Message Franking via Committing Authenticated Encryption*

Paul Grubbs[1], Jiahui Lu[2], and Thomas Ristenpart[1]

[1] Cornell Tech [2] Shanghai Jiao Tong University

Abstract

We initiate the study of message franking, recently introduced in Facebook's end-to-end encrypted message system. It targets verifiable reporting of abusive messages to Facebook without compromising security guarantees. We capture the goals of message franking via a new cryptographic primitive: compactly committing authenticated encryption with associated data (AEAD). This is an AEAD scheme for which a small part of the ciphertext can be used as a cryptographic commitment to the message contents. Decryption provides, in addition to the message, a value that can be used to open the commitment. Security for franking mandates more than that required of traditional notions associated with commitment. Nevertheless, and despite the fact that AEAD schemes are in general not committing (compactly or otherwise), we prove that many in-use AEAD schemes can be used for message franking by using secret keys as openings. An implication of our results is the first proofs that several in-use symmetric encryption schemes are committing in the traditional sense. We also propose and analyze schemes that retain security even after openings are revealed to an adversary. One is a generalization of the scheme implicitly underlying Facebook's message franking protocol, and another is a new construction that offers improved performance.

Keywords: authenticated encryption, encrypted messaging

1 Introduction

Encrypted messaging systems are now used by more than a billion people, due to the introduction of popular, industry-promoted products including WhatsApp [65], Signal [66], and Facebook Messenger [31]. These use specialized (non-interactive) key exchange protocols, in conjunction with authenticated encryption, to protect messages. Many tools are based on the Signal protocol [48], which itself was inspired by elements of the off-the-record (OTR) messaging protocol [20]. A primary design goal is end-to-end security: intermediaries including the messaging service providers, or those with access to their systems, should not be able to violate confidentiality or integrity of user messages.

End-to-end security can be at odds with other security goals. A well-known example is dealing with filtering and reporting spam in the context of encrypted email [42, 61]. Similar issues arise in modern encrypted messaging systems. For example, in Facebook's system when one user sends harassing messages, phishing links, malware attachments, etc., the recipient should be able to report the malicious behavior so that Facebook can block or otherwise penalize the sender. But end-to-end confidentiality means that Facebook must rely on users sending examples of malicious

*A preliminary version of this work appeared in the proceedings of CRYPTO 2017. This is the full version.

Scheme	MO security	Sender binding	Rec. binding	Enc	Dec	Ver
Encode-then-Encipher (Ideal)		✓	✓	–	–	–
Encrypt-then-HMAC (one key)		✓	✓	2+1	2+1	2+1
HMAC-then-CBC		✓	✓	2+1	2+1	2+1
CtE1	✓	✓	✓	3+1	3+1	1+1
CtE2 (Facebook)	✓	✓	✓	3+2	3+2	1+1
CEP	✓	✓	✓	2+1	2+1	1+1

Figure 1: Summary of schemes investigated in this work. The columns indicate whether the scheme meets multiple-opening (MO) security, sender binding, and receiver binding. The last three columns indicate the number of cryptographic passes over a bit string of length equal to the message plus the number of passes needed to handle the associated data, for each of the three main operations. We omit comparisons with concrete encode-then-encipher constructions, which vary in the number of passes required.

messages. How can the provider know that the reported message was the one sent? Reports could, in turn, become a vector for abuse should they allow a malicious reporter to fabricate a message and convince the provider it was the one sent (see also [42]).

Facebook messenger recently introduced an approach for verifiable abuse reporting that they refer to as *message franking* [32,51]. The idea is to include in the report a cryptographic proof that the reported message was the one sent, encrypted, by the particular sender. They offer a protocol (discussed below) and a sensible, but informal and vague, discussion of security goals. At present it is ultimately not clear what message franking provides, whether their approach is secure, and if there exist better constructions. Given the critical role message franking will play for messaging services moving forward, more study is needed.

We therefore initiate the formal study of message franking. We introduce the notion of compactly committing authenticated encryption with associated data (AEAD) as the cryptographic primitive of merit that serves as the basis for message franking. We provide security definitions, show how several widely used AEAD schemes can already serve as compactly committing AEAD, give an analysis of (a generalization of) the scheme underlying Facebook's protocol, and design a new scheme that has superior performance. A summary of schemes treated in this work, and their efficiency, is shown in Figure 1.

Facebook's message franking protocol. Facebook's protocol works as follows, modulo a few details (see Section 3). A sender first generates a fresh key for HMAC [3], and applies HMAC to the message. It then encrypts the HMAC key and message using a conventional AEAD scheme with a symmetric key shared with (just) the recipient, and sends along the resulting ciphertext and the hash value to Facebook's servers. Facebook signs the hash and forwards on the whole package — signature, HMAC hash, and ciphertext — to the recipient, who decrypts and checks the validity of the HMAC output using the recovered HMAC key. Should the recipient want to report abuse, their software client sends the signature, message, HMAC hash, and HMAC key to Facebook who can now verify the signature and hash.

While descriptions of Facebook's protocol do not use the term commitment, intuitively that is the role played by HMAC. This may suggest viewing message franking as simply a construction of committing encryption [37]. But committing encryption views the entire ciphertext as the commitment and opens ciphertexts by revealing the secret key. Neither is true of the Facebook scheme.

A new primitive: compactly committing AEAD. We introduce a new cryptographic primitive that captures the properties targeted in verifiable abuse reporting. We refer to it as compactly committing AEAD. This is an AEAD scheme for which a small portion of the ciphertext can be used as a commitment to the message. Decryption reveals an opening for the message, and the scheme comes equipped with an additional verification algorithm that can check the commitment. This formalization has some similarity to one for non-AEAD symmetric encryption due to Gertner and Herzberg [37], but differs in important ways and their treatment does not suffice for message franking. (See Section 9 for more detailed comparisons).

Formalizing security for committing AEAD schemes requires care. Informally we want confidentiality, ciphertext integrity, and that some designated portion of a ciphertext is a binding commitment to its underlying plaintexts. While seemingly a straightforward adaptation of real-or-random style confidentiality and ciphertext integrity notions would suffice [56, 58, 60], this turns out to provide only a weaker form of security in which reporting abuse may invalidate security of the encryption moving forward. In short, this is because the opening might reveal cryptographic key material, e.g., if the secret key is itself used as the opening. We refer to this as single-opening (SO) security. We formalize also multiple-opening (MO) security notions which, in addition to the usual challenge oracles, gives the adversary the ability to obtain regular encryptions and decryptions (which, by our syntax, reveals the opening should a ciphertext be valid) under the target key. Analogously to previous AEAD treatments [60], we formalize this both via an all-in-one security game that simultaneously establishes confidentiality and integrity, and as separate notions for confidentiality and integrity. We prove them equivalent.

Standard integrity notions like INT-CTXT do not by themselves imply that the ciphertext is a binding commitment to the underlying message. We introduce a notion called receiver binding, which is similar to the binding notions from the commitment literature, notions from the robust encryption[1] literature [1, 33, 34], and the prior notion of binding for committing encryption due to Gertner and Herzberg. Importantly, we deal with the fact that only a portion of the ciphertext is committing, and other details such as associated data. Achieving receiver binding means that no computationally limited adversary can find two opening, message pairs that verify for the same committing portion of a ciphertext.

At first glance this seemed like the end of the story with regards to binding security. But in the message franking setting, schemes that are only receiver binding may spectacularly fail to ensure verifiable abuse reporting. In particular such schemes can suffer from the following attack: a sender carefully chooses a ciphertext so that an abusive message is correctly decrypted by the receiver, but verification with the resulting opening of that message fails. Such an attack is devastating because it prohibits an abusive message from being verified as such, allowing malicious senders to send abusive messages with impunity. We therefore formalize and target meeting a *sender* binding property that rules out such attacks.

Legacy schemes. With formal notions in place, we start by investigating whether existing, in-use AEAD schemes are compactly committing. For these legacy schemes the opening always includes the secret key. For some schemes the per-message randomness is also included in the opening, but for other schemes this can lead to subtle attacks. In each case we identify a small portion of the ciphertext to take as the committing portion. In this context proving receiver binding also proves the scheme to be committing in the more traditional sense.

As mentioned, AEAD schemes are not in general binding via simple counter-examples. We therefore analyze specific constructions, focusing on three important schemes. The first, Encode-then-Encipher [12], uses a variable-input-length tweakable block cipher to build an authenticated encryption scheme by padding messages with randomness and redundancy information (zero bits). We show that, modeling the underlying tweakable cipher as ideal, one can show that taking a security-parameter number of bits of the ciphertext as the commitment is both receiver and sender

[1]We compare and contrast our security notions with notions from the robust encryption literature in Appendix A.

binding. Verification re-encrypts the message and checks that the resulting ciphertext properly matches the commitment value.

We next investigate Encrypt-then-MAC constructions [9], which are particularly relevant here given that Signal [48], and in turn Facebook messenger, uses AES-CBC followed by HMAC for authenticated encryption of messages. In practice, one uses a key-derivation function to derive an encryption key and a MAC key. Interestingly, if one uses as opening those two separate keys, then a simple attack shows that this scheme is *not* receiver binding. If, however, one uses the input to the KDF as the opening, we can prove receiver binding assuming the KDF and MAC are collision resistant. Notably this rules out using CMAC [45], PMAC [18], and Carter-Wegman MACs [64], but Encrypt-then-HMAC suffices.

This means that in Facebook messenger the underlying encryption already suffices as a single-opening-secure committing AEAD scheme. Moreover, due to ratcheting [14, 27, 49] Signal never reuses a symmetric key. Thus Facebook could have avoided the dedicated HMAC commitment. Admittedly they may be uncomfortable — for reason of psychological acceptability — with an architecture that sends decryption keys to Facebook despite the fact that this represents no harm to future or past communications.

We finally investigate MAC-then-Encrypt, the mode of operation underlying TLS 1.2 and before. The binding properties of MAC-then-Encrypt were briefly investigated in a recent paper that used TLS 1.2 records as commitments [63], including a brief proof sketch of receiver binding when taking the entire ciphertext as the commitment. We expand on their proof sketch and provide a full proof for the scheme instantiated with CBC-mode and HMAC (the instantiation used in TLS), taking a small constant number of ciphertext blocks as the committing portion.

Commit-then-Encrypt constructions. We next turn to analyzing generic constructions that combine a commitment with an existing AE scheme. We provide a generalization of the Facebook scheme, and show that it is multiple-opening secure and both sender and receiver binding, assuming only that the underlying AEAD scheme is sound and the commitment is unique. HMAC is a unique commitment, thereby giving us the first formal security analysis of Facebook's message franking scheme. One can also use a non-malleable commitment [30]. If one instead uses a malleable commitment, then the scheme will not achieve ciphertext integrity.

We also offer an alternative composition that removes the need for non-malleable commitments, and also can improve performance in the case that associated data is relatively long. Briefly, we use a commitment to the associated data and message as the associated data for the underlying AEAD scheme. This indirectly binds the encryption ciphertext to the associated data, without paying the cost of twice processing it.

Both these constructions are multiple-opening secure, since the commitment opening is independent of the underlying AE keys. This is intuitively simple but the proof requires care — commitments play a role in achieving CTXT and so we must show that unopened encryptions, despite using the same keys as opened encryptions, retain ciphertext integrity. See the body for details.

The Committing Encrypt-and-PRF (CEP) scheme. The generic constructions that meet multiple-opening security are slower than existing (single-opening secure) AEAD schemes, since they require an additional cryptographic pass over the message. This represents approximately a 1.5x slowdown both for encryption and decryption. For the expected workload in messaging applications that consists primarily of relatively short plaintexts, this may not matter, but if one wants to use committing AEAD for large plaintexts such as image and video attachments or in streaming settings (e.g., a committing version of TLS) the overhead will add up quickly.

We therefore offer a new AEAD scheme, called Committing Encrypt-and-PRF (CEP) that simultaneously enjoys multiple-opening security while also retaining the two-pass performance of standard AEAD schemes. As an additional bonus we make the scheme nonce-based [58], meaning that it is derandomized and only needs to be used with non-repeating nonces. (We formalize nonce-based committing AEAD in the body; it is largely similar as the randomized variant.)

The basic idea is to adapt an Encrypt-and-PRF style construction to be compactly committing and multiple-opening. To do so we derive one-time use PRF keys from the nonce, and compute a tag that is two-part. The commitment value for the ciphertext is the output of a keyed hash that is simultaneously a PRF when the key is private and collision resistant when it is adversarially chosen. The latter is critical since receiver binding requires, in this context, a collision-resistance property. If one stopped here, then the scheme would not be secure, since openings reveal the PRF's key, rendering it only CR, and CR is not enough to prevent future ciphertext forgeries. We therefore additionally run a one-time PRF (with key that is never opened) over this commitment value to generate a tag that is also checked during decryption. Ultimately we prove that the scheme achieves our notions of sender binding, receiver binding, and multiple-opening confidentiality and ciphertext integrity.

We strove to make the scheme simple and fast. Instantiated with a stream cipher such as AES-CTR-mode or ChaCha20, we require just a single secret key and use the stream cipher to generate not only the one-time keys for the PRFs but also a pad for encrypting the message. Because we need a collision-resistant PRF, our suggested instantiation is HMAC, though other multi-property hash functions [10] would work as well.

Future directions. Our work has focused on the symmetric encryption portion of messaging protocols, but one can also ask how the landscape changes if one holistically investigates the public-key protocols or key exchange in particular. Another important direction is to understand the potential tension between committing AEAD and security in the face of selective opening attacks (SOA) [7, 8]. Our current definitions do not model SOAs. (An SOA would allow, for example, a compromise of the full cryptographic key, not just the ability to get openings.) While it may seem that committing encryption and SOA security are at odds, we actually conjecture that this is not fundamental (particularly in the random oracle model), and future work will be able to show SOA-secure compactly committing AEAD.

2 Preliminaries

We fix some alphabet Σ, e.g., $\Sigma = \{0,1\}$. For any $x \in \Sigma^*$ let $|x|$ denote its length. We write $x \leftarrow\!\!\$\ X$ to denote uniformly sampling from a set X. We write $X \,\|\, Y$ to denote concatenation of two strings. For a string X of n bits, we will write $X[i, \ldots, j]$ for $i < j \leq n$ to mean the substring of X beginning at index i and ending at index j. For notational simplicity, we assume that one can unambiguously parse $Z = X \,\|\, Y$ into its two parts, even for strings of varying length. For strings $X, Y \in \{0,1\}^*$ we write $X \oplus Y$ to denote taking the XOR of $X[1, \ldots, \min\{|X|, |Y|\}] \oplus Y[1, \ldots, \min\{|X|, |Y|\}]$.

We use code-based games (q.v., [13]) to formalize security notions. A game G is a sequence of pseudocode statements, with variables whose type will be clear from context. Variables are implicitly initialized to appropriate defaults for their type (zero for integers, empty set for sets, etc.). Each variable is a random variable in the probability distribution defined by the random coins used to execute the game. We write $\Pr[G \Rightarrow y]$ to denote the event (over the random coins of G) that the game outputs a value y. Associated to this pseudocode is some fixed RAM model of computation where most operations are unit cost. We will use "big-O" notation $\mathcal{O}(\cdot)$ to hide only small constants that do not materially impact the interpretation of our results. For a game G and scheme S, we will sometimes use the terminology "a G_S adversary" to refer to an adversary in the game G instantiated with the scheme S. If we denote the adversary $\mathcal{A}$, we will write $G_S^{\mathcal{A}}$ to denote the game G instantiated with the scheme S and specific adversary $\mathcal{A}$, and $\Pr\left[\, G_S^{\mathcal{A}} \Rightarrow \mathsf{out} \,\right]$ to denote the probability over some sample space (usually the random coins used by the game and the adversary) that game G instantiated with scheme S and adversary $\mathcal{A}$ outputs out.

We will work in the random oracle model (ROM) [11] and the ideal cipher model (ICM). In the ROM, algorithms and adversaries are equipped with an oracle that associates to each input a random output of some length that will vary by, and be clear from, context. In the ICM, algorithms and adversaries are equipped with a pair of oracles. The first takes input a key, a tweak, and a message, all bit strings of some lengths k, t, and n, respectively. Each key, tweak pair selects a random permutation on $\{0,1\}^n$. The second oracle takes as input a key, a tweak, and an n-bit value, and returns the inverse of the permutation selected by the key and tweak applied to the value.

Symmetric encryption. A nonce-based authenticated encryption with associated data (AEAD) scheme $\mathsf{SE} = (\mathsf{kg}, \mathsf{enc}, \mathsf{dec})$ consists of a triple of algorithms. Associated to it are a key space $\mathcal{K} \subseteq \Sigma^*$, nonce space $\mathcal{N} \subseteq \Sigma^*$, header space $\mathcal{H} \subseteq \Sigma^*$, message space $\mathcal{M} \subseteq \Sigma^*$, and ciphertext space $\mathcal{C} \subseteq \Sigma^*$. The randomized key generation algorithm kg outputs a secret key $K \in \mathcal{K}$. Canonically kg selects $K \leftarrow\!\!\$\, \mathcal{K}$ and outputs K. Encryption enc is deterministic and takes as input a four-tuple $(K, N, H, M) \in (\Sigma^*)^4$ and outputs a ciphertext C or a distinguished error symbol $\perp$. We require that $\mathsf{enc}(K, N, H, M) \neq \perp$ if $(K, N, H, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{M}$. Decryption dec is deterministic and takes as input a tuple $(K, N, H, C) \in (\Sigma^*)^4$ and outputs a message M or $\perp$. An SE scheme is correct if for any $(K, N, H, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{M}$ it holds that $\mathsf{dec}(K, N, H, \mathsf{enc}(K, N, H, M)) = M$.

Some schemes that we will analyze predate the viewpoint of nonce-based encryption, including generic compositions that utilize CTR or CBC mode. A randomized SE scheme $\mathsf{SE} = (\mathsf{kg}, \mathsf{enc}, \mathsf{dec})$ is the same as a nonce-based SE scheme except that we omit nonces everywhere, and have enc take an additional input, the coins, that are assumed to be drawn from some coin space $\mathcal{R} \subseteq \sigma^*$. Correctness now is met if for any $(K, H, M, R) \in \mathcal{K} \times \mathcal{H} \times \mathcal{M} \times \mathcal{R}$ it holds that $\mathsf{dec}(K, H, \mathsf{enc}(K, H, M; R)) = M$. We will focus on schemes that are public-coin, meaning the ciphertext includes R explicitly. This is true, for example, of CTR or CBC mode encryption. For notational simplicity, we will assume for such schemes that enc outputs R concatenated with the remainder of the ciphertext. Below we will occasionally refer to plain authenticated encryption (AE) which does not handle associated data. This will be defined identically to AEAD above, but with the associated data H removed.

Message authentication codes. A message authentication code (MAC) is a tuple of algorithms $\mathsf{Mac} = (\mathsf{kg}, \mathsf{tag}, \mathsf{ver})$. Associated to a MAC is a key space $\mathcal{K} \subseteq \Sigma^*$, message space[2] $\mathcal{M} \subseteq \Sigma^* \times \Sigma^*$, and tag space $\mathcal{T} \subseteq \Sigma^*$. The key generation procedure kg is the same as the one used for symmetric encryption. The deterministic tag generation algorithm $\mathsf{tag}(K, M)$ takes as input a key $K \in \mathcal{K}$ and message $M \in \mathcal{M}$ and outputs a tag $T \in \mathcal{T}$. The deterministic verification procedure $\mathsf{ver}(K, M, T)$ takes as input a key $K \in \mathcal{K}$, message $M \in \mathcal{M}$, and tag $T \in \mathcal{T}$. It outputs true if $\mathsf{tag}(K, M) = T$ and false otherwise.

The standard security notion for MACs is existential unforgeability under chosen-message attack (UF-CMA). We use a multi-user variant that generalizes it to a setting in which an adversary can interact with multiple instances of the MAC. Game MU-UF-CMA$_{\mathsf{Mac}}$ is shown in Figure 2. The MU-UF-CMA$_{\mathsf{Mac}}$ advantage of an adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\text{mu-uf-cma}}_{\mathsf{Mac}}(\mathcal{A}) = \Pr\left[\, \text{MU-UF-CMA}^{\mathcal{A}}_{\mathsf{Mac}} \Rightarrow \mathsf{true} \,\right] .$$

[2]Looking ahead, we will use a MAC that takes pairs of strings as messages. See Section 8.

```
$\underline{\text{MU-UF-CMA}^{\mathcal{A}}_{\mathsf{Mac}}}$:
$\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Tag},\mathbf{Ver}}$
Return $\mathsf{win}$

$\underline{\mathbf{Ver}(S, M, T)}$:
If $\mathtt{K}[S] = \perp$ then $K[S] \leftarrow\!\!\$\ \mathsf{Kg}$
$b \leftarrow (\mathsf{tag}(K[S], M) = T)$
If $M \notin \mathcal{M}[S] \wedge b = 1$ then
    $\mathsf{win} \leftarrow \mathsf{true}$
Return $b$

$\underline{\mathbf{Tag}(S, M)}$:
If $\mathtt{K}[S] = \perp$ then $\mathtt{K}[S] \leftarrow\!\!\$\ \mathsf{Kg}$
$\mathcal{M}[S] \leftarrow \mathcal{M}[S] \cup M$
Return $\mathsf{tag}(K[S], M)$
```

Figure 2: Multi-user UF-CMA security for a MAC scheme $\mathsf{Mac} = (\mathsf{kg}, \mathsf{tag}, \mathsf{ver})$.

Nonce-based pseudorandom generators. A nonce-based pseudorandom generator G is a deterministic algorithm that takes as input a key K, a nonce N, and an output length ℓ. It outputs a string of length ℓ bits. The PRG advantage of an adversary $\mathcal{A}$ against G is defined by

$$\mathbf{Adv}^{\mathrm{prg}}_{G}(\mathcal{A}) = \left|\Pr\left[\, K \leftarrow\!\!\$\ \{0,1\}^k \colon \mathcal{A}^{G(K,\cdot,\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, \mathcal{A}^{R(\cdot,\cdot)} \Rightarrow 1 \,\right]\right|$$

where R works as follows. On query N, ℓ it checks if a previous query N, ℓ' was submitted. If $\ell' < \ell$ it picks a new random string of length $\ell - \ell'$, appends it to the previous returned string for N, records it (in a table indexed by N), and returns the concatenated random string. If no previous query exists, then it picks a random string of length ℓ, records it, and returns it. We call a PRG adversary $\mathcal{A}$ nonce-respecting if all its queries use a unique nonce N. We can build a nonce-based pseudorandom generator $G[E]$ from a block cipher $E \colon \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^n$ in CTR mode with $IV \leftarrow E_K(N)$. That is, on input K, N, ℓ, the PRG $G[E]$ outputs an ℓ-bit string P where the i^{th} n-bit block is $E_K(E_K(N) + i)$, truncating the last block if necessary. (Addition is in the field $\mathrm{GF}(2^n)$.) One can adapt existing techniques (e.g., [58, Th. 3]) in a straightforward way to prove the following lemma. Note that a tighter, but slightly messier, bound can be proven using the fact that at most σ/n queries to E are required to generate σ bits of PRG output.

