Nonces are Noticed: AEAD Revisited

Mihir Bellare[1], Ruth Ng[1], and Björn Tackmann[2]

[1] Department of Computer Science and Engineering
University of California San Diego, USA
{mihir,ring}@eng.ucsd.edu
[2] IBM Research – Zurich, Rüschlikon, Switzerland
bta@zurich.ibm.com

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 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 [14, 11, 33], 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, 22] (used in TLS), OCB [50, 47, 35], CAESAR candidates [17] 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 $\underline{\text{T}}$ransmit $\underline{\text{N}}$once 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 $\textsc{Enc}$ (taking nonce N, message M and header H to return a ciphertext C_2) and decryption oracle $\textsc{Dec}$ (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 $\textsc{Enc}$ 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 $\textsc{Dec}$ 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 $\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}$ be the class of $\underline{\text{u}}$nique-$\underline{\text{n}}$once adversaries, meaning ones that do not reuse a nonce across their $\textsc{Enc}$ queries. We refer to AE2[$\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}$]-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, 17, 35, 31, 53, 43, 26, 25, 18, 30], 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 $\mathsf{SE1.nl}$. *All our transforms have zero ciphertext overhead.* One challenge in achieving this is that nonce lengths like $\mathsf{SE1.nl} = 96$ 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 $\mathsf{SE1}$ into a basic-AE2-secure NBE2 scheme $\mathsf{SE2}$. (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 $\mathsf{SE1}$, the idea of scheme $\mathsf{SE2} = \mathbf{HN1}[\mathsf{SE1}, \mathsf{F}]$ 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 $\mathsf{SE1.mccl}$ to be non-trivial, like at least 128, which is not true for all $\mathsf{SE1}$. Scheme $\mathsf{SE2} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ 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 $\mathsf{SE2} = \mathbf{HN3}[\mathsf{SE1}, \mathsf{F}]$ uses the result of PRF F on the actual nonce as a derived nonce under which to run $\mathsf{SE1}$. This is similar to SIV [51, 43]; the difference is to achieve AE2 rather than

NBE2 scheme	**AE2-security provided**	
	Basic	Advanced
$\mathbf{HN1}[\mathsf{SE1},\mathsf{F}]$	Yes	Yes
$\mathbf{HN2}[\mathsf{SE1},\ell,\mathsf{E},\mathsf{Spl}]$	Yes	Yes if $\ell \geq 128$
$\mathbf{HN3}[\mathsf{SE1},\mathsf{F}]$	Yes	No
$\mathbf{HN4}[\mathsf{SE1},\ell,\mathsf{F}]$		Yes
$\mathbf{HN5}[\mathsf{TE},\ell,\ell_t]$		Yes

Fig. 1. Security attributes of the NBE2 schemes defined by our Hide-Nonce (HN) transforms. In the table $\mathsf{SE1}$ 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.

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 [32, 52]. 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[$\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}nmh}}$]-security where $\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}nmh}}$ is the class of $\underline{\text{u}}$nique-$\underline{\text{n}}$once, $\underline{\text{m}}$essage, $\underline{\text{h}}$eader adversaries, meaning ones that do not repeat a query to their Enc oracle. The presence of well-analyzed advanced-AE1-secure NBE1 schemes [51, 28, 26, 25, 18] 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, 43] 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 [14] 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 [28] yields correspondingly fast, advanced-AE2-secure NBE2.

