H is an (ϵ_1, ϵ_2)-AXU function family. We use the union bound over all sessions and all q_d decryption queries to give the second term in (36).

In G_6, both N and P are now being selected at random. This means that we can bound the probability that any one call to VF_5 (without the boxed code) will set win to true. In each call to VF_5, the two conditions under which win will be set to true are independent. We multiply their probabilities to get $2^{-\ell} 2^{-\mathsf{E.bl}+\ell} = 2^{-\mathsf{E.bl}}$. Using the union bound, we get (37). ∎

We now recall the theorem stated in Section 7, showing that $\mathsf{CAU2}$ is secure if H is an (ϵ_1, ϵ_2)-AXU function family and E is an sPRP. This follows from Lemma H.2 quite naturally. We sketch the proof below.

Theorem 7.1. *Let* $\mathsf{CAU2} = \mathbf{CAU2}[\mathsf{E}, \mathsf{H}, \ell]$ *be the NBE2 scheme defined above where* H *is an* (ϵ_1, ϵ_2)*-AXU function family. Let* $A \in \mathcal{A}^{\text{ae2}}_{\text{u-n}}$ *be an adversary making* q_n *calls to its* NEW *oracle,* q_e *calls to its* ENC *oracle per session and* q_d *calls to its* DEC *oracle per session. The total number of message blocks passed to the encryption oracle by* A *for any single session does not exceed* Q' *and the lengths of* C_2, H *passed to the decryption oracle by* A *do not exceed* ℓ_1', ℓ_2*, respectively. Then let* $Q = Q' + q_e + 1$ *and* $\ell_1 = \ell_1' + \mathsf{E.bl}$*. Then we can construct adversary* B *such that:*

$$\begin{aligned}\mathbf{Adv}^{\text{ae2}}_{\mathsf{CAU2}}(A) \leq & 2\mathbf{Adv}^{\text{sprp}}_{\mathsf{E}}(B) + q_n\left(\frac{Q^2 + Q + q_d^2 + 4q_dQ + 3q_d + Q2^{-\ell}}{2^{\mathsf{E.bl}+1}}\right) \\ & + q_n(q_eq_d + q_d^2 + q_d2^{-\ell}) \cdot \epsilon_1(\ell_1, \ell_2) + q_n(q_d^2 - q_d) \cdot \epsilon_2(\ell_1, \ell_2)\end{aligned}$$

B makes q_n queries to its NEW *oracle, no more than Q queries to its* FN *oracle for each user and no more than q_d queries to its* DEC *oracle for each user.*

Proof: (Sketch) Let B, C be the adversaries defined in Theorem H.3 and Theorem H.4 respectively. Combining the results of these theorems with Lemma H.2, we get the following bound:

$$\mathbf{Adv}^{\text{ae2}}_{\mathsf{CAU2}}(A) \leq \mathbf{Adv}^{\text{prf}}_{\mathsf{E}}(B) + \mathbf{Adv}^{\text{sprp}}_{\mathsf{E}}(C)$$

$$+ q_n\left(\frac{Q^2 + Q + q_d^2 + 4q_dQ + 3q_d + Q2^{-\ell}}{2^{\mathsf{E.bl}+1}}\right)$$
$$+ q_n(q_eq_d + q_d^2 + q_d2^{-\ell}) \cdot \epsilon_1(\ell_1, \ell_2) + q_n(q_d^2 - q_d) \cdot \epsilon_2(\ell_1, \ell_2)$$

Notice that adversary B can also play that sPRP game – it just ignores the FnInv oracle. The only difference between the two games is whether Fn, in the "ideal" case, will sample randomly from $\{0,1\}^{\mathsf{E.bl}}$ with replacement or without. We can bound this using a birthday collision bound. This gives us:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{E}}(B) \le \mathbf{Adv}^{\mathrm{sprp}}_{\mathsf{E}}(B) + \frac{q_nQ(Q+1)}{2^{\mathsf{E.bl}+1}}$$

The adversary B randomly picks one of B, C to run so $\mathbf{Adv}^{\mathrm{sprp}}_{\mathsf{E}}(B) = \frac{1}{2}\big(\mathbf{Adv}^{\mathrm{sprp}}_{\mathsf{E}}(B) + \mathbf{Adv}^{\mathrm{sprp}}_{\mathsf{E}}(C)\big)$. From here, we can derive the theorem statement. ∎