Lemma 1 *Let $E \colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let $G[E]$ be a nonce-based pseudorandom generator constructed from E as described in the previous paragraph. Let $\mathcal{A}$ be a nonce-respecting PRG adversary against $G[E]$ making at most q queries and whose output lengths sum to at most σ. Then one can give an explicit adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\mathrm{prg}}_{G[E]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prf}}_{E}(\mathcal{B}) + \sigma^2/2^n$. Adversary $\mathcal{B}$ makes at most σ queries and runs in time that of $\mathcal{A}$.*

Tweakable ciphers. A tweakable cipher $\mathrm{TC} = (\widetilde{E}, \widetilde{D})$ is a pair of algorithms. Associated to any tweakable cipher is a key space $\mathcal{K} \subseteq \Sigma^*$, tweak space $\mathcal{T} \subseteq \Sigma^*$, and message and ciphertext space $\mathcal{M} \subseteq \Sigma^*$. Keys are generated as random draws from $\mathcal{K}$. Encryption $\widetilde{E}$ takes a key, tweak, and message triple $(K, T, M) \in (\Sigma^*)^3$ and outputs a ciphertext $C \in \mathcal{M}$ or an error symbol $\perp$. Decryption $\widetilde{D}$ takes a key, tweak, and ciphertext triple $(K, T, C) \in (\Sigma^*)^3$ and outputs a message $M \in \mathcal{M}$ or an error symbol $\perp$. Below we will sometimes write the key K as a subscript of $\widetilde{E}$ or $\widetilde{D}$ and the tweak T as a superscript. Intuitively, a tweakable cipher is a family of permutations over $\mathcal{M}$ indexed by key and tweak pairs $(K, T) \in \mathcal{K} \times \mathcal{T}$. This is in contrast to a standard block cipher, which is a family of permutations indexed only by a key $K \in \mathcal{K}$. We say that TC has *variable input length* if the family of permutations is additionally indexed by the input length.

Pseudorandom functions. For a function $F : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^n$ and adversary $\mathcal{A}$ we define the *pseudorandom function* (PRF) advantage of $\mathcal{A}$ to be

$$\mathbf{Adv}^{\mathrm{prf}}_{F}(\mathcal{A}) = \left|\Pr\left[\, \mathcal{A}^{F(K,\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, \mathcal{A}^{R(\cdot)} \Rightarrow 1 \,\right]\right| .$$

In the (implicit) game for the left-hand term, the key K is drawn uniformly from $\mathcal{K}$ by the challenger.

In the game for the right-hand term the function R is drawn uniformly from **Func**, the space of all functions that output n bits.[3] Informally, we say the function F is a PRF if $\mathbf{Adv}_F^{\mathrm{prf}}()$ is small for all efficient adversaries. Below we will sometimes refer to the left-hand experiment as the "real world" and the other as the "ideal world".

In proofs it will be convenient to use multi-user PRF security [4]. We define the MU-PRF advantage of an adversary $\mathcal{A}$ to be

$$\mathbf{Adv}_F^{\mathrm{mu\text{-}prf}}(\mathcal{A}) = \left|\Pr\left[\,\mathcal{A}^{\overline{F}(\cdot,\cdot)} \Rightarrow 1\,\right] - \Pr\left[\,\mathcal{A}^{\overline{R}(\cdot,\cdot)} \Rightarrow 1\,\right]\right| .$$

where $\overline{F}$ on input a key identifier $S \in \{0,1\}^*$ and a message M, checks if there is a key associated to S, and if not chooses a fresh one $K[S] \leftarrow\!\!\$\ \{0,1\}^k$. It then returns $F(K[S], M)$. The oracle $\overline{R}$ on input a key identifier $S \in \{0,1\}^*$ and a message M, checks if there is a random function associated to S, and if not chooses a fresh one $R[S] \leftarrow\!\!\$\ \mathrm{Func}$. It returns $R[S](M)$. Note that MU-PRF security is implied by PRF security via a standard argument.

Collision-resistance. For a function $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^n$ and adversary $\mathcal{A}$, define the *collision-resistance* (CR) advantage as

$$\mathbf{Adv}_F^{\mathrm{cr}}(\mathcal{A}) = \Pr\left[\,((x_1,x_2),(x_1',x_2')) \leftarrow\!\!\$\ \mathcal{A} \colon \begin{array}{c} F(x_1,x_2) = F(x_1',x_2'), \\ (x_1,x_2) \neq (x_1',x_2') \end{array}\,\right] .$$

Informally, we say F is collision-resistant if $\mathbf{Adv}_F^{\mathrm{cr}}()$ is small for all efficient adversaries. Following prior work [59], we omit the alternate definition of collision-resistance that is used for unkeyed primitives like hash functions, in which the adversary needs to output a collision in a function that is randomly sampled from a family of functions.

Commitment schemes with verification. A commitment scheme with verification $\mathsf{CS} = (\mathsf{Com}, \mathsf{VerC})$ consists of two algorithms.[4] Associated to any commitment scheme is an opening space $\mathcal{K}_f \subseteq \Sigma^*$, a message space $\mathcal{M} \subseteq \Sigma^*$, and a commitment space $\mathcal{C} \subseteq \Sigma^*$. The algorithm Com is randomized and takes as input a $M \in \Sigma^*$ and outputs a pair $(K, C) \in \mathcal{K}_f \times \mathcal{C}$ or an error symbol $\perp$. We assume that Com returns $\perp$ with probability one if $M \notin \mathcal{M}$. The algorithm VerC is deterministic. It takes input a tuple $(K, C, M) \in \Sigma^*$ and outputs a bit. We assume that VerC returns 0 if its input $(K, C, M) \notin \mathcal{K}_f \times \mathcal{C} \times \mathcal{M}$. We assume that the commitment values C are of some fixed length (typically denoted by t).

A commitment scheme is correct if for all $M \in \mathcal{M}$, $\Pr[\mathsf{VerC}(\mathsf{Com}(M), M) = 1] = 1$ where the probability is over the coins used by Com. We can formalize the binding security notion of our commitment scheme as a game. Formally, the game $\mathrm{vBIND}_{\mathsf{CS}}^{\mathcal{A}}$ first runs the adversary $\mathcal{A}$ who outputs a tuple (K_c, M, K_c', M', C). The game then runs $b \leftarrow \mathsf{VerC}(K_c, C, M)$ and $b' \leftarrow \mathsf{VerC}(K_c', C, M')$. The game outputs true if $M \neq M'$ and $b = b' = 1$ and false otherwise. To a commitment scheme CS and adversary $\mathcal{A}$ we associate the vBIND advantage

$$\mathbf{Adv}_{\mathsf{CS}}^{\mathrm{v\text{-}bind}}(\mathcal{A}) = \Pr\left[\,\mathrm{vBIND}_{\mathsf{CS}}^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] .$$

The probability is over the coins used by the game.

Commitment schemes should enjoy a hiding property as well. Traditionally this is formalized as a left-or-right indistinguishability notion (q.v., [6]). For our purposes we will target a stronger notion, analogous to real-or-random (ROR) security for symmetric encryption. It asks that a commitment be indistinguishable from a random bit string while the opening remaining secret. Game $\mathrm{ROR1}_{\mathsf{CS}}^{\mathcal{A}}$ runs an adversary $\mathcal{A}$ and gives it access to an oracle **Com** to which it can query

[3]We are abusing the formalism here by sampling R from an infinite set; we do so for notational consistency and simplicity.

[4]We will not use the alternate definition of commitments with opening [21].

messages. The oracle computes $(K_c, C) \leftarrow_{\$} \mathsf{Com}(M)$ and returns C. The adversary outputs a bit, and the game outputs true if the bit is one. Game $\mathrm{ROR0}_{\mathsf{CS}}^{\mathcal{A}}$ is similar except that the oracle returns a string of random bits of length $|C|$ and the game outputs true if the adversary outputs zero. We define the advantage by $\mathbf{Adv}_{\mathsf{CS}}^{\text{cs-ror}}(\mathcal{A}) = \left|\Pr\left[\,\mathrm{ROR1}_{\mathsf{CS}}^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] - \Pr\left[\,\mathrm{ROR0}_{\mathsf{CS}}^{\mathcal{A}} \Rightarrow \mathsf{false}\,\right]\right|$.

HMAC is a good commitment. Any PRF that is also collision-resistant meets our security goals for commitments. In particular, one can build a commitment scheme $\mathsf{CS}[F] = (\mathsf{Com}, \mathsf{VerC})$ works from any function $F\colon \mathcal{K} \times \{0,1\}^* \to \{0,1\}^n$ as follows. Commitment $\mathsf{Com}(M)$ chooses a fresh value $K \leftarrow_{\$} \mathcal{K}$, computes $C \leftarrow F(K, M)$ and outputs (K, C). Verification $\mathsf{VerC}(K, C, M)$ outputs one if $F(K, M) = C$ and zero otherwise. Then the following theorem captures the security of this commitment scheme, which rests on the collision resistance and PRF security of F.

Theorem 1 *Let F be a function and $\mathsf{CS}[F]$ be the commitment scheme built from it as described above. Then for any $ROR_{\mathsf{CS}[F]}$ adversary $\mathcal{A}$ making at most q queries and $\mathrm{vBIND}_{\mathsf{CS}[F]}$ adversary $\mathcal{A}'$, we construct an explicit pair of adversaries $\mathcal{B}, \mathcal{B}'$ in the proof below so that*

$$\mathbf{Adv}_{\mathsf{CS}[F]}^{\text{cs-ror}}(\mathcal{A}) \le \mathbf{Adv}_{F}^{\text{mu-prf}}(\mathcal{B}) \quad \text{and} \quad \mathbf{Adv}_{\mathsf{CS}[F]}^{\text{v-bind}}(\mathcal{A}') \le \mathbf{Adv}_{F}^{\text{cr}}(\mathcal{B}') \ .$$

The adversary $\mathcal{B}$ runs in time that of $\mathcal{A}$ and makes the same number of oracle queries as $\mathcal{A}$. Adversary $\mathcal{B}'$ runs in time that of $\mathcal{A}'$.

Proof: Adversary $\mathcal{B}$ is straightforward - when $\mathcal{A}$ makes an oracle query on value q, it queries its mu-prf oracle with q and a fresh key identifier. The adversary $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs, and the first result follows immediately.

Next, we will bound the vBIND advantage of F using an adversary $\mathcal{B}'$ that simply runs $\mathcal{A}'$ giving it a description of the function F. When $\mathcal{A}'$ outputs (K_c, M, K_c', M', C), $\mathcal{B}'$ runs $\mathsf{VerC}(K_c, C, M)$ and $\mathsf{VerC}(K_c', C, M')$ and outputs $((K_c, M), (K_c', M'))$ if both calls to VerC return 1. Thus,

$$\Pr\left[\,\mathrm{vBIND}_{F}^{\mathcal{A}'} \Rightarrow \mathsf{true}\,\right] \le \mathbf{Adv}_{F}^{\text{cr}}(\mathcal{B}')$$

and the second result follows as well. ▮

As the underlying function needs to be both CR and a good PRF, a suitable candidate would be HMAC [5], i.e., $F(K, M) = \mathsf{HMAC}(K, M)$. Other multi-property hash functions [10] could be used as well. The Facebook franking scheme (discussed in Section 3) uses a non-standard HMAC-based commitment based on $F(K, M) = \mathsf{HMAC}(K, M \,\|\, K)$. We will assume HMAC remains a PRF when used in this non-standard way. One can substantiate this assumption directly in the random oracle model [29], or using techniques from the key-dependent message literature [19, 38].

3 Message Franking and End-to-End Encryption

In end-to-end encrypted messaging services there exists a tension between message privacy and reporting abusive message contents to service providers. The latter is important to flag abusive accounts, but reports need to be verifiable, meaning that the provider can check the contents of the allegedly abusive message *and* be certain that it was the message sent. Otherwise abuse-reporting mechanisms could themselves be abused to make false accusations.

A recipient can send the allegedly abusive plaintext to the service provider, but message privacy guarantees that the provider does not know whether the alleged message was in fact the one sent.[5]

[5]Of course, if the recipient is running a trusted client, then this assertion could be trusted. We are concerned with the case that the client is potentially subverted.

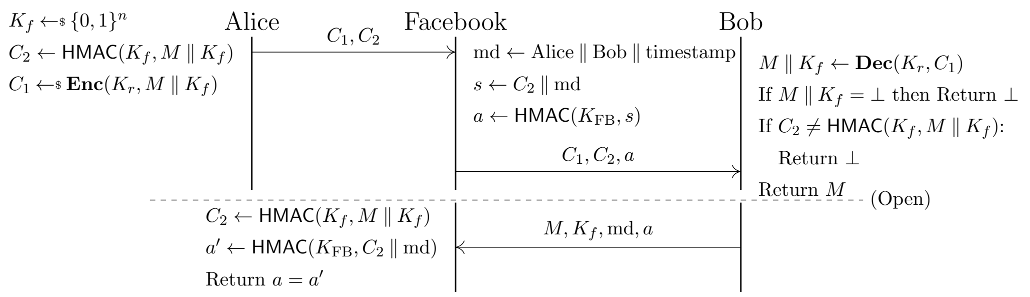
$K_f \leftarrow\!\!\$\ \{0,1\}^n$
Alice
C_1, C_2
Facebook
Bob
$C_2 \leftarrow \mathsf{HMAC}(K_f, M \parallel K_f)$
$C_1 \leftarrow\!\!\$\ \mathbf{Enc}(K_r, M \parallel K_f)$
md ← Alice ‖ Bob ‖ timestamp
$s \leftarrow C_2 \parallel \mathrm{md}$
$a \leftarrow \mathsf{HMAC}(K_{\mathrm{FB}}, s)$
C_1, C_2, a
$M \parallel K_f \leftarrow \mathbf{Dec}(K_r, C_1)$
If $M \parallel K_f = \perp$ then Return $\perp$
If $C_2 \neq \mathsf{HMAC}(K_f, M \parallel K_f)$:
Return $\perp$
Return M
(Open)
$C_2 \leftarrow \mathsf{HMAC}(K_f, M \parallel K_f)$
$a' \leftarrow \mathsf{HMAC}(K_{\mathrm{FB}}, C_2 \parallel \mathrm{md})$
Return $a = a'$
M, K_f, md, a

Figure 3: Facebook's message franking protocol [51]. The key K_r is a one-time-use symmetric key derived as part of the record layer protocol. The top portion is the sending of an encrypted message to the recipient. The bottom portion is the abuse reporting protocol.

A seeming solution would be for the service to log ciphertexts, and have the recipient disclose the secret key to allow the provider to decrypt the ciphertext. Not only is this impractical due to the storage requirements, but it also does *not* guarantee that the decrypted message is correct. It could be that the recipient chose a key that somehow decrypts the (legitimate) ciphertext to a fake message. Ultimately what is required for this to work is for the encryption to be committing: no computationally efficient adversary can find a secret key that decrypts the ciphertext to anything but the originally encrypted message.

Facebook's approach. Facebook recently detailed a new cryptographic mechanism [32, 51] targeting verifiable abuse reporting on Facebook messenger, which uses end-to-end encryption based on Signal [66]. The basic idea is to force the sender to provide a commitment, sent in the clear, to the plaintext message. A diagram of Facebook's protocol, that they call "message franking" (as in "speaking frankly"), is shown in Figure 3. The sender first applies HMAC with a fresh key K_f to the concatenation of the message and K_f to produce a value C_2, and then encrypts using an AEAD scheme the message and K_f to produce a ciphertext C_1 using a key K_r shared with the recipient. Then (C_1, C_2) is sent to Facebook. Facebook applies HMAC with its own secret key K_{FB} to C_2 to get a tag a, and sends to the recipient (C_1, C_2, a). The recipient decrypts C_1, recovers the message M and key K_f and checks the value $C_2 = \mathsf{HMAC}(K_f, M \parallel K_f)$. To report abuse, the recipient sends M, K_f, and a to Facebook. Facebook recomputes $\mathsf{HMAC}(K_f, M \parallel K_f)$ and checks the tag a.

It is clear that the sender is using HMAC as a cryptographic commitment to the message. (This terminology is not used in their technical specifications.) The use of HMAC by Facebook to generate the tag a is simply to forego having to store commitments, instead signing them so that they can be outsourced to recipients for storage and verified should an abuse report come in.

There are interesting security issues that could arise with Facebook's scheme, and cryptographic abuse reporting in general, that are orthogonal to the ones discussed here. In particular, binding Facebook's tag to the communicating parties seems crucial: otherwise a malicious party could create a sock-puppet (i.e. fake) account, send itself an abusive message, then accuse a victim of having sent it.

While the design looks reasonable, and the Facebook white paper provides some informal discussion about security, there has been no formal analysis to date. It is also not clear what security properties the main cryptographic construction — combining a commitment with AEAD — should satisfy. We rectify this by introducing, in the following section, the notion of committing AEAD. This will allow us not only to analyze Facebook's franking scheme, but to suggest alternative designs, including ones that are legacy-compatible with existing deployed AEAD schemes and that do not require adding an additional dedicated commitment.

4 Committing AEAD

Formally, a committing AEAD scheme $\mathsf{CE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ver})$ is a four-tuple of algorithms. Associated to a scheme is a key space $\mathcal{K} \subseteq \Sigma^*$, header space $\mathcal{H} \subseteq \Sigma^*$, message space $\mathcal{M} \subseteq \Sigma^*$, ciphertext space $\mathcal{C} \subseteq \Sigma^*$, opening space $\mathcal{K}_f \subseteq \Sigma^*$, and franking tag space $\mathcal{T} \subseteq \Sigma^*$.

- ***Key generation***: The randomized key generation algorithm Kg outputs a secret key $K \in \mathcal{K}$. We write $K \leftarrow\$ \mathsf{Kg}$ to denote executing key generation.
- ***Encryption***: Encryption Enc is randomized. The input to encryption is a triple $(K, H, M) \in (\Sigma^*)^3$ and the output is a pair $(C_1, C_2) \in \mathcal{C} \times \mathcal{T}$ or a distinguished error symbol $\perp$. Unlike with regular symmetric encryption, the output includes two components: a ciphertext C_1 and a franking tag C_2. We also refer to C_2 as the commitment. We require that $\mathsf{Enc}(K, H, M) \neq \perp$ if $(K, H, M) \in \mathcal{K} \times \mathcal{H} \times \mathcal{M}$. We write $(C_1, C_2) \leftarrow\$ \mathsf{Enc}(K, H, M)$ to denote executing encryption.
- ***Decryption***: Decryption, which is deterministic, takes as input a tuple $(K, H, C_1, C_2) \in (\Sigma^*)^4$ and outputs a message, opening value pair $(M, K_f) \in \mathcal{M} \times \mathcal{K}_f$ or $\perp$. We write $(M, K_f) \leftarrow \mathsf{Dec}(K, H, C_1, C_2)$ to denote executing decryption.
- ***Verification***: Verification, which is deterministic, takes as input a tuple $(H, M, K_f, C_2) \in (\Sigma^*)^4$ and outputs a bit. For $(H, M, K_f, C_2) \notin \mathcal{H} \times \mathcal{M} \times \mathcal{K}_f \times \mathcal{T}$, we assume that Ver outputs 0. We write $b \leftarrow \mathsf{Ver}(H, M, K_f, C_2)$ to denote executing verification.

We will often place K in the subscript of relevant algorithms. For example, $\mathsf{Enc}_K(H, M) = \mathsf{Enc}(K, H, M)$ and $\mathsf{Dec}_K(H, C_1, C_2) = \mathsf{Dec}(K, H, C_1, C_2)$.

We require that CE schemes output ciphertexts whose lengths are determined solely by the length of the header and message. Formally this means that there exists a function $\mathsf{clen}\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ such that for all $(K, H, M) \in \mathcal{K} \times \mathcal{H} \times \mathcal{M}$ it holds that $\Pr[(|C_1|, |C_2|) = \mathsf{clen}(|H|, |M|)] = 1$ where $(C_1, C_2) \leftarrow\$ \mathsf{Enc}_K(H, M)$ and the probability is over the coins used by encryption.

We say a CE scheme has *decryption correctness* if for all $(K, H, M) \in \mathcal{K} \times \mathcal{H} \times \mathcal{M}$ it holds that $\Pr[\mathsf{Dec}(K, H, C_1, C_2) = M] = 1$ where the probability is taken over the coins used to compute $(C_1, C_2) \leftarrow\$ \mathsf{Enc}(K, H, M)$.

We say that a scheme has *commitment correctness* if for all $(K, H, M) \in \mathcal{K} \times \mathcal{H} \times \mathcal{M}$ it holds that $\Pr[\mathsf{Ver}(H, M, K_f, C_2) = 1] = 1$ where the probability is taken over the random variables used in the experiment

$$(C_1, C_2) \leftarrow\$ \mathsf{Enc}_K(H, M)\ ;\ (M, K_f) \leftarrow \mathsf{Dec}_K(H, C_1, C_2)\ ;\ \text{Return } (K_f, C_2)$$

Our formulation of CE schemes is a generalization of that for conventional (randomized) AEAD schemes in the following sense. One can consider an AEAD scheme as a CE scheme that has encryption output the entire ciphertext as C_2, decryption output an empty string for the opening value, and has verify always return one.