Dedicated transforms. While our generic transforms are already able, with low overhead, to immunize GCM [40, 22] —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 [19]. The formalization of Namprempre, Rogaway and Shrimpton (NRS) [44] 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 [28], online AE [23, 29], committing AE [24, 21], 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 [25] to an advanced-AE2-secure NBE2 scheme, the bounds we get are inferior to those of [18]. Bridging this gap to get advanced-AE2-secure NBE2 with security bounds like [18] 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, 30] 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 $1 \leq i \leq j \leq |Z|$, 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 $0 \leq i < 2^n$ 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 $x \leftarrow\!\!\$\, X$ 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 $y \leftarrow\!\!\$\, A^{\mathrm{O}_1,\cdots}(x_1, \ldots)$ 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 [15]. 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 $\perp$.

```
Game $\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}$

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 $(\mathbf{Y}[i,X] = \perp)$ then
    $Y_0 \leftarrow\$ \{0,1\}^{\mathsf{F.ol}}$
    $Y_1 \leftarrow \mathsf{F.Ev}(K_i, X)$
    $\mathbf{Y}[i,X] \leftarrow Y_b$
Return $\mathbf{Y}[i,X]$

procedure Finalize$(b')$
Return $(b = b')$
```

```
Game $\mathbf{G}^{\mathrm{stprp}}_{\mathsf{TE}}$

procedure Initialize
$b \leftarrow\$ \{0,1\}$

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

procedure Fn$(i, T, X)$
If $(\mathbf{Y}[i,T,X] = \perp)$ then
    $Y_0 \leftarrow\$ \{0,1\}^{|X|} \setminus \mathcal{Y}_{i,T}$ ; $Y_1 \leftarrow \mathsf{TE.Ev}(K_i, T, X)$
    $\mathbf{Y}[i,T,X] \leftarrow Y_b$ ; $\mathbf{X}[i,T,Y_b] \leftarrow X$
    $\mathcal{X}_{i,T} \leftarrow \mathcal{X}_{i,T} \cup \{X\}$ ; $\mathcal{Y}_{i,T} \leftarrow \mathcal{Y}_{i,T} \cup \{Y_b\}$
Return $\mathbf{Y}[i,T,X]$

procedure FnInv$(i, T, Y)$
If $(\mathbf{X}[i,T,Y] = \perp)$ then
    $X_0 \leftarrow\$ \{0,1\}^{|Y|} \setminus \mathcal{X}_{i,T}$ ; $X_1 \leftarrow \mathsf{TE.In}(K_i, T, Y)$
    $\mathbf{X}[i,T,Y] \leftarrow X_b$ ; $\mathbf{Y}[i,T,X_b] \leftarrow Y$
    $\mathcal{X}_{i,T} \leftarrow \mathcal{X}_{i,T} \cup \{X_b\}$ ; $\mathcal{Y}_{i,T} \leftarrow \mathcal{Y}_{i,T} \cup \{Y\}$
Return $\mathbf{X}[i,T,Y]$

procedure Finalize$(b')$
Return $(b = b')$
```

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

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 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 $\mathsf{F.In}: \{0,1\}^{\mathsf{F.kl}} \times \{0,1\}^{\mathsf{F.ol}} \to \mathsf{F.D} \cup \{\perp\}$ 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) $\mathsf{F.In}(K,y) = \perp$ 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 $\perp$. It is required that any $\textsc{Fn}(i,X)$ 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 [37, 28] 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 $\textsc{Fn}$ and an inversion oracle $\textsc{FnInv}$. 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 $\textsc{Fn}(i,T,X)$ or $\textsc{FnInv}(i,T,Y)$ 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 $\mathbf{Adv}^{\mathrm{stprp}}_{\mathsf{TE}}(A) = 2\Pr[\mathbf{G}^{\mathrm{stprp}}_{\mathsf{TE}}(A)] - 1$.

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 $H \in \mathsf{SE1.HS}$.

```
Game $\mathbf{G}^{\mathrm{ae1}}_{\mathsf{SE1}}$

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

procedure NEW
$v \leftarrow v+1$ ; $K_v \leftarrow\$ \mathsf{SE1.KS}$

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

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

```
Game $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE2}}$

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

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

procedure ENC$(i, N, M, H)$
If $(b = 1)$ then
    $C_2 \leftarrow \mathsf{SE2.Enc}(K_i, N, M, H)$
    Else $C_2 \leftarrow\$ \mathsf{SE2.CS}(|N|, |M|, |H|)$
$\mathbf{M}[i, C_2, H] \leftarrow M$ ; Return $C_2$

