# Committing Authenticated Encryption: Generic Transforms with Hash Functions*

Shan Chen[1] and Vukašin Karadžić[2]

[1] Southern University of Science and Technology, Shenzhen, China**
dragoncs16@gmail.com
[2] Technische Universität Darmstadt, Darmstadt, Germany
vukasin.karadzic@tu-darmstadt.de

**Abstract.** Recent applications and attacks have highlighted the need for authenticated encryption (AE) schemes to achieve the so-called committing security beyond privacy and authenticity. As a result, several generic solutions have been proposed to transform a non-committing AE scheme to a committing one, for both basic unique-nonce security and advanced misuse-resistant (MR) security. We observe that all existing practical generic transforms are subject to at least one of the following limitations: (i) not committing to the entire encryption context, (ii) involving non-standard primitives, (iii) not being a black-box transform, (iv) providing limited committing security. Furthermore, so far, there has been no generic transform that can directly elevate a basic AE scheme to a committing AE scheme that offers MR security. Our work fills these gaps by developing black-box generic transforms that crucially rely on hash functions, which are well standardized and widely deployed.

First, we construct three basic transforms that combine AE with a single hash function, which we call HtAE, AEaH and EtH. They all guarantee strong security, and EtH can be applied to both AE and basic privacy-only encryption schemes. Next, for MR security, we propose two advanced hash-based transforms that we call AEtH and chaSIV. AEtH is an MRAE-preserving transform that adds committing security to an MR-secure AE scheme. chaSIV is the *first* generic transform that can directly elevate basic AE to one with both committing and MR security; moreover, chaSIV also works with arbitrary privacy-only encryption schemes. Both of them feature a simple design and ensure strong security.

For performance evaluation, we compare our transforms to similar existing ones, both in theory and through practical implementations. The results show that our AEaH achieves the highest practical efficiency among basic transforms, while AEtH excels in MRAE-preserving transforms. Our MRAE-lifting transform chaSIV demonstrates comparable performance to MRAE-preserving ones and surpasses them for messages larger than approximately 360 bytes; for longer messages, it even outperforms the benchmark, non-committing standardized AES-GCM-SIV.

**Keywords:** Authenticated Encryption · Committing Security · Misuse Resistance · Generic Transform · Hash Function · Random Oracle Model



** Shan Chen is affiliated with both the Research Institute of Trustworthy Autonomous Systems and the Department of Computer Science and Engineering of SUSTech.

# Table of Contents

## 1 Introduction

Symmetric encryption is widely used in practice to ensure the privacy and authenticity of data. Such encryption schemes are often referred to as *authenticated encryption (AE)* [4], which were initially designed to address practical shortcomings of basic encryption schemes that provide only privacy. Their syntax also evolves from probabilistic [15] to nonce-based with associated data [27], with nonces to avoid reliance on good randomness for their security and associated data to capture unencrypted but authenticated data.

In the past few years, a series of applications and attacks [1, 11, 16, 23] motivated another level of security for symmetric encryption. Such so-called *committing security* requires the ciphertext to be a *commitment* to the encryption inputs, e.g., to the key $K$, to the message $M$, or even to all inputs including the nonce $N$ and associated data $AD$. For example, the latter most secure case requires that no efficient adversary can find two distinct tuples $(K_1, N_1, AD_1, M_1) \neq (K_2, N_2, AD_2, M_2)$ such that they are encrypted to the same ciphertext. Authenticated encryption schemes that further satisfy committing security are often called *committing authenticated encryption*.

It has been shown in recent works [1, 9, 11, 16, 23, 24] that almost none of the existing AE schemes achieves committing security, including the most popular ones, e.g., AES-GCM [12], ChaCha20/Poly1305 [5, 6], CCM [34], OCB [29], AES-GCM-SIV [17], etc. In order to construct committing AE schemes, one approach is to directly add committing security to existing AE schemes through dedicated modifications. For example, Bellare and Hoang [2] proposed specific adjustments to AES-GCM and AES-GCM-SIV, resulting in schemes that achieve both AE security (privacy and authenticity) and committing security. However, such dedicated schemes require a deep understanding of the modified constructions, which could make them difficult for practitioners, especially non-experts, to implement correctly, as committing AE schemes are not yet standardized.

The other approach is to build committing AE from *generic transforms*. Although some of the above dedicated schemes may perform better in practice, the generically transformed schemes enjoy good modularity and flexibility. Such schemes are usually simpler, and they allow the component primitives to be instantiated with any constructions that satisfy the required syntax and security.

In this work, we ask how easy one can generically transform authenticated encryption (and even basic privacy-only encryption) to committing authenticated encryption, using standardized primitives. Before delving into our constructions, we first discuss existing generic transforms proposed in previous works and motivate our work on generic transforms and the use of hash functions.

### 1.1 Generic Transforms in Previous Works

We summarize the existing generic transforms in Table 1. The study of committing security for authenticated encryption schemes is initialized by Farshim *et al.* [13]. They proposed a generic transform EtM built from a pseudorandom generator (PRG), an AE scheme and a message authentication code (MAC). Their construction guarantees ciphertexts committing to the key by relying on non-standard security from PRG and MAC, e.g., PRG being right collision-resistant. Soon after, Grubbs *et al.* [16] proposed two generic commit-then-encrypt constructions CtE1 and CtE2 that combine a commitment scheme with AE. Such schemes ensure committing security to the message, in order to satisfy the security goals of message franking, a mechanism that verifies abuse reports. Their CtE1 transform relies on standard commitments, while CtE2 requires a non-standard property (i.e., unique commitments) for its commitment component. Then, a follow-up work [11] on message franking introduced a more efficient transform based on a new primitive called encryptment, but the resulting committing security still targets the message only. Later, Albertini *et al.* [1] explored vulnerabilities in real-world systems resulting from lack of key commitment and proposed a simple generic solution called CommitKey$\Pi$. This construction makes use of a collision-resistant pseudorandom function (PRF), a non-standard primitive, and provides committing security to both the key and the nonce.

To better mitigate the potential risks arising from insufficient committing security and to ease the burden on practitioners in understanding the nuances of different committing targets, recent efforts focus on building schemes with strong context-committing security. That is, the resulting ciphertext must commit to the entire *context*: key, nonce, associated data, and message.

Bellare and Hoang [2] proposed the first generic transform that provides the above strong context-committing security. Their transform, which we denote by HtE $\circ$ UtC, consists of two component transforms: first, UtC adds key commitment to an AE scheme, and then HtE elevates the key-committing AE

| Construction | Committing Target | Transform Method | Non-Standard Primitive | Black-Box Transform |
|---|---|---|---|---|
| EtM [13] | $K$ | PRG-then-Encrypt-then-MAC | PRG, MAC | ✓ |
| CtE1/CtE2 [16] | $M$ | Commit-then-AE | Commit for CtE2 | ✓ |
| CE [11] | $M$ | Encryptment-then-AE | Encryptment | ✓ |
| CommitKeyΠ [1] | $K, N$ | PRF-then-AE | PRF | ✓ |
| N1 [33] | $K, N, AD, M$ | Encrypt-and-MAC (N1 [26]) | Encrypt, MAC | ✓ |
| EaM/SIV [8] | $K, N, AD, M$ | (N1, A2)/A4 [26] | Encrypt, MAC | ✓ |
| KEtM [8] | $K, N, AD, M$ | KDF-then-(N2, A6) [26] | KDF, MAC | ✓ |
| HtE ∘ UtC [2] | $K, N, AD, M$ | Hash-then-PRF-then-AE | PRF | ✓ |
| HtE ∘ RtC [2] | $K, N, AD, M$ | Hash-then-PRF-then-AE-then-Hash | PRF | ✓ |
| CTX [9] | $K, N, AD, M$ | AE-then-Hash | - | ✗ |
| CTY [3] | $K, N, AD, M$ | AE-then-Hash | - | ✗ |
| PACT/comPACT [7] | $K, N, AD, M$ | AE-and-Hash-then-Encipher | - | ✓ |
| HtAE (this work) | $K, N, AD, M$ | Hash-then-AE | - | ✓ |
| AEaH (this work) | $K, N, AD, M$ | AE-and-Hash | - | ✓ |
| EtH (this work) | $K, N, AD, M$ | Encrypt-then-Hash | - | ✓ |
| AEtH (this work) | $K, N, AD, M$ | Hash-then-AE | - | ✓ |
| chaSIV (this work) | $K, N, AD, M$ | PRF-and-Hash-then-Encrypt | - | ✓ |

Table 1: Comparison of practical generic transforms to construct committing AE. The "Non-Standard Primitive" column lists the component primitives that are tailored to ensure some form of committing security but are not yet standardized, whereas "-" indicates that all component primitives are already standardized. The red-colored component primitive imposes a hard limit on the achievable committing security of the associated transforms.

to a context-committing one. The authors also proposed another transform HtE ∘ RtC to add context-committing security to a misuse-resistant AE scheme. The security of both UtC and RtC relies on that of a non-standard primitive that they introduced and named committing PRFs.

In a work contemporary to [2], Chan and Rogaway [9] proposed a generic transform called CTX that also achieves context-committing security. CTX is very simple and efficient: it just adds a hash computation to an AE scheme. A recent work by Bellare and Hoang [3] improved CTX to an even more efficient transform called CTY, in which the associated data input to the underlying AE is omitted. However, both CTX and CTY require the underlying AE to produce a separate authentication tag, rendering them *not* black-box transforms. As a result, they cannot accommodate AE schemes that use the authentication tag for decryption or do not produce a separate authentication tag, including those based on design paradigms such as synthetic IV (SIV), MAC-then-Encrypt (MtE) and Encode-then-Encipher (EtE). The latter has become more important lately, as NIST plans to develop a new block cipher mode of operation called *accordion mode* [10], with one of its main applications being the construction of authenticated encryption schemes using the EtE paradigm.

The above CTX/CTY limitation was also observed by a very recent work by Bhattacharjee *et al.* [7]. To support arbitrary AE schemes, they proposed two generic transforms, PACT and comPACT, with PACT further preserving the advanced security of the underlying AE in the presence of nonce misuse. At the end of these transforms, instead of hashing the AE authentication tag as in CTX/CTY, they encrypt the authentication tag (or one block of the AE ciphertext) with a block cipher. As a result, PACT and comPACT incur no extra ciphertext expansion, i.e., the resulting ciphertext is of the same length as the AE ciphertext; their committing security, however, is also limited to at most half the block size (e.g., up to 64 bits when AES is used) due to birthday attacks, and it cannot be expanded.

There are two other recent works [8, 33] that explored the possibility to develop generic transforms by generically composing an encryption scheme and a MAC. As shown in Table 1, their resulting transforms all involve primitives that have not been standardized. Naito *et al.* [25] also proposed a generic transform, which exploits message redundancy to achieve beyond-birthday-bound committing security. Though theoretically intriguing, their transform is not well-suited for practical use, as the sufficient message redundancy required to achieve good committing security is typically unavailable. Also, their

transform demands specific adjustments to the underlying encryption scheme based on the exact message redundancy, which is unrealistic or at least cumbersome in practice.

### 1.2 Motivation

Similar to classical AE generic composition (e.g., Encrypt-then-Mac) [4], black-box generic transforms for committing AE are important to investigate in both theory and practice. They provide modularity, flexibility, broader applicability, and robustness in cryptographic protocols. However, so far, no existing generic transform for committing AE serves as a good counterpart to Encrypt-then-MAC. From our discussion on previous works, all existing practical generic transforms suffer from at least one of the following limitations: (i) not context-committing, (ii) involving primitives not yet standardized, (iii) not being a black-box transform, (iv) providing limited committing security. Our goal is to develop practical black-box context-committing generic transforms using standard primitives, such that their committing security can be expanded as needed.

Furthermore, we observe that, all practical instantiations of the existing generic transforms that ensure context-committing security were proved secure under *idealized* assumptions.[3] For example, security proofs of CTX/CTY instantiations were performed in the random oracle model, and the security of the practical committing PRF instantiation proposed by [2] (that underlies UtC and RtC transforms) was proved in the ideal cipher model.

Since idealized assumptions are currently inevitable for practical constructions and hash functions are well standardized and widely deployed, it is both natural and appealing to develop generic transforms that combine AE with hash functions and prove their security in the random oracle model. Additionally, given that hash-based commitment schemes are both straightforward and efficient, hash functions serve as attractive candidates for constructing simple and practical schemes with committing security. These considerations motivate our focus on using hash functions to develop generic transforms, which aligns with the design choices of existing transforms such as CTX/CTY and PACT/comPACT.

We emphasize that our work is not aimed at developing a committing AE scheme with the best practical performance, as generic transforms typically fall short in that regard, and dedicated schemes like those proposed in [2] are better suited for that purpose. Nevertheless, we strive to make our generic transforms as straightforward and efficient as possible, ensuring they are easy for practitioners to understand and implement, thereby facilitating the adoption and deployment of committing AE. The simplicity of our design is particularly useful, given that committing AE schemes are currently not yet standardized and practitioners may need to develop their own implementations.[4] As a result, our work focuses on exploring various simple paradigms for constructing generic transforms using hash functions, rather than minimizing the ciphertext expansion (i.e., ciphertext length minus plaintext length) as was done in [3,9]. Even so, as we will show, some of our transforms do exhibit compact ciphertext expansion.

Finally, to the best of our knowledge, there is currently no generic transform that can directly elevate an AE scheme to a committing one with advanced misuse-resistant security. Such a transform can be highly useful in practice when misuse-resistant AE schemes are either unavailable (e.g., in Mozilla NSS or libsodium) or inefficiently implemented (e.g., in wolfSSL) within the chosen cryptographic library. Our work also aims to fill this gap.

### 1.3 Our Contributions

**Basic transforms.** As motivated above, we first explore black-box generic transforms that combine AE with a single hash function. To this end, similar to the classical generic composition approaches presented in the design of authenticated encryption schemes [4], we also consider three transform methods: Hash-then-AE (HtAE), AE-and-Hash (AEaH) and Encrypt-then-Hash (EtH). For each transform method, we propose a hash-based transform. We call these transforms *basic transforms* and naturally name them HtAE, AEaH, EtH, respectively. Their encryption processes are illustrated in Figure 1.

As shown in the figure, all basic transforms follow a very straightforward and clean design. We note that the first two transforms are built from an AE scheme, while the last EtH transform can directly

[3] Note that it is possible to prove instantiations of generic transforms secure in the standard model, e.g., HtE∘UtC with its underlying committing PRF instantiated with hardcore predicates [2, 13], but such constructions are only theoretical.

[4] We remark that standardization can be a lengthy process. For example, NIST's lightweight cryptography, started in 2015, selected Ascon in 2023, published a draft standard in 2024 [32], and will require more time for widespread adoption.

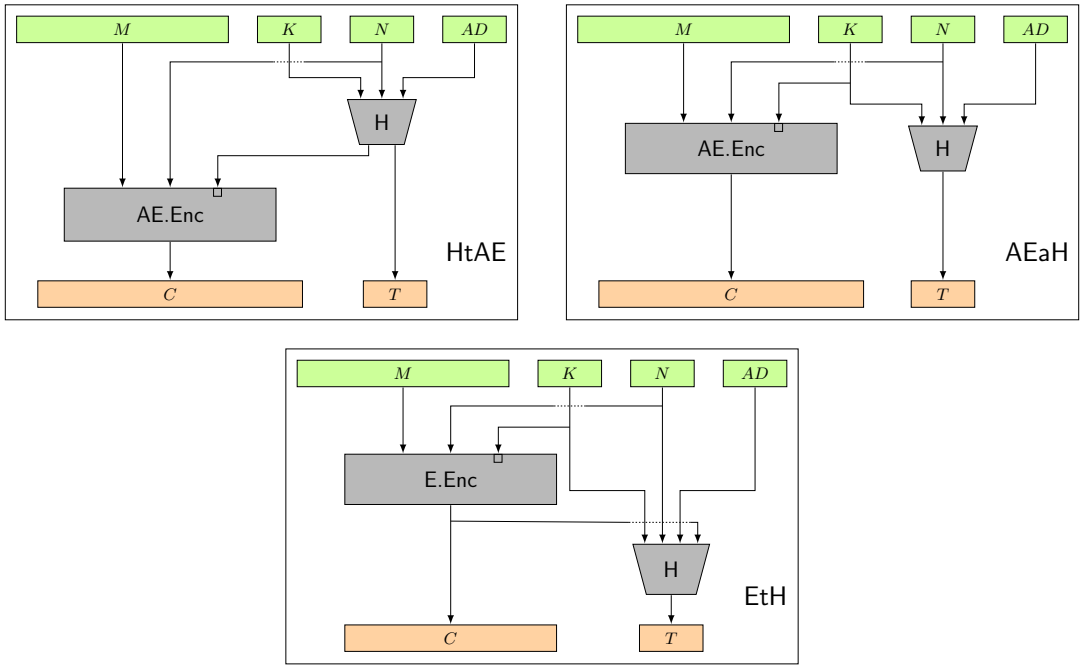


Fig. 1: Our three basic transforms HtAE, AEaH and EtH.

elevate a basic privacy-only encryption scheme to a committing AE scheme. From the figure, one can also observe that the encryption and hash computations in our second transform AEaH are independent by design and hence fully parallelizable, while the other two transforms have to run their two component primitives sequentially.

We prove that all basic transforms achieve both AE security and strong committing security. For committing security, in addition to the primary context-committing (CMT) notion, we also consider the context-discovery (CDY) notion proposed in [24]. CMT and CDY are very relevant notions; they can be viewed as analogous to collision-resistance and preimage-resistance for hash functions. As shown in [24], CDY can be implied by CMT when the encryption algorithm is "context-compressing", i.e., ciphertexts are decryptable under multiple contexts. However, using generic transforms to construct AE with strong committing security usually results in larger ciphertext expansion, which may lead to "non-context-compressing" encryption algorithms. Therefore, we also include CDY for our security analysis. CDY security was also analyzed in [8] for their transforms built from generic compositions. However, to the best of our knowledge, so far no AE-based transforms have been analyzed for CDY security.