Compactly committing AEAD. In our formalism, a ciphertext has two components. A scheme may output $C_1 = \varepsilon$ and a C_2 value that therefore consists of the entire ciphertext. This embodies the traditional viewpoint on committing AEAD, in which the entire ciphertext is viewed as the commitment. We refer to this as "traditionally committing encryption" (see Appendix A). But we are more general, and in particular our formalism allows schemes with *compact* commitments, by which we mean schemes for which $|C_2|$ is small. In particular we will want $|C_2|$ to be linear in the key size, rather than linear in the message length. One can make any CE scheme compact

```
$\underline{\text{MO-REAL}^{\mathcal{A}}_{\mathsf{CE}}}$:
$K \leftarrow\!\!\$\ \mathsf{Kg}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalEnc}}$
Return $b'$

$\underline{\mathbf{Enc}(H, M)}$
$(C_1, C_2) \leftarrow\!\!\$\ \mathsf{Enc}_K(H, M)$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{(H, C_1, C_2)\}$
Return $(C_1, C_2)$

$\underline{\mathbf{Dec}(H, C_1, C_2)}$
If $(H, C_1, C_2) \notin \mathcal{Y}_1$ then
    Return $\bot$
$(M, K_f) \leftarrow \mathsf{Dec}_K(H, C_1, C_2)$
Return $(M, K_f)$

$\underline{\mathbf{ChalEnc}(H, M)}$
$(C_1, C_2) \leftarrow\!\!\$\ \mathsf{Enc}_K(H, M)$
Return $(C_1, C_2)$
```

```
$\underline{\text{MO-RAND}^{\mathcal{A}}_{\mathsf{CE}}}$:
$K \leftarrow\!\!\$\ \mathsf{Kg}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalEnc}}$
Return $b'$

$\underline{\mathbf{Enc}(H, M)}$
$(C_1, C_2) \leftarrow\!\!\$\ \mathsf{Enc}_K(H, M)$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{(H, C_1, C_2)\}$
Return $(C_1, C_2)$

$\underline{\mathbf{Dec}(H, C_1, C_2)}$
If $(H, C_1, C_2) \notin \mathcal{Y}_1$ then
    Return $\bot$
$(M, K_f) \leftarrow \mathsf{Dec}_K(H, C_1, C_2)$
Return $(M, K_f)$

$\underline{\mathbf{ChalEnc}(H, M)}$
$(\ell_1, \ell_2) \leftarrow \mathsf{clen}(|H|, |M|)$
$(C_1, C_2) \leftarrow\!\!\$\ \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$
Return $(C_1, C_2)$
```

```
$\underline{\text{MO-CTXT}^{\mathcal{A}}_{\mathsf{CE}}}$:
$K \leftarrow\!\!\$\ \mathsf{Kg}$ ; $\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalDec}}$
Return $\mathsf{win}$

$\underline{\mathbf{Enc}(H, M)}$
$(C_1, C_2) \leftarrow\!\!\$\ \mathsf{Enc}_K(H, M)$
$\mathcal{Y} \leftarrow \mathcal{Y} \cup \{(H, C_1, C_2)\}$
Return $(C_1, C_2)$

$\underline{\mathbf{Dec}(H, C_1, C_2)}$
Return $\mathsf{Dec}_K(H, C_1, C_2)$

$\underline{\mathbf{ChalDec}(H, C_1, C_2)}$
If $(H, C_1, C_2) \in \mathcal{Y}$ then
    Return $\bot$
$(M, K_f) \leftarrow \mathsf{Dec}_K(H, C_1, C_2)$
If $M \neq \bot$ then
    $\mathsf{win} \leftarrow \mathsf{true}$
Return $(M, K_f)$
```

Figure 4: Confidentiality (left two games) and ciphertext integrity (rightmost) games for committing AEAD.

by hashing the ciphertext with a collision-resistant (CR) hash function, but we will show compact schemes that have better performance.

Single versus multiple openings. In some protocols, we may wish to use a CE scheme so that multiple different ciphertexts, encrypted under the same secret key, can be opened without endangering the privacy or integrity of other unopened ciphertexts. In other contexts, the CE scheme's opening need only be "single-use" — the secret key will not continue to be used after an opening. An example of the latter is Signal, which due to ratcheting effectively has a fresh secret key per message. As we will now discuss, whether one wants single-opening or multiple-opening CE must be reflected in the security definitions.

Confidentiality. We want our CE schemes to provide message confidentiality. We will in fact adapt the stronger real-or-random notion from the AE literature [60] to CE. At a high level we ask that no adversary can distinguish between legitimate CE encryptions and (pairs of) random bit strings. A complexity arises in the multi-opening case, where we want confidentiality to hold even after openings occur. We handle this by giving the attacker an additional pair of oracles, one for encryption and decryption. We must take care to avoid trivial wins, of course, separating use of the real oracles from the challenge ones. We also additionally require that the adversary can only query its decryption oracle on valid ciphertexts returned from the encryption oracle. This all is formalized in the games MO-REAL$^{\mathcal{A}}_{\mathsf{CE}}$ and MO-RAND$^{\mathcal{A}}_{\mathsf{CE}}$ shown in Figure 4. We measure the multiple-openings real-or-random (MO-ROR) advantage of an adversary $\mathcal{A}$ against a scheme CE by

$$\mathbf{Adv}^{\text{mo-ror}}_{\mathsf{CE}}(\mathcal{A}) = \left|\Pr\left[\,\text{MO-REAL}^{\mathcal{A}}_{\mathsf{CE}} \Rightarrow 1\,\right] - \Pr\left[\,\text{MO-RAND}^{\mathcal{A}}_{\mathsf{CE}} \Rightarrow 1\,\right]\right| \ .$$

The single-opening ROR (SO-ROR) games REAL$^{\mathcal{A}}_{\mathsf{CE}}$ and RAND$^{\mathcal{A}}_{\mathsf{CE}}$ are identical to MO-REAL$^{\mathcal{A}}_{\mathsf{CE}}$ and MO-RAND$^{\mathcal{A}}_{\mathsf{CE}}$ in Figure 4 except that we omit the **Enc** and **Dec** oracles. We measure the single-openings real-or-random (ROR) advantage of an adversary $\mathcal{A}$ against a scheme CE by

$$\mathbf{Adv}^{\text{ror}}_{\mathsf{CE}}(\mathcal{A}) = \left|\Pr\left[\,\text{REAL}^{\mathcal{A}}_{\mathsf{CE}} \Rightarrow 1\,\right] - \Pr\left[\,\text{RAND}^{\mathcal{A}}_{\mathsf{CE}} \Rightarrow 1\,\right]\right| \ .$$

```
s-BIND$_{\mathsf{CE}}^{\mathcal{A}}$:
$(K, H, C_1, C_2) \leftarrow\!\!\$\ \mathcal{A}$
$(M', K_f) \leftarrow \mathsf{Dec}(K, H, C_1, C_2)$
If $M' = \bot$ then Return false
$b \leftarrow \mathsf{Ver}(H, M', K_f, C_2)$
If $b = 0$ then
    Return true
Return false
```

```
r-BIND$_{\mathsf{CE}}^{\mathcal{A}}$:
$((H, M, K_f), (H', M', K_f'), C_2) \leftarrow\!\!\$\ \mathcal{A}$
$b \leftarrow \mathsf{Ver}(H, M, K_f, C_2)$
$b' \leftarrow \mathsf{Ver}(H', M', K_f', C_2)$
If $(H, M) = (H', M')$ then
    Return false
Return $(b = b' = 1)$
```

Figure 5: Binding security games for committing AEAD. Sender binding (left game) models a setting where a malicious sender wants to send a message, but prevent commitment opening from succeeding. Receiver binding (right game) models a setting where a sender and recipient collude to open a ciphertext to different messages.

Ciphertext integrity. We also want our CE schemes to enjoy ciphertext integrity. As with confidentiality, we will lift the standard (randomized) AEAD security notions to the multiple-opening and single-opening CE settings. The game MO-CTXT$_{\mathsf{CE}}^{\mathcal{A}}$ is shown in Figure 4. The adversary can obtain encryptions and decryptions under the secret key, and its goal is to query a valid ciphertext to a challenge decryption oracle. That ciphertext must not have been returned by the encryption oracle. We measure the multiple-openings ciphertext integrity (MO-CTXT) advantage of an adversary $\mathcal{A}$ against a scheme CE by

$$\mathbf{Adv}_{\mathsf{CE}}^{\text{mo-ctxt}}(\mathcal{A}) = \Pr\left[\,\text{MO-CTXT}_{\mathsf{CE}}^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] .$$

As with confidentiality, we can also specify a single-opening version of security by removing the decryption oracle **Dec** from game MO-CTXT$_{\mathsf{CE}}^{\mathcal{A}}$. Let the resulting game be CTXT$_{\mathsf{CE}}^{\mathcal{A}}$. We measure the single-openings ciphertext integrity (CTXT) advantage of an adversary $\mathcal{A}$ against a scheme CE by

$$\mathbf{Adv}_{\mathsf{CE}}^{\text{ctxt}}(\mathcal{A}) = \Pr\left[\,\text{CTXT}_{\mathsf{CE}}^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] .$$

All-in-one notions. We have given separate confidentiality and ciphertext integrity notions. As with traditional AEAD security, however, we can alternatively give an all-in-one notion that simultaneously captures confidentiality and integrity goals. We defer the details to Appendix B.

Security for AEAD. Given the fact that CE schemes encompass (randomized) AEAD schemes as well (see our comments above), we note that the ROR and CTXT notions apply to standard (randomized) AE schemes. As a slight abuse of notation, we will therefore use ROR and CTXT and their associated games and advantage measures for the security of traditional AE schemes.

Binding security notions. We introduce two security notions for binding: *sender binding* and *receiver binding*. Sender binding ensures the sender of a message is bound to the message it actually sent. In abuse-reporting scenarios, this prevents the sender of an abusive message from generating a bogus commitment that does not give the receiver the ability to report the message. The pseudocode game s-BIND on the left-hand-side of Figure 5 formalizes this requirement. To an adversary $\mathcal{A}$ and CE scheme CE we associate the sender binding advantage

$$\mathbf{Adv}_{\mathsf{CE}}^{\text{s-bind}}(\mathcal{A}) = \Pr\left[\,\text{s-BIND}_{\mathsf{CE}}^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] .$$

A CE scheme can generically meet sender binding by running Ver during Dec and having Dec return $\bot$ if Ver returns 0. We omit the proof of this, which follows by inspection. But legacy AEAD schemes do not do this, and one needs to check sender binding. For new schemes we will see more efficient ways to achieve sender binding.

The second security notion, receiver binding, is a lifting of the more traditional binding notion from commitment schemes (see Section 2). This definition is important in abuse reporting, where it formalizes the intuition that a malicious receiver should not be able to accuse a non-abusive sender of having said something abusive. A malicious receiver could do this by opening one of the sender's ciphertexts to an abusive message instead of the one the sender intended.

The pseudocode game r-BIND is shown on the right in Figure 5. It has an adversary output a pair of triples containing associated data, a message, and an opening. The adversary outputs a franking tag C_2 as well. The adversary wins if verification succeeds on both triples with C_2 and the header/message pairs differ. To a CE scheme CE and adversary $\mathcal{A}$ we associate the receiver binding advantage

$$\mathbf{Adv}^{\text{r-bind}}_{\mathsf{CE}}(\mathcal{A}) = \Pr\left[\,\text{r-BIND}^{\mathcal{A}}_{\mathsf{CE}} \Rightarrow \mathsf{true}\,\right] .$$

It is important to note that r-BIND security does not imply s-BIND security. These notions are, in fact, orthogonal. Moreover, our MO-ROR and MO-CTXT notions do not generically imply either of the binding notions.

Discussion. Our definitions also allow associated data, sometimes referred to as headers. This puts committing AEAD on equal footing with modern authenticated encryption with associated data (AEAD) schemes [56], which require it. That said, modern AEAD schemes are increasingly formalized as nonce-based, meaning that instead of allowing internal randomness, a non-repeating value (the nonce) is an explicit input and encryption is deterministic. Existing systems relevant to abuse complaints use randomized AEAD (e.g., Signal [48]) that do not meet nonce-based AEAD security. That said, we will explore nonce-based committing AEAD in Section 7.

5 Are Existing AEAD Schemes Committing?

In this section we study whether existing AEAD schemes meet our security goals for CE. We believe it is important to study legacy schemes for several reasons. If existing AEAD schemes are also traditionally committing when the key is used as the opening, it will have important implications for deployed protocols (such as Facebook's franking scheme) that implicitly rely on binding properties of symmetric encryption. It is also helpful for protocol designers who may want to build a protocol on top of existing legacy encryption (e.g. [63]). If well-tested, mature implementations of AEAD can be used as CE schemes without code changes, the attack surface of new protocol implementations is minimized.

Security of encryption schemes under key misuse has been studied in prior work on *robust encryption* [1, 33, 34, 52]. Informally, an encryption scheme is robust if it is difficult to find a key (other than the one used to encrypt) which correctly decrypts a ciphertext. In [34] Farshim et al. introduce a "best-possible" robustness notion (FROB security) for symmetric primitives. In Appendix A we show that when associated data is excluded, FROB implies r-BIND security for traditionally committing schemes. We also show that with associated data, FROB and r-BIND are incomparable. Since our positive results in this section include associated data, the proofs of FROB security for encrypt-then-MAC (EtM) and MAC-then-encrypt (MtE) variants from [34] do not imply our results. See Appendix A for more discussion of the connection between robustness and binding in authenticated encryption.

We only examine the binding properties of schemes, since past work has shown they meet standard definitions for confidentiality and integrity in the single-opening setting. Our positive results below imply the schemes are binding when the entire ciphertext is the commitment (i.e. they are traditionally committing). We will actually prove stronger statements that show the schemes are still binding when the commitment is only a substring of the full ciphertext (i.e. they

```
Enc(K, H, M):
    $R \leftarrow_{\$} \{0,1\}^r$
    $C \leftarrow \widetilde{E}^H_K(M \| R \| 0^s)$
    $\ell \leftarrow l + r + s - t$
    $C_1 \leftarrow C[1, \ldots, \ell]$
    $C_2 \leftarrow C[\ell + 1, \ldots, l + r + s]$
    Return $(C_1, C_2)$
```

```
Dec(K, H, C_1, C_2):
    $M' \| R' \| Z \leftarrow \widetilde{D}^H_K(C_1 \| C_2)$
    If $Z \neq 0^s$ then
        Return $\perp$
    Return $(M', (R', K))$
```

```
Ver(H, M, K_f, C_2):
    $R \| K \leftarrow K_f$
    $\ell \leftarrow l + r + s - t$
    $C \leftarrow \widetilde{E}^H_K(M \| R \| 0^s)$
    Return $C[\ell + 1, \ldots, l + r + s] = C_2$
```

Figure 6: Encode-then-encipher as a committing AEAD scheme where the commitment is the final t bits of the ciphertext. $\widetilde{E}^H_K$ and $\widetilde{D}^H_K$ refer to encryption and decryption for a tweakable blockcipher TC where the header H is the tweak and K is the key.

are compactly committing). We begin by proving that encode-then-encipher satisfies our binding notions in the ideal cipher model. Then, we will prove the EtM generic composition satisfies our binding notions if the MAC used in EtM is a collision-resistant PRF. We will prove MtE meets our binding notions in the random oracle and ideal cipher model. We will conclude with some simple attacks that break binding for real-world modes using Carter-Wegman MACs (GCM and ChaCha20/Poly1305).

5.1 Committing Encode-then-Encipher

The Encode-then-Encipher (EtE) construction of Bellare and Rogaway shows how to achieve AE security for messages given only a variable-input-length PRP [12]. Their construction is simple: given a variable-input-length tweakable cipher TC $= (\widetilde{E}, \widetilde{D})$ and key $K \in \mathcal{K}$, encrypt a message $M \in \mathcal{M}$ ($|\mathcal{M}| = 2^l$) with header $H \in \mathcal{H}$ by first drawing a random string $R \leftarrow_{\$} \{0,1\}^r$ and computing $c = \widetilde{E}^H_K(M \| R \| 0^s)$ (using the header H as the tweak). Decrypting a ciphertext M works by first running $M' = \widetilde{D}^H_K(C)$ and checking whether the last s bits of M' are all zero. If they are, we call the message "valid" and output M, else we output $\perp$. For compactness, we commit to only the last t bits of the ciphertext. We must include the randomness used to encrypt in the opening of the commitment.

To analyze r-BIND security, we will assume that TC is an ideal tweakable cipher—that is, a fresh uniformly random permutation is drawn for each distinct key and tweak pair input to encryption or decryption. Instantiating the ideal tweakable cipher used in our proof is not straightforward. An AEZ variant [44] (which uses a collision-resistant hash for associated data) could be used. Doubts have been raised about the security of AEZ, however [25], and with suitable modifications more mature variable-input-length tweakable cipher constructions such as CMC [39] or EME [40] could perhaps be used instead. We note that while these constructions have been proven to be strong pseudorandom permutations, it is not currently known if either are indifferentiable from an ideal tweakable cipher.

Now we will prove encode-then-encipher when used according to the pseudocode in Figure 6 meets r-BIND security when a length-t substring of the ciphertext is used as the commitment. Our bound is in terms of the number of ideal cipher queries made during the course of the game, both by the adversary and by the game itself during Ver. The scheme achieves perfect s-BIND security: the advantage of any adversary is zero because the output of decryption is re-computed in Ver.

Theorem 2 *Let* EtE[TC] *be the scheme defined above using an ideal tweakable cipher* TC *and parameters $s, t > 0$. Let $\mathcal{A}$ be a* r-BIND$_{\text{EtE[TC]}}$ *adversary and let q be the total number of queries made by the adversary and by the game itself during Ver. Then* $\mathbf{Adv}^{\text{r-bind}}_{\text{EtE[TC]}}(\mathcal{A}) \leq \frac{q}{2^{s-1}} + \frac{q^2}{2^{t-1}}$.

```
$G_0$ $\boxed{G_1}$:
$B \leftarrow \{P \| 0^s \mid P \in \{0,1\}^{l+r}\}$
$((H, M, K_f), (H', M', K'_f), C_2) \leftarrow\$ \mathcal{A}^{\widetilde{E},\widetilde{D}}$
$b \leftarrow \mathsf{Ver}(H, M, K_f, C_2)$
$b' \leftarrow \mathsf{Ver}(H', M', K'_f, C_2)$
If $(H, M) = (H', M')$ then
    Return $\mathsf{false}$
Return $(b = b' = 1)$

$\widetilde{E}(K, H, M)$:
If $\mathrm{R}[K, H, M] \neq \perp$:
  return $\mathrm{R}[K, H, M]$
$C \leftarrow\$ \mathrm{Rng}[K, H]$
If $M \in B$ and $C \in \mathbb{V}$ then
    $\mathsf{bad}_{12} \leftarrow \mathsf{true}$
$\ell \leftarrow l + r + s - t$
$C_2 \leftarrow C[\ell + 1, \ldots, l + r + s]$
$\mathbb{V} \overset{\cup}{\leftarrow} \{S \| C_2 \mid S \in \{0,1\}^{\ell}\}$
$\mathrm{R}[K, H, M] \leftarrow C$
$\mathrm{Rng}[K, H] \leftarrow \mathrm{Rng}[K, H] \setminus \{C\}$
$\mathrm{D}[K, H, C] \leftarrow M$
$\mathrm{Dom}[K, H] \leftarrow \mathrm{Dom}[K, H] \setminus \{M\}$
Return $\mathrm{R}[K, H, M]$

$\widetilde{D}(K, H, C)$:
If $\mathrm{D}[K, H, C] \neq \perp$:
  return $\mathrm{D}[K, H, C]$
$M \leftarrow\$ \mathrm{Dom}[K, H]$
If $M \in B$ then
  $\mathsf{bad}_{01} \leftarrow \mathsf{true}$
  $\boxed{M \leftarrow\$ \mathrm{Dom}[K, H] \setminus B}$
$\mathrm{D}[K, H, C] \leftarrow M$
$\mathrm{Dom}[K, H] \leftarrow \mathrm{Dom}[K, H] \setminus \{M\}$
$\mathrm{R}[K, H, M] \leftarrow C$
$\mathrm{Rng}[K, H] \leftarrow \mathrm{Rng}[K, H] \setminus \{C\}$
Return $\mathrm{D}[K, H, C]$

$\mathsf{Ver}(H, M, K_f, C_2)$:
$R \| K \leftarrow K_f$
$\ell \leftarrow l + r + s - t$
$C \leftarrow \widetilde{E}(K, H, M \| R \| 0^s)$
Return $C[\ell + 1, \ldots, l + r + s] = C_2$
```

```
$G_2$:
$B \leftarrow \{P \| 0^s \mid P \in \{0,1\}^{l+r}\}$
$((H, M, K_f), (H', M', K'_f), C_2) \leftarrow\$ \mathcal{A}^{\widetilde{E},\widetilde{D}}$
$b \leftarrow \mathsf{Ver}(H, M, K_f, C_2)$
$b' \leftarrow \mathsf{Ver}(H', M', K'_f, C_2)$
If $(H, M) = (H', M')$ then
    Return $\mathsf{false}$
Return $(b = b' = 1)$

$\widetilde{E}(K, H, M)$:
If $\mathrm{R}[K, H, M] \neq \perp$ then
  return $\mathrm{R}[K, H, M]$
$C \leftarrow\$ \mathrm{Rng}[K, H]$
If $M \in B$ and $C \in \mathbb{V}$ then
    $\mathsf{bad}_{12} \leftarrow \mathsf{true}$
    $\boxed{C \leftarrow\$ \mathrm{Rng}[K, H] \setminus \mathbb{V}}$
$\ell \leftarrow l + r + s - t$
$C_2 \leftarrow C[\ell + 1, \ldots, l + r + s]$
$\mathbb{V} \overset{\cup}{\leftarrow} \{S \| C_2 \mid S \in \{0,1\}^{\ell}\}$
$\mathrm{R}[K, H, M] \leftarrow C$
$\mathrm{Rng}[K, H] \leftarrow \mathrm{Rng}[K, H] \setminus \{C\}$
$\mathrm{D}[K, H, C] \leftarrow M$
$\mathrm{Dom}[K, H] \leftarrow \mathrm{Dom}[K, H] \setminus \{M\}$
Return $\mathrm{R}[K, H, M]$

$\widetilde{D}(K, H, C)$:
If $\mathrm{D}[K, H, C] \neq \perp$:
  return $\mathrm{D}[K, H, C]$
$M \leftarrow\$ \mathrm{Dom}[K, H] \setminus B$
$\mathrm{D}[K, H, C] \leftarrow M$
$\mathrm{Dom}[K, H] \leftarrow \mathrm{Dom}[K, H] \setminus \{M\}$
$\mathrm{R}[K, H, M] \leftarrow C$
$\mathrm{Rng}[K, H] \leftarrow \mathrm{Rng}[K, H] \setminus \{C\}$
Return $\mathrm{D}[K, H, C]$

$\mathsf{Ver}(H, M, K_f, C_2)$:
$R \| K \leftarrow K_f$
$\ell \leftarrow l + r + s - t$
$C \leftarrow \widetilde{E}(K, H, M \| R \| 0^s)$
Return $C[\ell + 1, \ldots, l + r + s] = C_2$
```

Figure 7: EtE security games for proof of Theorem 2.

Proof: The games for this proof are in Figure 7. Game G_0 is a rewriting of the game r-BIND$^{\mathcal{A}}_{\text{EtE}}$. In all games we use lazy sampling for the ideal ciphers.