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 $(b = b')$
```

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

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

AE1 game and advantage. Let $\mathsf{SE1}$ be an NBE1 scheme and A an adversary. We associate to them the game $\mathbf{G}^{\mathrm{ae1}}_{\mathsf{SE1}}(A)$ 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 $\mathbf{Adv}^{\mathrm{ae1}}_{\mathsf{SE1}}(A) = 2\Pr[\mathbf{G}^{\mathrm{ae1}}_{\mathsf{SE1}}(A)] - 1$. 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 ENC(i, N, M, H) query of A satisfy $1 \leq i \leq v$, $N \in \mathsf{SE1.NS}$, $M \in \mathsf{SE1.MS}$ 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, $\textsc{Enc}$ 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 $\textsc{Enc}$. It is required that any $\textsc{Dec}(i, N, C_1, H)$ query of A satisfy $1 \leq i \leq v$, $N \in \mathsf{SE1.NS}$ 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, $\textsc{Dec}$ 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 $\textsc{Enc}$, 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 $\mathcal{A}_1^{\text{ae1}}$ allows us to recover the single-user setting. Different security notions in the literature are then captured as AE1[$\mathcal{A}$]-security for different classes of adversaries $\mathcal{A}$, as we illustrate below:

- $\mathcal{A}_{\text{u-n}}^{\text{ae1}}$ is the class of adversaries whose $\textsc{Enc}$ queries never repeat a user-nonce pair. AE1[$\mathcal{A}_{\text{u-n}}^{\text{ae1}} \cap \mathcal{A}_1^{\text{ae1}}$]-security is thus AEAD as defined in [46, 48].
- AE1[$\mathcal{A}_{\text{u-n}}^{\text{ae1}}$]-security is the extension of this to the multi-user setting as defined in [16], which we have referred to as basic AE1-security in Section 1.
- Adversaries in $\mathcal{A}_{\text{u-nmh}}^{\text{ae1}} \supseteq \mathcal{A}_{\text{u-n}}^{\text{ae1}}$ are allowed to re-use a user-nonce pair across $\textsc{Enc}$ queries as long as they never repeat an entire query. AE1[$\mathcal{A}_{\text{u-nmh}}^{\text{ae1}} \cap \mathcal{A}_1^{\text{ae1}}$]-security is misuse resistant AE [51].
- AE1[$\mathcal{A}_{\text{u-nmh}}^{\text{ae1}}$]-security is the extension of this to the multi-user setting [18], which we have referred to as advanced-AE1-security in Section 1.
- Adversaries in $\mathcal{A}_{\text{r-n}}^{\text{ae1}}$ pick the nonces in their $\textsc{Enc}$ 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 [12].) No restriction is placed on how the adversary picks nonces in $\textsc{Dec}$ queries. AE1[$\mathcal{A}_{\text{r-n}}^{\text{ae1}} \cap \mathcal{A}_1^{\text{ae1}}$]-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[$\mathcal{A}_{\text{priv}}^{\text{ae1}} \cap \mathcal{A}_{\text{u-n}}^{\text{ae1}}$]-security. AE1[$\mathcal{A}_{\text{priv}}^{\text{ae1}} \cap \mathcal{A}_{\text{u-n}}^{\text{ae1}} \cap \mathcal{A}_1^{\text{ae1}}$]-security is $\mathsf{IND\$\text{-}CPA}$ security, as defined in [46].

Further adversary classes can be defined to capture limited nonce reuse [18] or other resource restrictions.

The following says that AE1[$\mathcal{A}^{\mathrm{ae1}}_{\mathrm{u\text{-}n}}$]-security implies AE1[$\mathcal{A}^{\mathrm{ae1}}_{\mathrm{r\text{-}n}}$]-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 1. *Let* $\mathsf{SE1}$ *be an NBE1 scheme. Given adversary* $A_{\mathrm{rn}} \in \mathcal{A}^{\mathrm{ae1}}_{\mathrm{r\text{-}n}}$ *making at most* u New *queries and at most* q Enc *queries per user, we construct adversary* $A_{\mathrm{un}} \in \mathcal{A}^{\mathrm{ae1}}_{\mathrm{u\text{-}n}}$ *such that*

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