For performance, we compare our two AE-based transforms with other similar existing transforms, and show that HtAE and AEaH are both more efficient than their comparable counterparts. Our comparison also implies that AEaH achieves the highest efficiency. To understand the practical performance of our basic transforms and some relevant existing ones, we implement these transforms using OpenSSL and wolfSSL libraries,[5] and measure their overhead relative to the widely used (non-committing) AE scheme, AES-GCM [12]. Our results show that, for all tested message lengths (16 to 2048 bytes), the total cost of the most efficient transform, AEaH, is at most three times that of AES-GCM.

**Advanced transforms.** Next, to achieve (nonce-)misuse-resistant (MR) security, we propose two advanced hash-based transforms.

Our first transform is designed to add committing security to AE schemes that already provide MR security. Such a so-called MRAE-preserving transform follows the AE-then-Hash (AEtH) style, and hence we name it AEtH. Its encryption process is illustrated in Figure 2 on the left. As shown in the figure, instead of hashing the entire AE ciphertext, AEtH hashes only the last block (or any block) of it to improve efficiency while preserving security.

Next, we construct the *first* generic transform that can directly "lift" a basic (unique-nonce) AE scheme to one with both committing and MR security. Our design of such a MRAE-lifting transform is

[5] While OpenSSL is arguably the most widely used cryptographic library, wolfSSL, despite its lower conventional popularity, still powers cryptography in billions of devices [21], making it a valuable second choice for our experiments.

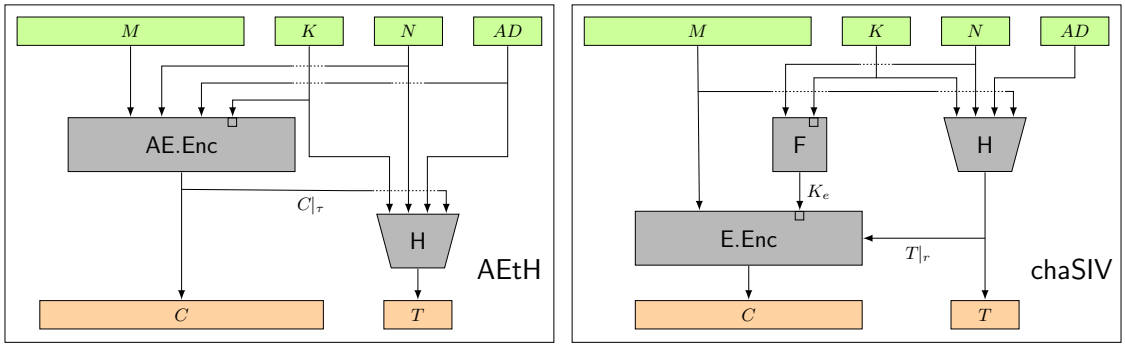


Fig. 2: Our two advanced transforms AEtH and chaSIV. $X|_m$ denotes last $m$ bits of a bitstring $X$. $\tau$ denotes the ciphertext expansion of the AE scheme AE and $r$ denotes the nonce size of the encryption scheme E.

very simple. It can be viewed as a composition of HtAE and the SIV mode [30]: just replace the MAC in SIV with a hash function and add a PRF to re-key the underlying encryption scheme per distinct nonce. We therefore name this transform chaSIV, short for ***c**ommitting **ha**sh-based **SIV***, with its encryption process illustrated in Figure 2 on the right. As shown in the figure, chaSIV can be built from a basic privacy-only encryption scheme, so like EtH it can be applied to both AE and privacy-only encryption.

We prove that both of our advanced transforms achieve the desired strong committing misuse-resistant security. Then, we compare their concrete security when instantiated with consistent standardized primitives. The results show that our chaSIV, despite its simple design and MRAE-lifting power, achieves security comparable to that of our MRAE-preserving transform AEtH.

Finally, we compare the performance of our advanced transforms with other similar existing MRAE-preserving transforms. Both the theoretical and implementation results show that CTX [9] and our AEtH (essentially a black-box generalization of CTX) exhibit the highest efficiency. In OpenSSL, our MRAE-lifting transform chaSIV demonstrates similar performance to CTX and AEtH and surpasses them for messages longer than about 360 bytes; surprisingly, for messages longer than 1024 bytes, our committing AE chaSIV even outperforms the benchmark, non-committing standardized misuse-resistant AE AES-GCM-SIV [17]. Then, in wolfSSL, chaSIV outperforms all other transforms across all message sizes, including the non-committing benchmark AES-SIV [19], the only misuse-resistant AE available in wolfSSL; the result is mainly due to the slow implementation of AES-SIV, which is used by all MRAE-preserving transforms.

## 2 Notation and Definitions

In the section, we introduce some notation, recall preliminary definitions of the building blocks, and define certain notions that we will use in this work.

**Notation.** Let $\varepsilon$ denote the empty string and $\{0,1\}^*$ denote the set of all finite binary strings. For a binary string $X$, let $|X|$ denote its bit length. For a binary string $X$, where $|X| \geq n$, the last (i.e., rightmost) $n$ bits of $X$ is denoted by $X|_n$. Let the special symbol $\perp$ (which is outside $\{0,1\}^*$) denote uninitialized state or failure. Let $x \leftarrow y$ denote assigning $y$ to $x$. For a randomized algorithm A, let $x \xleftarrow{\$} \mathsf{A}$ denote running A and assigning the output to $x$; if A is deterministic, we write $x \leftarrow \mathsf{A}$ instead. For a finite set $\mathcal{S}$, let $|\mathcal{S}|$ denote its size and $S \xleftarrow{\$} \mathcal{S}$ denote sampling $S$ from $\mathcal{S}$ uniformly at random. For two sets $\mathcal{X}$ and $\mathcal{Y}$, let $\mathcal{X} \xleftarrow{\cup} \mathcal{Y}$ denote $\mathcal{X} \leftarrow \mathcal{X} \cup \mathcal{Y}$. For a real number $r$, let $\lceil r \rceil$ denote the smallest integer that is larger than or equal to $r$. For a function H that takes only one input, $\mathsf{H}(x_1, x_2, \ldots, x_n)$ indicates that the inputs $x_1, x_2, \ldots, x_n$ are injectively encoded to a single input before being fed into H; note that they can be simply concatenated when the first $n-1$ inputs are of fixed lengths. For an adversary $\mathcal{A}$ (an algorithm), the notation $\mathcal{A}^O$ denotes that $\mathcal{A}$ has oracle access to $O(\cdot)$.

### 2.1 Pseudorandom Functions and Hash Functions

**Pseudorandom functions (PRFs).** For a function $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^n$, consider two PRF games associated with an adversary $\mathcal{A}$. In the "real" world, $\mathcal{A}$ has oracle access to the function

| Game $\mathsf{CR}_{\mathsf{H}}^{\mathcal{A}}$ |
|---|
| $(X_1, X_2) \xleftarrow{\$} \mathcal{A}$ |
| **return** $\mathsf{H}(X_1) = \mathsf{H}(X_2)$ |

| Game $\mathsf{rCR}_{\mathsf{H}'}^{\mathcal{A}}$ |
|---|
| $(X_1, X_2) \xleftarrow{\$} \mathcal{A}$ ; $(Y_{1,L}, Y_{1,R}) \leftarrow \mathsf{H}'(X_1)$ ; $(Y_{2,L}, Y_{2,R}) \leftarrow \mathsf{H}'(X_2)$ |
| **return** $Y_{1,R} = Y_{2,R}$ |

Fig. 3: The collision-resistance game (**left**) for a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^v$, and the right-collision-resistance game (**right**) for a hash function $\mathsf{H}' : \{0,1\}^* \to \{0,1\}^v \times \{0,1\}^w$.

$\mathsf{F}_K(\cdot) = \mathsf{F}(K, \cdot)$, where $K$ is randomly sampled from $\{0,1\}^\kappa$; in the "ideal" world, it has oracle access to a random function $\mathsf{f}(\cdot)$, where $\mathsf{f}$ is uniformly sampled from $\mathcal{F}$, the family of all functions with domain $\{0,1\}^*$ and range $\{0,1\}^n$. In the end, $\mathcal{A}$ outputs a bit $b'$ guessing which world it was in. Its PRF advantage measure is defined as

$$\mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{A}) = \Pr\Big[K \xleftarrow{\$} \{0,1\}^\kappa : \mathcal{A}^{\mathsf{F}_K} \Rightarrow 1\Big] - \Pr\Big[\mathsf{f} \xleftarrow{\$} \mathcal{F} : \mathcal{A}^{\mathsf{f}} \Rightarrow 1\Big].$$

$\mathsf{F}$ is called a PRF if the above advantage is sufficiently small for any efficient adversary $\mathcal{A}$.

**Hash functions.** An (unkeyed) hash function $\mathsf{H} : \{0,1\}^* \to \mathcal{Y}$ maps a binary string $X \in \{0,1\}^*$ to an element $Y$ from the message digest space $\mathcal{Y}$. In this work, we consider two types of digest spaces $\mathcal{Y} = \{0,1\}^u$ and $\mathcal{Y} = \{0,1\}^v \times \{0,1\}^w = \{0,1\}^{v+w}$, where $u$, $v$, $w$ are integers. That is, the second type is a special case of the first type, with the digest space explicitly divided into two parts.

As shown in Figure 3, we consider two collision resistance notions for hash functions. The first security game $\mathsf{CR}_{\mathsf{H}}^{\mathcal{A}}$ defines the "regular" collision resistance for a hash function $\mathsf{H}$ with digest space $\mathcal{Y} = \{0,1\}^u$, where the adversary wins by finding a collision, i.e., two distinct messages hashing to the same digest. Then, the second security game $\mathsf{rCR}_{\mathsf{H}'}^{\mathcal{A}}$ defines a so-called *right collision resistance* notion for a hash function $\mathsf{H}'$ with digest space $\mathcal{Y} = \{0,1\}^v \times \{0,1\}^w$, where the adversary wins by finding a collision only on the right part of the digest space $\{0,1\}^w$. Their advantage measures are defined respectively as

$$\mathbf{Adv}_{\mathsf{H}}^{\mathrm{cr}}(\mathcal{A}) = \Pr\Big[\mathsf{CR}_{\mathsf{H}}^{\mathcal{A}} \Rightarrow 1\Big] \quad \text{and} \quad \mathbf{Adv}_{\mathsf{H}'}^{\mathrm{rcr}}(\mathcal{A}) = \Pr\Big[\mathsf{rCR}_{\mathsf{H}'}^{\mathcal{A}} \Rightarrow 1\Big].$$

From the above definitions, it is clear that right collision resistance implies collision resistance. It is generally believed that the practical SHA-2/3 hash families satisfy both collision resistance notions, but the security of right collision resistance is restricted by the right-part digest length $w$.

### 2.2 Symmetric Encryption

A (nonce-based) *symmetric encryption (SE)* scheme $\mathsf{SE}$ specifies a pair of deterministic algorithms $(\mathsf{Enc}, \mathsf{Dec})$ and is associated with a key space $\{0,1\}^\kappa$, nonce space $\mathcal{N} \subseteq \{0,1\}^*$, associated data space $\mathcal{AD} \subseteq \{0,1\}^*$, and message space $\mathcal{M} \subseteq \{0,1\}^*$. Its encryption algorithm $\mathsf{SE.Enc}$ takes as input a key $K \in \{0,1\}^\kappa$, nonce $N \in \mathcal{N}$, associated data $AD \in \mathcal{AD}$, message $M \in \mathcal{M}$, and outputs a ciphertext $C \in \{0,1\}^*$. Its decryption algorithm $\mathsf{SE.Dec}$ takes as input a key $K$, nonce $N \in \mathcal{N}$, associated data $AD \in \mathcal{AD}$, ciphertext $C \in \{0,1\}^*$, and outputs either a message $M \in \mathcal{M}$ or the special symbol $\perp$ indicating decryption failure. Correctness requires that $\mathsf{SE.Dec}(K, N, AD, \mathsf{SE.Enc}(K, N, AD, M)) = M$ holds for all $K \in \{0,1\}^\kappa$, $N \in \mathcal{N}$, $AD \in \mathcal{AD}$, $M \in \mathcal{M}$.

The above SE syntax actually coincides with the syntax of an *authenticated encryption (AE)* scheme $\mathsf{AE}$. For simplicity, we also abuse the same SE syntax to capture a basic nonce-based encryption scheme $\mathsf{E}$, for which the associated data is ignored and correctness requires that $\mathsf{E.Dec}(K, N, \mathsf{E.Enc}(K, N, M)) = M$ holds for all $K \in \{0,1\}^\kappa$, $N \in \mathcal{N}$, $M \in \mathcal{M}$. A basic nonce-based encryption scheme $\mathsf{E}$ is said to be *tidy* [26] if $M \leftarrow \mathsf{E.Dec}(K, N, C)$ implies $\mathsf{E.Enc}(K, N, M) = C$. Tidiness is a natural requirement for practical nonce-based encryption schemes to avoid unnecessary degeneration and mitigate security risks. Correctness and tidiness together imply that functions $\mathsf{E.Enc}(K, N, \cdot)$ and $\mathsf{E.Dec}(K, N, \cdot)$ are the inverse of each other.

We assume a symmetric encryption scheme $\mathsf{SE}$ (and also implicitly $\mathsf{E}$) is further associated with a linear-time computable ciphertext-length function $\mathsf{SE.clen}$ such that $|\mathsf{SE.Enc}(K, N, AD, M)| = \mathsf{SE.clen}(|M|)$ holds for all $K, N, AD, M$. In this work, we consider only "natural" SE schemes that satisfy $\min_{M \in \mathcal{M}}\{\mathsf{SE.clen}(|M|) - |M|\} = \tau > 0$, where the constant $\tau$ is called *ciphertext expansion* of $\mathsf{SE}$. We note that, in order to achieve practical security, ciphertext expansions of real-world deployed AE schemes are usually of sufficient length, e.g., 96 or 128 bits.

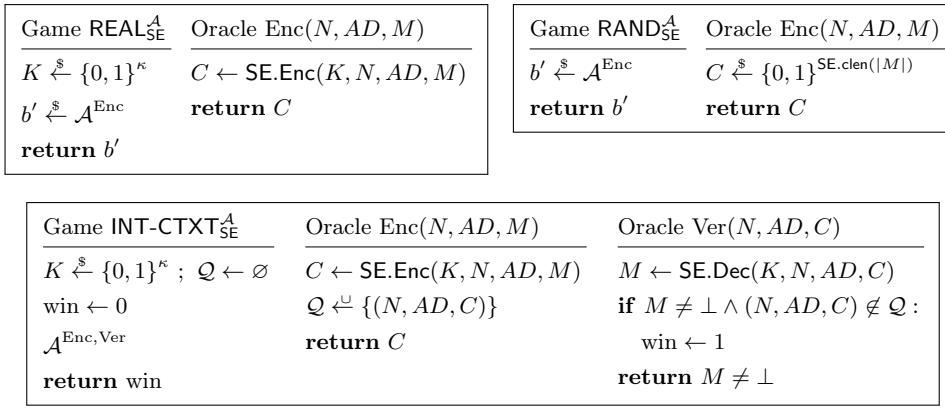


Fig. 4: The privacy games, real (**top left**) and random (**top right**), and the authenticity game (**bottom**) for a symmetric encryption scheme $\mathsf{SE}$.

For security of an SE scheme, in addition to AE security notions (i.e., privacy and authenticity), we also consider two committing security notions: context committing (CMT) and context discovery (CDY). Here "context" refers to the encryption input $(K, N, AD, M)$. These committing security notions can be viewed as analogous to collision-resistance and preimage-resistance for hash functions. If a symmetric encryption scheme is expected to ensure both AE security and committing security, it is called *committing authenticated encryption*.

**AE security.** We recall the basic privacy and authenticity notions as defined in [27], for which the adversary is required to be *nonce-respecting*, i.e., it never repeats a nonce for its encryption queries. Besides, we also consider the stronger nonce-misuse resistant notions and a weaker privacy notion.

For privacy, consider the two privacy games shown in Figure 4 that define the notion of IND$-CPA: indistinguishability from random bits under chosen-plaintext attack. In both games the adversary $\mathcal{A}$ has access to an encryption oracle Enc, which returns the encrypted ciphertext in the "real" world $\mathsf{REAL}^{\mathcal{A}}_{\mathsf{SE}}$ and returns random bits in the "ideal" world $\mathsf{RAND}^{\mathcal{A}}_{\mathsf{SE}}$. In the end, $\mathcal{A}$ outputs a bit $b'$ guessing which world it was in. Its IND$-CPA advantage measure is defined as

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{SE}}(\mathcal{A}) = \Pr\left[\mathsf{REAL}^{\mathcal{A}}_{\mathsf{SE}} \Rightarrow 1\right] - \Pr\left[\mathsf{RAND}^{\mathcal{A}}_{\mathsf{SE}} \Rightarrow 1\right].$$

If the adversary is restricted to make *at most one* query to its encryption oracle, the resulting weaker privacy notion is named one-time IND$-CPA (or simply OT-IND$-CPA), with its advantage measure denoted by $\mathbf{Adv}^{\text{ot-ind\$-cpa}}_{\mathsf{SE}}(\mathcal{A})$.