We will use two game transitions to move to a game in which the adversary's probability of success is zero. The first transition is from G_0 to G_1. Game G_1 is the same as G_0 but with the boxed statement included. The boxed code sets a flag if the random plaintext chosen as the decryption of a query is valid (i.e. it ends with s zero bits). Importantly, note that decrypting the ciphertext of a previously-encrypted valid point will *not* set this flag, because it will return the point stored in our table D. By the fundamental lemma of game-playing [13] we have that

$$\Pr[\, G_0 \Rightarrow \mathsf{true} \,] \leq \Pr[\, G_1 \Rightarrow \mathsf{true} \,] + \Pr[\, G_1 \text{ sets } \mathsf{bad}_{01} \,] \ .$$

To bound the rightmost term, observe that after i queries which do not set the flag, there are at most $2^{l+r+s} - i$ points remaining which can be sampled. Of those points which can still be sampled after i queries, 2^{l+r} will set the flag. Thus, the probability of setting the flag on the ith query is at most $\frac{2^{l+r}}{2^{l+r+s}-i}$, and a union bound gives us

$$\Pr[\, G_1 \text{ sets } \mathsf{bad}_{01} \,] \leq \sum_{i=1}^{q} \frac{2^{l+r}}{2^{l+r+s} - i} \ .$$

The largest term in this summation is upper-bounded by $\frac{2^{l+r}}{2^{l+r+s-1}}$, so we can rewrite as

$$\Pr[\, G_1 \text{ sets } \mathsf{bad}_{01} \,] \leq \sum_{i=1}^{q} \frac{1}{2^{s-1}} = \frac{q}{2^{s-1}} \ .$$

The next transition is from G_1 to G_2. Game G_2 is the same as G_1 except for the boxed code in encryption, which re-samples the ciphertext C if the one previously sampled has the same t-bit suffix as any previous ciphertext. We can again use the fundamental lemma of game-playing to get that

$$\Pr[\, G_1 \Rightarrow \mathsf{true} \,] \leq \Pr[\, G_2 \Rightarrow \mathsf{true} \,] + \Pr[\, G_2 \text{ sets } \mathsf{bad}_{12} \,] \ .$$

To bound the rightmost term, we need to count the number of points in the set $\mathbb{V}$ after i failed queries. When we generate a ciphertext with some t-bit suffix, we add all points with that suffix to $\mathbb{V}$. After i queries there are at most $i2^{l+r+s-t}$ strings in $\mathbb{V}$, and $2^{l+r+s} - i$ points left to sample. This upper bound holds even if the adversary queries with multiple different keys. A union bound over the queries gives us

$$\Pr[\, G_2 \text{ sets } \mathsf{bad}_{12} \,] \leq \sum_{i=1}^{q} \frac{i2^{l+r+s-t}}{2^{l+r+s} - i} \ .$$

Each term in the summation is upper-bounded by $\frac{i2^{l+r+s-t}}{2^{l+r+s-1}}$, so rewriting and cancelling gives us

$$\Pr[\, G_2 \text{ sets } \mathsf{bad}_{12} \,] \leq \sum_{i=1}^{q} \frac{i}{2^{t-1}} \leq \frac{q^2}{2^{t-1}} \ .$$

To complete the proof, note that $\Pr\left[\, G_2^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right] = 0$, since none of its queries can output a ciphertext which decrypts to a valid message under two different keys. Thus,

$$\mathbf{Adv}^{\text{r-bind}}_{\text{EtE[TC]}}(\mathcal{A}) \leq \Pr[\, G_1 \text{ sets } \mathsf{bad}_{01} \,] + \Pr[\, G_2 \text{ sets } \mathsf{bad}_{12} \,] \leq \frac{q}{2^{s-1}} + \frac{q^2}{2^{t-1}} \ .$$

▮

```
$\underline{\mathsf{Enc}(K, H, M)}$:
$K^e \leftarrow \mathsf{KDF}_K(0)$
$K^m \leftarrow \mathsf{KDF}_K(1)$
$R \leftarrow\!\!\$\ \mathcal{R}$
$R \,\|\, C \leftarrow \mathsf{enc}_{K^e}(M\ ;\ R)$
$T \leftarrow F_{K^m}(H \,\|\, R \,\|\, C)$
Return $(R \,\|\, C, T)$
```

```
$\underline{\mathsf{Dec}(K, H, C_1, C_2)}$:
$R \,\|\, C \leftarrow C_1$
$K^e \leftarrow \mathsf{KDF}_K(0)$
$K^m \leftarrow \mathsf{KDF}_K(1)$
$T' \leftarrow F_{K^m}(H \,\|\, R \,\|\, C_1)$
If $T' \neq C_2$ then Return $\perp$
$M \leftarrow \mathsf{dec}_{K^e}(C_1)$
If $M = \perp$ then Return $\perp$
Return $(M, (R, K))$
```

```
$\underline{\mathsf{Ver}(H, M, (R, K), C_2)}$:
$K^e \leftarrow \mathsf{KDF}_K(0)$
$K^m \leftarrow \mathsf{KDF}_K(1)$
$C \leftarrow \mathsf{enc}_{K^e}(M\ ;\ R)$
$T \leftarrow F_{K^m}(H \,\|\, R \,\|\, C)$
Return $T = C_2$
```

Figure 8: Committing AEAD scheme $\mathsf{EtM}[\mathsf{KDF}, F, \mathsf{SE}]$ that composes an encryption scheme $\mathsf{SE} = (\mathsf{kg}, \mathsf{enc}, \mathsf{dec})$ using random coins from $\mathcal{R}$, a MAC F, and that derives keys via a function KDF.

5.2 Encrypt-then-MAC

The classic Encrypt-then-MAC (EtM) construction composes a symmetric encryption scheme and a message authentication code (MAC), by first encrypting the message, then computing the MAC over the ciphertext and any associated data. We first prove that EtM is binding if a collision-resistant key derivation function (KDF) is used to derive separate encryption and authentication keys. Then, we give a counterexample that shows EtM is not binding if encryption and MAC keys are specified directly and not derived via a KDF.

Committing EtM. We analyze EtM as a committing AEAD scheme in the case that the encryption and authentication keys are derived via a KDF that is a collision-resistant pseudorandom function. The scheme $\mathsf{EtM}[\mathsf{KDF}, F, \mathsf{SE}]$ is detailed in Figure 8. Beyond the functions F and KDF, the scheme also makes use of a public-coin randomized symmetric encryption algorithm $\mathsf{SE} = (\mathsf{kg}, \mathsf{enc}, \mathsf{dec})$ that does not use associated data and whose key generation is a random selection of some fixed-length bit string. It is important that the scheme is public coin, as we require that the randomness is recoverable during decryption so it can be included in the opening.

There are deployed protocols which use a collision-resistant KDF to derive encryption and authentication keys for EtM. The Signal protocol [48], for example, uses HKDF to derive keys for use with CTR mode encryption combined with HMAC. An "opening" phase would be added to the protocol using these keys, and a new analysis would likely be needed to verify the protocol remains secure when keys are exposed by opening a ciphertext. We leave the details as an interesting open problem for future work. The following theorem proves the committing EtM construction in Figure 8 meets r-BIND if the MAC and key derivation function are both collision-resistant PRFs. The s-BIND security of $\mathsf{EtM}[\mathsf{KDF}, F, \mathsf{SE}]$ is perfect because verification re-encrypts the plaintext to check the tag.

Theorem 3 *Let* $\mathsf{EtM} = \mathsf{EtM}[\mathsf{KDF}, F, \mathsf{SE}]$ *be the EtM construction using functions* F *and* KDF *as well as encryption scheme* SE*. Let* $\mathcal{A}$ *be any* r-BIND$_{\mathsf{EtM}}$ *adversary. Then there exist adversaries* $\mathcal{B}$ *and* $\mathcal{C}$*, each that run in time that of* $\mathcal{A}$*, such that* $\mathbf{Adv}^{\text{r-bind}}_{\mathsf{EtM}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{cr}}_{F}(\mathcal{B}) + \mathbf{Adv}^{\text{cr}}_{\mathsf{KDF}}(\mathcal{C})$.

Proof: In this proof we will refer to Figure 9. We will use two game hops to transition to a game in which the adversary's probability of winning is zero.

Start with game G_0, which is a syntactic rewriting of r-BIND$^{\mathcal{A}}_{\mathsf{EtM}}$. Thus $\Pr\left[\,\text{r-BIND}^{\mathcal{A}}_{\mathsf{EtM}} \Rightarrow \mathsf{true}\,\right] = \Pr\left[\,G_0 \Rightarrow \mathsf{true}\,\right]$. From G_0 we will transition to the game G_1, which is identical to G_0 except for the boxed code which sets $\tilde{K}^m$ to be different from K^m (in particular, it sets $\tilde{K}^m$ to be the bitwise complement of K^m) if flag bad is set. The flag is set only if $K^m = \tilde{K}^m$. The games G_0 and G_1 are

```
$G_0$ $\boxed{G_1}$:
$(P, P', C_2) \leftarrow\!\!\$\ \mathcal{A}$
$(H, M, K_f) \leftarrow P$
$(\tilde{H}, \tilde{M}, \tilde{K_f}) \leftarrow P'$
If $(H, M) = (\tilde{H}, \tilde{M})$ then
    Return $\mathsf{false}$
$IV, K \leftarrow K_f$
$K^e \leftarrow \mathsf{KDF}_K(0)$
$K^m \leftarrow \mathsf{KDF}_K(1)$
$C \leftarrow \mathsf{enc}_{K^e}(IV; M)$
$T \leftarrow F_{K^m}(H \parallel IV \parallel C)$
$b \leftarrow (T = C_2)$
$\tilde{IV}, \tilde{K} \leftarrow \tilde{K_f}$
$\tilde{K}^e \leftarrow \mathsf{KDF}_{\tilde{K}}(0)$
$\tilde{K}^m \leftarrow \mathsf{KDF}_{\tilde{K}}(1)$
If $K^m = \tilde{K}^m$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$
    $\boxed{\tilde{K}^m = \neg K^m}$
$\tilde{C} \leftarrow \mathsf{enc}_{\tilde{K}^e}(\tilde{IV}; \tilde{M})$
$\tilde{T} \leftarrow F_{\tilde{K}^m}(\tilde{H} \parallel \tilde{IV} \parallel \tilde{C})$
If $T = \tilde{T}$ then
    $\mathsf{bad}_1 \leftarrow \mathsf{true}$
$\tilde{b} \leftarrow (\tilde{T} = C_2)$
return $b \wedge \tilde{b}$
```

```
$G_2$:
$(P, P', C_2) \leftarrow\!\!\$\ \mathcal{A}$
$(H, M, K_f) \leftarrow P$
$(\tilde{H}, \tilde{M}, \tilde{K_f}) \leftarrow P'$
If $(H, M) = (\tilde{H}, \tilde{M})$ then
    Return $\mathsf{false}$
$IV, K \leftarrow K_f$
$K^e \leftarrow \mathsf{KDF}_K(0)$
$K^m \leftarrow \mathsf{KDF}_K(1)$
$C \leftarrow \mathsf{enc}_{K^e}(IV; M)$
$T \leftarrow F_{K^m}(H \parallel IV \parallel C)$
$b \leftarrow (T = C_2)$
$\tilde{IV}, \tilde{K} \leftarrow \tilde{K_f}$
$\tilde{K}^e \leftarrow \mathsf{KDF}_{\tilde{K}}(0)$
$\tilde{K}^m \leftarrow \mathsf{KDF}_{\tilde{K}}(1)$
If $K^m = \tilde{K}^m$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$
    $\tilde{K}^m = \neg K^m$
$\tilde{C} \leftarrow \mathsf{enc}_{\tilde{K}^e}(\tilde{IV}; \tilde{M})$
$\tilde{T} \leftarrow F_{\tilde{K}^m}(\tilde{H} \parallel \tilde{IV} \parallel \tilde{C})$
If $T = \tilde{T}$ then
    $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    $\boxed{\tilde{T} = \neg T}$
$\tilde{b} \leftarrow (\tilde{T} = C_2)$
return $b \wedge \tilde{b}$
```

Figure 9: Game for Theorem 3.

identical-until-bad, so by the fundamental lemma of code-based games we can write

$$\Pr[\, G_0 \Rightarrow \mathsf{true} \,] \leq \Pr[\, G_1 \Rightarrow \mathsf{true} \,] + \Pr[\, \mathsf{bad} \text{ is set} \,] \ .$$

We can bound the probability bad is set using a reduction to the collision-resistance of KDF. Call this reduction $\mathcal{C}$, and we have $\Pr[\, \mathsf{bad} \text{ is set} \,] \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{KDF}}(\mathcal{C})$.

Next we transition from game G_1 to G_2. Game G_2 is the same as G_1 except for the boxed code, which sets $\tilde{T}$ to be the bitwise complement of T (ensuring they are different) if flag bad_1 is set. The flag is set only if $T = \tilde{T}$. Using the same lemma as above gives us that

$$\Pr[\, G_1 \Rightarrow \mathsf{true} \,] \leq \Pr[\, G_2 \Rightarrow \mathsf{true} \,] + \Pr[\, \mathsf{bad}_1 \text{ is set} \,] \ .$$

As above, we can bound the probability the flag is set by a reduction to the collision-resistance of F. Call this reduction $\mathcal{B}$, and we have $\Pr[\, \mathsf{bad}_1 \text{ is set} \,] \leq \mathbf{Adv}^{\mathrm{cr}}_{F}(\mathcal{B})$.

We conclude by noting that $\Pr[\, G_2 \Rightarrow \mathsf{true} \,] = 0$ since for both b and $\tilde{b}$ to be true, T and $\tilde{T}$ must be equal. The second game transition ensures they are not equal. Combining and rewriting gives us the bound. ▮

Two-key EtM is not binding. The use of a KDF to derive the encryption and MAC keys above is requisite to achieve receiver binding security. Consider omitting the KDF steps, and instead letting keys be a pair (K^e, K^m) where each component is chosen randomly. The opening output by encryption and used by verification is instead $(R, (K^e, K^m))$. The rest of the scheme remains the same as that in Figure 8. But it is easy to break the receiver binding for this two-key variant: have an adversary $\mathcal{A}$ choose an arbitrary header H, message M, keys (K^e, K^m), and randomness R, and compute $R \| C \leftarrow \mathsf{enc}_{K^e}(M; R)$ and then $T \leftarrow F_{K^m}(H \| R \| C)$. It then chooses another key $\widetilde{K}^e \neq K^e$,

```
Enc(K, H, M):
  $K^e, K^m \leftarrow K$
  $IV \leftarrow\$ \{0,1\}^n$
  $T \leftarrow \mathrm{RO}_{K^m}(H \parallel M)$
  $C \leftarrow \mathrm{CBC}_{K^e}(\mathrm{Pad}_n(M \parallel T)\ ;\ IV)$
  $\ell \leftarrow \mathrm{Pad}_n(M \parallel T)/n$
  $C' \parallel C_{\ell-2} \parallel C_{\ell-1} \parallel C_\ell \leftarrow C$
  Return $(C', IV \parallel C_{\ell-2} \parallel C_{\ell-1} \parallel C_\ell)$
```

```
Dec(K, H, C_1, C_2):
  $K^e, K^m \leftarrow K$
  $IV \parallel C_{\ell-2} \parallel C_{\ell-1} \parallel C_\ell \leftarrow C_2$
  $C_f \leftarrow C_{\ell-2} \parallel C_{\ell-1} \parallel C_\ell$
  $M \parallel T \leftarrow \mathrm{CBC}^{-1}_{K^e}(C_1 \parallel C_f\ ;\ IV)$
  $T' \leftarrow \mathrm{RO}_{K^m}(H \parallel M)$
  If $T \neq T'$ then Return $\perp$
  Return $(M, (K^e, K^m))$
```

```
Ver(H, M, K_f, C_2):
  $K^e, K^m \leftarrow K_f$
  $IV \parallel C'_{\ell-2} \parallel C'_{\ell-1} \parallel C'_\ell \leftarrow C_2$
  $T \leftarrow \mathrm{RO}_{K^m}(H \parallel M)$
  $P \leftarrow \mathrm{Pad}_n(M \parallel T)$
  $C \leftarrow \mathrm{CBC}_{K^e}(P\ ;\ IV)$
  $C' \parallel C''_{\ell-2} \parallel C''_{\ell-1} \parallel C''_\ell \leftarrow C$
  Return $\bigwedge_{i=\ell-2}^{\ell} (C''_i = C'_i)$
```

Figure 10: Committing authenticated encryption based on MtE composition of CBC mode and a MAC modeled as a random oracle. The length ℓ is defined to be $\mathrm{Pad}_n(M \| T)/n$. The function Pad is the standard (min-length) PKCS#7 padding used in TLS. The notation $\mathrm{CBC}_K(\cdot\ ;\ IV)$ and $\mathrm{CBC}_K^{-1}(\cdot\ ;\ IV)$ means CBC mode encryption and decryption with key K and initialization vector IV.

and computes $\widetilde{M} \leftarrow \mathsf{dec}_{\widetilde{K}^e}(R \parallel C)$. Finally, it outputs $(H, (R, (K^e, K^m))), (H, (R, (\widetilde{K}^e, K^m))), T)$. This adversary will win the r-BIND game with probability close to one, assuming SE is such that decrypting the same ciphertext under different keys yields distinct plaintexts with overwhelming probability. A similar counterexample was given by Farshim et al. [34] to separate two notions of robustness for authenticated encryption.

5.3 MAC-then-Encrypt

The MAC-then-encrypt mode generically composes a MAC and an encryption scheme by first computing the MAC of the header and message, then appending the MAC to the message and encrypting them both. The pseudocode in Figure 10 uses for concreteness CBC mode encryption and we refer to this committing AEAD scheme as MtE. We will also assume the MAC is suitable to be modeled as a keyed random oracle; HMAC-SHA256 is one such [29]. CBC with HMAC in an MtE mode is a common cipher suite for modern TLS connections, which motivated these choices. Prior work has investigated the security of MtE in the sense of CTXT [46,55] and its ROR security is inherited directly from the encryption mode. We do not allow the empty message to be encrypted in our scheme. Below we will assume that the block size of n bits for the cipher underlying CBC mode, and that our MACs have output length $2n$ bits. With suitable modifications to the scheme, the result below generalizes to other parameter choices as well. Instead of including the IV in the opening (as in EtM), we include the IV in the commitment. Because the first ciphertext block in CBC mode is the XOR of the IV and the first plaintext block, putting the IV in the opening would give the attacker the ability to create a "free" collision in the first block of the ciphertext. Since we disallow the empty message, in the worst case (where the message is a single bit) this would allow the adversary to break r-BIND security after only about $q^2/2^{n+1}$ random oracle queries.

Unlike with Encrypt-then-MAC, we are able to prove the two-key version of MtE secure in the sense of receiver binding. The binding security of MtE in the case where keys are derived via a KDF follows as a corollary, though we believe better bounds can be achieved in that case. Our bound below is in terms of the number of ideal cipher and random oracle queries made both by the adversary and by the game itself during Ver. A sketch of an argument that MtE is binding (in the traditional sense where the entire ciphertext is the commitment) appeared in [63]. Their approach, which only relied on modeling the MAC as a RO and made no assumptions about CBC mode, led to a rather loose bound. We instead additionally model the cipher underlying CBC as ideal.

Theorem 4 *Let* MtE *be the scheme defined above using a random oracle and an ideal cipher in CBC mode. Let* n *be the block size of the ideal cipher, and let* $2n$ *be the output length of the random oracle. For any adversary* $\mathcal{A}$ *in game* r-BIND_{MtE} *where at most* q_i *queries are made to the ideal cipher (including those made by* Ver*) and* q_r *queries are made to the random oracle, it holds that* $\mathbf{Adv}^{\text{r-bind}}_{\mathsf{MtE}}(\mathcal{A}) < q_i q_r / 2^{2n}$.

Proof: Note first that the way we've specified the committing portion of MtE ciphertexts guarantees that the entire MAC will be encrypted in the commitment. Fixing a key K_1^e, message M, header H, and IV IV determines the ciphertext output *except* for the blocks containing bits of the MAC. If the commitment output by the adversary is a valid encryption of some MAC under some key, it can be decrypted at most q_i different ways given q_i queries to the ideal cipher underlying CBC mode. Since fixing the other key K_2^e fixes the decryption of the commitment, it also fixes the string T which must be $\text{RO}_{K^m}(H \,\|\, M)$. By a union bound the probability of hitting the right string T is at most $\frac{q_r}{2^{2n}}$ using q_r queries to the RO. A second union bound over the decryptions of the string C_2 yields the result. ∎

The s-BIND advantage against compactly-committing MtE is zero, since the commitment along with the output of a successful call to Dec uniquely defines the inputs to Ver. Thus, no other ciphertext can be computed in Ver other than the one previously decrypted in Dec, because the inputs to Ver are fixed by Dec.

MtE with CTR mode. With some modifications, our proof also applies when CTR mode is used instead of CBC. Changing to CTR mode does require some care, but even in the two-key case MtE seems to resist the attack against two-key EtM described in the previous section. We leave a more detailed treatment of the r-BIND security of MtE in CTR mode as an open problem for future work.

5.4 Some Non-binding AEAD schemes

In this section we will briefly detail attacks which break the receiver binding security of some deployed AEAD schemes. In particular, typical schemes that use MACs which are not collision resistant, such as Carter-Wegman MACs, do not suffice. For completeness we spell out an example of breaking the receiver binding of GCM [50], an encrypt-then-MAC style construction that uses a Carter-Wegman MAC.

A slight simplification of the GCM MAC is the function F shown in Figure 11 applied to a ciphertext. (We ignore associated data for simplicity.) It uses a key K for a block cipher E with block size n, as well as a nonce N. An initial point $P_0 \leftarrow E_K(0^n)$ and a pad $R \leftarrow E_K(N)$ are computed. GCM uses an ϵ-almost XOR universal (ϵ-AXU) [62] hash function computed by considering a ciphertext of m encrypted message blocks an m-degree polynomial defined over a finite field $\mathbb{F}$. The field is a particular representation of $\text{GF}(2^{128})$. This polynomial is evaluated at the encryption point P_0 and the result is XORed with the pad R. The GCM AEAD scheme encrypts the message using CTR mode encryption using E_K and a random 96-bit IV concatenated with a 32-bit counter initially set at one, and then MACs the resulting ciphertext $C = C_1, \ldots, C_m$ to generate a tag $T = F(K, IV \,\|\, 0^{32}, C_1, \ldots, C_m)$.

```
$F(K, N, (C_1, \ldots, C_m))$:
$P_0 \leftarrow E_K(0^n)$
$R \leftarrow E_K(N)$
$S \leftarrow \sum_{i=1}^{m} C_i P_0^{m-i}$
$T \leftarrow R \oplus S$
Return $T$
```

Figure 11: A simplified description of the CW MAC used in GCM.