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 $\mathsf{SE2.Dec} : \mathsf{SE2.KS} \times \{0,1\}^* \times \mathsf{SE2.HS} \to \mathsf{SE2.MS} \cup \{\bot\}$ 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 $\bot$. (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 $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE2}}(A)$ 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 $\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE2}}(A) = 2\Pr[\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE2}}(A)] - 1$. 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 $\textsc{Enc}(i,N,M,H)$ query of A satisfy $1 \le i \le v$, $N \in \mathsf{SE2.NS}$, $M \in \mathsf{SE2.MS}$ 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. When $b = 0$, 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 $\bot$ everywhere, and holds ciphertexts returned by Enc. It is required that any $\textsc{Dec}(i, C_2, H)$ query of A satisfy $1 \le i \le v$ and $H \in \mathsf{SE2.HS}$. When the challenge bit b is 1, the decryption oracle will perform decryption as stipulated by $\mathsf{SE2.Dec}$, using the key for the indicated user i. When $b = 0$, $\textsc{Dec}$ 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 $\textsc{Enc}$ 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 $\textsc{Enc}$ 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 $\textsc{Enc}$ queries uniformly and independently at random from $\mathsf{SE2.NS}$. 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 [24], 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 $\mathsf{SE2}$ is nonce-recovering if there exists a deterministic nonce-plus-message recovery algorithm $\mathsf{SE2.NMR}$ 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 $C_1 \leftarrow \mathsf{SE1}.\mathsf{Enc}(K, N, M, H)$ 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 $M \leftarrow \mathsf{SE1}.\mathsf{Dec}(K, N, C_1, H)$. 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 AE1[$\mathcal{A}^{\mathrm{ae1}}_{\mathrm{u\text{-}n}}$]-secure, or it might even be AE1[$\mathcal{A}^{\mathrm{ae1}}_{\mathrm{u\text{-}nmh}}$]-secure. Regardless, we claim that NBE2 scheme $\mathsf{SE}_{\mathrm{TN}} = \mathbf{TN}[\mathsf{SE1}]$ fails to be AE2[$\mathcal{A}^{\mathrm{ae2}}_{\mathrm{priv}} \cap \mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}$]-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 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 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[$\mathcal{A}^{\mathrm{ae1}}_{\text{u-n}}$]-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[$\mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}$]-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 $\mathcal{A}^{\mathrm{ae2}}_{\text{u-n}}$— 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 $\mathcal{A}^{\mathrm{ae2}}_{\text{u-nmh}}$.

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 [22]. We refer to this value as the minimal core-ciphertext length of the scheme $\mathsf{SE1}$, 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 $\mathsf{SE1.mccl}$ to be non-trivial to provide security.

All transforms here use two keys, meaning the key for the constructed NBE2 scheme $\mathsf{SE2}$ is a pair consisting of a key for a PRF and a key for $\mathsf{SE1}$. 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 $\mathsf{SE2}$ constructed by any of our transforms from an NBE1 scheme $\mathsf{SE1}$, 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 $\mathsf{SE2}$ than in $\mathbf{TN}[\mathsf{SE1}]$. 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 the full version.

Theorem 2. *Let* $\mathsf{SE}_{\mathrm{HN1}} = \mathbf{HN1}[\mathsf{SE1},\mathsf{F}]$ *be obtained as above. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}$*, 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*

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

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, 35]— so in the absence of stealing, the nonce would be padded before enciphering, leading to ciphertext overhead. Second, while we show here (Theorem 3) that the scheme preserves basic security regardless of the amount ℓ stolen, we show later (Theorem 6) that it preserves even advanced security if ℓ is non-trivial (128 bits or more). We now proceed to the full description.