For authenticity, consider the authenticity game $\mathsf{INT\text{-}CTXT}^{\mathcal{A}}_{\mathsf{SE}}$ also shown in Figure 4 that defines ciphertext integrity (INT-CTXT). Here, the adversary $\mathcal{A}$ wins by forging a new tuple $(N, AD, C)$ that can be successfully decrypted. Its INT-CTXT advantage measure is defined as

$$\mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{SE}}(\mathcal{A}) = \Pr\left[\mathsf{INT\text{-}CTXT}^{\mathcal{A}}_{\mathsf{SE}} \Rightarrow 1\right].$$

Note that the above also defines privacy and authenticity security for a nonce-based encryption scheme $\mathsf{E}$ when associated data is ignored [28]. In this work, when such encryption schemes are concerned, we implicitly refer to those that ensure only privacy but not authenticity. These basic encryption schemes usually serve as a component of AE schemes and hence have simpler and more efficient instantiations than AE schemes.

**MRAE security.** We also consider the stronger (nonce-)misuse-resistant (MR) AE security, for which the adversary is no longer required to be nonce-respecting. However, to avoid trivial wins, the adversary is not allowed to repeat a query to its encryption oracle. The misuse-resistant AE security notions are named MR-IND$-CPA and MR-INT-CTXT respectively, with advantage measures defined in the same way and denoted by $\mathbf{Adv}^{\text{mr-ind\$-cpa}}_{\mathsf{SE}}(\mathcal{A})$ and $\mathbf{Adv}^{\text{mr-int-ctxt}}_{\mathsf{SE}}(\mathcal{A})$.

**Context-committing security.** The context-committing game shown in Figure 5 defines the CMT-3 notion proposed by [2] that we simply call CMT, which is conceptually simpler than but security-wise equivalent to their strongest CMT-4 notion. The adversary wins by finding two different contexts

Game $\mathsf{CMT}^{\mathcal{A}}_{\mathsf{SE}}$

$((K_1, N_1, AD_1, M_1), (K_2, N_2, AD_2, M_2)) \xleftarrow{\$} \mathcal{A}$

$C_1 \leftarrow \mathsf{SE.Enc}(K_1, N_1, AD_1, M_1)$ ; $C_2 \leftarrow \mathsf{SE.Enc}(K_2, N_2, AD_2, M_2)$

**return** $C_1 = C_2 \wedge (K_1, N_1, AD_1) \neq (K_2, N_2, AD_2)$

Fig. 5: The context-committing game for a symmetric encryption scheme $\mathsf{SE}$.

Game $\mathsf{CDY}^{\mathcal{A}}_{\mathsf{SE},\mathsf{S}}$

$C \xleftarrow{\$} \mathsf{S}$ ; $(K, N, AD, M) \xleftarrow{\$} \mathcal{A}(C)$

**return** $\mathsf{SE.Enc}(K, N, AD, M) = C$

Context selector $\mathsf{S}_\$$

$K \xleftarrow{\$} \{0,1\}^\kappa$ ; $N \xleftarrow{\$} \mathcal{N}$ ; $AD \xleftarrow{\$} \mathcal{AD}$ ; $M \xleftarrow{\$} \mathcal{M}$

**return** $\mathsf{SE.Enc}(K, N, AD, M)$

Fig. 6: The context-discovery game parameterized by a context selector $\mathsf{S}$ for a symmetric encryption scheme $\mathsf{SE}$ (**left**) and the context selector $\mathsf{S}_\$$ (**right**).

encrypting to the same ciphertext. Its advantage measure is defined as

$$\mathbf{Adv}^{\mathrm{cmt}}_{\mathsf{SE}}(\mathcal{A}) = \Pr\left[\mathsf{CMT}^{\mathcal{A}}_{\mathsf{SE}} \Rightarrow 1\right].$$

**Context-discovery security.** We follow [8] to consider a simplified version of the context-discovery security notion proposed by [24], which we call CDY. As shown in Figure 6, the CDY game is parameterized by a so-called context selector $\mathsf{S}$. The adversary is challenged with a ciphertext $C$ selected by $\mathsf{S}$ and wins by finding a context $(K, N, AD, M)$ that encrypts to $C$. We define its advantage measure as:

$$\mathbf{Adv}^{\mathrm{cdy}}_{\mathsf{SE},\mathsf{S}}(\mathcal{A}) = \Pr\left[\mathsf{CDY}^{\mathcal{A}}_{\mathsf{SE},\mathsf{S}} \Rightarrow 1\right].$$

In Figure 6 we also specify a context selector $\mathsf{S}_\$$ that we think is of particular interest. $\mathsf{S}_\$$ just samples the challenge context uniformly at random, with this context selector $\mathsf{S}_\$$ the CDY notion can be viewed as *one-wayness* of the encryption algorithm. For simplicity, we write CDY\$ to denote this notion and its advantage measure is therefore defined as:

$$\mathbf{Adv}^{\mathrm{cdy\$}}_{\mathsf{SE}}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{cdy}}_{\mathsf{SE},\mathsf{S}_\$}(\mathcal{A}) = \Pr\left[\mathsf{CDY}^{\mathcal{A}}_{\mathsf{SE},\mathsf{S}_\$} \Rightarrow 1\right].$$

The above context selector $\mathsf{S}_\$$ also occurred in the proof of Theorem 1 in [24]. This theorem shows that CMT implies CDY\$ for symmetric encryption schemes that are "context-compressing", i.e., ciphertexts are decryptable under multiple contexts. This property is satisfied by many standardized AE schemes (e.g., $\mathsf{AES\text{-}GCM}$) and their unstandardized CMT-secure variants (e.g., those proposed in [2]). Nevertheless, it is still interesting to explore both CMT and CDY\$ security for committing AE schemes that are generically transformed from AE schemes, e.g., $\mathsf{HtE} \circ \mathsf{UtC}$ [2], $\mathsf{CTX}$ [9], $\mathsf{CTY}$ [3], and the hash-based constructions that we will present in this work. Such schemes may involve larger ciphertext expansion due to the added committing security, and hence may not be context-compressing, especially when the $\mathcal{AD}$ size is small.

We note that Bhaumik *et al.* [8] proved CDY security for some SE schemes constructed from generic compositions, but their proofs can only work with context selectors that have *no* access to the underlying MAC component. As a result, the context selectors considered in [8] are actually quite restrictive, e.g., they cannot even capture the above $\mathsf{S}_\$$ that encrypts a random context; also, such selectors may not produce a valid challenge ciphertext, for which CDY security trivially holds and is hence meaningless. Therefore, we focus on context selector $\mathsf{S}_\$$ when analyzing CDY security of our constructions, but one can easily adapt our CDY\$ security proofs to handle arbitrary context selectors that have *no* access to the encryption algorithm.

## 3 Basic Transforms

We begin by exploring generic transforms that combine an AE scheme with a hash function, aiming for the resulting schemes to achieve both AE security and committing security. We refer to constructions in this section as *basic transforms*, as the more advanced MRAE security is not considered here.

| $\mathsf{HtAE.Enc}(K, N, AD, M)$ | $\mathsf{HtAE.Dec}(K, N, AD, (C, T))$ |
|---|---|
| $(K_e, T) \leftarrow \mathsf{H}(K, N, AD)$ | $(K_e, T') \leftarrow \mathsf{H}(K, N, AD)$ |
| $C \leftarrow \mathsf{AE.Enc}(K_e, N, \varepsilon, M)$ | **if** $T' \neq T$ : **return** $\perp$ |
| **return** $(C, T)$ | $M \leftarrow \mathsf{AE.Dec}(K_e, N, \varepsilon, C)$ |
| | **return** $M$ |

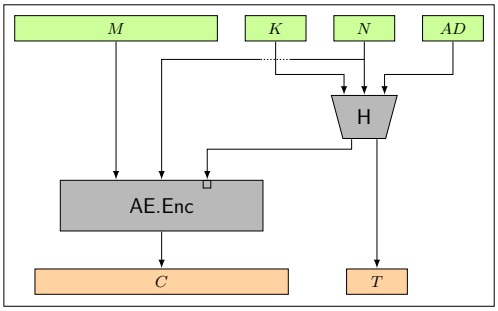


Fig. 7: The pseudocode (**left**) of HtAE transform and the graphical illustration (**right**) of its encryption algorithm.

Similar to the classical generic composition paradigms for the design of authenticated encryption schemes [4], we also consider three transform methods: Hash-then-AE (HtAE), AE-and-Hash (AEaH) and Encrypt-then-Hash (EtH). The latter transform is named Encrypt-then-Hash because it can be applied to both AE and basic privacy-only encryption schemes.

In the following, we first present our basic transforms and their security results individually, comparing each to similar existing transforms. Then, we compare the security guarantees of our basic transforms, deferring their performance comparison to a later section, where we also compare the performance of advanced transforms.

### 3.1 Hash-then-AE (HtAE)

For the Hash-then-AE (HtAE) transform, we present the HtAE construction. HtAE is built from a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa \times \{0,1\}^t$ and an authenticated encryption scheme AE with key space $\{0,1\}^\kappa$. For simplicity, we consider HtAE with the same key space $\{0,1\}^\kappa$. Its pseudocode and the graphical illustration of its encryption algorithm HtAE.Enc are shown in Figure 7.

To encrypt, HtAE first computes the hash function H with the input $(K, N, AD)$ to derive an symmetric key $K_e$ and a committing tag $T$, then initializes the underlying AE scheme AE with key $K_e$ and uses it to encrypt the input message $M$ with nonce $N$ and empty associated data, and finally outputs the derived AE ciphertext $C$ together with $T$. To decrypt, HtAE also computes the hash function H to derive an symmetric key $K_e$ and a committing tag $T'$, then checks if this tag $T'$ is equal to the input committing tag $T$ and outputs $\perp$ on failure; if the check passes, it uses AE to decrypt the input ciphertext $C$ under key $K_e$ with nonce $N$ and empty associated data, and finally outputs the decrypted message $M$.

We present the security guarantees of HtAE in the following theorems, with proofs in Appendix B.1∼B.4.

**Theorem 1.** *Let the underlying hash function of* HtAE *be modeled as a random oracle H. For any efficient adversary $\mathcal{A}$ making $q_e$ and $q_h$ queries respectively to oracles Enc and H, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{HtAE}}(\mathcal{A}) \leq q_e \mathbf{Adv}^{\text{ot-ind\$-cpa}}_{\mathsf{AE}}(\mathcal{B}) + \frac{q_h}{2^\kappa}.$$

**Theorem 2.** *Let the underlying hash function of* HtAE *be modeled as a random oracle H. For any efficient adversary $\mathcal{A}$ making $q_e$, $q_v$, $q_h$ queries respectively to oracles Enc, Ver, H, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{HtAE}}(\mathcal{A}) \leq (q_e + q_v) \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{AE}}(\mathcal{B}) + \frac{q_h}{2^\kappa}.$$

**Theorem 3.** *For any efficient adversary $\mathcal{A}$, there exists an efficient adversary $\mathcal{B}$ (that can be constructed from $\mathcal{A}$) such that*

$$\mathbf{Adv}^{\text{cmt}}_{\mathsf{HtAE}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{rcr}}_{\mathsf{H}}(\mathcal{B}).$$

*Remark 1.* The above theorem reduces CMT security of HtAE to right collision resistance of H to avoid idealized assumptions, but one can also easily prove it in the random oracle model. We follow the same style when analyzing CMT security of our other transforms.

| $\mathsf{AEaH.Enc}(K, N, AD, M)$ | $\mathsf{AEaH.Dec}(K, N, AD, (C, T))$ |
|---|---|
| $C \leftarrow \mathsf{AE.Enc}(K, N, \varepsilon, M)$ | $M \leftarrow \mathsf{AE.Dec}(K, N, \varepsilon, C)$ |
| $T \leftarrow \mathsf{H}(K, N, AD)$ | **if** $M = \perp \vee \mathsf{H}(K, N, AD) \neq T$ : |
| **return** $(C, T)$ | **return** $\perp$ |
| | **return** $M$ |

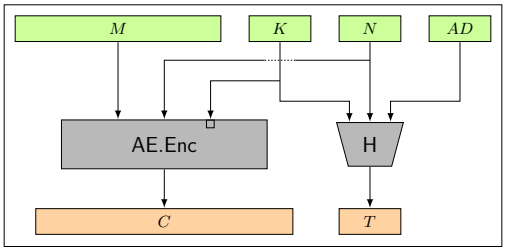


Fig. 8: The pseudocode (**left**) of AEaH transform and the graphical illustration (**right**) of its encryption algorithm.

**Theorem 4.** *Let the underlying hash function of* HtAE *be modeled as a random oracle $H$. For any efficient adversary $\mathcal{A}$ making $q_h$ random oracle queries, we have*

$$\mathbf{Adv}^{\mathrm{cdy\$}}_{\mathsf{HtAE}}(\mathcal{A}) \leq \frac{1}{|\mathcal{AD}|} \cdot \frac{q_h + 1}{2^\kappa} + \frac{q_h + 1}{2^t}.$$

*Remark 2.* As implied in the proof, if there is no associated data (i.e., $\mathcal{AD} = \varnothing$), the above security bound still holds without the $|\mathcal{AD}|$ term.

**Comparison with** CommitKeyΠ **and** HtE ∘ UtC**.** Among the existing transforms, CommitKeyΠ [1] and HtE ∘ UtC [2] follow a similar paradigm to our HtAE. They are also designed to first derive a *fresh* encryption key $K_e$ for each distinct nonce and then feed this $K_e$ to the underlying AE. As discussed below, our HtAE can be viewed as a simpler hash-based counterpart.

- Instead of computing a single hash, CommitKeyΠ uses two separate collision-resistant PRFs to derive the encryption key $K_e$ and the committing tag $T$, respectively. It is quite similar to our HtAE, as in practice collision-resistant PRFs can be instantiated with hash functions. However, CommitKeyΠ incurs one more primitive call (and hence is slower than HtAE) and does not ensure CMT security.
- HtE∘UtC improved CommitKeyΠ by first computing a hash and then calling a more efficient primitive called committing PRF. Nevertheless, this transform still makes two primitive calls to derive $K_e$ and $T$. As we will show later in Section 5, our HtAE outperforms HtE∘UtC even when the committing PRF is instantiated with the practical block-cipher-based construction proposed in [2]. More importantly, unlike hash functions, their practical instantiation is not yet standardized.

### 3.2 AE-and-Hash (AEaH)

Then, for the AE-and-Hash (AEaH) transform, we present the AEaH construction. AEaH is built from a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^t$ and an authenticated encryption scheme AE with key space $\{0,1\}^\kappa$. Again, for simplicity, we consider AEaH with the same key space $\{0,1\}^\kappa$. Its pseudocode and the graphical illustration of its encryption algorithm AEaH.Enc are shown in Figure 8.

To encrypt, AEaH performs two separate computations: (i) using the underlying AE scheme AE to encrypt the input message $M$ under the input key $K$ with the input nonce $N$ and empty associated data, to derive an AE ciphertext $C$, (ii) computing the hash function $\mathsf{H}(K, N, AD)$, to derive a committing tag $T$. Finally, it outputs $(C, T)$ as its ciphertext. To decrypt, AEaH performs two similar separate computations: (iii) using AE to decrypt the input ciphertext $C$ under the input key $K$ with the input nonce $N$ and empty associated data, to derive a decrypted message $M$, (iv) computing $\mathsf{H}(K, N, AD)$, to derive a committing tag $T'$. Then, it checks if AE decryption succeeds (i.e., $M \neq \perp$) and if $T'$ is equal to the input committing tag $T$, and finally it outputs $M$ if the check passes or outputs $\perp$ otherwise.

We note that computations (i) and (ii) are independent of each other, as are (iii) and (iv), so in practice they can be executed *in parallel*.

The security guarantees of AEaH are presented in the following theorems, with proofs in Appendix B.5∼B.8.

**Theorem 5.** *Let the underlying hash function of* AEaH *be modeled as a random oracle $H$. For any efficient adversary $\mathcal{A}$ making $q_e$ and $q_h$ queries respectively to oracles Enc and H, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathrm{ind\$\text{-}cpa}}_{\mathsf{AEaH}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\mathrm{ind\$\text{-}cpa}}_{\mathsf{AE}}(\mathcal{B}) + \frac{q_h}{2^\kappa},$$

*where $\mathcal{B}$ makes at most $q_e + 1$ encryption queries and its one extra encryption query (if any) is on a $\kappa$-bit message.*

**Theorem 6.** *Let the underlying hash function of* AEaH *be modeled as a random oracle H. For any efficient adversary $\mathcal{A}$ making $q_e$, $q_v$, $q_h$ queries respectively to oracles Enc, Ver, H, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{AEaH}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{AE}}(\mathcal{B}) + \frac{q_v}{2^t},$$

*where $\mathcal{B}$ makes $q_e$ encryption queries and at most $q_v + q_h$ verification queries.*

**Theorem 7.** *For any efficient adversary $\mathcal{A}$, there exists an efficient adversary $\mathcal{B}$ (that can be constructed from $\mathcal{A}$) such that*

$$\mathbf{Adv}^{\text{cmt}}_{\mathsf{AEaH}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{cr}}_{\mathsf{H}}(\mathcal{B}).$$

**Theorem 8.** *Let the underlying hash function of* AEaH *be modeled as a random oracle H and let $\tau$ be the ciphertext expansion of the underlying AE scheme* AE*. For any efficient adversary $\mathcal{A}$ making $q_h$ random oracle queries, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{cdy\$}}_{\mathsf{AEaH}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ot-ind\$-cpa}}_{\mathsf{AE}}(\mathcal{B}) + \frac{q_h + 1}{2^\tau} + \frac{q_h + 1}{2^t}.$$

**Comparison with** comPACT**.** The recently proposed comPACT [7] is the most similar existing transform to AEaH. In fact, it computes the AE ciphertext $C$ and H digest $T$ in the same way as in AEaH. However, comPACT uses the H digest $T$ as an encryption key rather than a committing tag. More precisely, it applies a block cipher under key $T$ to encrypt the authentication tag (or at most one last block) of the derived AE ciphertext $C$, denoted by $T_C$, to produce $T^*_C$; it then outputs $C$ with $T_C$ replaced by the modified tag $T^*_C$.