A straightforward way to consider GCM as a compactly committing AEAD is to have encryption output as the commitment portion C_2 the tag T, and the

```
CtE1-Enc(K, H, M)
    $(K_f, C_2) \leftarrow\$ \mathsf{Com}(H \| M)$
    $C_1 \leftarrow\$ \mathsf{enc}_K(C_2, M \| K_f)$
    Return $(C_1, C_2)$
```

```
CtE2-Enc(K, H, M)
    $(K_f, C_2) \leftarrow\$ \mathsf{Com}(H \| M)$
    $C_1 \leftarrow\$ \mathsf{enc}_K(H, M \| K_f)$
    Return $(C_1, C_2)$
```

```
CtE1-Dec(K, H, C_1, C_2)
    $(M \| K_f) \leftarrow \mathsf{dec}_K(C_2, C_1)$
    If $M = \perp$ then Return $\perp$
    $b \leftarrow \mathsf{VerC}(K_f, C_2, H \| M)$
    If $b = 0$ then
        Return $\perp$
    Return $(M, K_f)$
```

```
CtE2-Dec(K, H, C_1, C_2)
    $(M \| K_f) \leftarrow \mathsf{dec}_K(H, C_1)$
    If $M = \perp$ then Return $\perp$
    $b \leftarrow \mathsf{VerC}(K_f, C_2, H \| M)$
    If $b = 0$ then
        Return $\perp$
    Return $(M, K_f)$
```

Figure 12: Algorithms for two Commit-then-Encrypt variants. Facebook's scheme uses CtE2 with an HMAC-based commitment. CtE1-Ver and CtE2-Ver both just output $\mathsf{VerC}(H, M, K_f, C_2)$.

rest of the ciphertext as the first portion C_1. Decryption works as usual for GCM, but additionally outputs (IV, K) as the opening. Verification works by recomputing encryption and checking that the resulting tag matches the commitment value C_2. We denote this scheme simply by $\mathsf{GCM} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ver})$ below.

We now give an r-BIND$_{\mathsf{GCM}}$ adversary $\mathcal{A}$. We ignore associated data for simplicity. To win, $\mathcal{A}$ must output $((M, (IV, K), (M', (IV', K'), T)$ so that $\mathsf{Ver}(M, (IV, K), T) = \mathsf{Ver}(M', (IV', K'), T) = 1$. We will build an $\mathcal{A}$ that chooses messages such that $|M| = |M'|$. The adversary $\mathcal{A}$ will start by choosing a ciphertext $C_1, \ldots, C_m$ such that

$$F(K, IV, C_1, \ldots, C_m) = F(K', IV', C_1, \ldots, C_m) \tag{1}$$

and letting M (resp. M') be the CTR-mode decryption of $C_1, \ldots, C_m$ under IV, K (resp. IV', K'). Choosing the ciphertext such that condition 1 holds is straightforward, as plugging in for the definition of F and rearranging, the adversary must solve the equation

$$\left[\sum_{i=1}^{m} C_i (P^{m-i} + (P')^{m-i})\right] + (E_K(N) + E_{K'}(N')) = 0$$

where $P \leftarrow E_K(0^n)$ and $P' \leftarrow E_{K'}(0^n)$. For example, pick arbitrary $C_1, \ldots, C_{m-1}$ and solve for the C_m that satisfies the equation.

This attack works even if associated data is used, or if the whole ciphertext is used as the commitment. A very similar attack works on ChaCha20/Poly1305 [15]; a small tweak is required to handle the fact that not every member of $\mathbb{F}_{2^{130}-5}$ is a valid ciphertext block.

6 Composing Commitment and AEAD

In the last section we saw that existing AEAD schemes already realize (compactly) committing AEAD in some cases. These schemes, however, only realize single-opening security as the opening includes the secret key. We now turn to schemes that achieve multi-opening committing AEAD, and focus specifically on schemes that generically compose AEAD with a commitment scheme.

Commit-then-Encrypt. We start with a simple general construction, what we call the Commit-then-Encrypt scheme.[6] It combines a commitment scheme $\mathsf{CS} = (\mathsf{Com}, \mathsf{VerC})$ with an AEAD scheme $\mathsf{SE} = (\mathsf{kg}, \mathsf{enc}, \mathsf{dec})$. Formally the scheme CtE1[CS, SE] = (kg, CtE1-Enc, CtE1-Dec, CtE1-Ver) works as shown in Figure 12.

The CtE1 scheme produces a commitment value to the message and associated data H, and then encrypts the message along with the opening of the commitment. It uses as associated data

[6]This name was also used in [37], but the scheme is distinct. See Section 9.

during encryption the commitment value, but not H. This nevertheless binds the underlying AEAD ciphertext to H as well as C_2 — as we will show tampering with either will be detected and rejected during decryption. One could additionally include H in the associated data for enc, but this would be less efficient. Should a protocol want H to not be in the commitment scope, one can instead include H only as associated data within enc and omit it from the commitment.

Theorem 5 *[*CtE1 *confidentiality] Let* $\text{CtE1} = \text{CtE1}[\mathsf{CS}, \mathsf{SE}]$. *Let $\mathcal{A}$ be an* $\text{MO-ROR}_{\text{CtE1}}$ *adversary making at most q queries to its oracles. Then we give adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{C}$ such that*

$$\mathbf{Adv}^{\text{mo-ror}}_{\text{CtE1}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ror}}_{\mathsf{SE}}(\mathcal{B}_1) + \mathbf{Adv}^{\text{ror}}_{\mathsf{SE}}(\mathcal{B}_2) + \mathbf{Adv}^{\text{cs-ror}}_{\mathsf{CS}}(\mathcal{C}) \ .$$

The adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{C}$ all run in the same amount of time as $\mathcal{A}$ with an $\mathcal{O}(q)$ overhead and make the same number of queries as $\mathcal{A}$.

Proof: We will use a sequence of game hops, bounding the distinguishing advantage of each hop. Let game G_0 be the same as $\text{MO-REAL}^{\mathcal{A}}_{\text{CtE1}}$. We can do a purely syntactic rewriting of this game to remove the Dec calls from **Dec**, instead indexing the message M and opening K_f in a table with the (H, C_1, C_2) tuple corresponding to the encryption query in which M and K_f were created.

Next, transition to the game G_1, which replaces C_1 values generated during **Enc** and **ChalEnc** queries in G_0 with random bits. The distinguishing advantage between G_0 and G_1 can be upper-bounded using a reduction to the ROR security of SE. Call this reduction $\mathcal{B}_1$.

Next, transition to the game G_2, which replaces the C_2 values output by **ChalEnc** with random bits. Crucially, we do not also replace C_2 values output by **Enc** — the adversary gets openings for these commitments, so it could distinguish the games with probability 1 by running Ver on the output. We can bound the distinguishing advantage between G_2 and G_1 using a reduction $\mathcal{C}$ to the real-or-random security of CS. This reduction is straightforward except we have $\mathcal{C}$ run Com on queries to **Enc**.

At this point, $\mathcal{A}$'s **ChalEnc** oracle is outputting random bits, as in game $\text{MO-RAND}^{\mathcal{A}}_{\text{CtE1}}$. We can do on more game hop using ROR security of SE to put real calls to Enc back into **Enc**. Call the reduction for this hop $\mathcal{B}_2$. Finally, we can undo our initial lossless game transition to have **Dec** call decryption instead of using a table lookup for decryption. Our final game is exactly $\text{MO-RAND}^{\mathcal{A}}_{\text{CtE1}}$. ▮

Theorem 6 *[*CtE1 *integrity] Let* $\text{CtE1} = \text{CtE1}[\mathsf{CS}, \mathsf{SE}]$. *Let $\mathcal{A}$ be an* $\text{MO-CTXT}_{\text{CtE1}}$ *adversary making at most q queries to its oracles. Then we give adversaries $\mathcal{B}$, $\mathcal{C}$ such that*

$$\mathbf{Adv}^{\text{mo-ctxt}}_{\text{CtE1}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ctxt}}_{\mathsf{SE}}(\mathcal{B}) + \mathbf{Adv}^{\text{v-bind}}_{\mathsf{CS}}(\mathcal{C})$$

Adversary $\mathcal{B}$ runs in the same amount of time as $\mathcal{A}$ with an $\mathcal{O}(q)$ overhead. Adversary $\mathcal{C}$ runs in the same amount of time as $\mathcal{A}$ does. The adversaries also make the same number of queries to their oracles as $\mathcal{A}$ does.

Proof: Games for this proof are in Figure 13. We start with game $\text{MO-CTXT}^{\mathcal{A}}_{\text{CtE1}}$ of Figure 13. This is a rewriting of $\text{MO-CTXT}^{\mathcal{A}}_{\mathsf{CE}}$ of Figure 4 with the encryption and decryption algorithms of CtE1. We then transition to game G_0 of Figure 13. In this game, decryption in **Dec** of previously-encrypted values is done by table lookup. If a ciphertext is submitted to **Dec** that successfully decrypts but was not present in the table, we set flag win to true. This strictly increases the probability of $\mathcal{A}$ winning the game. Then, we have that

$$\mathbf{Adv}^{\text{mo-ctxt}}_{\text{CtE1}[\mathsf{CS},\mathsf{SE}]}(\mathcal{A}) \leq \Pr\left[\, \text{G}^{\mathcal{A}}_0 \Rightarrow \mathsf{true} \,\right]$$

```
$\underline{\text{MO-CTXT}^{\mathcal{A}}_{\text{CtE1}}}$:
$K \leftarrow\!\!\$\ \mathsf{Kg}$ ; $\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalDec}}$
Return $\mathsf{win}$

$\underline{\mathbf{Enc}(H, M)}$
$(K_f, C_2) \leftarrow\!\!\$\ \mathsf{Com}(H \parallel M)$
$C_1 \leftarrow\!\!\$\ \mathsf{enc}_K(C_2, M \parallel K_f)$
$\mathcal{Y} \leftarrow \mathcal{Y} \cup \{(H, C_1, C_2)\}$
Return $(C_1, C_2)$

$\underline{\mathbf{Dec}(H, C_1, C_2)}$
$P \leftarrow \mathsf{dec}_K(C_2, C_1)$
If $P = \perp$ then Return $\perp$
$(M \parallel K_f) \leftarrow P$
$b \leftarrow \mathsf{VerC}(K_f, C_2, H \parallel M)$
If $b = 0$ then
    Return $\perp$
Return $(M, K_f)$

$\underline{\mathbf{ChalDec}(H, C_1, C_2)}$
If $(H, C_1, C_2) \in \mathcal{Y}$ then
    Return $\perp$
$P \leftarrow \mathsf{dec}_K(C_2, C_1)$
If $P = \perp$ then Return $\perp$
$(M \parallel K_f) \leftarrow P$
$b \leftarrow \mathsf{VerC}(K_f, C_2, H \parallel M)$
If $b = 0$ then
    Return $\perp$
If $M \neq \perp$ then
    $\mathsf{win} \leftarrow \mathsf{true}$
Return $(M, K_f)$
```

```
$\underline{G_0}$:
$K \leftarrow\!\!\$\ \mathsf{Kg}$ ; $\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalDec}}$
Return $\mathsf{win}$

$\underline{\mathbf{Enc}(H, M)}$
$(K_f, C_2) \leftarrow\!\!\$\ \mathsf{Com}(H \parallel M)$
$C_1 \leftarrow\!\!\$\ \mathsf{enc}_K(C_2, M \parallel K_f)$
$\mathcal{Y} \leftarrow \mathcal{Y} \cup \{(H, C_1, C_2)\}$
$D[H, C_1, C_2] \leftarrow (M, K_f)$
Return $(C_1, C_2)$

$\underline{\mathbf{Dec}(H, C_1, C_2)}$
If $D[H, C_1, C_2] \neq \perp$ then
    Return $D[H, C_1, C_2]$
$P \leftarrow \mathsf{dec}_K(C_2, C_1)$
If $P = \perp$ then Return $\perp$
$(M \parallel K_f) \leftarrow P$
$b \leftarrow \mathsf{VerC}(K_f, C_2, H \parallel M)$
If $b = 0$ then
    Return $\perp$
$\mathsf{win} \leftarrow \mathsf{true}$
Return $(M, K_f)$

$\underline{\mathbf{ChalDec}(H, C_1, C_2)}$
If $(H, C_1, C_2) \in \mathcal{Y}$ then
    Return $\perp$
$P \leftarrow \mathsf{dec}_K(C_2, C_1)$
If $P = \perp$ then Return $\perp$
$(M \parallel K_f) \leftarrow P$
$b \leftarrow \mathsf{VerC}(K_f, C_2, H \parallel M)$
If $b = 0$ then
    Return $\perp$
If $M \neq \perp$ then
    $\mathsf{win} \leftarrow \mathsf{true}$
Return $(M, K_f)$
```

Figure 13: Games for CtE1 integrity proof of Theorem 6.

Note that for win to be set in either decryption oracle with a query (H, C_1, C_2), it must be that no previous encryption query (H, M) for some M returned (C_1, C_2). Let the winning decryption oracle query be on the values (H^*, C_1^*, C_2^*). We partition the probability of setting win into two cases, either (C_1^*, C_2^*) is distinct from all encryption outputs, or it is not and H^* is not the header for the encryption query that returned C_1^*, C_2^*. Let win_H be the event that $\mathcal{A}$ wins with a query where H^* is a different header, and win_C be the event that $\mathcal{A}$ wins with a query where (C_1^*, C_2^*) is distinct. Then

$$\Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right] \leq \Pr\left[\, \mathsf{win}_H \,\right] + \Pr\left[\, \mathsf{win}_C \,\right] \;.$$

We'll first bound $\Pr[\, \mathsf{win}_C \,]$. In this case we will construct an adversary in the CTXT game of SE, $\mathcal{B}$. This adversary simulates G_0 for $\mathcal{A}$, as follows. When $\mathcal{A}$ queries (H, M) to **Enc**, $\mathcal{B}$ first generates a commitment and opening K_f, C_2. Then, $\mathcal{B}$ queries $\mathsf{enc}(C_2, M \parallel K_f)$. It stores the result in a table, then outputs C_1, C_2 to $\mathcal{A}$. It simulates **Dec** and **ChalDec** queries that are outputs of previous **Enc** queries by consulting its table and outputting either the proper value (for **Dec**) or

$\perp$ (for **ChalDec**). When $\mathcal{A}$ queries **Dec** or **ChalDec** with a value not in the table, $\mathcal{B}$ submits (C_2, C_1) as a forgery to its decryption oracle. Our $\mathcal{B}$ perfectly simulates G_0 for $\mathcal{A}$. Since $\mathcal{A}$'s query must be a successful forgery for win to be set in G_0, $\mathcal{B}$'s output is a successful forgery and $\mathcal{B}$ will break CTXT in this reduction with probability at least $\Pr[\,\mathsf{win}_C\,]$. Thus,

$$\Pr[\,\mathsf{win}_C\,] \leq \mathbf{Adv}^{\mathrm{ctxt}}_{\mathsf{SE}}(\mathcal{B}) \; .$$

To bound $\Pr[\,\mathsf{win}_H\,]$ and complete the proof we can build another reduction using a $\mathrm{vBIND}_{\mathsf{CS}}$ (as in Section 2) adversary $\mathcal{C}$. The adversary $\mathcal{C}$ simulates $\mathcal{A}$'s view of G_0 as $\mathcal{B}$ did, except $\mathcal{C}$ generates a random encryption key and computes enc and dec internally. When $\mathcal{A}$ makes a query (H, C_1, C_2) to **Dec** or **ChalDec** where H is not the header input to the encryption query that output C_1, C_2, $\mathcal{C}$ fetches from its stored values the message M and opening K_f corresponding to C_2, as well as H_0, the header part of the encryption query that produced C_1, C_2. In its game $\mathcal{C}$ outputs $((H, M, K_f), (H_0, M, K_f), C_2)$. The environment of G_0 is perfectly simulated by $\mathcal{C}$. Since in this case the winning query differs in the header, to win in G_0, $\mathcal{A}$'s winning query must cause Ver to output 1. In this case, $\mathcal{A}$ has broken binding of CS, and so will $\mathcal{C}$ in the reduction. Thus,

$$\Pr[\,\mathsf{win}_H\,] \leq \mathbf{Adv}^{\mathrm{v\text{-}bind}}_{\mathsf{CS}}(\mathcal{C})$$

and to conclude we can sum this and the upper-bound for $\mathcal{B}$ to get

$$\Pr\left[\,\mathrm{G}_0^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] \leq \mathbf{Adv}^{\mathrm{ctxt}}_{\mathsf{SE}}(\mathcal{B}) + \mathbf{Adv}^{\mathrm{v\text{-}bind}}_{\mathsf{CS}}(\mathcal{C}) \; .$$

∎

The receiver binding security of CtE1 is trivially implied by the security of the underlying commitment scheme, as captured by the next theorem.

Theorem 7 *[*CtE1 *receiver binding] Let* $\mathrm{CtE1} = \mathrm{CtE1}[\mathsf{CS}, \mathsf{SE}]$. *Let* $\mathcal{A}$ *be an* $\mathrm{r\text{-}BIND}_{\mathrm{CtE1}}$ *adversary. Then* $\mathbf{Adv}^{\mathrm{r\text{-}bind}}_{\mathrm{CtE1}}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{v\text{-}bind}}_{\mathsf{CS}}(\mathcal{A})$.

We conclude the section by noting CtE1 meets s-BIND security, since it runs Ver during decryption.

Facebook's scheme. The Facebook franking scheme (Section 3) is almost, but not quite, an instantiation of CtE1 using HMAC as the commitment scheme CS. One difference is that their franking scheme does not bind C_2 to C_1 by including C_2 in the associated data during encryption. The other difference is that the Facebook scheme builds a commitment from HMAC by first generating a random secret key, then using it to evaluate HMAC on the concatenation of the message and the key itself (see Figure 3 for a diagram). Assuming HMAC remains a collision-resistant PRF when evaluated on its own key, we can prove Facebook's non-standard construction is a secure commitment (see Theorem 1). To analyze Facebook's scheme, then, we introduce the scheme $\mathrm{CtE2}[\mathsf{SE}, \mathsf{CS}]$ that works as shown in Figure 12. Note that Facebook does not discuss how to handle associated data, and so their scheme is CtE2 using CS instantiated with HMAC and requiring $H = \varepsilon$.

There are two benefits to the approach of CtE1: (1) proving ciphertext integrity does not require any special properties of the commitment scheme, and (2) it is more efficient because associated data is cryptographically processed once, rather than twice. We therefore advocate CtE1, but analyze CtE2 here since it is already in use.

CtE2 is *not* secure assuming just that CS is hiding and binding. The reason is that such commitments can be malleable and this allows easy violation of ciphertext integrity. Specifically, consider a commitment scheme $\mathsf{CSBad} = (\mathsf{ComBad}, \mathsf{VerBad})$ built using a standard commitment scheme $\mathsf{CS} = (\mathsf{Com}, \mathsf{VerC})$. Algorithm $\mathsf{ComBad}(M)$ runs $(K_c, C) \leftarrow\!\!\$\; \mathsf{Com}(M)$ and then outputs $(K_c, C \,\|\, 1)$. Algorithm $\mathsf{VerBad}(M, K_f, C \,\|\, b)$ runs $\mathsf{VerC}(M, K_f, C)$ and outputs the result. An easy reduction shows that CSBad is both hiding and binding, assuming CS is too. But it's clear that CtE2[$\mathsf{SE}, \mathsf{CSBad}$] does *not* enjoy ciphertext integrity. The adversary simply obtains one ciphertext, flips the last bit, and submits to the challenge decryption oracle to win.

This shows that standard commitments with hiding and binding properties are insufficient to instantiate CtE2. But if a scheme CS has unique commitments, then we can in fact show security of CtE2. A scheme has *unique commitments* if for any pair $(K_c, M) \in \mathcal{K}_f \times \mathcal{M}$ it holds that there is a single commitment value $C \in \mathcal{C}$ for which $\mathsf{Ver}(K_c, C, M) = 1$. All hash-based CS schemes, including the HMAC one used by Facebook's franking scheme, have unique commitments. If one wanted to use a scheme that does not have unique commitments, then one would need the commitment to satisfy a form of non-malleability [30]. The following sequence of theorems captures the security of CtE2 assuming a unique commitment scheme.

Theorem 8 *[*CtE2 *confidentiality] Let* CtE2 $=$ CtE2[CS, SE]. *Let $\mathcal{A}$ be an* MO-ROR$_{\text{CtE2}}$ *adversary, making at most q queries to its oracles, and assume* CS *has unique commitments. Then we give adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{C}$ such that*

$$\mathbf{Adv}^{\text{mo-ror}}_{\text{CtE2}[\mathsf{CS},\mathsf{SE}]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ror}}_{\mathsf{SE}}(\mathcal{B}_1) + \mathbf{Adv}^{\text{ror}}_{\mathsf{SE}}(\mathcal{B}_2) + \mathbf{Adv}^{\text{cs-ror}}_{\mathsf{CS}}(\mathcal{C})$$

The adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{C}$ all run in the same amount of time as $\mathcal{A}$, with an $\mathcal{O}(q)$ overhead, and make the same number of queries as $\mathcal{A}$.

This proof uses a sequence of game transitions similar to the corresponding real-or-random proof for CtE1, so we will omit it.

Theorem 9 *[*CtE2 *integrity] Let* CtE2 $=$ CtE2[CS, SE]. *Let $\mathcal{A}$ be an* MO-CTXT$_{\text{CtE2}}$ *adversary and assume* CS *has unique commitments. Then we give adversaries $\mathcal{B}$, $\mathcal{C}$ such that*

$$\mathbf{Adv}^{\text{mo-ctxt}}_{\text{CtE2}[\mathsf{CS},\mathsf{SE}]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ctxt}}_{\mathsf{SE}}(\mathcal{B}) + \mathbf{Adv}^{\text{v-bind}}_{\mathsf{CS}}(\mathcal{C}) \ .$$

Adversaries $\mathcal{B}$ and $\mathcal{C}$ both run in the same amount of time as $\mathcal{A}$, with an $\mathcal{O}(q)$ overhead. The adversaries also make the same number of queries to their oracles as $\mathcal{A}$ does.

Proof: We begin here, as in the integrity proof for CtE1, by transitioning to a game in which decryption in **Dec** of previously-encrypted values is done by table lookup. If a ciphertext is submitted to **Dec** that successfully decrypts but was not present in the table, we set flag win to true. We can set win to true in two places in our game. Here our proof diverges from the one above for MO-CTXT$_{\text{CtE1}}$, for the following reason. Take some output of **Enc** and its header, (H, C_1, C_2). Let C_2' be some other valid commitment returned by a different call to **Enc**. Because the commitments are *not* included in the associated data of encryption, it does not violate CTXT of SE to query (H, C_1, C_2') to decryption! Our reduction $\mathcal{B}$ from the previous proof no longer applies. Here is where we must use the fact that CS has unique commitments. Because fixing H and C_1 fixes K_f, there exists only one C_2 value for which $\mathsf{Ver}(K_f, C_2, H \parallel M)$ will output 1 during decryption. Thus, the query (H, C_1, C_2') cannot set win to true. Our reduction to CTXT, $\mathcal{B}$, therefore behaves as above except it only submits a decryption query to its forgery oracle if C_1 differs from the one it output to $\mathcal{A}$ in **Enc**. The reduction $\mathcal{C}$ to vBIND$_{\mathsf{CS}}$ security behaves identically, and the result follows. ∎

Theorem 10 *[*CtE2 *binding] Let* CtE2 $=$ CtE2[CS, SE]. *Let $\mathcal{A}$ be an* r-BIND$_{\text{CtE2}}$ *adversary. Then we give an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{r-bind}}_{\text{CtE1}[\mathsf{CS},\mathsf{SE}]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{v-bind}}_{\mathsf{CS}}(\mathcal{B}) \ .$$

The adversary $\mathcal{B}$ runs in the same amount of time as $\mathcal{A}$ and makes the same number of queries to its oracles as $\mathcal{A}$.