```
$\mathsf{SE}_{\mathrm{HN1}}.\mathsf{Enc}((K_{\mathsf{F}}, K_1), N, M, H)$
    $C_1 \leftarrow \mathsf{SE1.Enc}(K_1, N, M, H)$
    $x \leftarrow C_1[1..\mathsf{F.il}]$ ; $P \leftarrow \mathsf{F}(K_{\mathsf{F}}, x)$
    $Y \leftarrow P \oplus N$ ; $C_2 \leftarrow Y \| C_1$
    Return $C_2$

$\mathsf{SE}_{\mathrm{HN1}}.\mathsf{Dec}((K_{\mathsf{F}}, K_1), C_2, H)$
    If $(|C_2| < \mathsf{SE1.nl} + \mathsf{F.il})$ then return $\perp$
    $Y \| C_1 \leftarrow C_2$ ; $x \leftarrow C_1[1..\mathsf{F.il}]$ ; $P \leftarrow \mathsf{F}(K_{\mathsf{F}}, x)$
    $N \leftarrow P \oplus Y$ ; $M \leftarrow \mathsf{SE1.Dec}(K_1, N, C_1, H)$
    Return $M$
```

```
$\mathsf{SE}_{\mathrm{HN2}}.\mathsf{Enc}((K_{\mathsf{E}}, K_1), N, M, H)$
    $C_1 \leftarrow \mathsf{SE1.Enc}(K_1, N, M, H)$
    $(x, y) \leftarrow \mathsf{Spl.Ev}(\ell, C_1)$
    $C_{2,1} \leftarrow \mathsf{E.Ev}(K_{\mathsf{E}}, N \| x)$
    $C_2 \leftarrow C_{2,1} \| y$
    Return $C_2$

$\mathsf{SE}_{\mathrm{HN2}}.\mathsf{Dec}((K_{\mathsf{E}}, K_1), C_2, H)$
    If $(|C_2| < \mathsf{E.bl})$ then return $\perp$
    $N \| x \leftarrow \mathsf{E.In}(K_{\mathsf{E}}, C_2[1..\mathsf{E.bl}])$
    $y \leftarrow C_2[\mathsf{E.bl}+1..|C_2|]$ ; $C_1 \leftarrow \mathsf{Spl.In}(x, y)$
    If $(C_1 = \perp)$ then return $\perp$
    $M \leftarrow \mathsf{SE1.Dec}(K_1, N, C_1, H)$ ; Return $M$
```

```
$\mathsf{SE}_{\mathrm{HN3}}.\mathsf{Enc}((K_{\mathsf{F}}, K_1), N, M, H)$
    $N_1 \leftarrow \mathsf{F.Ev}(K_{\mathsf{F}}, N)$
    $C_1 \leftarrow \mathsf{SE1.Enc}(K_1, N_1, M, H)$
    $C_2 \leftarrow N_1 \| C_1$ ; Return $C_2$