The extra tag encryption in comPACT is adopted to reduce ciphertext expansion; however, as mentioned in the Introduction, the maximum committing security of the resulting scheme is thus restricted to half the block size. In contrast, our AEaH can provide stronger committing security by choosing a hash function with large enough digest size. By construction, it is obvious that AEaH has higher efficiency than comPACT, and our AEaH is fully parallelizable.

### 3.3 Encrypt-then-Hash (EtH)

Finally, we present the EtH construction for the Encrypt-then-Hash transform. Here, we use "Encrypt" to emphasize that our EtH transform can be applied to both AE schemes and basic privacy-only encryption schemes. In this section, we adopt the syntax and security of a basic encryption scheme E to describe our construction and analyze the security results, but one can also easily adapt the construction and results to capture arbitrary AE schemes.

EtH is built from a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^t$ and a symmetric encryption scheme E with key space $\{0,1\}^\kappa$. Again, for simplicity, we consider EtH with the same key space $\{0,1\}^\kappa$. Its pseudocode and the graphical illustration of its encryption algorithm EtH.Enc are shown in Figure 9.

To encrypt, EtH first uses the underlying AE scheme E to encrypt the input message $M$ under the input key $K$ with the input nonce $N$, to derive an E ciphertext $C$; then it computes the hash function H with input $(K, N, AD, C)$, to derive a committing tag $T$; finally it outputs $(C, T)$ as its ciphertext. To decrypt, EtH first computes the hash function H to derive a committing tag $T'$, then checks if this tag $T'$ is equal to the input committing tag $T$ and outputs $\perp$ on failure; if the check passes, it uses E to decrypt the input ciphertext $C$ under key $K$ with nonce $N$ and outputs the decrypted message $M$.

We present the security guarantees of EtH in the following theorems, with proofs in Appendix B.9∼B.12. We note that our security results for EtH do *not* require the underlying basic encryption scheme E to be tidy.

**Theorem 9.** *Let the underlying hash function of* EtH *be modeled as a random oracle H. For any efficient adversary $\mathcal{A}$ making $q_e$ and $q_h$ queries respectively to oracles Enc and H, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{EtH}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{E}}(\mathcal{B}) + \frac{q_h}{2^\kappa},$$

*where $\mathcal{B}$ makes at most $q_e + 1$ encryption queries and its one extra encryption query (if any) is on a $\kappa$-bit message.*

| $\mathsf{EtH}.\mathsf{Enc}(K, N, AD, M)$ | $\mathsf{EtH}.\mathsf{Dec}(K, N, AD, (C, T))$ |
|---|---|
| $C \leftarrow \mathsf{E}.\mathsf{Enc}(K, N, M)$ | $T' \leftarrow \mathsf{H}(K, N, AD, C)$ |
| $T \leftarrow \mathsf{H}(K, N, AD, C)$ | **if** $T' \neq T$ : **return** $\perp$ |
| **return** $(C, T)$ | $M \leftarrow \mathsf{E}.\mathsf{Dec}(K, N, C)$ |
| | **return** $M$ |

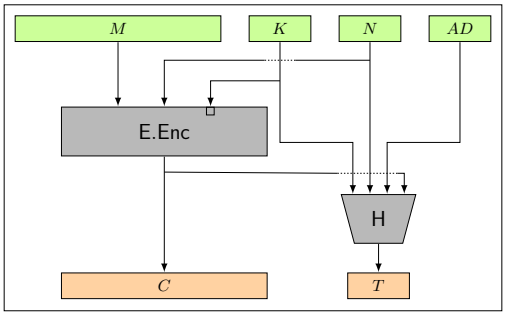


Fig. 9: The pseudocode (**left**) of EtH transform and the graphical illustration (**right**) of its encryption algorithm.

**Theorem 10.** *Let the underlying hash function of* EtH *be modeled as a random oracle H. For any efficient adversary $\mathcal{A}$ making $q_e$, $q_v$, $q_h$ queries respectively to oracles Enc, Ver, H, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{EtH}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{E}}(\mathcal{B}) + \frac{q_h}{2^\kappa} + \frac{q_v}{2^t},$$

*where $\mathcal{B}$ makes $q_e + 1$ encryption queries and its one extra encryption query is on a $\kappa$-bit message.*

*Remark 3.* The above theorem reduces the INT-CTXT security of EtH to the IND$-CPA security of E, because the underlying basic encryption scheme E may not have authenticity security. However, if our Encrypt-then-Hash transform is built from an authenticated encryption scheme AE, one can perform a proof (omitted here) very similar to that of Theorem 6 to reduce the INT-CTXT security of our transform to the INT-CTXT security of the underlying AE.

**Theorem 11.** *For any efficient adversary $\mathcal{A}$, there exists an efficient adversary $\mathcal{B}$ (that can be constructed from $\mathcal{A}$) such that*

$$\mathbf{Adv}^{\text{cmt}}_{\mathsf{EtH}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{cr}}_{\mathsf{H}}(\mathcal{B}).$$

**Theorem 12.** *Let the underlying hash function of* EtH *be modeled as a random oracle H. For any efficient adversary $\mathcal{A}$ making $q_h$ random oracle queries, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{cdy\$}}_{\mathsf{EtH}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{E}}(\mathcal{B}) + \frac{q_h + 1}{2^\kappa} + \frac{q_h + 1}{2^t},$$

*where $\mathcal{B}$ makes two encryption queries and one of them is on a $\kappa$-bit message.*

**Comparison with CTY.** Among the existing transforms, CTY [3] is the most similar to EtH. It also follows the Encrypt-then-Hash paradigm, but it is built from AE schemes that produce a separate authentication tag and hashes only this tag rather than the entire AE ciphertext.

With practical instantiations, CTY usually has better performance than EtH, even when the underlying encryption scheme of EtH is instantiated with basic privacy-only encryption. For example, as demonstrated by our implementation results in Section 5, CTY instantiated with AES-GCM runs faster than EtH instantiated with AES-CTR (i.e., AES in counter mode) for common message sizes, because hashing the entire ciphertext (as in EtH) is typically slower than MACing it within AE (as in CTY).

However, CTY is not a black-box transform and cannot be applied to basic privacy-only encryption schemes; our EtH, by contrast, can work with arbitrary AE and basic privacy-only encryption schemes, making it a more powerful generic transform. Additionally, when instantiated with basic encryption schemes that do not incur ciphertext expansion (e.g., with AES-CTR), our EtH exhibits the same compact ciphertext expansion as CTY.

### 3.4 Security Comparison

Now, we compare the security guarantees of our three basic transforms: HtAE, AEaH, EtH, when instantiated with the same or consistent component primitives.

**AE security.** All of our basic transforms achieve AE security.

To achieve IND$-CPA security, HtAE requires only *one-time* IND$-CPA security for its underlying AE, while AEaH and EtH require the regular IND$-CPA security. This is because, for each HtAE encryption with a distinct nonce, a fresh AE key $K_e$ is derived and used solely for encrypting that one message.

For INT-CTXT security, the guarantees for our basic transforms are similarly aligned, except for EtH with basic privacy-only encryption instantiation. In this case, as remarked for Theorem 10, the INT-CTXT security of EtH has to rely on the IND$-CPA security of the underlying basic encryption scheme E.

**Committing security.** All of our basic transforms achieve committing security.

To achieve CMT security, AEaH and EtH rely on collision resistance of the underlying hash function, while HtAE requires the stronger right collision resistance. As a result, to ensure the same level of CMT security, the hash function used in HtAE must have a larger digest size compared to the other two transforms, leading to more constraint choices for hash instantiations.

For CDY$ security, however, HtAE has a better security bound than the other two transforms. As shown in Theorem 4, its CDY$ security does not rely on the security of its underlying AE, which means HtAE can still ensure context-discovery security even if AE is broken. This is because revealing the AE key $K_e$ used in HtAE does not help with determining the input key $K$. In contrast, AEaH and EtH directly use the same input key $K$ for encryption, so breaking the underlying encryption scheme would reveal the input message, key and nonce. As discussed in the end of Section 2.2, CDY$ security is of particular interest when the associated data space $\mathcal{AD}$ has small size, and in this case breaking the encryption scheme compromises CDY$ security. Therefore, AEaH and EtH have to rely on the privacy security of the underlying encryption scheme to ensure context-discovery security; however, they require only very minimal one-time or two-time IND$-CPA security for privacy.

## 4 Advanced Transforms

In this section, we present our two *advanced* generic transforms. The first simpler transform adds committing security to an AE scheme while preserving its misuse-resistance security, while the second transform directly "lifts" a basic encryption scheme to a committing AE that is secure against nonce misuse.

In the following, we first present our advanced transforms and their security results individually, comparing the first with similar existing MRAE-preserving transforms. Then, we instantiate our transforms with consistent standardized component primitives and compare their security guarantees.

### 4.1 MRAE-Preserving Transform AEtH

For our MRAE-preserving transform, we present the AEtH construction, which follows the AE-then-Hash paradigm. AEtH is built from a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^t$ and an authenticated encryption scheme AE with key space $\{0,1\}^\kappa$. For simplicity, we consider AEtH with the same key space $\{0,1\}^\kappa$. Its pseudocode and the graphical illustration of its encryption algorithm AEtH.Enc are shown in Figure 10.

As shown in the figure, AEtH is very similar to our basic EtH transform (see Figure 9), except that the basic encryption scheme E in EtH is replaced with AE in AEtH and only the last $\tau$ bits of the AE ciphertext is fed into the hash function. Here $\tau$ refers to the ciphertext expansion of the underlying AE, but it can be adjusted for stronger security when the length of the AE ciphertext is guaranteed to have more than $\tau$ bits.

To encrypt, AEtH first uses the underlying AE scheme AE to encrypt the input message $M$ under the input key $K$ with the input nonce $N$ and associated data $AD$, to derive an AE ciphertext $C$; then it computes the hash function H with input $(K, N, AD, C|_\tau)$, to derive a committing tag $T$; finally it outputs $(C, T)$ as its ciphertext. To decrypt, AEtH first computes the hash function H to derive a committing tag $T'$, then checks if this tag $T'$ is equal to the input committing tag $T$ and outputs $\perp$ on failure; if the check passes, it uses AE to decrypt the input ciphertext $C$ under key $K$ with the input nonce $N$ and associated data $AD$, and outputs the decrypted message $M$.

The security guarantees of AEtH are presented in the following theorems, with proofs in Appendix B.13∼B.16.

**Theorem 13.** *Let the underlying hash function of* AEtH *be modeled as a random oracle H. For any efficient adversary $\mathcal{A}$ making $q_e$ and $q_h$ queries respectively to oracles Enc and H, there exists an efficient*

| $\mathsf{AEtH.Enc}(K, N, AD, M)$ | $\mathsf{AEtH.Dec}(K, N, AD, (C, T))$ |
|---|---|
| $C \leftarrow \mathsf{AE.Enc}(K, N, AD, M)$ | $T' \leftarrow \mathsf{H}(K, N, AD, C\|_\tau)$ |
| $T \leftarrow \mathsf{H}(K, N, AD, C\|_\tau)$ | **if** $T' \neq T$ : **return** $\perp$ |
| **return** $(C, T)$ | $M \leftarrow \mathsf{AE.Dec}(K, N, AD, C)$ |
| | **return** $M$ |

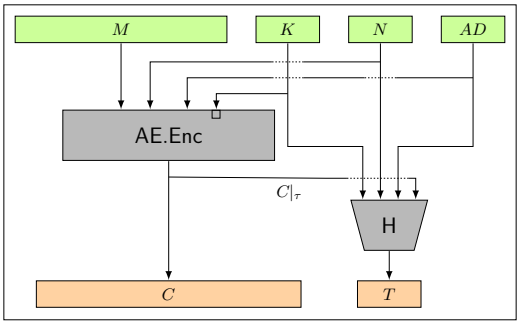


Fig. 10: The pseudocode (**left**) of AEtH transform and the graphical illustration (**right**) of its encryption algorithm.

*adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{mr-ind\$-cpa}}_{\mathsf{AEtH}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\text{mr-ind\$-cpa}}_{\mathsf{AE}}(\mathcal{B}) + \frac{q_e^2}{2^{\tau+1}} + \frac{q_h}{2^\kappa},$$

*where $\mathcal{B}$ makes at most $q_e + 1$ encryption queries and its one extra encryption query (if any) is on a $\kappa$-bit message.*

*Remark 4.* We remark that the $\frac{q_e^2}{2^{\tau+1}}$ term is suboptimal with respect to the allowed number of encryptions, e.g., it restricts $q_e \leq 2^{48.5}$ for $\tau = 128$ (the typical expansion size) with a $2^{-32}$ advantage. This, however, could be circumvented by (i) padding all messages with $l$ bits before encryption and then feeding $\tau + l$ bits of the derived AE ciphertext into H, or (ii) excluding short messages from the message space, which ensures sufficiently long ciphertexts. We note that some other advanced transforms such as CTX [9] and PACT [7] also suffer from the above suboptimal term.[6] The above circumvention ideas can be applied to CTX, but fail for PACT, as the expansion size $\tau$ of PACT is fixed to the block size by design due to the added block cipher encryption.

**Theorem 14.** *Let the underlying hash function of* AEtH *be modeled as a random oracle $H$. For any efficient adversary $\mathcal{A}$ making $q_e$, $q_v$, $q_h$ queries respectively to oracles Enc, Ver, H, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{mr-int-ctxt}}_{\mathsf{AEtH}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\text{mr-int-ctxt}}_{\mathsf{AE}}(\mathcal{B}),$$

*where $\mathcal{B}$ makes $q_e$ encryption queries and at most $q_v + q_h$ verification queries.*

**Theorem 15.** *For any efficient adversary $\mathcal{A}$, there exists an efficient adversary $\mathcal{B}$ (that can be constructed from $\mathcal{A}$) such that*

$$\mathbf{Adv}^{\text{cmt}}_{\mathsf{AEtH}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{cr}}_{\mathsf{H}}(\mathcal{B}).$$

**Theorem 16.** *Let the underlying hash function of* AEtH *be modeled as a random oracle $H$ and let $\tau$ be the ciphertext expansion of the underlying AE scheme* AE*. For any efficient adversary $\mathcal{A}$ making $q_h$ random oracle queries, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{cdy\$}}_{\mathsf{AEtH}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ot-ind\$-cpa}}_{\mathsf{AE}}(\mathcal{B}) + \frac{q_h + 1}{2^\tau} + \frac{q_h + 1}{2^t}.$$

**Comparison with existing MRAE-preserving transforms.** There are three existing transforms that also preserve MRAE security: HtE ∘ RtC [2], CTX [9] and PACT [7]. We compare them to our AEtH as follows:

– The HtE ∘ RtC transform follows the same approach as HtE ∘ UtC, with RtC applied to a misuse-resistant AE scheme rather than a basic one. As a result, HtE ∘ RtC still suffers from the expensive cost to re-key the underlying AE per distinct nonce, resulting in worse performance than our AEtH. Furthermore, like HtE ∘ UtC, the HtE ∘ RtC transform also makes use of the non-standard committing PRF primitive.

[6] The term did not appear explicitly in the original work that presents CTX [9], since the authors proved only AE security in the basic nonce-respecting setting. However, due to the very similar design of CTX and our AEtH, our security proof can be directly applied to CTX to derive almost identical bounds.

| chaSIV.Enc$(K, N, AD, M)$ | chaSIV.Dec$(K, N, AD, (C, T))$ |
|---|---|
| $K_e \leftarrow \mathsf{F}(K, N)$ | $K_e \leftarrow \mathsf{F}(K, N)$ |
| $T \leftarrow \mathsf{H}(K, N, AD, M)$ | $N_e \leftarrow T\|_r$ |
| $N_e \leftarrow T\|_r$ | $M \leftarrow \mathsf{E.Dec}(K_e, N_e, C)$ |
| $C \leftarrow \mathsf{E.Enc}(K_e, N_e, M)$ | **if** $M = \perp$ : **return** $\perp$ |
| **return** $(C, T)$ | $T' \leftarrow \mathsf{H}(K, N, AD, M)$ |
| | **if** $T' \neq T$ : **return** $\perp$ |
| | **return** $M$ |

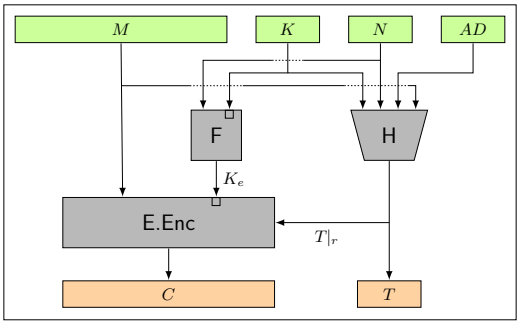


Fig. 11: The pseudocode (**left**) of chaSIV[F, H, E] transform and the graphical illustration (**right**) of its encryption algorithm.

- Our AEtH can be viewed as a black-box generalization of the CTX transform, where CTX is restricted to hash a separate authentication tag of the underlying AE ciphertext rather than any part of it. The performance of CTX and AEtH are hence almost the same, with CTX saving $\tau$-bit ciphertext output.
- The black-box transform PACT [7] is the heavier variant of comPACT, aiming to preserve misuse-resistance security. The only difference between PACT and comPACT is that, in PACT, the underlying AE inputs $AD$ rather than an empty string as associated data, where the same difference also applies to CTX and CTY. As with comPACT, PACT applies an additional block cipher encryption after hashing, leading to slower performance than our AEtH.