The proof of this theorem is identical to the proof of Theorem 7. Finally, note that CtE2 achieves s-BIND security because it verifies the commitment during decryption.

7 Nonce-based Committing AEAD and the CEP Construction

The committing AEAD schemes we've examined thus far have all been randomized. Cryptographers have advocated that modern AEAD schemes should be designed as nonce-based instead. Here one replaces internal randomness during encryption with an input, called the nonce. Security should hold as long as the nonce never repeats throughout the course of encrypting messages with a particular key.

We formalize nonce-based committing AEAD and provide a construction of it that additionally achieves a number of attractive properties. It achieves a multiple-opening security notion suitably modified to the nonce-based setting. It is faster than the other multiple-opening schemes, requiring only two cryptographic passes during encryption and decryption, and a single one during verification. It also reduces ciphertext stretch compared to the schemes of Section 6, since the opening will be recomputed in the course of decryption and so does not need to be sent in the encryption.

Nonce-based committing AEAD. A nonce-based CE scheme is a tuple of algorithms $\mathrm{nCE} = (\mathrm{Kg}, \mathrm{Enc}, \mathrm{Dec}, \mathrm{Ver})$. We define it exactly like CE schemes (Section 4) except for the following differences. In addition to the other sets, we associate to any nCE scheme a nonce space $\mathcal{N} \subseteq \Sigma^*$. Encryption and decryption are now defined as follows:

- *Encryption*: Encryption Enc is deterministic and takes as input a tuple $(K, N, H, M) \in (\Sigma^*)^4$ and outputs a pair $(C_1, C_2) \in \mathcal{C} \times \mathcal{T}$ or a distinguished error symbol $\perp$. We require that for any $(K, N, H, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{M}$ it is the case that $\mathrm{Enc}(K, N, H, M) \neq \perp$.
- *Decryption*: Decryption Dec is deterministic. It takes as input a quintuple $(K, N, H, C_1, C_2) \in (\Sigma^*)^5$ and outputs a message, opening value pair $(M, K_f) \in \mathcal{M} \times \mathcal{K}_f$ or $\perp$.

Key generation and verification are unchanged relative to randomized CE schemes. As for randomized schemes, we assume that the length of ciphertexts are dictated only by the lengths of the header and message. We will often write $\mathrm{Enc}_K^N(H, M)$ for $\mathrm{Enc}(K, N, H, M)$ and $\mathrm{Dec}_K^N(H, C_1, C_2)$ for $\mathrm{Dec}(K, N, H, C_1, C_2)$.

Nonce-based security. We adapt the confidentiality and integrity security notions from Section 4 to the nonce-based setting. For a scheme nCE, we measure the nonce-based multiple-openings real-or-random $\text{MO-nROR}_{\text{nCE}}$ advantage of an adversary $\mathcal{A}$ (using the games $\text{MO-nREAL}_{\text{nCE}}$ and $\text{MO-nRAND}_{\text{nCE}}$ in Figure 14) as

$$\mathbf{Adv}^{\text{mo-nror}}_{\text{nCE}}(\mathcal{A}) = \left|\Pr\left[\,\text{MO-nREAL}^{\mathcal{A}}_{\text{nCE}} \Rightarrow 1\,\right] - \Pr\left[\,\text{MO-nRAND}^{\mathcal{A}}_{\text{nCE}} \Rightarrow 1\,\right]\right| \,.$$

An adversary is *nonce-respecting* if its queries never repeat the same N across a pair of encryption queries (two queries to **Enc**, two to **ChalEnc**, or one to each). We will assume nonce-respecting $\text{MO-nRAND}_{\text{nCE}}$ adversaries.

For a scheme nCE, we measure the nonce-based multiple-openings real-or-random $\text{MO-nCTXT}_{\text{nCE}}$ advantage of an adversary $\mathcal{A}$ (using the game $\text{MO-nCTXT}_{\text{nCE}}$ in Figure 14) as

$$\mathbf{Adv}^{\text{mo-nctxt}}_{\text{nCE}}(\mathcal{A}) = \Pr\left[\,\text{MO-nCTXT}^{\mathcal{A}}_{\text{nCE}} \Rightarrow 1\,\right] \,.$$

As with randomized committing AEAD, we can provide single-opening versions of the above definitions, and can give an all-in-one version of nonce-based MO and SO security. We omit the

```
$\text{MO-nREAL}^{\mathcal{A}}_{\text{nCE}}$:
$K \leftarrow\!\!\$\ \mathsf{Kg}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalEnc}}$
Return $b'$

$\mathbf{Enc}(N, H, M)$
$(C_1, C_2) \leftarrow \text{Enc}^N_K(H, M)$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{(N, H, C_1, C_2)\}$
Return $C$

$\mathbf{Dec}(N, H, C_1, C_2)$
If $(N, H, C_1, C_2) \notin \mathcal{Y}_1$ then
    Return $\perp$
$(M, K_f) \leftarrow \text{Dec}^N_K(H, C_1, C_2)$
Return $(M, K_f)$

$\mathbf{ChalEnc}(N, H, M)$
$(C_1, C_2) \leftarrow \text{Enc}^N_K(H, M)$
$\mathcal{Y}_2 \leftarrow \mathcal{Y}_2 \cup \{(N, H, C_1, C_2)\}$
Return $C$
```

```
$\text{MO-nRAND}^{\mathcal{A}}_{\text{nCE}}$:
$K \leftarrow\!\!\$\ \mathsf{Kg}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalEnc}}$
Return $b'$

$\mathbf{Enc}(N, H, M)$
$(C_1, C_2) \leftarrow \text{Enc}^N_K(H, M)$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{(N, H, C_1, C_2)\}$
Return $C$

$\mathbf{Dec}(N, H, C_1, C_2)$
If $(N, H, C_1, C_2) \notin \mathcal{Y}_1$ then
    Return $\perp$
$(M, K_f) \leftarrow \text{Dec}^N_K(H, C_1, C_2)$
Return $(M, K_f)$

$\mathbf{ChalEnc}(N, H, M)$
$(\ell_1, \ell_2) \leftarrow \mathsf{clen}(|H|, |M|)$
$(C_1, C_2) \leftarrow\!\!\$\ \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$
$\mathcal{Y} \leftarrow \mathcal{Y} \cup \{(N, H, C_1, C_2)\}$
Return $(C_1, C_2)$
```

```
$\text{MO-nCTXT}^{\mathcal{A}}_{\text{nCE}}$:
$K \leftarrow\!\!\$\ \mathsf{Kg}$ ; $\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalDec}}$
Return $\mathsf{win}$

$\mathbf{Enc}(N, H, M)$
$(C_1, C_2) \leftarrow \text{Enc}^N_K(H, M)$
$\mathcal{Y} \leftarrow \mathcal{Y} \cup \{(N, H, C_1, C_2)\}$
Return $C$

$\mathbf{Dec}(N, H, C_1, C_2)$
Return $\text{Dec}^N_K(H, C_1, C_2)$

$\mathbf{ChalDec}(N, H, C_1, C_2)$
If $(N, H, C_1, C_2) \in \mathcal{Y}$ then
    Return $\perp$
$(M, K_f) \leftarrow \text{Dec}^N_K(H, C_1, C_2)$
If $M \neq \perp$ then
    $\mathsf{win} \leftarrow \mathsf{true}$
Return $(M, K_f)$
```

Figure 14: Confidentiality (left two games) and ciphertext integrity (rightmost) games for nonce-based committing AEAD.

details for the sake of brevity.

The sender binding notion s-BIND for nonce-based schemes is the same as for randomized schemes except that the adversary also outputs a nonce N, which is used with Dec. Because verification is unchanged, receiver binding security is formalized exactly the same for randomized and nonce-based committing AEAD.

The Committing Encrypt-and-PRF scheme. One can analyze some traditional nonce-based AEAD schemes to show they are compactly committing. As one example, it is easy to see that the EtE construction (Section 5.1) works just as well with non-repeating nonces, but with only single-opening security. The other schemes in Section 5 do not, but can be easily modified to by replacing IV with $E_K(N)$. Here we focus on a new scheme that will have better overall performance and security. Unlike the legacy schemes studied in Section 5 it will be provably secure for multiple openings. At the same time, it will be more efficient than the schemes in Section 6.

Let G be a nonce-based PRG as defined in Section 2. Let $F, F^{\text{cr}}\colon \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^t$ be a PRF and a collision-resistant PRF, respectively, both as defined in Section 2. The scheme $\text{CEP}[G, F, F^{\text{cr}}] = (\text{CEP-Kg}, \text{CEP-Enc}, \text{CEP-Dec}, \text{CEP-Ver})$ is in the style of an Encrypt-and-PRF construction. The key space for CEP is $\mathcal{K} = \{0,1\}^k$ and key generation simply outputs a random draw from it. Encryption starts by using the nonce with the key K to derive a pad P from the nonce-based PRG G. Part of this pad will be used to encrypt the message. The initial $2n$ bits are used as two one-time keys for F^{cr} and F. Finally it computes a binding value for H, M and applies F to that commitment value to generate a tag. Detailed pseudocode is given in Figure 15.

We will need F^{cr} to both be CR (for binding) as well as secure as a one-time PRF (for confidentiality). This rules out some otherwise desirable choices such as CMAC [45], PMAC [57] and Carter-Wegman-style PRFs such as Poly1305 [16] and UMAC [17]. These PRFs are some of the fastest available, but would make CEP vulnerable to binding attacks. (See also the discussion in Section 5.4.)

The most obvious choice is HMAC, for which formal analyses support it being a secure PRF for

```
CEP-Enc$^N_K(H, M)$:
$m \leftarrow \lceil |M|/n \rceil$
$P \leftarrow G(K, N, m + 2n)$
$C_1 \leftarrow (P_2 \parallel \cdots \parallel P_{m+1}) \oplus M$
$C_2 \leftarrow F^{\mathrm{cr}}_{P_0}(H \parallel M)$
$T \leftarrow F_{P_1}(C_2)$
Return $(C_1 \parallel T, C_2)$
```

```
CEP-Dec$^N_K(H, C_1 \parallel T, C_2)$:
$m \leftarrow \lceil |C_1|/n \rceil$
$P \leftarrow G(K, N, m + 2n)$
$M \leftarrow (P_2 \parallel \cdots \parallel P_{m+1}) \oplus C_1$
$C_2' \leftarrow F^{\mathrm{cr}}_{P_0}(H \parallel M)$
$T' \leftarrow F_{P_1}(C_2')$
If $T \neq T' \vee C_2' \neq C_2$ then
    Return $\perp$
Return $(M, P_0)$
```

```
CEP-Ver$(H, M, K_f, C_2)$:
$C_2' \leftarrow F^{\mathrm{cr}}_{K_f}(H \parallel M)$
If $C_2' \neq C_2$ then Return 0
Return 1
```

Figure 15: The nonce-based committing AEAD scheme CEP$[G, F, F^{\mathrm{cr}}]$. For the pad P generated by G, the notation P_i refers to the ith n-bit block of P, i.e. $P[in, \ldots, (i+1)n - 1]$.

a key secret [2,3] and CR for adversarially chosen keys of the same length (assuming the underlying hash function is CR). Other multi-property hash functions [10] would also suffice.

The reason we use G both for CTR mode and for key derivation is speed. This ensures that we need ever only use a single key with G; in some environments rekeying can be very expensive. Any nonce-based pseudorandom generator can be used to instantiate G, e.g., ChaCha-20 [15].

One might wonder why have a tag T as well as the commitment value C_2. The reason is that to achieve multi-opening security, we must disclose the key used with F^{cr}, rendering the unforgeability of C_2 values moot. If one instead omitted T and only checked $C_2' = C_2$ to attempt to achieve unforgeability, then there exists a straightforward MO-nCTXT attack that obtains a ciphertext for a nonce N, queries it to **Dec** to get the key for F^{cr}, and then uses that to forge a new ciphertext to be submitted to **ChalDec**. The application of F under a distinct key provides ciphertext integrity even after an adversary obtains openings (keys for F^{cr}). Similarly, dropping the check during decryption that $C_2' = C_2$ also leads to an attack, but this time on sender binding.

Comparisons. Before getting into the formal security analysis in the next section, we first compare CEP to the generic composition constructions that also achieve multiple-opening security. The first benefit over other schemes is that it is nonce-based, making it suitable for stateful as well as randomized settings (see also Rogaway's discussion of the benefits of nonce-based encryption [58]).

The second is that ciphertext expansion is reduced by a security parameter number of bits compared to the generic composition constructions, because in CEP we do not need to transport an explicit opening — the recipient recomputes it pseudorandomly from the secret key. Consequently, CEP ciphertexts are shorter than Facebook's by 256 bits.

The third is that encryption and decryption both save an entire cryptographic pass over the associated data and message. For Facebook's chosen algorithms (HMAC for the commitment, plus Encrypt-then-MAC using AES-CBC and HMAC), this means that CEP offers more than a 50% speed-up for both algorithms.[7] While in some messaging settings encryption and decryption may not be particularly performance-sensitive operations, any cost savings is desirable. In other contexts, such as if one starts using committing encryption on larger files (images, videos) sent over messaging applications or if one wants abuse reporting for streaming communications, performance will be very important.

CEP achieves the stronger multiple-opening security goal, setting it apart from the legacy committing AEAD schemes from Section 5. At the same time, CEP has equivalent or better performance than those schemes. With respect to EtM and MtE, verification is reduced from two cryptographic passes to one.

[7]HMAC, with suitable choice of hash function, is slower than AES. If AES-NI is available, then the speed-up will be even larger, since the HMAC passes will be the performance bottleneck.

8 Analysis of CEP

In this section we will analyze the nonce-based committing AEAD security of the CEP construction. Before stating and proving the main results, we first recast the $F \circ F^{\mathrm{cr}}$ function as a MAC scheme; this will make our proofs more modular.

The MAC scheme $\mathsf{Mac}[F \circ F^{\mathrm{cr}}] = (\mathsf{kg}, \mathsf{tag}, \mathsf{ver})$ is defined as follows. Key generation outputs a single random n-bit string. Tag generation algorithm is $\mathsf{tag}(P_1, (P_0, X)) = F_{P_1}(F^{\mathrm{cr}}_{P_0}(X))$ for random key P_1 and any pair of strings (P_0, X). In CEP the X will be the concatenation of the associated data H and message M. The output of tag is a t-bit string. The verification procedure $\mathsf{ver}(P_1, (P_0, X), T)$ re-runs $\mathsf{tag}(P_1, (P_0, X))$ and returns true if its output is equal to T, and false otherwise.

Note that we have taken P_0 to be a message. That's because we require that $F \circ F^{\mathrm{cr}}$ realizes a good MAC even when P_0 is known to the adversary. Since the value P_1 changes for every encryption in CEP, it is easier to reduce to a multi-user unforgeability under chosen message attack (see Section 2). We therefore state a result about the MU-UF-CMA security of our MAC $F \circ F^{\mathrm{cr}}$. The proof uses standard techniques, but we include it for completeness.

Lemma 2 *Let $\mathcal{A}$ be an* MU-UF-CMA$_{F\circ F^{\mathrm{cr}}}$ *adversary making at most q queries. Then we give in the proof explicit adversaries $\mathcal{B}, \mathcal{C}$ such that*

$$\mathbf{Adv}^{\text{mu-uf-cma}}_{F\circ F^{\mathrm{cr}}}(\mathcal{A}) \le \mathbf{Adv}^{\text{mu-prf}}_{F}(\mathcal{B}) + \mathbf{Adv}^{\mathrm{cr}}_{F^{\mathrm{cr}}}(\mathcal{C}) + \frac{q}{2^t} \, .$$

Adversary $\mathcal{B}$ makes at most q queries and runs in time that of $\mathcal{A}$ plus an $\mathcal{O}(q)$ overhead. Adversary $\mathcal{C}$ runs in time that of $\mathcal{A}$ plus an $\mathcal{O}(q)$ overhead.

Proof: Begin with the game G_0, which is the same as MU-UF-CMA$^{\mathcal{A}}_{F\circ F^{\mathrm{cr}}}$ in Figure 2. Define a new game G_1 which is the same as G_0 except with every call to $F_{\mathsf{K}[S]}$ replaced by a call to a random function R which is drawn uniformly at random for each new key identifier queried. We can upper-bound the difference in $\mathcal{A}$'s advantage in these two games using a reduction $\mathcal{B}$ to the MU-PRF security of F. The reduction $\mathcal{B}$ has access to an oracle $\mathbf{O}(\cdot,\cdot)$ which takes a key identifier and a message. It uses $\mathbf{O}$ to simulate $\mathcal{A}$'s **Tag** oracle on input $(S, (P, M))$ by returning the output of $\mathbf{O}(S, F^{\mathrm{cr}}_P(M))$. It records the message M in table $\mathcal{M}[S]$ as well. It simulates $\mathcal{B}$'s **Ver** oracle on inputs $(S, (P, M), T)$ by returning $(M \notin \mathcal{M}[S] \wedge (\mathbf{O}(S, F^{\mathrm{cr}}_P(M)) = T))$. Then we have that

$$\Pr[\, G_0 \Rightarrow \mathsf{true}\,] \le \Pr[\, G_1 \Rightarrow \mathsf{true}\,] + \mathbf{Adv}^{\text{mu-prf}}_{F}(\mathcal{B})$$

Next, define a game G_2 which is identical to G_1 except in **Tag** and **Ver**, $F^{\mathrm{cr}}_P(M)$ is not run at all—tags are generated by applying the random function R for key identifier S to the concatenation of P and message M directly. Since R's output (and, therefore, each tag generated) in both G_2 and G_1 is a uniformly random string, the only way the tag outputs could change in G_2 is if $\mathcal{A}$ finds a collision in F^{cr}. We can upper-bound the difference in $\mathcal{A}$'s success probability in G_1 and G_2 using a reduction $\mathcal{C}$ to the collision-resistance of F^{cr}. The reduction $\mathcal{C}$ runs the code of game G_1 exactly, except it also records the values of $F^{\mathrm{cr}}_P(M)$ it sees throughout queries to **Tag** and **Ver**. If it sees a collision in F^{cr} at any point, it outputs it. We have that

$$\Pr[\, G_1 \Rightarrow \mathsf{true}\,] \le \Pr[\, G_2 \Rightarrow \mathsf{true}\,] + \mathbf{Adv}^{\mathrm{cr}}_{F^{\mathrm{cr}}}(\mathcal{C})$$

Let $(P, M), (P', M')$ be a collision in F^{cr}. In G_2, the query $\mathbf{Ver}(S, P', M', \mathbf{Tag}(S, P, M))$ will only set win to true with probability $\frac{1}{2^t}$ since the outputs of the random function, $R(P' \,\|\, M')$ and $R(P \,\|\, M)$, are equal with that probability. With this, we can complete the proof by noting that $\Pr[\, G_2 \Rightarrow \mathsf{true}\,] \le \frac{q}{2^t}$, since the only way to set win to true in G_2 is to correctly guess the output of a random function on a previously unqueried point. In q queries this happens with probability at most $\frac{q}{2^t}$ by a union bound. Summing and rewriting yields the bound. ∎

Security of CEP. We are now in position to formally analyze the confidentiality, ciphertext integrity, and binding of CEP. We give theorems and proofs for each in turn.

Theorem 11 *[CEP confidentiality] Let* CEP = CEP$[G, F, F^{\mathrm{cr}}]$. *Let $\mathcal{A}$ be an* MO-nROR$_{\mathrm{CEP}}$ *adversary making q queries and whose queried messages total at most σ bits. Let the length of the keys for F and F^{cr} be n bits. Then we give explicit adversaries $\mathcal{B}, \mathcal{C}, \mathcal{D}$ below such that*

$$\mathbf{Adv}^{\text{mo-nror}}_{\mathrm{CEP}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\mathrm{prg}}_{G}(\mathcal{B}) + 2 \cdot \mathbf{Adv}^{\text{mu-prf}}_{F}(\mathcal{C}) + \mathbf{Adv}^{\text{mu-prf}}_{F^{\mathrm{cr}}}(\mathcal{D})$$

Adversary $\mathcal{B}$ is nonce-respecting, makes at most q queries to its oracle, and the sum of its total outputs requested is $\sigma + 2qn$ bits. Adversary $\mathcal{C}$ makes at most q queries to its oracle, and never repeats a key identifier. Adversary $\mathcal{D}$ make at most q queries to its oracle and never repeats a key identifier. All adversaries run in the same amount of time as $\mathcal{A}$ with an $\mathcal{O}(q)$ overhead.

Proof: The proof uses a sequence of game transitions. We will start with the first game, G_0, which is the same as MO-nREAL$_{\mathrm{CEP}[G,F,F^{\mathrm{cr}}]}$. Game G_1 is the same as G_0 except that G is replaced by a routine $R(\cdot,\cdot)$ that outputs random bit strings, in a prefix-preserving way, for any query. We construct a PRG adversary $\mathcal{B}_0$ against G that uses its oracle to simulate $\mathcal{A}$'s oracles in game G_0. On query $\mathbf{Enc}(N, H, M)$, it queries its oracle with inputs N and $\lceil |M|/n \rceil + 2n$ and uses the output to run the rest of the CEP-Enc procedure. It simulates $\mathcal{A}$'s other oracles similarly. We have that

$$\Pr[\,\mathrm{G}_0 \Rightarrow 1\,] \leq \Pr[\,\mathrm{G}_1 \Rightarrow 1\,] + \mathbf{Adv}^{\mathrm{prg}}_{G}(\mathcal{B}_0)\;.$$

Game G_2 is the same as G_1 except that we replace F by random functions, one per nonce used. We construct a reduction $\mathcal{C}_0$ to MU-PRF security of F by replacing calls to it in $\mathbf{Enc}(N, H, M)$ with calls to $\mathcal{C}_0$'s oracle $\mathcal{R}$ with inputs $(N, F^{\mathrm{cr}}_{P_0}(H \,\|\, M))$. We replace calls to F in $\mathcal{A}$'s other oracles similarly. Standard arguments give us that

$$\Pr[\,\mathrm{G}_1 \Rightarrow 1\,] \leq \Pr[\,\mathrm{G}_2 \Rightarrow 1\,] + \mathbf{Adv}^{\text{mu-prf}}_{F}(\mathcal{C}_0)\;.$$

The next game G_3 replaces C_2 values returned by **ChalEnc** by random values. Here we must be careful, as the C_2 values returned by **Enc** are *not* pseudorandom — we may release the associated P_1 values later due to a decryption query. Therefore we reduce to the MU-PRF security of F^{cr}, but where the reduction itself generates keys for **Enc** queries and only uses its own oracle for **ChalEnc** queries' use of F^{cr}. This step relies on the fact that $\mathcal{A}$ is nonce-respecting, lest **Enc** and **ChalEnc** could end up using the same F^{cr} keys.