$\mathsf{SE}_{\mathrm{HN3}}.\mathsf{Dec}((K_{\mathsf{F}}, K_1), C_2, H)$
    If $(|C_2| < \mathsf{F.ol})$ then return $\perp$
    $N_1 \| C_1 \leftarrow C_2$ ; $M \leftarrow \mathsf{SE1.Dec}(K_1, N_1, C_1, H)$
    Return $M$
```

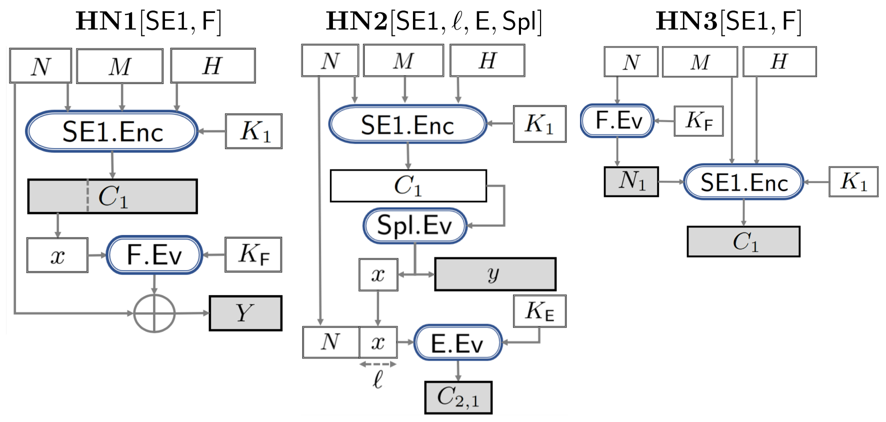

Fig. 4. Top: Encryption and decryption algorithms of the NBE2 schemes constructed by our basic transforms. From top to bottom: $\mathsf{SE}_{\mathrm{HN1}} = \mathbf{HN1}[\mathsf{SE1}, \mathsf{F}]$, $\mathsf{SE}_{\mathrm{HN2}} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ and $\mathsf{SE}_{\mathrm{HN3}}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

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 2 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 the full version of this paper [12].

Theorem 3. *Let* $\mathsf{SE}_{\mathrm{HN2}} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ *be obtained as above. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}$*, 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*

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

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$[\mathcal{A}^{\mathrm{ae1}}_{\mathrm{r\text{-}n}}]$-secure. (Recall this is AE1-security for the class of adversaries that pick the nonce at random.) By Proposition 1 this is implied by its being AE1$[\mathcal{A}^{\mathrm{ae1}}_{\mathrm{u\text{-}n}}]$-secure (that is, basic-AE1-secure). Assuming additionally that F is a PRF, the following says that **HN3**$[\mathsf{SE1},\mathsf{F}]$ is AE2$[\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}]$-secure (that is, basic-AE2-secure). The proof is in the full version.

Theorem 4. *Let* $\mathsf{SE}_{\mathrm{HN3}} = \mathbf{HN3}[\mathsf{SE1},\mathsf{F}]$ *be obtained as above. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}$ *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* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\mathrm{r\text{-}n}}$ *and* B *such that*

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

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$[\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}nmh}}]$-security. We discuss various transforms for this purpose.

Advanced security of **HN1***.* We showed in Theorem 2 that **HN1** preserves basic security. It turns out that it also preserves advanced security. The following says that if the starting NBE1 scheme $\mathsf{SE1}$ 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 the full version.

Theorem 5. *Let* $\mathsf{SE}_{\mathrm{HN1}} = \mathbf{HN1}[\mathsf{SE1},\mathsf{F}]$ *be obtained as above. Then, given adversary* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}nmh}}$*, 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* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\mathrm{u\text{-}nmh}}$ *and* B *such that*

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

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*.*

```
$\underline{\mathsf{SE}_{\mathrm{HN4}}.\mathsf{Enc}((K_{\mathsf{F}}, K_1), N, M, H)}$
$N_1 \leftarrow \mathsf{F.Ev}(K_{\mathsf{F}}, (N, M, H))$
$C_1 \leftarrow \mathsf{SE1.Enc}(K_1, N_1, N \| M, H)$
$C_2 \leftarrow N_1 \| C_1$
Return $C_2$

$\underline{\mathsf{SE}_{\mathrm{HN4}}.\mathsf{Dec}((K_{\mathsf{F}}, K_1), C_2, H)}$
If $(|C_2| < \mathsf{F.ol})$ then return $\bot$
$N_1 \| C_1 \leftarrow C_2$ ; $X \leftarrow \mathsf{SE1.Dec}(K_1, N_1, C_1, H)$
If $(X = \bot)$ then return $\bot$
$N \| M \leftarrow X$ ; $T \leftarrow \mathsf{F.Ev}(K_{\mathsf{F}}, (N, M, H))$
If $(T = N_1)$ then return $M$ else return $\bot$

$\underline{\mathsf{SE}_{\mathrm{HN5}}.\mathsf{Enc}(K_{\mathsf{TE}}, N, M, H)}$
$C_2 \leftarrow \mathsf{TE.Ev}(K_{\mathsf{TE}}, H, 0^{\ell_t} \| N \| M)$
Return $C_2$