### 4.2 MRAE-Lifting Transform chaSIV

Finally, we present our MRAE-lifting transform named chaSIV, which stands for ***c**ommitting **h**ash-based SIV*. Like our EtH transform, this advanced chaSIV can also be applied to both AE schemes and basic privacy-only encryption schemes. In this section, we adopt the syntax and security of a basic encryption scheme E to describe our construction and analyze the security results, but one can easily adapt the construction and results to capture arbitrary AE schemes.

As its name indicates, chaSIV follows the synthetic IV (SIV) paradigm [30]: it just replaces the MAC in SIV with a hash function and adds a PRF to re-key the underlying encryption scheme per distinct nonce. As a result, in addition to hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^t$ and an encryption scheme E with key space $\{0,1\}^\kappa$, chaSIV also makes use of a PRF $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^r \to \{0,1\}^\kappa$. For chaSIV to be properly defined, we require $t \geq r$, i.e., the committing tag size must be at least the nonce size, which holds for all natural and practical choices of $t$ and $r$. As with our previous transforms, we consider chaSIV with the same key space $\{0,1\}^\kappa$. Its pseudocode and the graphical illustration of its encryption algorithm chaSIV.Enc are shown in Figure 11.

To encrypt, chaSIV first uses PRF F under the input key $K$ to derive the encryption key $K_e$ for the underlying E. Then, it hashes the whole context with H to derive a committing tag $T$. The last $r$ bits of $T$ are then used as a nonce for E to encrypt the input message $M$ and get ciphertext $C$. Finally, it outputs $(C, T)$ as its ciphertext. To decrypt, chaSIV derives $K_e$ as is done in encryption and takes the last $r$ bits of the input committing tag $T$ as nonce $N_e$. Then, it uses E to decrypt the input ciphertext $C$ under the key $K_e$ with nonce $N_e$. If decryption fails, it outputs $\perp$; otherwise, chaSIV gets the decrypted message $M$ and computes the hash function H over $(K, N, AD, M)$ to derive $T'$. Finally, it checks if $T' = T$ and outputs $\perp$ on failure; otherwise, it outputs $M$.

The security guarantees of chaSIV are presented in the following theorems, with proofs in Appendix B.17∼B.20.

**Theorem 17.** *Let the underlying hash function of* chaSIV *be modeled as a random oracle H. For any efficient adversary $\mathcal{A}$ making $q_h$ queries to oracle H and querying oracle Enc with $Q_e$ distinct nonces, where the $i$-th nonce is repeated $R_i$ times for $i \in \{1, \ldots, Q_e\}$, there exist efficient adversaries $\mathcal{B}$ and $\mathcal{C}$ such that*

$$\mathbf{Adv}^{\text{mr-ind\$-cpa}}_{\mathsf{chaSIV}}(\mathcal{A}) \leq Q_e \cdot \mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{E}}(\mathcal{B}) + 2 \cdot \mathbf{Adv}^{\text{prf}}_{\mathsf{F}}(\mathcal{C}) + \frac{\sum_{i=1}^{Q_e} R_i^2}{2^r} + \frac{q_h}{2^\kappa},$$

*where $\mathcal{B}$ makes at most $\max_{i \in \{1,\ldots,Q_e\}}\{R_i\}$ encryption queries and $\mathcal{C}$ makes at most $Q_e + 1$ PRF oracle queries.*

*Remark 5.* Note that here we do not have the suboptimal quadratic term that appeared in Theorem 13. Instead, we have $\frac{\sum_{i=1}^{Q_e} R_i^2}{2^r}$, which typically allows for more encryptions for a common nonce size, e.g., $r = 96$, since in practice the number of repetitions is usually not high, e.g., hundreds for random nonces.

**Theorem 18.** *Let the underlying hash function of* chaSIV *be modeled as a random oracle H. For any efficient adversary $\mathcal{A}$ making $q_h, q_v$ queries respectively to oracles H, Ver and querying oracles Enc, Ver with a total of Q distinct nonces, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{mr-int-ctxt}}_{\mathsf{chaSIV}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prf}}_{\mathsf{F}}(\mathcal{B}) + \frac{q_h}{2^\kappa} + \frac{q_v}{2^t},$$

*where $\mathcal{B}$ makes at most $Q + 1$ PRF oracle queries.*

*Remark 6.* The above theorem requires the underlying encryption scheme E to be *tidy*. In our entire work, this is the only theorem that relies on tidyness.

**Theorem 19.** *For any efficient adversary $\mathcal{A}$, there exists an efficient adversary $\mathcal{B}$ (that can be constructed from $\mathcal{A}$) such that*

$$\mathbf{Adv}^{\text{cmt}}_{\mathsf{chaSIV}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{cr}}_{\mathsf{H}}(\mathcal{B}).$$

**Theorem 20.** *Let the underlying hash function of* chaSIV *be modeled as a random oracle H. For any efficient adversary $\mathcal{A}$ making $q_h$ random oracle queries, there exists an efficient adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{cdy\$}}_{\mathsf{chaSIV}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prf}}_{\mathsf{F}}(\mathcal{B}) + \frac{q_h + 1}{2^\kappa} + \frac{q_h + 1}{2^t},$$

*where $\mathcal{B}$ makes only one PRF oracle query.*

### 4.3 Security Comparison

We now compare the security guarantees of our advanced transforms, AEtH and chaSIV, when instantiated with consistent component primitives. Since they respectively employ misuse-resistant AE and basic encryption schemes, which typically have quite different concrete security bounds, we focus on specific consistent primitives when comparing their security. In particular, we consider the misuse-resistant AE AE instantiated with AES-GCM-SIV, basic encryption scheme E instantiated with AES-CTR, and the PRF F of chaSIV instantiated with AES.

**MRAE security.** Both of our advanced transforms achieve MRAE security. We next compare their concrete security under the aforementioned instantiations.

As proved in [18], the dominating terms of the MRAE security bound for AES-GCM-SIV are $\frac{3Q}{2^{96}}$, $\frac{QB_{\max}^2}{2^{129}}$, $\frac{l_{\max}\sum_{i=1}^{Q} R_i^2}{2^{126}}$, where $Q$ is the number of distinct nonces queried across *both* encryption and decryption/verification oracles, $l_{\max}$ is the maximum message length in blocks, and $B_{\max}$ is the maximum number of blocks queried per nonce (i.e. $B_{\max} \leq \max_i\{R_i\} \cdot l_{\max}$). Plugging these terms into Theorem 13 (with $\tau = 128$) and Theorem 14 yields the dominating terms of the MRAE security bound for our AEtH: $\frac{3Q}{2^{95}}$, $\frac{QB_{\max}^2}{2^{128}}$, $\frac{l_{\max}\sum_{i=1}^{Q} R_i^2}{2^{125}}$, $\frac{q_e^2}{2^{129}}$.

Then, for $B$ queried blocks, the dominating term of both the IND$-CPA security bound for AES-CTR and the PRF security bound for AES is $\frac{B^2}{2^{128}}$, as shown in [20,28]. Plugging these terms into Theorem 17 (with $r = 96$) and Theorem 18 yields the dominating terms of the MRAE security bound for our chaSIV: $\frac{Q_e B_{\max}^2}{2^{128}}$, $\frac{\sum_{i=1}^{Q_e} R_i^2}{2^{96}}$, $\frac{3(Q+1)^2}{2^{128}}$.

By comparing the above dominating terms, we observe that AEtH and chaSIV achieve very similar concrete security.[7] When nonce repetitions are considered, $\frac{3(Q+1)^2}{2^{128}}$ is better than $\frac{q_e^2}{2^{129}}$, e.g., with at least 3 repetitions per nonce on average; this aligns with our remarks for Theorem 13 and Theorem 17. Therefore, chaSIV has only one slightly worse term compared to AEtH: for $l_{\max} < 2^{29}$ we could have $\frac{\sum_{i=1}^{Q_e} R_i^2}{2^{96}} > \frac{l_{\max}\sum_{i=1}^{Q} R_i^2}{2^{125}}$. However, for common nonce repetitions in practice, e.g., at most $2^{10}$ on average, $\frac{\sum_{i=1}^{Q_e} R_i^2}{2^{96}}$ is typically good enough, allowing $Q_e = 2^{44}$ with a $2^{-32}$ advantage, very close to the $Q$ limit imposed by the $\frac{3(Q+1)^2}{2^{128}}$ term.

[7] We remark that actually the existing MRAE-preserving transforms HtE ∘ RtC [2], CTX [9], PACT [7] all show very similar MRAE concrete security, as their security bounds are mostly dominated by the MRAE security of the underlying AE.

**Committing security.** Both of our advanced transforms achieve committing security. Their CMT security bounds are identical, both relying on the collision resistance of the underlying hash function. They offer very similar CDY$ security, with AEtH relying on one-time IND$-CPA security of its underlying AE, while chaSIV relies on PRF security (with a single query) of its underlying F; in concrete instantiations, these IND$-CPA and PRF bounds become minor. The dominating terms for CDY$ security of AEtH is determined by the sizes of the committing tag and ciphertext expansion, while those for chaSIV are similarly determined by the sizes of the committing tag and the key.

## 5 Performance Evaluation

Finally, we evaluate the performance of our basic and advanced transforms, along with relevant existing transforms, both in theory and through implementations.

**Theoretical evaluation.** For theoretical performance evaluation, we focus on the following factors of the considered transforms: ciphertext expansion, number of cryptographic passes over specific targets, key consistency across encryptions/decryptions, and parallelizability. We also include the minimal assumptions made by the transforms on the component encryption primitive to capture their primitive applicability. The results are summarized in Table 2.

As shown in the table, HtE ∘ UtC exhibits the worst efficiency among the AE-based basic transforms because it involves more computation passes and, more crucially, re-keys its AE component for each encryption/decryption. Similarly, HtE ∘ RtC runs the slowest among the MRAE-preserving advanced transforms.

To reduce ciphertext expansion, CTY and comPACT introduce an additional cryptographic pass over $T_C$, the separate authentication tag (or the last block) of the derived AE ciphertext. This extra pass, besides requiring more computation, renders such transforms non-parallelizable. Furthermore, to support black-box transforms, comPACT encrypts $T_C$ rather than hashing it, using a non-fixed key. Similar operations are performed by their advanced transform variants, CTX and PACT, but the cryptographic pass over $T_C$ is now employed by all MRAE-preserving transforms to achieve misuse-resistant security. This makes all such MRAE-preserving transforms non-parallelizable.

As motivated in the Introduction, our transforms prioritize black-boxness and design simplicity over minimizing ciphertext expansion. As shown in Table 2, when ciphertext expansion is not a primary concern, our AEaH exhibits the best performance among the AE-based basic transforms; additionally, AEaH is fully parallelizable, making it even more attractive. Similarly, our AEtH, along with CTX, outperforms other MRAE-preserving advanced transforms.

Among the transforms presented in the table, our EtH and chaSIV are the only ones that support basic privacy-only encryption schemes, and using these schemes results in compact ciphertext expansion. We observe that, when instantiated with AE, EtH guarantees the same security as AEaH but exhibits strictly worse performance. Therefore, EtH is primarily designed to accommodate basic encryption schemes. However, it is useful to apply chaSIV to both basic and authenticated encryption schemes, as AE schemes may not be misuse-resistant and hence cannot be accommodated by MRAE-preserving transforms. The performance of EtH and chaSIV is compared to other transforms through implementations, as we will show shortly.

Finally, we note that while the primary standardized misuse-resistant AE, AES-GCM-SIV, requires re-keying across distinct nonces, some schemes do support key consistency. For example, Deoxys-II [22], the winner of the CAESAR competition for the *defense-in-depth* category, is one such scheme. This suggests that advanced fixed-key transforms, e.g., CTX and our AEtH, may offer potential performance benefits when used with suitable AE schemes.

**Implementation results.** To assess the practical efficiency of our transforms and relevant others, we implemented their encryption algorithms using both OpenSSL (version 3.3.2) and wolfSSL (version 5.7.0). For basic transforms we used AES-GCM speed (CPU cycles per byte) as a benchmark, and for advanced transforms we used AES-GCM-SIV (resp. AES-SIV) in OpenSSL (resp. wolfSSL). Performance is tested with message lengths ranging from 16 to 2048 bytes, using a 5-byte associated data that matches the length of the authenticated but unencrypted part of the TLS 1.3 header. Further testing details are presented in Appendix A. The results are presented in Figure 12 for basic transforms and Figure 13 for advanced ones.

For all AE-based transforms, we instantiate the basic (unique-nonce) AE with AES-GCM; the misuse-resistant AE is instantiated with AES-GCM-SIV in OpenSSL and AES-SIV in wolfSSL, as the latter is the only misuse-resistant AE available in wolfSSL. The basic privacy-only scheme E in EtH and chaSIV is

| Construction | Min. Assm. on Enc. Primitive | Ciphertext Expansion | Passes over $(K, N, AD, M, C, T_C)$ | Fixed Key | Parallel. |
|---|---|---|---|---|---|
| HtE ∘ UtC [2] | AE-secure | $\tau + t$ | $(2, 2, 1, 1, 0, 0)$ | ✗ | ✗ |
| CTY [3] | AE-secure | $t$ | $(1, 1, 1, 1, 0, 1)$ | ✓ | ✗ |
| comPACT [7] | AE-secure | $\tau$ | $(1, 1, 1, 1, 0, 1)$ | (✗) | (✗) |
| HtAE (this work) | AE-secure | $\tau + t$ | $(1, 1, 1, 1, 0, 0)$ | ✗ | ✗ |
| AEaH (this work) | AE-secure | $\tau + t$ | $(1, 1, 1, 1, 0, 0)$ | ✓ | ✓ |
| EtH (this work) | IND$-CPA-secure | $t$ | $(1, 1, 1, 1, 1, 0)$ | ✓ | ✗ |
| HtE ∘ RtC [2] | MRAE-secure | $\tau + t$ | $(2, 2, 1, 1, 0, 1)$ | ✗ | ✗ |
| CTX [3] | MRAE-secure | $t$ | $(1, 1, 2, 1, 0, 1)$ | ✓ | ✗ |
| PACT [7] | MRAE-secure | $\tau$ | $(1, 1, 2, 1, 0, 1)$ | (✗) | (✗) |
| AEtH (this work) | MRAE-secure | $\tau + t$ | $(1, 1, 2, 1, 0, 1)$ | ✓ | ✗ |
| chaSIV (this work) | IND$-CPA-secure | $t$ | $(2, 2, 1, 2, 0, 0)$ | ✗ | ✗ |

Table 2: Theoretical performance evaluation of our generic transforms and relevant others. Ciphertext expansion is measured in terms of bit length, where $\tau$ is the ciphertext expansion of the component encryption scheme and $t$ is the size of the committing tag. For the ciphertext $C$ derived from the component encryption scheme, let $T_C$ denote its separate authentication tag (if any) or its last block. The (✗) symbol indicates partial satisfaction of the key consistency (as shown in the "Fixed Key" column) or parallelizability properties.

instantiated with AES-CTR in OpenSSL and AES-GCM in wolfSSL, as AES-CTR is significantly slower than AES-GCM in wolfSSL. All encryption schemes use a 128-bit key, i.e., $\kappa = 128$. For HtE∘UtC/HtE∘RtC, we instantiate their underlying committing PRF with the CX scheme proposed and suggested by the authors in [2]. The block ciphers used in CX and comPACT/PACT are instantiated with AES under a 128-bit key. The PRF F in chaSIV is also instantiated with AES under a 128-bit key; if the input nonce is shorter than the block size, it is automatically padded by the libraries. The hash functions are instantiated with truncated SHA-512 in all implemented transforms, with a 128- or 256-bit output depending on the transform.

As shown in Figure 12, our implementation results demonstrate that AEaH has the highest efficiency among basic transforms, which matches our theoretical discussions. For all tested message lengths, the overhead of AEaH is no more than twice the benchmark. Furthermore, we note that the AEaH transform is implemented *sequentially*, so we expect that an implementation fully utilizing its parallelizability could perform even better. For EtH, one may notice the sudden increase in its efficiency overhead. This is

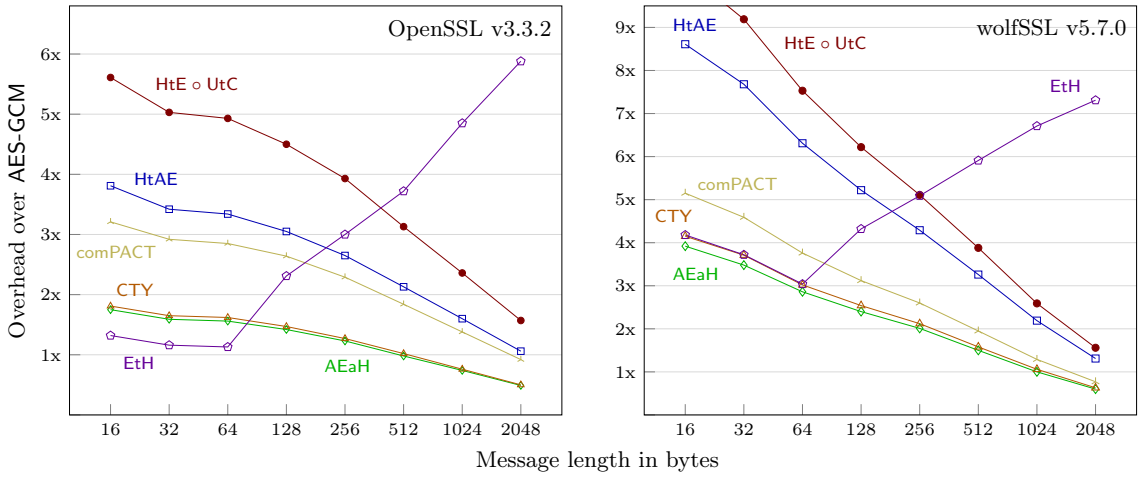


Fig. 12: Encryption speed overhead of basic transforms compared to AES-GCM. Smaller is better.