Let $\mathcal{D}$ be the MU-PRF adversary against F^{cr} that starts by running $\mathcal{A}$, and simulates its oracles as follows. For **Enc** and **Dec** it simply runs the code of game G_2's procedures. For **ChalEnc**, it works as in G_2 except that the F^{cr} execution is replaced by a query to $\mathcal{D}$'s oracle on $(N, H \,\|\, M)$. This gives that

$$\Pr[\,\mathrm{G}_2 \Rightarrow 1\,] \leq \Pr[\,\mathrm{G}_3 \Rightarrow 1\,] + \mathbf{Adv}^{\text{mu-prf}}_{F^{\mathrm{cr}}}(\mathcal{D})\;.$$

Game G_3 returns a random string of bits in response to any **ChalEnc** oracle query, because $\mathcal{A}$ is nonce-respecting and so the random pad P is always a uniform fresh choice, the families of random functions that replaced F and F^{cr} are only ever used on a single point. Game G_3 is not equivalent yet to MO-nRAND$_{\mathrm{nCE}}$ since the **Enc** and **Dec** oracles are using random pads and random functions, as opposed to G and F. But symmetric arguments to the ones used above give specific adversaries $\mathcal{C}_1$ and then $\mathcal{B}_1$ such that

$$\Pr[\,\mathrm{G}_3 \Rightarrow 1\,] \leq \Pr\left[\,\text{MO-nRAND}^{\mathcal{A}}_{\mathrm{nCE}} \Rightarrow 1\,\right] + \mathbf{Adv}^{\text{mu-prf}}_{F}(\mathcal{C}_1) + \mathbf{Adv}^{\mathrm{prg}}_{F}(\mathcal{B}_1)\;.$$

Finally a standard argument gives adversaries $\mathcal{B}, \mathcal{C}$ such that

$$\begin{aligned}\mathbf{Adv}_F^{\text{mu-prf}}(\mathcal{C}_0) + \mathbf{Adv}_F^{\text{mu-prf}}(\mathcal{C}_1) &\leq 2 \cdot \mathbf{Adv}_F^{\text{mu-prf}}(\mathcal{C}) \quad \text{and} \\ \mathbf{Adv}_G^{\text{prg}}(\mathcal{B}_0) + \mathbf{Adv}_G^{\text{prg}}(\mathcal{B}_1) &\leq 2 \cdot \mathbf{Adv}_B^{\text{prg}}(\mathcal{B}) \ .\end{aligned}$$

∎

The next theorem captures the ciphertext integrity of the scheme.

Theorem 12 *[*CEP *ciphertext integrity] Let* $\text{CEP} = \text{CEP}[G, F, F^{\text{cr}}]$. *Let* $\mathcal{A}$ *be an* MO-nCTXT$_{\text{CEP}}$ *adversary making at most* q *queries with query inputs totalling at most* σ *bits. Let* $F \circ F^{\text{cr}}$ *be the MAC described above, with keys of length* $2n$ *bits. Then we give adversaries* $\mathcal{B}, \mathcal{C}$ *such that*

$$\mathbf{Adv}_{\text{CEP}}^{\text{mo-nctxt}}(\mathcal{A}) \leq \mathbf{Adv}_G^{\text{prg}}(\mathcal{B}) + \mathbf{Adv}_{F \circ F^{\text{cr}}}^{\text{mu-uf-cma}}(\mathcal{C}) \ .$$

Adversary $\mathcal{B}$ *runs in time that of* $\mathcal{A}$ *plus at most* $\mathcal{O}(q)$ *overhead and makes* q *queries totaling at most* $\sigma + 2nq$ *bits. Adversary* $\mathcal{C}$ *makes at most* q *queries and runs in time that of* $\mathcal{A}$ *plus at most* $\mathcal{O}(q)$ *overhead.*

Proof: Let $\text{G}_0 = \text{MO-nCTXT}_{\text{CEP}}^{\mathcal{A}}$. By a standard reduction to the security of G we can transition to a game G_1 in which G is replaced by a routine $R(\cdot,\cdot)$ returning prefix-preserving random bit strings. We have that

$$\Pr\left[\,\text{G}_0^{\mathcal{A}} \Rightarrow 1\,\right] \leq \Pr\left[\,\text{G}_1^{\mathcal{A}} \Rightarrow 1\,\right] + \mathbf{Adv}_G^{\text{prg}}(\mathcal{B}) \ .$$

We modify game G_1 to obtain game G_2, shown in Figure 16. The differences are that: (1) queries to **Dec** on tuples $(N, H, C_1 \,\|\, T, C_2)$ for which there was a previous query to $\mathbf{Enc}(N, H, M)$ that returned $C_1 \,\|\, T, C_2$ simply reply with (M, P_0) without bothering to do decryption, and (2) we set win to true if any other query to **Dec** successfully decrypts. The first difference is without loss, since the **Dec** in G_1 would have anyway returned (M, P_0). The second difference only increases the adversary's probability of success. Thus

$$\Pr\left[\,\text{G}_1^{\mathcal{A}} \Rightarrow 1\,\right] \leq \Pr\left[\,\text{G}_2^{\mathcal{A}} \Rightarrow 1\,\right] \ .$$

We now bound $\mathcal{A}$'s probability of success in G_2 by its ability to forge against $F \circ F^{\text{cr}}(P_1, P_0, H \,\|\, M)$. Notice that in G_2 the adversary only ever uses P_1 values for F and that, by our earlier transition, these are uniformly random, independent bit strings for each N. The MU-UF-CMA$_{F \circ F^{\text{cr}}}$ (as in Figure 2) adversary $\mathcal{C}$, shown in Figure 16, simulates the environment of $\mathcal{A}$ and uses its **Tag** and **Ver** oracles to perform tagging and verification using $F \circ F^{\text{cr}}$. (Recall that verification works by simply re-executing $F \circ F^{\text{cr}}$ and checking that the computed and submitted tags are equal.) We will argue that

$$\Pr\left[\,\text{G}_2^{\mathcal{A}} \Rightarrow 1\,\right] = \mathbf{Adv}_{F \circ F^{\text{cr}}}^{\text{mu-uf-cma}}(\mathcal{C}) \ .$$

To do so we need to show that anytime win would have been set in G_2, the corresponding query to $\mathbf{Ver}(N^*, (P_0^*, H^* \,\|\, M^*), T^*)$ is a successful forgery (where we have added asterisks to distinguish this winning query from other queries). To be a successful forgery the MU-UF-CMA game must return true, which is clear by inspection, and it must be that there was not a previous query $\mathbf{Tag}(N^*, (P_0^*, H^* \,\|\, M^*))$ that returned T^*. Suppose otherwise, and let $N, (P_0, H \,\|\, M)$ be that query. Let the return from the corresponding **Enc** query on inputs (N, H, M) be the pair $(C_1 \,\|\, T, C_2)$. Then by assumption $H = H^*$, and $M = M^*$. We also have that $N = N^*$, $P_0 = P_0^*$, and $P = P^*$. In turn, since F is deterministic $T = T^*$ and $C_2 = C_2^*$. Finally, $C = M \oplus P = M^* \oplus P^* = C_1^*$. Thus, $(N, H, C_1 \,\|\, T, C_2) = (N^*, H^*, C_1^* \,\|\, T^*, C_2^*)$, implying that either $\mathsf{V}[N^*, H^*, C_1^* \,\|\, T^*, C_2^*] \neq \perp$ (for **Dec**) or $(N^*, H^*, C_1^* \,\|\, T^*, C_2^*) \in \mathcal{Y}$ (for **ChalDec**), which is a contradiction since it means that **Ver** could not have been called. ∎

```
$\underline{\mathrm{G}_2:}$

$\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalDec}}$
Return $\mathsf{win}$

$\underline{\mathbf{Enc}(N, H, M):}$

$(P_0, P_1, P) \leftarrow\!\!\$\ R(N, 2n + |M|)$
$C_1 \leftarrow M \oplus P$
$C_2 \leftarrow F^{\mathrm{cr}}_{P_0}(H \parallel M)$
$T \leftarrow F_{P_1}(C_2)$
$\mathsf{V}[N, H, C_1 \parallel T, C_2] \leftarrow (M, P_0)$
Return $(C_1 \parallel T, C_2)$

$\underline{\mathbf{Dec}(N, H, C_1 \parallel T, C_2):}$

If $\mathsf{V}[N, H, C_1 \parallel T, C_2] \neq \perp$ then
    Return $\mathsf{V}[N, H, C_1 \parallel T, C_2]$
$(P_0, P_1, P) \leftarrow\!\!\$\ R(N, 2n + |C_1|)$
$M \leftarrow C_1 \oplus P$
$C_2' \leftarrow F^{\mathrm{cr}}_{P_0}(H \parallel M)$
$T' \leftarrow F_{P_1}(C_2')$
If $T' \neq T$ then Return $\perp$
$\mathsf{win} \leftarrow \mathsf{true}$
Return $(M, P_0)$

$\underline{\mathbf{ChalDec}(N, H, C_1 \parallel T, C_2):}$

If $(N, H, C_1 \parallel T, C_2) \in \mathcal{Y}$ then
    Return $\perp$
$(P_0, P_1, P) \leftarrow\!\!\$\ R(N, 2n + |C_1|)$
$M \leftarrow C_1 \oplus P$
$C_2' \leftarrow F^{\mathrm{cr}}_{P_0}(H \parallel M)$
$T' \leftarrow F_{P_1}(C_2')$
If $T' \neq T$ then Return $\perp$
$\mathsf{win} \leftarrow \mathsf{true}$
Return $(M, P_0)$
```

```
$\underline{\mathcal{C}^{\mathbf{Tag},\mathbf{Ver}}:}$

$\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalDec}}$
Return $\mathsf{win}$

$\underline{\mathbf{Enc}(N, H, M):}$

$(P_0, P_1, P) \leftarrow\!\!\$\ R(N, 2n + |M|)$
$C_1 \leftarrow M \oplus P$
$C_2 \leftarrow F^{\mathrm{cr}}_{P_0}(H \parallel M)$
$T \leftarrow \mathbf{Tag}(N, (P_0, H \parallel M))$
$\mathsf{V}[N, H, C_1 \parallel T, C_2] \leftarrow (M, P_0)$
Return $(C_1 \parallel T, C_2)$

$\underline{\mathbf{Dec}(N, H, C_1 \parallel T, C_2):}$

If $\mathsf{V}[N, H, C_1 \parallel T, C_2] \neq \perp$ then
    Return $\mathsf{V}[N, H, C_1 \parallel T, C_2]$
$(P_0, P_1, P) \leftarrow\!\!\$\ R(N, 2n + |C_1|)$
$M \leftarrow C_1 \oplus P$
$b \leftarrow \mathbf{Ver}(N, (P_0, H \parallel M), T)$
If $b \neq b'$ then Return $\perp$
Return $(M, P_0)$

$\underline{\mathbf{ChalDec}(N, H, C_1 \parallel T, C_2):}$

If $(N, H, C_1 \parallel T, C_2) \in \mathcal{Y}$ then
    Return $\perp$
$\mathcal{N} \leftarrow \mathcal{N} \cup \{N\}$
$(P_0, P_1, P) \leftarrow R(N, 2n + |C_1|)$
$M \leftarrow C_1 \oplus P$
$b \leftarrow \mathbf{Ver}(N, (P_0, H \parallel M), T)$
If $b \neq b'$ then Return $\perp$
Return $(M, P_0)$
```

Figure 16: Games for proof of MO-nCTXT in Theorem 12. The functions **Ver** and **Tag** are verification and tag generation oracles, respectively, for $\mathcal{C}$'s MU-UF-CMA game as in Figure 2.

Finally, we turn to binding. Recall that any scheme that effectively runs commitment verification during decryption achieves sender binding. The check that $C_2' = C_2$ during decryption accomplishes this, and so the scheme is perfectly sender binding. For receiver binding, a simple reduction gives the following theorem showing that the CR of $F^{\rm cr}$ implies binding of CEP.

Theorem 13 *[CEP receiver binding] Let* $\mathrm{CEP} = \mathrm{CEP}[G, F, F^{\rm cr}]$. *Let $\mathcal{A}$ be any* r-BIND$_{\rm CEP}$ *adversary. Then we give an adversary $\mathcal{B}$ such that* $\mathbf{Adv}^{\text{r-bind}}_{\rm CEP}(\mathcal{A}) \le \mathbf{Adv}^{\rm cr}_{F^{\rm cr}}(\mathcal{B})$ *and $\mathcal{B}$ runs in the same amount of time as $\mathcal{A}$.*

Proof: The adversary $\mathcal{B}$ runs $\mathcal{A}$ until it outputs $((H, M, K_f), (H', M', K_f'), C_2)$. Since r-BIND$^{\mathcal{A}}_{\rm CEP}$ outputs 1 only if $F^{\rm cr}_{K_f}(H \parallel M) = F^{\rm cr}_{K_f'}(H' \parallel M') = C_2$, a output for which r-BIND$^{\mathcal{A}}_{\rm CEP} \Rightarrow 1$ is also a valid collision in $F^{\rm cr}$. Thus, if $\mathcal{B}$ outputs whatever $\mathcal{A}$ does, the result follows.

9 Related Work

The primary viewpoint in the literature has been that committing encryption is undesirable either because one wants deniability [20,22,54] or due to the theoretical challenges associated with proving encryption confidentiality in the face of adaptive compromises [23]. Thus while *non*-committing encryption has received significant attention (e.g., [22–24,26,28,35,36,43,47,53,54,67–70]), there is a dearth of literature on building purposefully committing encryption.

We are aware of only one previous work on building committing encryption schemes, due to Gertner and Herzberg [37]. They give definitions that are insufficient for the message franking setting (in particular they do not capture server binding or multiple opening security). They do not analyze AE schemes, and focus only on building asymmetric primitives.

Our receiver binding security property is related to the concept of robust encryption, introduced by Abdalla et al. [1]. They give two security notions for public-key encryption (PKE). The stronger, called strong robustness, asks that an adversarially-chosen ciphertext should only correctly decrypt under at most one legitimate secret key. Mohassel [52] showed efficient ways of adapting existing PKE schemes to be robust. Farshim et al. [33] subsequently pointed out that some applications require robustness to adversarially generated secret keys, and introduced a notion called complete robustness. In a later work, Farshim, Orlandi, and Rosie [34] adapt these robustness definitions to the setting of authenticated encryption, message authentication codes (MACs), and pseudorandom functions (PRFs). They show that in this context, the simpler full robustness notion of [33] is the strongest of those considered.

These prior notions, in particular the full robustness for AE notion from [34], do not suffice for formalizing binding for AEAD. First, it does not capture sender binding. Second, for receiver binding, it turns out that the most straightforward adaptation of full robustness to handle associated data fails to imply receiver binding. We provide a more detailed explanation in Appendix A.

Abdalla et al. [1] propose a generic composition of a commitment scheme and PKE scheme to achieve robustness and Farshim et al. [34] show a variant of this for the symmetric encryption setting. The latter construction commits to the key, not the message, and could not be used to achieve the multiple opening security targeted by our generic composition constructions.

Selective-opening security allows an adversary to adaptively choose to corrupt some senders that sent (correlated) encrypted messages [8] or to compromise the keys of a subset of receivers [41]. Bellare et al. [8] gave the first constructions of schemes secure against selective-opening attacks for sender corruptions. Non-committing encryption can be used to realize security for receiver corruptions. Our definitions do not model selective-opening attacks, and as mentioned in the introduction, assessing the viability of committing AEAD in selective-opening settings is an interesting open problem.

Acknowledgments

The authors would like to thank the anonymous reviewers of Crypto 2017 for their thoughtful comments, as well as Mihir Bellare for discussions about robust encryption and its relation to binding. This work was funded in part by NSF grant CNS-1330308.

References

[1] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In *TCC*, 2010.

[2] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In *CRYPTO*, 2006.

[3] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *CRYPTO*, 1996.

[4] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *FOCS*, 1996.

[5] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. HMAC: Keyed-hashing for message authentication. *Internet Request for Comment RFC*, 2104, 1997.

[6] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, 1997.

[7] Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In *EUROCRYPT*, 2012.

[8] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, 2009.

[9] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT*, 2000.

[10] Mihir Bellare and Thomas Ristenpart. Multi-property-preserving hash domain extension and the EMD transform. In – *ASIACRYPT*, 2006.

[11] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, 1993.

[12] Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In *ASIACRYPT*, 2000.

[13] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT*, 2006.

[14] Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Igors Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In *CRYPTO*, 2017.

[15] Daniel J. Bernstein. ChaCha, a variant of Salsa20. `https://cr.yp.to/chacha/chacha-20080128.pdf`.

[16] Daniel J Bernstein. The Poly1305-AES message-authentication code. In *FSE*, 2005.

[17] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and secure message authentication. In *CRYPTO*, 1999.

[18] John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *EUROCRYPT*, 2002.

[19] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *SAC*, 2002.

[20] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use PGP. In *ACM Workshop on Privacy in the Electronic Society*, 2004.

[21] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 1988.

[22] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *CRYPTO*, 1997.

[23] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC*, 1996.

[24] Ran Canetti, Oxana Poburinnaya, and Mariana Raykova. Optimal-rate non-committing encryption in a CRS model. *IACR Cryptology ePrint Archive, Report 2016/511*, 2016.

[25] Colin Chaigneau and Henri Gilbert. Is AEZ v4.1 sufficiently resilient against key-recovery attacks? *IACR Transactions on Symmetric Cryptology*, 2016.

[26] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *ASIACRYPT*, 2009.

[27] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the Signal messaging protocol. *IACR ePrint Archive, Report 2016/1013*, 2016.

[28] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, 2000.

[29] Yevgeniy Dodis, Thomas Ristenpart, John Steinberger, and Stefano Tessaro. To hash or not to hash again?(in) differentiability results for H^2 and HMAC. In *CRYPTO*, 2012.

[30] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Review*, 2003.

[31] Facebook. Facebook Messenger app. `https://www.messenger.com/`, 2016.

[32] Facebook. Messenger Secret Conversations technical whitepaper. `https://fbnewsroomus.files.wordpress.com/2016/07/secret_conversations_whitepaper-1.pdf`, 2016.

[33] Pooya Farshim, Benoît Libert, Kenneth G Paterson, and Elizabeth A Quaglia. Robust encryption, revisited. In *PKC*. 2013.

[34] Pooya Farshim, Claudio Orlandi, and Razvan Rosie. Security of symmetric primitives under incorrect usage of keys. *IACR Transactions on Symmetric Cryptology*, 2017(1):449–473, 2017.

[35] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. *IACR Cryptology ePrint Archive, Report 2008/534*, 2008.

[36] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In *CRYPTO*, 2009.

[37] Yitchak Gertner and Amir Herzberg. Committing encryption and publicly-verifiable signcryption. *IACR Cryptology ePrint Archive, Report 2003/254*, 2003.

[38] Shai Halevi and Hugo Krawczyk. Security under key-dependent inputs. In *ACM CCS*, 2007.

[39] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In *CRYPTO*, 2003.

[40] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In *CT-RSA*, 2004.

[41] Carmit Hazay, Arpita Patra, and Bogdan Warinschi. Selective opening security for receivers. In *ASIACRYPT*, 2015.

[42] Michael Hearn. Modern anti-spam and E2E crypto. `https://moderncrypto.org/mail-archive/messaging/2014/000780.html`.

[43] Brett Hemenway, Rafail Ostrovsky, and Alon Rosen. Non-committing encryption from Φ-hiding. In *TCC (1)*, 2015.

[44] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In *EUROCRYPT*, 2015.

[45] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-key CBC MAC. In *FSE*, 2003.

[46] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In *CRYPTO*, 2001.

[47] Feiyu Lei, Wen Chen, and Kefei Chen. A non-committing encryption scheme based on quadratic residue. In *ISCIS*, 2006.

[48] Moxie Marlinspike. libsignal protocol (Java). `https://github.com/WhisperSystems/libsignal-protocol-java`, 2016.

[49] Moxie Marlinspike and Trevor Perrin. The Double Ratchet algorithm. `https://whispersystems.org/docs/specifications/doubleratchet/doubleratchet.pdf`.

[50] David McGrew and John Viega. The Galois/counter mode of operation (GCM). *Submission to NIST Modes of Operation Process*, 20, 2004.

[51] Jon Millican. Challenges of E2E Encryption in Facebook Messenger. Real World Cryptography conference, 2017. `https://www.realworldcrypto.com/rwc2017/program`.

[52] Payman Mohassel. A closer look at anonymity and robustness in encryption schemes. In *ASIACRYPT*, 2010.

[53] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, 2002.

[54] Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In *CRYPTO*, 2011.

[55] Kenneth G Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In *ASIACRYPT*, 2011.

[56] Phillip Rogaway. Authenticated-encryption with associated-data. In *ACM CCS*, 2002.

[57] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *ASIACRYPT*. Springer, 2004.

[58] Phillip Rogaway. Nonce-based symmetric encryption. In *FSE*, 2004.

[59] Phillip Rogaway. Formalizing human ignorance. In *VIETCRYPT*. 2006.

[60] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In *EUROCRYPT*, 2006.

[61] Mark Dermot Ryan. Enhanced certificate transparency and end-to-end encrypted mail. In *NDSS*. The Internet Society, 2014.

[62] Douglas R Stinson. Universal hashing and authentication codes. *Designs, Codes and Cryptography*, 1994.

[63] Liang Wang, Rafael Pass, abhi shelat, and Thomas Ristenpart. Secure channel injection and anonymous proofs of account ownership. Cryptology ePrint Archive, Report 2016/925, 2016. `http://eprint.iacr.org/2016/925`.

[64] Mark N Wegman and J Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 1981.

[65] Whatsapp. Whatsapp. `https://www.whatsapp.com/`, 2016.

[66] Wikipedia. Signal (software). `https://en.wikipedia.org/wiki/Signal_(software)`, 2016.