$\underline{\mathsf{SE}_{\mathrm{HN5}}.\mathsf{Dec}(K_{\mathsf{TE}}, C_2, H)}$
$X \leftarrow \mathsf{TE.In}(K_{\mathsf{TE}}, H, C_2)$
If $X[1..\ell_t] \neq 0^{\ell_t}$ then return $\bot$
$N \| M \leftarrow X[(\ell_t + 1)..|X|]$ ; Return $M$
```

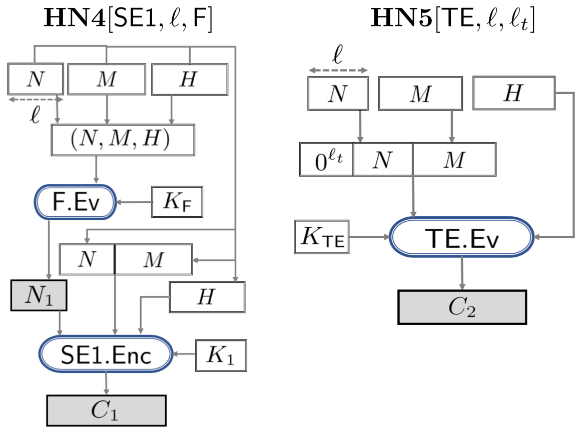

Fig. 5. Top: Encryption and decryption algorithms of the NBE2 schemes constructed by our advanced transforms. From top to bottom: $\mathsf{SE}_{\mathrm{HN4}} = \mathbf{HN4}[\mathsf{SE1}, \ell, \mathsf{F}]$ and $\mathsf{SE}_{\mathrm{HN5}} = \mathbf{HN5}[\mathsf{TE}, \ell, \ell_t]$. Bottom: Diagrams illustrating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

Advanced security of **HN2**. We showed in Theorem 3 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[$\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}nmh}}$]-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 $\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}nmh}}$-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 the full version of this paper, we prove this theorem and describe the attack alluded to above in detail, showing that the bound in Theorem 6 is tight [12].

Theorem 6. *Let* $\mathsf{SE}_{\text{HN2}} = \mathbf{HN2}[\mathsf{SE1}, \ell, \mathsf{E}, \mathsf{Spl}]$ *be obtained as above. Then, given adversary* $A_2 \in \mathcal{A}^{\text{ae2}}_{\text{u-nmh}}$*, making* q_n *queries to its* New *oracle and* q_e *queries per user to its* Enc *oracle, we construct adversaries* $A_1 \in \mathcal{A}^{\text{ae1}}_{\text{u-nmh}}$ *and* B *such that*

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

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 $\mathsf{SE1.mccl} \geq 128$ (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[$\mathcal{A}^{\text{ae2}}_{\text{u-nmh}}$]-security is achieved, assuming $\mathsf{SE1}$ is AE1[$\mathcal{A}^{\text{ae1}}_{\text{u-nmh}}$]-secure. The latter assumption, however, is not satisfactory here because AE1[$\mathcal{A}^{\text{ae1}}_{\text{u-nmh}}$]-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[$\mathcal{A}^{\text{ae1}}_{\text{r-n}} \cap \mathcal{A}^{\text{ae1}}_{\text{priv}}$]-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 $\mathsf{SE}_{\text{HN4}} = \mathbf{HN4}[\mathsf{SE1}, \ell, \mathsf{F}]$ whose encryption and decryption algorithms are shown in Figure 5. A key (K_{F}, K_1) for SE_{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 $\mathsf{SE}_{\text{HN4}}.\mathsf{KS} = \{0,1\}^{\mathsf{F.kl}} \times \mathsf{SE1.KS}$. The message and header spaces are unchanged, and the nonce space is $\mathsf{SE}_{\text{HN4}}.\mathsf{NS} = \{0,1\}^{\ell}$. The parsing in the second line of the decryption algorithm SE_{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 [43]. 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 AE1$[\mathcal{A}^{\mathrm{ae1}}_{\mathrm{r\text{-}n}} \cap \mathcal{A}^{\mathrm{ae1}}_{\mathrm{priv}}]$-security. By Proposition 1 this is implied by its being AE1$[\mathcal{A}^{\mathrm{ae1}}_{\mathrm{u\text{-}n}} \cap \mathcal{A}^{\mathrm{ae1}}_{\mathrm{priv}}]$-secure. Assuming additionally that F is a PRF, the following says that $\mathbf{HN4}[\mathsf{SE1}, \ell, \mathsf{F}]$ is AE2$[\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}nmh}}]$-secure. The proof is in the full version [12].

Theorem 7. *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* $A_2 \in \mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}nmh}}$ *making* q_n *queries to its* New *oracle and* q_e, q_d *encryption and decryption queries for each user, respectively, we construct adversaries* $A_1 \in \mathcal{A}^{\mathrm{ae1}}_{\mathrm{r\text{-}n}} \cap \mathcal{A}^{\mathrm{ae1}}_{\mathrm{priv}}$ *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 [14] to provide advanced-AE2-security. Instantiation via a fast tweakable cipher like AEZ [28] 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 [28] 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 $\ell_z \geq 0$ be the number of bits of redundancy we introduce to provide authenticity [14]. Our transform defines NBE2 scheme $\mathsf{SE}_{\mathrm{HN5}} = \mathbf{HN5}[\mathsf{TE}, \ell, \ell_z]$ 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 $\ell_z + |N| + |M|$, so $\mathsf{SE}_{\mathrm{HN5}}.\mathsf{CS}(\ell_n, \ell_m, \ell_h) = \{0,1\}^{\ell_z + \ell + \ell_m}$. 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_z bits, which is effectively optimal [28].

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_z is sufficiently large.

Theorem 8. *Let* $\mathsf{SE}_{\mathrm{HN5}} = \mathbf{HN5}[\mathsf{TE}, \ell, \ell_z]$ *be obtained as above. Let* $A \in \mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}nmh}}$ *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* $\ell_2 \geq \ell_z$ *per user to its* Dec *oracle. We construct adversary* B *such that*

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

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 Bellare and Tackmann [16], 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 $1 \leq \ell < \mathsf{E.bl}$. 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 $\mathsf{H.ol} = \mathsf{H.kl} = \mathsf{E.bl}$. The full description of CAU1.Enc and CAU1.Dec is in the full version [12].

AES-GCM, as proposed by McGrew and Viega [40] and standardized by NIST [22], 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[$\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}$]-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 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] [13].

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 slightly different 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 [36, 16, 34, 2] and others). We refer the reader to the full version of this paper for a description of these notions of security [12].

Theorem 9. *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* $A \in \mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}$ *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:*

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

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 the full version of this paper. Future work can apply the techniques used in recent work to improve upon this bound [16, 38, 30].

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.