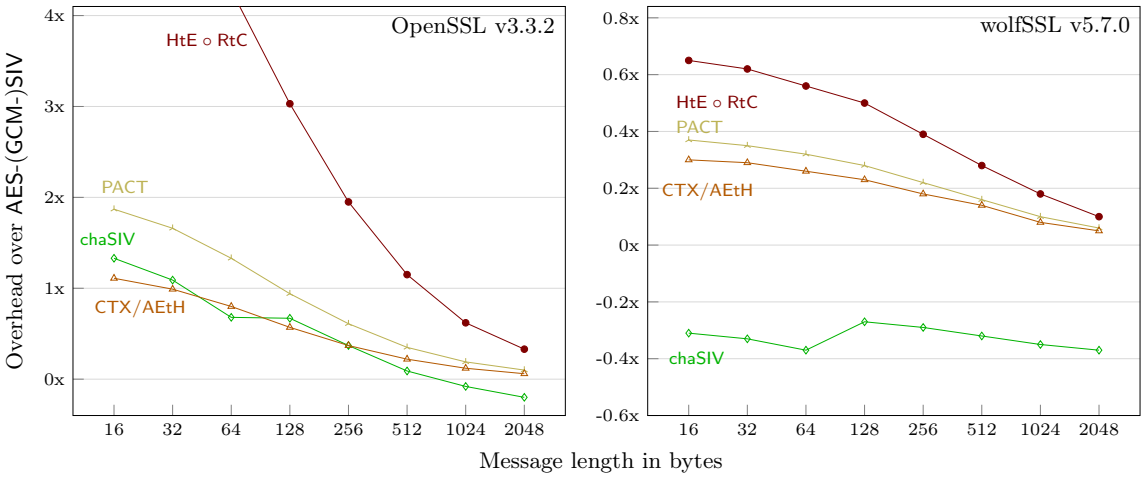


Fig. 13: Encryption speed overhead of advanced transforms compared to AES-GCM-SIV in OpenSSL and AES-SIV in wolfSSL. Smaller is better.

because for message length of 79 bytes the hash function input reaches 112 bytes,[8] which triggers a doubling of the padded input processed by SHA-512, causing the hashing cost to surpass the underlying MACing cost of the AE component in other transforms. However, for short messages (e.g., less than 64 bytes), EtH shows the highest efficiency.

For advanced transforms, as shown in Figure 13, our results demonstrate that AEtH/CTX and chaSIV achieve the best performance across all tested message sizes. Their overhead is within 1.5 times the benchmark in both libraries. In wolfSSL, due to the slow AES-SIV implementation, we notice that chaSIV, which does not rely on AES-SIV, outperforms all other transforms by a large margin. In OpenSSL, chaSIV outperforms AEtH/CTX for messages longer than about 360 bytes, and even surpasses the benchmark AES-GCM-SIV for messages exceeding 1024 bytes. The results show that our chaSIV not only is the first committing MRAE-lifting transform but also has great practical potential.

**Implementation benefits of our transforms.** During our implementation process, we found that our transforms are very easy to implement. If the practitioners are familiar with the cryptographic library they use, implementing our transforms would take just a couple of lines of code per functionality (e.g., encryption algorithm). Unfortunately, this is not the case for some of the other transforms. For example, implementing the decryption algorithm of CTY is tricky and inefficient (e.g., in OpenSSL) or even impossible (e.g., in wolfSSL) using the standard API functions in commonly used cryptographic libraries. This is because the API functions for AES-GCM decryption are not designed for (or do not allow at all) decrypting only the core part of the ciphertext (i.e., without the authentication tag) and recalculating the original authentication tag, which is needed for a final verification check in CTY.

## 6 Conclusion

Our work explored hash-based black-box generic transforms to add strong committing security to authenticated encryption schemes. To this end, we proposed three basic transforms and two advanced transforms, all exhibiting a simple design and strong security. As part of this effort, we developed the first generic transform that directly elevates a basic encryption scheme to a committing and misuse-resistant authenticated encryption scheme. By focusing on a straightforward design and relying crucially on efficient and well-standardized hash functions, we hope that our transforms will promote the adoption and deployment of committing authenticated encryption.

**Acknowledgments.** Shan Chen is funded by the research start-up grant from the Southern University of Science and Technology. Vukašin Karadžić is funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1119 – 236615297.

[8] Apart from the 79 message bytes, there is a 16-byte key, 12-byte nonce and 5-byte additional data, all of which sums up to 112 bytes.

## References


1. Albertini, A., Duong, T., Gueron, S., Kölbl, S., Luykx, A., Schmieg, S.: How to abuse and fix authenticated encryption without key commitment. In: Butler, K.R.B., Thomas, K. (eds.) USENIX Security 2022: 31st USENIX Security Symposium. pp. 3291–3308. USENIX Association, Boston, MA, USA (Aug 10–12, 2022)
2. Bellare, M., Hoang, V.T.: Efficient schemes for committing authenticated encryption. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022, Part II. Lecture Notes in Computer Science, vol. 13276, pp. 845–875. Springer, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-07085-3_29
3. Bellare, M., Hoang, V.T.: Succinctly-committing authenticated encryption. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology – CRYPTO 2024, Part IV. Lecture Notes in Computer Science, vol. 14923, pp. 305–339. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68385-5_10
4. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: ASIACRYPT 2000. pp. 531–545. Springer (2000)
5. Bernstein, D.J.: The Poly1305-AES message-authentication code. In: International workshop on fast software encryption. pp. 32–49. Springer (2005)
6. Bernstein, D.J., et al.: ChaCha, a variant of Salsa20. In: Workshop record of SASC. vol. 8, pp. 3–5. Citeseer (2008)
7. Bhattacharjee, A., Bhaumik, R., Dhar, C.: Universal context commitment without ciphertext expansion. Cryptology ePrint Archive, Report 2024/1382 (2024), https://eprint.iacr.org/2024/1382
8. Bhaumik, R., Chakraborty, B., Choi, W., Dutta, A., Govinden, J., Shen, Y.: The committing security of MACs with applications to generic composition. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology – CRYPTO 2024, Part IV. Lecture Notes in Computer Science, vol. 14923, pp. 425–462. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68385-5_14
9. Chan, J., Rogaway, P.: On committing authenticated-encryption. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) ESORICS 2022: 27th European Symposium on Research in Computer Security, Part II. Lecture Notes in Computer Science, vol. 13555, pp. 275–294. Springer, Copenhagen, Denmark (Sep 26–30, 2022). https://doi.org/10.1007/978-3-031-17146-8_14
10. Chen, Y.L., Davidson, M., Dworkin, M., Kang, J., Kelsey, J., Sasaki, Y., Sönmez Turan, M., Chang, D., Mouha, N., Thompson, A.: Proposal of requirements for an Accordion mode: Discussion draft for the NIST Accordion mode workshop 2024 (2024)
11. Dodis, Y., Grubbs, P., Ristenpart, T., Woodage, J.: Fast message franking: From invisible salamanders to encryptment. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part I. Lecture Notes in Computer Science, vol. 10991, pp. 155–186. Springer, Santa Barbara, CA, USA (Aug 19–23, 2018). https://doi.org/10.1007/978-3-319-96884-1_6
12. Dworkin, M.J.: Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC (2007)
13. Farshim, P., Orlandi, C., Roşie, R.: Security of symmetric primitives under incorrect usage of keys. IACR Transactions on Symmetric Cryptology **2017**(1), 449–473 (2017). https://doi.org/10.13154/tosc.v2017.i1.449-473
14. Fischlin, M., Mittelbach, A.: An overview of the hybrid argument. Cryptology ePrint Archive, Paper 2021/088 (2021), https://eprint.iacr.org/2021/088, https://eprint.iacr.org/2021/088
15. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences **28**(2), 270–299 (1984)
16. Grubbs, P., Lu, J., Ristenpart, T.: Message franking via committing authenticated encryption. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 66–97. Springer, Santa Barbara, CA, USA (Aug 20–24, 2017). https://doi.org/10.1007/978-3-319-63697-9_3
17. Gueron, S., Langley, A., Lindell, Y.: AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption. RFC 8452 (Apr 2019). https://doi.org/10.17487/RFC8452, https://www.rfc-editor.org/info/rfc8452
18. Gueron, S., Lindell, Y.: Better bounds for block cipher modes of operation via nonce-based key derivation. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1019–1036 (2017)
19. Harkins, D.: Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES). RFC 5297 (Oct 2008). https://doi.org/10.17487/RFC5297, https://www.rfc-editor.org/info/rfc5297
20. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Proceedings of the twenty-first annual ACM symposium on theory of computing. pp. 44–61 (1989)
21. wolfSSL Inc.: Case studies, https://www.wolfssl.com/docs/case-studies/. Accessed on 28.01.2025
22. Jean, J., Nikolic, I., Peyrin, T., Seurin, Y.: The deoxys AEAD family. Journal of Cryptology **34**(3), 31 (Jul 2021). https://doi.org/10.1007/s00145-021-09397-w
23. Len, J., Grubbs, P., Ristenpart, T.: Partitioning oracle attacks. In: USENIX Security 2021. pp. 195–212. USENIX Association (2021)

24. Menda, S., Len, J., Grubbs, P., Ristenpart, T.: Context discovery and commitment attacks - how to break CCM, EAX, SIV, and more. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 379–407. Springer, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30634-1_13
25. Naito, Y., Sasaki, Y., Sugawara, T.: Kivr: Committing authenticated encryption using redundancy and application to GCM, CCM, and more. In: International Conference on Applied Cryptography and Network Security. pp. 318–347. Springer (2024)
26. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering generic composition. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 257–274. Springer, Copenhagen, Denmark (May 11–15, 2014). https://doi.org/10.1007/978-3-642-55220-5_15
27. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM CCS 2002: 9th Conference on Computer and Communications Security. pp. 98–107. ACM Press, Washington, DC, USA (Nov 18–22, 2002). https://doi.org/10.1145/586110.586125
28. Rogaway, P.: Nonce-based symmetric encryption. In: International Workshop on Fast Software Encryption. pp. 348–358. Springer (2004)
29. Rogaway, P., Bellare, M., Black, J.: OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Transactions on Information and System Security (TISSEC) **6**(3), 365–403 (2003)
30. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 373–390. Springer (2006)
31. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), https://eprint.iacr.org/2004/332
32. Sönmez Turan, M., McKay, K., Chang, D., Kang, J., Kelsey, J.: Ascon-based lightweight cryptography standards for constrained devices: Authenticated encryption, hash, and extendable output functions. Tech. rep., National Institute of Standards and Technology (2024)
33. Struck, P., Weishäupl, M.: Constructing committing and leakage-resilient authenticated encryption. IACR Transactions on Symmetric Cryptology **2024**(1), 497–528 (2024). https://doi.org/10.46586/tosc.v2024.i1.497-528
34. Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC (CCM). Tech. rep. (2003)

## A Performance Testing Details

For each tested message length (ranging from 16 bytes to 2048 bytes) and every tested transform, we start the experiment by running the encryption 500 times to "warm-up" the cache, and then run 1000 encryptions and measure the average number of cycles per byte needed for one encryption call. We ran the experiment thirty times, discarded the five best and worst results and then calculated the average number of cycles needed over the remaining twenty runs. The resulting average serves as the final performance cost. As a benchmark for comparison, we measure the performance of AES-GCM for basic transforms, and AES-GCM-SIV/AES-SIV for advanced transforms in the same way described above. The tests are run on a notebook using Intel Core i5-8265U CPU (Skylake microarchitecture), with the base frequency of 1.6GHz and the hyper-threading, frequency scaling and turbo mode functionalities disabled.

## B Security Proofs

Our proofs follow the *game-based technique* (see [31] for a tutorial) and some proofs require a *hybrid argument* (see [14] for a tutorial). In the proofs, when the hash function $\mathsf{H}$ is modeled as a random oracle H, the random oracle H is instantiated via *lazy sampling*, i.e., an independent random output is sampled for each distinct H query and the same output is returned for the same query.

### B.1 Proof of Theorem 1 (IND$-CPA Security of HtAE)

*Proof.* We define three games $\mathbf{G}_0 \sim \mathbf{G}_2$ as shown in Figure 14.

Game $\mathbf{G}_0$ is the same as $\mathsf{REAL}^{\mathcal{A}}_{\mathsf{HtAE}}$ (see Figure 4) equipped with a random oracle H. Therefore, we have $\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathsf{REAL}^{\mathcal{A}}_{\mathsf{HtAE}} \Rightarrow 1\right]$.

Game $\mathbf{G}_1$ is the same as game $\mathbf{G}_0$, except that the AE key $K_e$ and committing tag $T$ are now sampled uniformly at random from the set $\{0,1\}^{\kappa} \times \{0,1\}^{t}$ in the encryption oracle Enc. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically until the bad event happens, where the adversary $\mathcal{A}$ ever queries H with an input that is prefixed with the key $K$ sampled in the beginning of the games. The only syntactical difference between the two games is the generation of $K_e$ and $T$. Recall that $\mathcal{A}$ is nonce-respecting, i.e., it never

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X) \wedge X = (K, *)$ : | $(K_e, T) \xleftarrow{\$} \{0,1\}^\kappa \times \{0,1\}^t$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | $\quad \mathsf{bad} \leftarrow \mathbf{true}$ | $\boxed{(K_e, T) \leftarrow \mathrm{H}(K, N, AD)}$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $C \leftarrow \mathsf{AE.Enc}(K_e, N, \varepsilon, M)$ |
| **return** $b'$ | **return** $\mathrm{T_H}[X]$ | **return** $(C, T)$ |

| Game $\mathbf{G}_2$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $T \xleftarrow{\$} \{0,1\}^t$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **return** $\mathrm{T_H}[X]$ | $C \xleftarrow{\$} \{0,1\}^{\mathsf{AE.clen}(\lvert M \rvert)}$ |
| **return** $b'$ | | **return** $(C, T)$ |

Fig. 14: Games $\mathbf{G}_0 \sim \mathbf{G}_2$ in the proof of Theorem 1, where $\mathbf{G}_0$ contains boxed content while $\mathbf{G}_1$ does not.

repeats a nonce for its encryption queries. Therefore, if $\mathsf{bad}$ does not occur, then in both games the $\mathsf{AE}$ key $K_e$ and committing tag $T$ are random strings that are independent of $\mathcal{A}$'s view, and hence games $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical-until-$\mathsf{bad}$. By the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr[\mathsf{bad}]. \tag{1}$$

Then, note that $K$ is independent of $\mathcal{A}$'s view, because $K_e$ and $T$ do not reveal any information about the random oracle input $K$. Since $K$ is sampled from $\{0,1\}^\kappa$ uniformly at random and $\mathcal{A}$ makes at most $q_\mathrm{h}$ random oracle queries, we have

$$\Pr[\mathsf{bad}] \leq \frac{q_\mathrm{h}}{2^\kappa}. \tag{2}$$

Game $\mathbf{G}_2$ is the same as game $\mathbf{G}_1$, except that in the Enc oracle both the committing tag $T$ and the $\mathsf{AE}$ ciphertext $C$ are sampled uniformly at random. To bound the gap between $\mathbf{G}_1$ and $\mathbf{G}_2$, we apply a hybrid argument as follows. Consider a sequence of hybrid games $\mathbf{H}_0, \mathbf{H}_1, \ldots, \mathbf{H}_{q_\mathrm{e}}$ such that $\mathbf{G}_1 = \mathbf{H}_0$ and $\mathbf{G}_2 = \mathbf{H}_{q_\mathrm{e}}$. Game $\mathbf{H}_i$ is the same as game $\mathbf{G}_1$, except that the output of the first $i$ Enc queries are replaced by independent random strings as in $\mathbf{G}_2$. By construction, consecutive games $\mathbf{H}_{i-1}$ and $\mathbf{H}_i$ are the same except for the $i$-th Enc query. We can construct an adversary $\mathcal{B}_i$ against the IND$-CPA security of the underlying $\mathsf{AE}$ such that it simulates $\mathcal{A}$'s view in $\mathbf{H}_{i-1}$ in the real world and $\mathbf{H}_i$ in the ideal world. To do so, $\mathcal{B}_i$ simulates $\mathcal{A}$'s views in $\mathbf{H}_{i-1}$ and $\mathbf{H}_i$ in the same way following their game procedures, except that, for the $i$-th Enc query $\mathcal{B}_i$ samples a random $T$ and calls its own encryption oracle $\mathrm{Enc}'$ to simulate the $\mathsf{AE}$ ciphertext $C$. Finally, $\mathcal{B}_i$ outputs the bit that $\mathcal{A}$ outputs. By construction, we have $\Pr\left[\mathbf{H}_{i-1}^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{H}_i^{\mathcal{A}} \Rightarrow 1\right] = \mathbf{Adv}^{\text{ot-ind\$-cpa}}_{\mathsf{AE}}(\mathcal{B}_i)$, because $\mathcal{B}_i$ makes only one $\mathrm{Enc}'$ query. With a hybrid argument, there exists an efficient adversary $\mathcal{B}$ such that

$$\begin{aligned} &\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathbf{H}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{H}_{q_\mathrm{e}}^{\mathcal{A}} \Rightarrow 1\right] \\ &= \sum_{i=1}^{q_\mathrm{e}} \left(\Pr\left[\mathbf{H}_{i-1}^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{H}_i^{\mathcal{A}} \Rightarrow 1\right]\right) \leq q_\mathrm{e} \mathbf{Adv}^{\text{ot-ind\$-cpa}}_{\mathsf{AE}}(\mathcal{B}). \end{aligned} \tag{3}$$

Finally, we observe that game $\mathbf{G}_2$ is identical to $\mathsf{RAND}^{\mathcal{A}}_{\mathsf{HtAE}}$ (see Figure 4) equipped with a random oracle H. This is because, in game $\mathbf{G}_2$, for each Enc query with a distinct nonce, its output is an independent uniform random string sampled from $\{0,1\}^{\mathsf{HtAE.clen}(|M|)} = \{0,1\}^{\mathsf{AE.clen}(|M|)+t}$. Therefore, we have

$$\mathbf{Adv}^{\text{ind\$-cpa}}_{\mathsf{HtAE}}(\mathcal{A}) = \Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right]. \tag{4}$$

The proof is concluded by combining equations (1)∼(4). □

### B.2 Proof of Theorem 2 (INT-CTXT Security of HtAE)

*Proof.* We define two games $\mathbf{G}_0$ and $\mathbf{G}_1$ as shown in Figure 15.