[67] Huafei Zhu, Tadashi Araragi, Takashi Nishide, and Kouichi Sakurai. Adaptive and composable non-committing encryptions. In *ACISP*, 2010.

[68] Huafei Zhu, Tadashi Araragi, Takashi Nishide, and Kouichi Sakurai. Universally composable non-committing encryptions in the presence of adaptive adversaries. In *SECRYPT*, 2010.

[69] Huafei Zhu and Feng Bao. Non-committing encryptions based on oblivious Naor-Pinkas cryptosystems. In *INDOCRYPT*, 2009.

[70] Huafei Zhu and Feng Bao. Error-free, multi-bit non-committing encryption with constant round complexity. In *Inscrypt*, 2010.

```
r-BIND$^{\mathcal{A}}_{\mathsf{SE}}$:
$((H, M, K), (H', M', K'), C) \leftarrow\!\$\ \mathcal{A}$
$\overline{M} \leftarrow \mathsf{dec}(K, H, C)$
$\overline{M}' \leftarrow \mathsf{dec}(K', H', C)$
Return $(M = \overline{M}) \wedge (M' = \overline{M}') \wedge ((H, M) \neq (H', M'))$
```

```
FROB$^{\mathcal{A}}_{\mathsf{SE}}$:
$((H, K), (H', K'), C) \leftarrow\!\$\ \mathcal{A}$
If $K = K'$ then   Return $\mathsf{false}$
$M \leftarrow \mathsf{Dec}(K, H, C)$
$M' \leftarrow \mathsf{Dec}(K', H', C)$
Return $(M \neq \perp) \wedge (M' \neq \perp)$
```

Figure 17: **(Left)** An equivalent formulation of receiver binding for the case of traditionally committing encryption schemes. **(Right)** The full robustness security game.

A Traditionally Committing Encryption and Robust Encryption

Traditionally committing encryption. Committing AEAD as formulated in the body explicitly aims to accommodate compactness. For traditionally committing encryption, in which the entire ciphertext is taken as the commitment, the secret key is the opening, and one opens by decrypting the ciphertext, then one can dispense with some complexities of the treatment given in the body. This will also allow us to compare with prior work directly, which does not consider compact commitments.

To consider traditionally committing encryption, we return to standard symmetric encryption schemes given by a tuple $\mathsf{SE} = (\mathsf{kg}, \mathsf{enc}, \mathsf{dec})$ as defined in Section 2. Then the single-opening security notions for ROR and CTXT apply (with the obvious syntactic tweak to change ciphertext to singletons as opposed to pairs). We give the variant of receiver binding on the left of Figure 17. This is equivalent to receiver binding for CE schemes for which: (1) the entire ciphertext is the commitment C_2; (2) the opening value is the secret key; and (3) $\mathsf{Ver}(H, M, K_f, C_2)$ works by running $M' \leftarrow \mathsf{Dec}(K_f, H, C_2)$, outputting one if $M = M'$, and outputting zero otherwise.

Robust encryption. Robust encryption, introduced by Abdalla et al. [1], targets schemes for which a ciphertext cannot decrypt under distinct keys to valid messages. Robustness is important when using anonymous encryption, and may also help render encryption more misuse resistant. Robustness is closely related to traditionally committing encryption, though there are several subtleties.

Figure 17 (right box) provides a game defining full robustness, following the formulation of robustness for AE schemes given in [34] with minor adaptations to our notation and to accommodate headers. The adversary $\mathcal{A}$ outputs a pair of keys K, K', a pair of headers H, H', and a ciphertext C. The adversary wins if the keys are distinct and the ciphertext decrypts to a valid message under (K, H) and (K', H'). The FROB$^{\mathcal{A}}_{\mathsf{SE}}$ advantage is defined by

$$\mathbf{Adv}^{\mathrm{frob}}_{\mathsf{SE}}(\mathcal{A}) = \Pr\left[\, \mathrm{FROB}^{\mathcal{A}}_{\mathsf{SE}} \Rightarrow \mathsf{true} \,\right] .$$

Receiver binding does not imply full robustness for traditionally committing encryption. The reason is that receiver binding requires messages be distinct, and so one can come up with a scheme that achieves receiver binding but not full robustness.

On the other hand, full robustness implies receiver binding for, crucially, schemes that do not have associated data. When dispensing with associated data, we remove from security games and schemes reference to headers. Consider an adversary $\mathcal{A}$ that achieves success in the r-BIND$_{\mathsf{SE}}$ game for some scheme SE which does not use headers. Then, for $\mathcal{A}$ to win it must output a pair $((M, K), (M', K'), C)$ for which C decrypts under K to M and K' to M' and $M \neq M'$ and neither equal $\perp$. It must be the case that $K \neq K'$ since otherwise decryption, which is deterministic, would produce the same message. Thus, this triple is a winning triple for full robustness: $K \neq K'$ and both keys decrypt C to valid messages. The following theorem formalizes this implication.

Theorem 14 *Let SE be any authenticated encryption scheme which does not use headers. Let $\mathcal{A}$ be an r-BIND$_{\mathsf{SE}}$ adversary. Then we construct an adversary $\mathcal{B}$ which runs in the same amount of time as $\mathcal{A}$ such that $\mathbf{Adv}^{\text{r-bind}}_{\mathsf{SE}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{frob}}_{\mathsf{SE}}(\mathcal{B})$.*

Note that this theorem implies that the FROB-secure AE schemes from Farshim et al. meet receiver binding security. It is simple to adapt those schemes and show they are committing authenticated encryption schemes (with no associated data) as we have defined them.

When associated data is again considered, things get more complicated because it is no longer the case that the same key, ciphertext pair must decrypt to the same plaintext — the associated header may differ. Indeed in this case we give a counter-example showing that receiver binding is not implied by full robustness.

Let $\mathsf{SE} = (\mathsf{kg}, \mathsf{enc}, \mathsf{dec})$ be a symmetric encryption scheme, with key space $\mathcal{K} = \{0,1\}^\kappa$ for some κ and ciphertext space $\mathcal{C}$. Assume it enjoys full robustness, real-or-random security, and ciphertext integrity. We use it to construct a new scheme, $\overline{\mathsf{SE}} = (\mathsf{kg}, \overline{\mathsf{enc}}, \overline{\mathsf{dec}})$, that will likewise enjoy full robustness, real-or-random security, and ciphertext integrity, yet is not receiver binding. Let $C^* \notin \mathcal{C}$ be a distinguished ciphertext. The ciphertext space for $\overline{\mathsf{SE}}$ is $\overline{\mathcal{C}} = \mathcal{C} \cup \{C^*\}$. The key space for $\overline{\mathsf{SE}}$ is $\overline{\mathcal{K}} = \{0,1\}^{\kappa+1}$. Key generation $\overline{\mathsf{kg}}$ runs kg and outputs $K \,\|\, 0$. Encryption $\overline{\mathsf{enc}}(K \,\|\, b, H, M)$ runs $\mathsf{enc}(K, H, M)$ and outputs the resulting ciphertext. Decryption $\overline{\mathsf{dec}}(K \,\|\, b, H, C)$ first checks if $K = 1^{\kappa+1}$ and $C = C^*$ for some distinguished ciphertext value C^*. If so, it outputs H as the message (we assume that $\mathcal{H} \subseteq \mathcal{M}$, meaning the header space is a subset of the message space). If instead $C = C^*$ but $K \neq 1^{\kappa+1}$, it outputs $\perp$. Otherwise it runs $\mathsf{dec}(K, H, C)$ and outputs the result.

The scheme $\overline{\mathsf{SE}}$ is not receiver binding: have an adversary output $(H, H, 1^{\kappa+1}), (H', H', 1^{\kappa+1}), C^*)$ for some arbitrary $H \neq H'$. At the same time, the artificial behavior of $\overline{\mathsf{SE}}$ on the all ones key does not compromise robustness, because the latter requires producing distinct keys. We show this formally in the next theorem.

Theorem 15 *Let $\overline{\mathsf{SE}}$ be the SE scheme constructed as defined above using scheme SE. Then for any FROB$_{\overline{\mathsf{SE}}}$ adversary $\mathcal{A}$ we give an FROB$_{\mathsf{SE}}$ adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\text{frob}}_{\overline{\mathsf{SE}}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{frob}}_{\mathsf{SE}}(\mathcal{B})$. Adversary $\mathcal{B}$ runs in time that of $\mathcal{A}$.*

Proof: Adversary $\mathcal{B}$ runs $\mathcal{A}$ to obtain output $((H, K \,\|\, b), (H', K' \,\|\, b'), C)$. Adversary $\mathcal{B}$ checks if $C = C^*$, and if so outputs an arbitrary triple of values (essentially giving up). Otherwise $\mathcal{B}$ outputs $((H, K), (H', K'), C)$. We will show that the only way for $\mathcal{B}$ to obtain advantage is by not using C^*, in which case a successful attack against $\overline{\mathsf{SE}}$ translates to one against SE.

First consider the case that $\mathcal{A}$'s output has $C \neq C^*$. Then for $\mathcal{A}$ to win against $\overline{\mathsf{SE}}$ it must be the case that $\mathcal{B}$ wins against SE — for $C \neq C^*$ decryption $\overline{\mathsf{dec}}$ just runs dec on the truncated key. Now consider the case that $C = C^*$. Then because winning requires that $K \neq K'$, it must be that either $K \neq 1^{\kappa+1}$ or $K' \neq 1^{\kappa+1}$ for $\mathcal{A}$ to succeed. But whichever is not equal to $1^{\kappa+1}$ cannot lead to an output other than $\perp$ from decryption, by construction of $\overline{\mathsf{dec}}$. Thus in this case $\mathcal{A}$ cannot win. ∎

It can be verified that $\overline{\mathsf{SE}}$ inherits the ROR security and CTXT integrity of SE (up to some small additive loss related to a key being randomly chosen as the all ones key). This means that enjoying these properties as well as full robustness is still not sufficient to imply receiver binding.

A stronger robustness notion. The fact that full robustness and receiver binding are orthogonal (when associated data is considered) raises the question of whether there exists a stronger notion

```
eFROB$^{\mathcal{A}}_{\mathsf{SE}}$:
$((H,K),(H',K'),C) \leftarrow\!\!\$\ \mathcal{A}$
$M \leftarrow \mathsf{dec}(K,H,C)$
$M' \leftarrow \mathsf{dec}(K',H',C)$
Return $(M \neq \bot) \wedge (M' \neq \bot) \wedge ((H,M,K) \neq (H',M',K'))$
```

Figure 18: A stronger notion of full robustness.

that implies both. In fact, we can relax the condition in full robustness that $K \neq K'$ to $(H,M,K) \neq (H',M',K')$. See Figure 18. We refer to this notion as 'even fuller robustness'. The eFROB$_{\mathsf{SE}}$ advantage of an adversary $\mathcal{A}$ is given by

$$\mathbf{Adv}^{\mathrm{efrob}}_{\mathsf{SE}}(\mathcal{A}) = \Pr\left[\,\mathrm{eFROB}^{\mathcal{A}}_{\mathsf{SE}} \Rightarrow \mathsf{true}\,\right] .$$

It is easy to verify that this notion is strictly stronger than both receiver binding and full robustness.

From the point of view of receiver binding, what this strengthening does is extend security to also prevent attacks that produce different secret keys, yet open the same ciphertext to the same message using the same header. This does not seem to be a meaningful attack in the context of abuse reporting, but may prove important elsewhere.

B All-in-one Confidentiality/Integrity Security Notions

Here we provide all-in-one versions of committing AEAD confidentiality and integrity security for both single-opening and multiple-opening security. We focus on multiple-openings security for nonce-based schemes; an analogous treatment for randomized schemes and/or single-openings is easily derived from the following.

The all-in-one games MO-nREAL-C$^{\mathcal{A}}_{\mathrm{nCE}}$ and MO-nRAND$^{\mathcal{A}}_{\mathrm{nCE}}$ are shown in Figure 4. We measure the multiple-openings real-or-random MO-nROR-C$_{\mathrm{nCE}}$ advantage of an adversary $\mathcal{A}$ against a scheme nCE by

$$\mathbf{Adv}^{\mathrm{mo\text{-}nror\text{-}ctxt}}_{\mathrm{nCE}}(\mathcal{A}) = |\Pr[\,\mathrm{MO\text{-}nREAL\text{-}C}_{\mathrm{nCE},\mathcal{A}} \Rightarrow 1\,] - \Pr[\,\mathrm{MO\text{-}nRAND\text{-}C}_{\mathrm{nCE},\mathcal{A}} \Rightarrow 1\,]| .$$

An adversary $\mathcal{A}$ is nonce-respecting if its queries never repeat the same N in its encryption queries (to either **Enc** or **ChalEnc**).

The next theorem shows that the all-in-one notion is implied by the two standalone notions.

Theorem 16 *Let* nCE *be a nonce-based committing AEAD scheme, let* $\mathcal{A}$ *be an* MO-nROR-C$_{\mathrm{nCE}}$ *adversary making at most* q *queries to its oracles. Then there exist specific adversaries* $\mathcal{B},\mathcal{C}$ *given in the proof below such that*

$$\mathbf{Adv}^{\mathrm{mo\text{-}nror\text{-}ctxt}}_{\mathrm{nCE}}(\mathcal{A}) \leq 2\cdot\mathbf{Adv}^{\mathrm{mo\text{-}ctxt}}_{\mathrm{nCE}}(\mathcal{B}) + \mathbf{Adv}^{\mathrm{mo\text{-}ror}}_{\mathrm{nCE}}(\mathcal{C}) .$$

Adversaries $\mathcal{B}$ *and* $\mathcal{C}$ *use at most the same number of queries as* $\mathcal{A}$*. Adversary* $\mathcal{C}$ *runs in time that of* $\mathcal{A}$*. Adversary* $\mathcal{B}$ *runs in time that of* $\mathcal{A}$ *plus an* $\mathcal{O}(q)$ *overhead.*

Proof: We start with a game G_0 = MO-nREAL-C$^{\mathcal{A}}_{\mathsf{CE}}$. Game G_1 is the same as G_0 except that: (1) **Dec** queries on tuples $(N,H,C_1,C_2) \notin \mathcal{Y}_1$ are returned with $\bot$, and (2) **ChalDec** queries are always answered with $\bot$. We construct an adversary $\mathcal{B}_0$ such that

$$\Pr[\,\mathrm{G}_0 \Rightarrow 1\,] = \Pr[\,\mathrm{G}_1 \Rightarrow 1\,] + \mathbf{Adv}^{\mathrm{mo\text{-}nctxt}}_{\mathrm{nCE}}(\mathcal{B}_0) . \quad (2)$$

Adversary $\mathcal{B}_0$ starts by running $\mathcal{A}$. It simulates **Enc** and **ChalEnc** queries as in the MO-nREAL-C$_{\mathrm{nCE}}$ game but using its own **Enc** oracle for encryption executions. It simulates **Dec** by checking if

```
$\underline{\text{MO-nREAL-C}^{\mathcal{A}}_{\text{nCE}}}$:
$K \leftarrow\!\!\$\ \mathsf{Kg}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textbf{Enc},\textbf{Dec},\textbf{ChalEnc},\textbf{ChalDec}}$
Return $b'$

$\underline{\textbf{Enc}(N,H,M)}$
$(C_1,C_2) \leftarrow \text{Enc}^N_K(H,M)$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{(N,H,C_1,C_2)\}$
Return $C$

$\underline{\textbf{Dec}(N,H,C_1,C_2)}$
If $(N,H,C_1,C_2) \in \mathcal{Y}_2$ then
    Return $\bot$
$(M,K_f) \leftarrow \mathsf{Dec}^N_K(H,C_1,C_2)$
Return $(M,K_f)$

$\underline{\textbf{ChalEnc}(N,H,M)}$
$(C_1,C_2) \leftarrow\!\!\$\ \text{Enc}^N_K(H,M)$
$\mathcal{Y}_2 \leftarrow \mathcal{Y}_2 \cup \{(N,H,C_1,C_2)\}$
Return $C$

$\underline{\textbf{ChalDec}(N,H,C_1,C_2)}$
If $(N,H,C_1,C_2) \in \mathcal{Y}_1 \cup \mathcal{Y}_2$ then
    Return $\bot$
$(M,K_f) \leftarrow \mathsf{Dec}^N_K(H,C_1,C_2)$
Return $(M,K_f)$
```

```
$\underline{\text{MO-nRAND-C}^{\mathcal{A}}_{\text{nCE}}}$:
$K \leftarrow\!\!\$\ \mathsf{Kg}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textbf{Enc},\textbf{Dec},\textbf{ChalEnc},\textbf{ChalDec}}$
Return $b'$

$\underline{\textbf{Enc}(N,H,M)}$
$(C_1,C_2) \leftarrow\!\!\$\ \text{Enc}^N_K(H,M)$
Return $C$

$\underline{\textbf{Dec}(N,H,C_1,C_2)}$
If $(N,H,C_1,C_2) \in \mathcal{Y}_2$ then
    Return $\bot$
$(M,K_f) \leftarrow \mathsf{Dec}^N_K(H,C_1,C_2)$
Return $(M,K_f)$

$\underline{\textbf{ChalEnc}(N,H,M)}$
$(\ell_1,\ell_2) \leftarrow \mathsf{clen}(|H|,|M|)$
$(C_1,C_2) \leftarrow\!\!\$\ \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$
$\mathcal{Y}_2 \leftarrow \mathcal{Y}_2 \cup \{(N,H,C_1,C_2)\}$
Return $(C_1,C_2)$

$\underline{\textbf{ChalDec}(N,H,C_1,C_2)}$
Return $\bot$
```

Figure 19: All-in-one confidentiality and ciphertext integrity games for nonce-based committing AEAD.

$(N,H,C_1,C_2) \in \mathcal{Y}_1$. If so, it queries its own **Dec** oracle and returns the result. Otherwise, it proceeds by checking if $(N,H,C_1,C_2) \in \mathcal{Y}_2$, returning $\bot$ if so, and otherwise makes a query to **ChalDec**(N,H,C_1,C_2) and returns the result. Finally it simulates **ChalDec** queries by replacing decryption with a call to its own **ChalDec** oracle. By construction $\mathcal{B}_0$ never queries its own **ChalDec** in the case that N,H,C_1,C_2 was returned by any of its encryption queries. It is straightforward to verify that for this $\mathcal{B}_0$ equality (2) holds.

At this stage game G_1 only runs decryption on values (N,H,C_1,C_2) that were returned by **Enc**. The next game G_2 is the same as G_1 except that **ChalEnc** queries are answered with random bits as in MO-nRAND-C_{nCE}. We construct an adversary $\mathcal{C}$ such that

$$\Pr[\,\text{G}_1 \Rightarrow 1\,] = \Pr[\,\text{G}_2 \Rightarrow 1\,] + \mathbf{Adv}^{\text{mo-nror}}_{\text{nCE}}(\mathcal{C})\ . \tag{3}$$

Adversary $\mathcal{C}$ starts by running $\mathcal{A}$. It simulates **Enc** queries as in the MO-nREAL-C_{CE} game, but using its own **Enc** oracle. Similarly it simulates **Dec** queries using its own **Dec** oracle for decryption. Notice that by our earlier transitions, it's the case that only tuples (N,H,C_1,C_2) returned by $\mathcal{C}$'s **Enc** queries are queried to its **Dec** oracle. To any **ChalDec** query, it replies with $\bot$. It forwards **ChalEnc** queries to its own **ChalEnc** oracle. It is straightforward to verify that for this $\mathcal{B}_0$ equality (3) holds.

Game G_2 is almost the same as MO-nRAND-C_{nCE}, the only difference is that **Dec** only performs decryption on points that were returned by a previous query to **Enc**. We therefore introduce one more game G_3 which is the same as G_2 except that we go back to performing decryption for any query to **Dec** besides those in the set $\mathcal{Y}_2$. To do so we give an adversary $\mathcal{B}_1$ such that

$$\Pr[\,\text{G}_2 \Rightarrow 1\,] = \Pr[\,\text{G}_3 \Rightarrow 1\,] + \mathbf{Adv}^{\text{mo-nctxt}}_{\text{nCE}}(\mathcal{B}_1)\ . \tag{4}$$

Adversary $\mathcal{B}_1$ starts by running $\mathcal{A}$. It simulates **Enc** and **ChalEnc** queries as in the MO-nRAND-$\mathrm{C}_{\mathrm{nCE}}$ game but using its own **Enc** oracle for encryption executions within **Enc**. It simulates **Dec** by checking if $(N, H, C_1, C_2) \in \mathcal{Y}_1$. If so, it queries its own **Dec** oracle and returns the result. Otherwise, it proceeds by checking if $(N, H, C_1, C_2) \in \mathcal{Y}_2$, returning $\perp$ if so, and otherwise makes a query to $\mathbf{ChalDec}(N, H, C_1, C_2)$ and returns the result. Finally it simulates **ChalDec** queries as in MO-nRAND-$\mathrm{C}_{\mathrm{nCE}}$, i.e., returning $\perp$ for any query. By construction $\mathcal{B}_1$ never queries its own **ChalDec** in the case that (N, H, C_1, C_2) was returned by any of its encryption queries. It is straightforward to verify that for this $\mathcal{B}_1$ equality (4) holds.

Finally, a standard argument gives a concrete $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{mo-nctxt}}_{\text{nCE}}(\mathcal{B}_0) + \mathbf{Adv}^{\text{mo-nctxt}}_{\text{nCE}}(\mathcal{B}_1) \leq 2 \cdot \mathbf{Adv}^{\text{mo-nctxt}}_{\text{nCE}}(\mathcal{B}) .$$

Combining all the equations above, and verifying the run-times and query complexity of the adversaries, proves the theorem. ∎

Next we give a theorem showing that the all-in-one notion implies the multiple-opening standalone notions. We omit the proofs, which are simple, for the sake of brevity.

Theorem 17 *Let* nCE *be a nonce-based committing AEAD scheme,* $\mathcal{A}_1$ *be an* MO-$\mathrm{nROR}_{\mathrm{nCE}}$ *adversary, and* $\mathcal{A}_2$ *be an* MO-$\mathrm{nCTXT}_{\mathrm{nCE}}$ *adversary. Then there exists concrete adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *such that* $\mathbf{Adv}^{\text{mo-nror}}_{\text{nCE}}(\mathcal{A}_1) \leq \mathbf{Adv}^{\text{mo-nror-ctxt}}_{\text{nCE}}(\mathcal{B}_1)$ *and* $\mathbf{Adv}^{\text{mo-nctxt}}_{\text{nCE}}(\mathcal{A}_2) \leq \mathbf{Adv}^{\text{mo-nror-ctxt}}_{\text{nCE}}(\mathcal{B}_2)$. *For* $x \in \{1, 2\}$, *adversary* $\mathcal{B}_x$ *uses at most as many queries as* $\mathcal{A}_x$ *and runs in time that of* $\mathcal{A}_x$.