```
$\underline{\mathsf{CAU2.Enc}(K, N, M, H)}$
$C_2 \leftarrow \mathsf{CAU1.Enc}(K, N, 0^{\ell} \| M, H)$
Return $C_2$
```

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

Fig. 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$[\mathcal{A}^{\mathrm{ae2}}_{\mathrm{u\text{-}n}}]$-secure variant of GCM.

8 A real-world perspective

In addition to bridging the gap between theory and usage, our framework allows us to formalize 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 [27]).

Acknowledgments. We thank the anonymous reviewers for their feedback and suggestions. Bellare was supported in part by NSF grants CNS-1526801 and CNS-1717640, ERC Project ERCC FP7/615074 and a gift from Microsoft. Ng was supported by DSO National Labs. Tackmann was supported in part by the Swiss National Science Foundation (SNF) via Fellowship No. P2EZP2_155566 and NSF grant CNS-1228890.

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

2. Abed, F., Fluhrer, S.R., Forler, C., List, E., Lucks, S., McGrew, D.A., Wenzel, J.: Pipelineable on-line encryption. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540. Springer.
3. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873. Springer.
4. Ashur, T., Dunkelman, O., Luykx, A.: Boosting authenticated encryption robustness with minimal modifications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403. Springer.
5. Aumasson, J., Babbage, S., Bernstein, D., Cid, C., Daemen, J., Dunkelman, O., Gaj, K., Gueron, S., Junod, P., Langley, A., McGrew, D., Paterson, K., Preneel, B., Rechberger, C., Rijmen, V., Robshaw, M., Sarkar, P., Schaumont, P., Shamir, A., Verbauwhede, I.: CHAE: Challenges in authenticated encryption. ECRYPT-CSA D1.1, Revision 1.05 (March 2017), `https://chae.cr.yp.to/whitepaper.html`
6. Barbosa, M., Farshim, P.: Indifferentiable authenticated encryption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991.
7. Barwell, G., Martin, D.P., Oswald, E., Stam, M.: Authenticated encryption in the face of protocol and side channel leakage. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624. Springer.
8. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807. Springer.
9. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: The cascade construction and its concrete security. In: 37th FOCS.IEEE Computer Society Press (Oct 1996).
10. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS. IEEE Computer Society Press (Oct 1997).
11. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976. Springer.
12. Bellare, M., Ng, R., Tackmann, B.: Nonces are noticed: AEAD revisited. Cryptology ePrint Archive Report 2019/624, (2019), `http://eprint.iacr.org/2019/624`
13. Bellare, M., Rogaway, P.: On the construction of variable-input-length ciphers. In: Knudsen, L.R. (ed.) FSE'99. LNCS, vol. 1636. Springer.
14. Bellare, M., Rogaway, P.: Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976. Springer.
15. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004. Springer.
16. Bellare, M., Tackmann, B.: The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814. Springer.
17. Bernstein, D.: CAESAR call for submissions, final (2014.01.27) (2014), `https://competitions.cr.yp.to/caesar-call.html`
18. Bose, P., Hoang, V.T., Tessaro, S.: Revisiting AES-GCM-SIV: Multi-user security, faster key derivation, and better bounds. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820. Springer.
19. CAESAR Committee: Cryptographic competitions: Caesar call for submissions, final (2014.01.27). `https://competitions.cr.yp.to/caesar-call.html`, accessed: 2018-07-23

20. Checkoway, S., Maskiewicz, J., Garman, C., Fried, J., Cohney, S., Green, M., Heninger, N., Weinmann, R.P., Rescorla, E., Shacham, H.: A systematic analysis of the juniper dual EC incident. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016.
21. Dodis, Y., Grubbs, P., Ristenpart, T., Woodage, J.: Fast message franking: From invisible salamanders to encryptment. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991. Springer.
22. Dworkin, M.: Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D (November 2007).
23. Fleischmann, E., Forler, C., Lucks, S.: McOE: A family of almost foolproof on-line authenticated encryption schemes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549. Springer.
24. Grubbs, P., Lu, J., Ristenpart, T.: Message franking via committing authenticated encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403. Springer.
25. Gueron, S., Langley, A., Lindell, Y.: AES-GCM-SIV: Specification and analysis. Cryptology ePrint Archive, Report 2017/168 (2017), `http://eprint.iacr.org/2017/168`
26. Gueron, S., Lindell, Y.: GCM-SIV: Full nonce misuse-resistant authenticated encryption at under one cycle per byte. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015.
27. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining your ps and qs: Detection of widespread weak keys in network devices. In: USENIX Security Symposium. vol. 8, p. 1 (2012).
28. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056. Springer.
29. Hoang, V.T., Reyhanitabar, R., Rogaway, P., Vizár, D.: Online authenticated-encryption and its nonce-reuse misuse-resistance. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215. Springer.
30. Hoang, V.T., Tessaro, S., Thiruvengadam, A.: The multi-user security of GCM, revisited: Tight bounds for nonce randomization. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018.
31. Iwata, T., Ohashi, K., Minematsu, K.: Breaking and repairing GCM security proofs. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417. Springer.
32. Joux, A.: 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`
33. Katz, J., Yung, M.: Unforgeable encryption and chosen ciphertext secure modes of operation. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978. Springer.
34. Krawczyk, H.: LFSR-based hashing and authentication. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839. Springer.
35. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733. Springer.
36. Kurosawa, K., Iwata, T.: TMAC: Two-key CBC MAC. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612. Springer.
37. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. Journal of Cryptology 24(3), 588–613 (Jul 2011).

38. Luykx, A., Mennink, B., Paterson, K.G.: Analyzing multi-key security degradation. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part II. LNCS, vol. 10625. Springer.
39. McGrew, D.: An interface and algorithms for authenticated encryption. IETF Network Working Group, RFC 5116 (January 2008).
40. McGrew, D.A., Viega, J.: The security and performance of the Galois/counter mode (GCM) of operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348. Springer.
41. Meyer, C.H., Matyas, S.M.: CRYPTOGRAPHY: A new dimension in computer data security: A guide for the design and implementation of secure systems. Wiley (1982).
42. Minematsu, K.: Authenticated encryption with small stretch (or, how to accelerate AERO). In: Liu, J.K., Steinfeld, R. (eds.) ACISP 16, Part II. LNCS, vol. 9723. Springer.
43. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering generic composition. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441. Springer.
44. Namprempre, C., Rogaway, P., Shrimpton, T.: AE5 security notions: Definitions implicit in the CAESAR call. Cryptology ePrint Archive, Report 2013/242 (2013), http://eprint.iacr.org/2013/242
45. Reddit: Hash of message as nonce? (2015), https://redd.it/3c504m, https://redd.it/3c504m
46. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM CCS 2002.
47. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329. Springer.
48. Rogaway, P.: Nonce-based symmetric encryption. In: Roy, B.K., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017. Springer.
49. Rogaway, P.: The evolution of authenticated encryption. Real World Cryptography Workshop, Stanford (January 2013), https://crypto.stanford.edu/RealWorldCrypto/slides/phil.pdf
50. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A block-cipher mode of operation for efficient authenticated encryption. In: Reiter, M.K., Samarati, P. (eds.) ACM CCS 2001.
51. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004. Springer.
52. Vaudenay, S., Vizár, D.: Under pressure: Security of caesar candidates beyond their guarantees. Cryptology ePrint Archive, Report 2017/1147 (2017), https://eprint.iacr.org/2017/1147
53. Wu, H., Preneel, B.: AEGIS: A fast authenticated encryption algorithm. In: Lange, T., Lauter, K., Lisonek, P. (eds.) SAC 2013. LNCS, vol. 8282. Springer.