Game $\mathbf{G}_0$ is the same as $\mathsf{INT\text{-}CTXT}^{\mathcal{A}}_{\mathsf{HtAE}}$ (see Figure 4) equipped with a random oracle H. Therefore, we have $\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] = \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{HtAE}}(\mathcal{A})$.

| Game $\mathbf{G}_0$, $\boxed{\mathbf{G}_1}$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X)$ | $(K_e, T) \leftarrow \mathrm{H}(K, N, AD)$ | $(K_e, T') \leftarrow \mathrm{H}(K, N, AD)$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | $\wedge\ X = (K, *)$ : | $C \leftarrow \mathsf{AE.Enc}(K_e, N, \varepsilon, M)$ | **if** $T' \neq T$ : **return false** |
| $\mathcal{Q} \leftarrow \varnothing$ | $\mathsf{bad} \leftarrow$ **true** | $\mathcal{Q} \xleftarrow{\cup} \{(N, AD, (C, T))\}$ | $M \leftarrow \mathsf{AE.Dec}(K_e, N, \varepsilon, C)$ |
| $\mathrm{win} \leftarrow 0$ | $\boxed{\textbf{abort}}$ | **return** $(C, T)$ | **if** $M \neq \perp \wedge (N, AD, (C, T)) \notin \mathcal{Q}$ : |
| $\mathcal{A}^{\mathrm{H,Enc,Ver}}$ | **if** $\mathrm{T_H}[X] = \perp$ : | | $\mathrm{win} \leftarrow 1$ |
| **return** win | $\mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^{\kappa+t}$ | | **return** $M \neq \perp$ |
| | **return** $\mathrm{T_H}[X]$ | | |

| Adv. $\mathcal{B}^{\mathrm{Enc}', \mathrm{Ver}'}$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | **if** $\mathcal{A}$ queried $\mathrm{H}(X)$ | **if** $(N, AD)$ is the $\hat{i}$-th distinct pair : | **if** $(N, AD)$ is the $\hat{i}$-th distinct pair : |
| $\hat{i} \xleftarrow{\$} [q_\mathrm{e} + q_\mathrm{v}]$ | $\wedge\ X = (K, *)$ : | $T \leftarrow \widehat{T}$ | **if** $T \neq \widehat{T}$ : **return false** |
| $\widehat{T} \xleftarrow{\$} \{0,1\}^t$ | **abort** | $C \leftarrow \mathrm{Enc}'(N, \varepsilon, M)$ | **return** $\mathrm{Ver}'(N, \varepsilon, C)$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp$ : | **else** | **else** |
| $\mathcal{A}^{\mathrm{H,Enc,Ver}}$ | $\mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^{\kappa+t}$ | $(K_e, T) \leftarrow \mathrm{H}(K, N, AD)$ | $(K_e, T') \leftarrow \mathrm{H}(K, N, AD)$ |
| | **return** $\mathrm{T_H}[X]$ | $C \leftarrow \mathsf{AE.Enc}(K_e, N, \varepsilon, M)$ | **if** $T \neq T'$ : **return false** |
| | | **return** $(C, T)$ | $M \leftarrow \mathsf{AE.Dec}(K_e, N, \varepsilon, C)$ |
| | | | **return** $M \neq \perp$ |

Fig. 15: Games $\mathbf{G}_0$, $\mathbf{G}_1$ and the adversary $\mathcal{B}$ in the proof of Theorem 2, where $\mathbf{G}_1$ aborts when bad occurs while $\mathbf{G}_0$ does not.

Game $\mathbf{G}_1$ is the same as game $\mathbf{G}_0$, except that it aborts when the bad event bad occurs, which is triggered when adversary $\mathcal{A}$ queries H with an input that is prefixed with the key $K$ sampled in the beginning of the games. Since $K$ is sampled from $\{0,1\}^\kappa$ uniformly at random and it is independent of $\mathcal{A}$'s view, the probability that bad occurs is at most $q_\mathrm{h}/2^\kappa$. By construction, $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical-until-bad. Therefore, by the Difference Lemma of the game-based proof technique [31], we have

$$\left|\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \Pr[\mathsf{bad}] \leq \frac{q_\mathrm{h}}{2^\kappa}. \tag{5}$$

Finally, we bound $\mathcal{A}$'s winning probability $\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}$ against the INT-CTXT security of the underlying AE. To simulate $\mathcal{A}$'s view in $\mathbf{G}_1$, adversary $\mathcal{B}$ first samples a random key $K \xleftarrow{\$} \{0,1\}^\kappa$. Then, for each random oracle query made by $\mathcal{A}$ on an input $X$ not prefixed with $K$, adversary $\mathcal{B}$ simulates $\mathrm{H}(X)$ via lazy sampling. For encryption and verification queries on input $(N, AD, \cdot)$ made by $\mathcal{A}$, we note that each distinct $(N, AD)$ pair determines an independent key $K_e$ for the underlying AE. Therefore, to simulate responses to such queries, $\mathcal{B}$ first randomly picks one of such distinct $(N, AD)$ pairs, say the $\hat{i}$-th distinct pair denoted by $(\widehat{N}, \widehat{AD})$, and then samples an independent random committing tag $\widehat{T} \xleftarrow{\$} \{0,1\}^t$. For each encryption or verification query on input $(N, AD, \cdot) \neq (\widehat{N}, \widehat{AD}, \cdot)$, $\mathcal{B}$ can easily simulate its response: it first derives an AE key $K_e$ and a committing tag $T$ by simulating the output of $\mathrm{H}(K, N, AD)$ via lazy sampling, then it uses $(K_e, T)$ to simulate the rest of the oracle. For a $(\widehat{N}, \widehat{AD}, \cdot)$ query, $\mathcal{B}$ does not simulate the random oracle output but instead calls its own INT-CTXT $\mathrm{Enc}'/\mathrm{Ver}'$ oracle to simulate the answer. A code-based description of $\mathcal{B}$ is shown in Figure 15.

Note that, if bad does not occur, then $\mathcal{A}$'s random oracle queries do not contain any information about the AE keys. Therefore, by construction of $\mathcal{B}$, it simulates $\mathcal{A}$'s view in $\mathbf{G}_1$ perfectly. Then, suppose $\mathcal{A}$ wins by querying $\mathrm{Ver}(N^*, AD^*, (C^*, T^*))$ and $\mathcal{B}$ happens to pick the correct $(N, AD)$ pair (i.e., $(\widehat{N}, \widehat{AD}) = (N^*, AD^*)$), we show that $\mathcal{B}$ also wins in its $\mathsf{INT\text{-}CTXT}_{\mathsf{AE}}^{\mathcal{B}}$ game. By definition of the winning condition of $\mathcal{A}$, we know $(C^*, T^*)$ was not output by any $\mathrm{Enc}(N^*, AD^*, \cdot)$ query and $\mathrm{Ver}(N^*, AD^*, (C^*, T^*)) = 1$. By construction of $\mathcal{B}$, the latter implies that $\widehat{T} = T^*$ and $\mathrm{Ver}'(N^*, \varepsilon, C^*) = 1$, so it suffices to show that $C^*$ was not output by any $\mathrm{Enc}'(N^*, \varepsilon, \cdot)$ query made by $\mathcal{B}$. This is true by observing that the committing tag for an $\mathrm{Enc}(N^*, AD^*, \cdot)$ query (i.e., an $\mathrm{Enc}(\widehat{N}, \widehat{AD}, \cdot)$ query) is $T^*$ (i.e., $\widehat{T}$). That is, if $C^*$ was output by an $\mathrm{Enc}'(N^*, \varepsilon, \cdot)$ query, then $\mathcal{A}$ must have queried $\mathrm{Enc}(N^*, AD^*, \cdot)$ that output $(C^*, T^*)$, which is a

contradiction. Therefore, $\mathcal{B}$ also wins. Since there are at most $q_e + q_v$ distinct $(N, AD)$ pairs, we have

$$\frac{1}{q_e + q_v} \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{AE}}^{\text{int-ctxt}}(\mathcal{B}). \tag{6}$$

The proof is concluded by combining equations (5) and (6). □

### B.3 Proof of Theorem 3 (CMT Security of HtAE)

*Proof.* In order to win the $\mathsf{CMT}_{\mathsf{HtAE}}^{\mathcal{A}}$ game (see Figure 5), $\mathcal{A}$ needs to output two tuples $(K_1, N_1, AD_1, M_1)$, $(K_2, N_2, AD_2, M_2)$ such that they are encrypted to the same ciphertext and $(K_1, N_1, AD_1) \neq (K_2, N_2, AD_2)$. By construction of HtAE, this implies that the hashes of $(K_1, N_1, AD_1)$ and $(K_2, N_2, AD_2)$ collide in the right part (i.e. the committing tag $T$), which breaks right collision resistance of H. Therefore, we can construct $\mathcal{B}$ as follows: it first invokes $\mathcal{A}$, then extracts $(K_1, N_1, AD_1)$ and $(K_2, N_2, AD_2)$ from $\mathcal{A}$'s output and outputs this pair. □

### B.4 Proof of Theorem 4 (CDY$ Security of HtAE)

*Proof.* In the $\mathsf{CDY}_{\mathsf{HtAE},\mathsf{S}_\$}^{\mathcal{A}}$ game (see Figure 6), $\mathcal{A}$ gets a challenge ciphertext $(C, T)$ output by the context selector $\mathsf{S}_\$$ and wins by outputting a context tuple $(K', N', AD', M')$ such that $\mathsf{HtAE.Enc}(K', N', AD', M') = (C, T)$. Therefore, if $\mathcal{A}$ wins, then the committing tag $T'$ derived from $(C', T') \leftarrow \mathsf{HtAE.Enc}(K', N', AD', M')$ must be equal to $T$. This can only happen in two cases: (1) when a random oracle query was made on the exact tuple $(K, N, AD)$ sampled by $\mathsf{S}_\$$ (together with a random $M$) to derive the challenge ciphertext $(C, T)$, or (2) when one of the random oracle queries (not on input $(K, N, AD)$) outputs a committing tag equal to $T$. The former occurs with probability at most $(q_h + 1)/2^\kappa |\mathcal{AD}|$, because by construction of HtAE the randomly sampled $K$ and $AD$ are independent of $\mathcal{A}$'s view; the latter occurs with probability at most $(q_h + 1)/2^t$, because the random oracle H outputs independent random committing tags for distinct inputs. Note that the denominators are $q_h + 1$ because $\mathcal{A}$ made $q_h$ random oracle queries and the challenger made one last H query to compute the encryption of $\mathcal{A}$'s output. Therefore, we have

$$\mathbf{Adv}_{\mathsf{HtAE}}^{\text{cdy\$}}(\mathcal{A}) \leq \frac{1}{|\mathcal{AD}|} \cdot \frac{q_h + 1}{2^\kappa} + \frac{q_h + 1}{2^t}. \tag{7}$$

□

### B.5 Proof of Theorem 5 (IND$-CPA Security of AEaH)

*Proof.* We define three games $\mathbf{G}_0 \sim \mathbf{G}_2$ as shown in Figure 16.

Game $\mathbf{G}_0$ is the same as $\mathsf{REAL}_{\mathsf{AEaH}}^{\mathcal{A}}$ (see Figure 4) equipped with a random oracle H. Therefore, we have $\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathsf{REAL}_{\mathsf{AEaH}}^{\mathcal{A}} \Rightarrow 1\right]$.

Game $\mathbf{G}_1$ is the same as game $\mathbf{G}_0$, except that the committing tag $T$ is now sampled randomly from the set $\{0,1\}^t$ in the encryption oracle Enc. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically unless the bad event happens, where the adversary $\mathcal{A}$ ever queries H with an input that is prefixed with the key $K$ used in Enc. The only syntactical difference between the two games is the generation of $T$. Recall that $\mathcal{A}$ is nonce-respecting, i.e., it never repeats a nonce for its encryption queries. Therefore, if bad does not occur, then in both games the committing tag $T$ is a random string that is independent of $\mathcal{A}$'s view, and hence games $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical-until-bad. By the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr[\mathsf{bad}]. \tag{8}$$

Next, we bound Pr[bad] by considering a game $\mathbf{G}_*$ also shown in Figure 16. Here $\mathbf{G}_*$ is just defined to clarify the bad event, such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers bad in game $\mathbf{G}_0/\mathbf{G}_1$. Formally, we define a set $\mathcal{Q}^*$ that records all the keys contained in the random oracle H queries made by $\mathcal{A}$, then let $\mathbf{G}_*$ return 1 if and only if $\mathcal{Q}^*$ contains the key $K$ used in Enc. Therefore, we have

$$\Pr[\mathsf{bad}] = \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]. \tag{9}$$

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}_1$ against the IND$-CPA security of the underlying AE. $\mathcal{B}_1$ starts by initializing the set $\mathcal{Q}^*$ and the table $\mathrm{T_H}$ (to an empty set and empty table,

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X) \wedge X = (K, *)$ : | $C \leftarrow \mathsf{AE.Enc}(K, N, \varepsilon, M)$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | $\quad \mathsf{bad} \leftarrow \mathbf{true}$ | $T \xleftarrow{\$} \{0,1\}^t$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $\boxed{T \leftarrow \mathrm{H}(K, N, AD)}$ |
| **return** $b'$ | **return** $\mathrm{T_H}[X]$ | **return** $(C, T)$ |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | $(K', *) \leftarrow X$ | $C \leftarrow \mathsf{AE.Enc}(K, N, \varepsilon, M)$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K'\}$ | $T \xleftarrow{\$} \{0,1\}^t$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | **return** $(C, T)$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **return** $\mathrm{T_H}[X]$ | |
| **return** $K \in \mathcal{Q}^*$ | | |

| Game $\mathbf{G}_2$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $C \xleftarrow{\$} \{0,1\}^{\mathsf{AE.clen}(\lvert M \rvert)}$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **return** $\mathrm{T_H}[X]$ | $T \xleftarrow{\$} \{0,1\}^t$ |
| **return** $b'$ | | **return** $(C, T)$ |

Fig. 16: Games $\mathbf{G}_0 \sim \mathbf{G}_2$ and $\mathbf{G}_*$ in the proof of Theorem 5, where $\mathbf{G}_0$ contains boxed content while $\mathbf{G}_1$ does not.

respectively), and then it runs $\mathcal{A}$. To answer $\mathcal{A}$'s encryption query on input $(N, AD, M)$, $\mathcal{B}_1$ calls its own encryption oracle with input $(N, \varepsilon, M)$ and gets back $C$, then samples a random committing tag $T$ from $\{0,1\}^t$ and returns $(C, T)$ back to $\mathcal{A}$. For $\mathcal{A}$'s random oracle query $X = (K', *)$, $\mathcal{B}_1$ records $K'$ in $\mathcal{Q}^*$ and answers the query via lazy sampling. After $\mathcal{A}$ terminates, $\mathcal{B}_1$ ignores its output and queries its own encryption oracle with input $(N^*, \varepsilon, 0^\kappa)$ and gets back $C^*$, where $N^*$ is a nonce different from those used in previous encryption queries. Then, for each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}_1$ computes $C_i \leftarrow \mathsf{AE.Enc}(K_i, N^*, \varepsilon, 0^\kappa)$ and checks if $C_i = C^*$ holds. If this holds for any of the keys in $\mathcal{Q}^*$, $\mathcal{B}_1$ guesses that it was in the real world $\mathsf{REAL}_{\mathsf{AE}}^{\mathcal{B}_1}$ and outputs 1; otherwise, it outputs 0, guessing that it was in the ideal world $\mathsf{RAND}_{\mathsf{AE}}^{\mathcal{B}_1}$.

If $\mathcal{B}_1$ was in the real world, then it simulates $\mathcal{A}$'s view in $\mathbf{G}_*$ perfectly and hence we have $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathsf{REAL}_{\mathsf{AE}}^{\mathcal{B}_1} \Rightarrow 1\right]$. Here we have an inequality to capture "false positives" in the above $C_i = C^*$ check. On the other hand, if $\mathcal{B}_1$ was in the ideal world, the probability that a random ciphertext $C^*$ equals $\mathsf{AE.Enc}(K_i, N^*, \varepsilon, 0^\kappa)$ for some $K_i \in \mathcal{Q}^*$ is at most $|\mathcal{Q}^*|/2^{\mathsf{AE.clen}(\kappa)}$. Since $|\mathcal{Q}^*| \leq q_{\mathrm{h}}$, we have $\Pr\left[\mathsf{RAND}_{\mathsf{AE}}^{\mathcal{B}_1} \Rightarrow 1\right] \leq q_{\mathrm{h}}/2^{\mathsf{AE.clen}(\kappa)}$. Recalling that $\mathbf{Adv}_{\mathsf{AE}}^{\text{ind\$-cpa}}(\mathcal{B}_1) = \Pr\left[\mathsf{REAL}_{\mathsf{AE}}^{\mathcal{B}_1} \Rightarrow 1\right] - \Pr\left[\mathsf{RAND}_{\mathsf{AE}}^{\mathcal{B}_1} \Rightarrow 1\right]$ and $\mathsf{AE.clen}(\kappa) \geq \kappa$, we have

$$\mathbf{Adv}_{\mathsf{AE}}^{\text{ind\$-cpa}}(\mathcal{B}_1) \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_{\mathrm{h}}}{2^{\mathsf{AE.clen}(\kappa)}} \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_{\mathrm{h}}}{2^\kappa}, \tag{10}$$

and it follows from equations (8)∼(10) that

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{AE}}^{\text{ind\$-cpa}}(\mathcal{B}_1) + \frac{q_{\mathrm{h}}}{2^\kappa}. \tag{11}$$

We continue by making a game hop from $\mathbf{G}_1$ to $\mathbf{G}_2$. As shown in Figure 16, game $\mathbf{G}_2$ is the same as game $\mathbf{G}_1$, except that now in the encryption oracle Enc the ciphertext $C$ is also sampled uniformly at random. We bound the gap between $\mathbf{G}_1$ and $\mathbf{G}_2$ by constructing an adversary $\mathcal{B}_2$ against the IND$-CPA security of the underlying $\mathsf{AE}$. $\mathcal{B}_2$ just simulates $\mathcal{A}$'s view as the challenger in $\mathbf{G}_1/\mathbf{G}_2$ does, but for $\mathcal{A}$'s encryption query, $\mathcal{B}_2$ calls its own encryption oracle to get $C$ and then finish the rest of the simulation. It is easy to see that $\mathcal{B}_2$ simulates $\mathcal{A}$'s view in $\mathbf{G}_1$ in the real world and $\mathbf{G}_2$ in the ideal world. In the end, adversary $\mathcal{B}_2$ outputs the bit that $\mathcal{A}$ outputs. Therefore, we have

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{AE}}^{\text{ind\$-cpa}}(\mathcal{B}_2). \tag{12}$$

Finally, we observe that game $\mathbf{G}_2$ is identical to $\mathsf{RAND}^{\mathcal{A}}_{\mathsf{AEaH}}$ (see Figure 4) equipped with a random oracle H. This is because, in game $\mathbf{G}_2$, for each Enc query with a distinct nonce, its output is an independent uniform random string sampled from $\{0,1\}^{\mathsf{HtAE.clen}(|M|)} = \{0,1\}^{\mathsf{AE.clen}(|M|)+t}$.

The proof is concluded by combining equations (11) and (12). □

### B.6 Proof of Theorem 6 (INT-CTXT Security of AEaH)

*Proof.* We define two games $\mathbf{G}_0$ and $\mathbf{G}_1$ as shown in Figure 17.

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X)$ | $C \leftarrow \mathsf{AE.Enc}(K, N, \varepsilon, M)$ | **if** $\mathrm{T}_T[(N, AD)] = \bot$ : |
| $\mathrm{T}_\mathrm{H}[\cdot] \leftarrow \bot$ | $\quad \wedge X = (K, *)$ : | **if** $\mathrm{T}_T[(N, AD)] = \bot$ : | $\quad \mathrm{T}_T[(N, AD)] \xleftarrow{\$} \{0,1\}^t$ |
| $\mathrm{T}_T[\cdot] \leftarrow \bot$ | $\quad \mathsf{bad} \leftarrow \mathbf{true}$ | $\quad \mathrm{T}_T[(N, AD)] \xleftarrow{\$} \{0,1\}^t$ | $T' \leftarrow \mathrm{T}_T[(N, AD)]$ |
| $\mathcal{Q} \leftarrow \varnothing$ | **if** $\mathrm{T}_\mathrm{H}[X] = \bot$ : | $T \leftarrow \mathrm{T}_T[(N, AD)]$ | $\boxed{T' \leftarrow \mathrm{H}(K, N, AD)}$ |
| $\mathrm{win} \leftarrow 0$ | $\quad \mathrm{T}_\mathrm{H}[X] \xleftarrow{\$} \{0,1\}^t$ | $\boxed{T \leftarrow \mathrm{H}(K, N, AD)}$ | **if** $T \neq T'$ : **return false** |
| $\mathcal{A}^{\mathrm{H},\mathrm{Enc},\mathrm{Ver}}$ | **return** $\mathrm{T}_\mathrm{H}[X]$ | $\mathcal{Q} \xleftarrow{\cup} \{(N, AD, (C, T))\}$ | $M \leftarrow \mathsf{AE.Dec}(K, N, \varepsilon, C)$ |
| **return** win | | **return** $(C, T)$ | **if** $M \neq \bot \wedge (N, AD, (C, T)) \notin \mathcal{Q}$ : |
| | | | $\quad \mathrm{win} \leftarrow 1$ |
| | | | **return** $M \neq \bot$ |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | $(K', *) \leftarrow X$ | $C \leftarrow \mathsf{AE.Enc}(K, N, \varepsilon, M)$ | **if** $\mathrm{T}_T[(N, AD)] = \bot$ : |
| $\mathrm{T}_\mathrm{H}[\cdot] \leftarrow \bot$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K'\}$ | **if** $\mathrm{T}_T[(N, AD)] = \bot$ : | $\quad \mathrm{T}_T[(N, AD)] \xleftarrow{\$} \{0,1\}^t$ |
| $\mathrm{T}_T[\cdot] \leftarrow \bot$ | **if** $\mathrm{T}_\mathrm{H}[X] = \bot$ : | $\quad \mathrm{T}_T[(N, AD)] \xleftarrow{\$} \{0,1\}^t$ | $T' \leftarrow \mathrm{T}_T[(N, AD)]$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | $\quad \mathrm{T}_\mathrm{H}[X] \xleftarrow{\$} \{0,1\}^t$ | $T \leftarrow \mathrm{T}_T[(N, AD)]$ | **if** $T \neq T'$ : **return false** |
| $\mathcal{A}^{\mathrm{H},\mathrm{Enc},\mathrm{Ver}}$ | **return** $\mathrm{T}_\mathrm{H}[X]$ | **return** $(C, T)$ | $M \leftarrow \mathsf{AE.Dec}(K, N, \varepsilon, C)$ |
| **return** $K \in \mathcal{Q}^*$ | | | **return** $M \neq \bot$ |

Fig. 17: Games $\mathbf{G}_0$, $\mathbf{G}_1$ and $\mathbf{G}_*$ in the proof of Theorem 6, where $\mathbf{G}_0$ contains boxed content while $\mathbf{G}_1$ does not.

Game $\mathbf{G}_0$ is the same as $\mathsf{INT\text{-}CTXT}^{\mathcal{A}}_{\mathsf{AEaH}}$ (see Figure 4) equipped with a random oracle H. (Note that the $\mathrm{T}_T$ operations can be ignored in $\mathbf{G}_0$.) Therefore, we have $\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] = \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{AEaH}}(\mathcal{A})$.

Game $\mathbf{G}_1$ is the same as game $\mathbf{G}_0$, except that the committing tag $T$ is now lazily sampled from the table $\mathrm{T}_T$ in both oracles, where the table is indexed by the queried $(N, AD)$ pair. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically unless the $\mathsf{bad}$ event happens, which occurs if the adversary $\mathcal{A}$ ever queries H with an input that is prefixed with the key $K$. The only syntactical difference between the two games are in the generation of $T$. If $\mathsf{bad}$ does not occur, in both games the committing tag $T$ is either randomly sampled for a new $(N, AD)$ pair, or gets the same previously sampled value if $(N, AD)$ is repeated; hence the two games are indeed identical-until-$\mathsf{bad}$. By the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr[\mathsf{bad}]. \tag{13}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by considering a game $\mathbf{G}_*$ also shown in Figure 17. $\mathbf{G}_*$ is defined such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers $\mathsf{bad}$ in $\mathbf{G}_0/\mathbf{G}_1$. Therefore, we have

$$\Pr[\mathsf{bad}] = \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]. \tag{14}$$

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}_1$ against the INT-CTXT security of the underlying $\mathsf{AE}$. $\mathcal{B}_1$ starts by initializing the set $\mathcal{Q}^*$ and tables $\mathrm{T}_\mathrm{H}, \mathrm{T}_T$ (to an empty set and empty tables, respectively), and then it runs $\mathcal{A}$. To answer $\mathcal{A}$'s encryption query on input $(N, AD, M)$, $\mathcal{B}_1$ calls its own encryption oracle $\mathrm{Enc}'$ with input $(N, \varepsilon, M)$ and gets back $C$, then lazily samples $T$ with the help of table $\mathrm{T}_T$. At the end, it returns $(C, T)$ back to $\mathcal{A}$. To answer $\mathcal{A}$'s verification query on input $(N, AD, (C, T))$, $\mathcal{B}_1$ lazily samples $T'$ with the help of the table $\mathrm{T}_T$ and checks if $T'$ equals $T$. If not, $\mathcal{B}_1$ returns **false**,

otherwise it calls its own verification oracle $\mathrm{Ver}'$ with input $(N, \varepsilon, C)$ and forwards the answer back to $\mathcal{A}$. For $\mathcal{A}$'s random oracle query $X = (K', *)$, $\mathcal{B}_1$ records $K'$ in $\mathcal{Q}^*$ and answers the query via lazy sampling using the table $\mathrm{T_H}$. After $\mathcal{A}$ terminates, $\mathcal{B}_1$ picks a nonce $N^*$ that is different from those used by $\mathcal{A}$ in encryption oracle queries; then, for each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}$ computes $C_i \leftarrow \mathsf{AE.Enc}(K_i, N^*, \varepsilon, 0^\kappa)$ and queries its own verification oracle with $(N^*, \varepsilon, C_i)$.

The above adversary $\mathcal{B}_1$ correctly simulates the $\mathcal{A}$'s view in $\mathbf{G}_*$, and we claim that, if $\mathcal{A}$ wins in $\mathbf{G}_*$, then $\mathcal{B}_1$ wins in its $\mathsf{INT\text{-}CTXT}^{\mathcal{B}_1}_{\mathsf{AE}}$ game. If $\mathcal{A}$ wins, then the $\mathsf{AE}$ key $K$ (sampled by $\mathcal{B}_1$'s challenger) must be included in the set $\mathcal{Q}^*$. Since $\mathcal{B}_1$ enumerated all keys in $\mathcal{Q}^*$ to compute ciphertexts with a new nonce, the ciphertext derived from the correct key $K$ must be a valid forgery for $\mathcal{B}_1$ to win in its game. Therefore, we have

$$\Pr\left[\mathbf{G}^{\mathcal{A}}_* \Rightarrow 1\right] \leq \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{AE}}(\mathcal{B}_1). \tag{15}$$

and it follows from equations (13)∼(15) that

$$\Pr\left[\mathbf{G}^{\mathcal{A}}_0 \Rightarrow 1\right] - \Pr\left[\mathbf{G}^{\mathcal{A}}_1 \Rightarrow 1\right] \leq \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{AE}}(\mathcal{B}_1). \tag{16}$$

Finally, we bound $\mathcal{A}$'s winning probability $\Pr\left[\mathbf{G}^{\mathcal{A}}_1 \Rightarrow 1\right]$ by constructing another INT-CTXT adversary $\mathcal{B}_2$ against the underlying $\mathsf{AE}$. Adversary $\mathcal{B}_2$ is the same as adversary $\mathcal{B}_1$, except that $\mathcal{B}_2$ terminates immediately after $\mathcal{A}$ terminates, i.e., $\mathcal{B}_2$ does not record keys in $\mathcal{Q}^*$ or make the final verification queries after $\mathcal{A}$ terminates. Clearly, $\mathcal{B}_2$ correctly simulates $\mathcal{A}$'s view in $\mathbf{G}_1$. Now, suppose $\mathcal{A}$ wins by querying $\mathrm{Ver}(N^*, AD^*, (C^*, T^*))$. By definition of the winning condition of $\mathcal{A}$, we know $(C^*, T^*)$ was not output by any $\mathrm{Enc}(N^*, AD^*, \cdot)$ query and $\mathrm{Ver}(N^*, AD^*, (C^*, T^*)) = 1$. By construction of $\mathcal{B}_2$, the latter implies that $T^* = \mathrm{T}_T[(N^*, AD^*)]$ and $\mathrm{Ver}'(N^*, \varepsilon, C^*) = 1$. There are two cases. If $C^*$ was not output by any $\mathrm{Enc}'(N^*, \varepsilon, \cdot)$ query made by $\mathcal{B}_2$, then $\mathcal{B}_2$ wins in its $\mathsf{INT\text{-}CTXT}^{\mathcal{B}_2}_{\mathsf{AE}}$ game. Otherwise, $\mathcal{A}$ must have queried $\mathrm{Enc}(N^*, \widehat{AD}, \cdot)$ with some $\widehat{AD} \neq AD^*$ that output $(C^*, \widehat{T})$, because any $\mathrm{Enc}(N^*, AD^*, \cdot)$ query will produce the committing tag $T^*$. Then, since $\mathcal{B}_2$ is nonce-respecting, it has never made an $\mathrm{Enc}(N^*, AD^*, \cdot)$ query to learn $\mathrm{T}_T[(N^*, AD^*)]$. Therefore, $\mathcal{B}_2$ can only guess $T^*$ when making verification queries. Since it can make $q_\mathrm{v}$ verification queries and $T^*$ is randomly sampled from $\{0,1\}^t$, the probability that $\mathcal{B}_2$ wins is at most $q_\mathrm{v}/2^t$. Therefore, we have

$$\Pr\left[\mathbf{G}^{\mathcal{A}}_1 \Rightarrow 1\right] \leq \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{AE}}(\mathcal{B}_2) + \frac{q_\mathrm{v}}{2^t}. \tag{17}$$

The proof is concluded by combining equations (16) and (17). □

### B.7 Proof of Theorem 7 (CMT Security of AEaH)

*Proof.* In order to win the $\mathsf{CMT}^{\mathcal{A}}_{\mathsf{AEaH}}$ game (see Figure 5), $\mathcal{A}$ needs to output two tuples $(K_1, N_1, AD_1, M_1)$, $(K_2, N_2, AD_2, M_2)$ such that they are encrypted to the same ciphertext and $(K_1, N_1, AD_1) \neq (K_2, N_2, AD_2)$. By construction of $\mathsf{AEaH}$, this implies that the hashes of $(K_1, N_1, AD_1)$ and $(K_2, N_2, AD_2)$ collide, which breaks the collision resistance of $\mathsf{H}$. Therefore, we can construct $\mathcal{B}$ as follows: it first invokes $\mathcal{A}$, then extracts $(K_1, N_1, AD_1)$ and $(K_2, N_2, AD_2)$ from $\mathcal{A}$'s output and outputs this pair. □

### B.8 Proof of Theorem 8 (CDY$ Security of AEaH)

*Proof.* We define two games $\mathbf{G}_0$ and $\mathbf{G}_1$ as shown in Figure 18.

Game $\mathbf{G}_0$ is the same as $\mathsf{CDY}^{\mathcal{A}}_{\mathsf{AEaH},\mathsf{S}_\$}$ (see Figure 6) equipped with a random oracle H. Therefore, we have $\Pr\left[\mathbf{G}^{\mathcal{A}}_0 \Rightarrow 1\right] = \mathbf{Adv}^{\text{cdy}}_{\mathsf{AEaH},\mathsf{S}_\$}(\mathcal{A}) = \mathbf{Adv}^{\text{cdy\$}}_{\mathsf{AEaH}}(\mathcal{A})$.

Game $\mathbf{G}_1$ is the same as game $\mathbf{G}_0$, except that the committing tag $T$, calculated by the challenger, is now independent of queries to the random oracle H. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically unless the $\mathsf{bad}$ event happens, where the random oracle H is ever queried with an input that is prefixed with the key $K$ sampled by the challenger. This is because, if $\mathsf{bad}$ does not occur, then $T$ in $\mathbf{G}_0$ is also independent of random oracle queries made by $\mathcal{A}$ and by the challenger (to compute $T'$). Therefore, by the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}^{\mathcal{A}}_0 \Rightarrow 1\right] - \Pr\left[\mathbf{G}^{\mathcal{A}}_1 \Rightarrow 1\right] \leq \Pr[\mathsf{bad}]. \tag{18}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by constructing a game $\mathbf{G}_*$ also shown in Figure 18. $\mathbf{G}_*$ is defined such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers $\mathsf{bad}$ in game $\mathbf{G}_0/\mathbf{G}_1$ when (1)

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\mathrm{H}(X)$ |
|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ ; $K \xleftarrow{\$} \{0,1\}^\kappa$ ; $N \xleftarrow{\$} \mathcal{N}$ ; $AD \xleftarrow{\$} \mathcal{AD}$ ; $M \xleftarrow{\$} \mathcal{M}$ | **if** $X = (K, *)$ : |
| $C \leftarrow \mathsf{AE.Enc}(K, N, \varepsilon, M)$ ; $T \xleftarrow{\$} \{0,1\}^t$ | $\mathsf{bad} \leftarrow \mathbf{true}$ |
| $\boxed{\mathrm{T_H}[(K, N, AD)] \leftarrow T}$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ |
| $(K', N', AD', M') \xleftarrow{\$} \mathcal{A}^{\mathrm{H}}(C, T)$ | **return** $\mathrm{T_H}[X]$ |
| $C' \leftarrow \mathsf{AE.Enc}(K', N', \varepsilon, M')$ ; $T' \leftarrow \mathrm{H}(K', N', AD')$ | |
| **return** $(C', T') = (C, T)$ | |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ |
|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ ; $K \xleftarrow{\$} \{0,1\}^\kappa$ ; $N \xleftarrow{\$} \mathcal{N}$ ; $AD \xleftarrow{\$} \mathcal{AD}$ ; $M \xleftarrow{\$} \mathcal{M}$ | $(K^*, *) \leftarrow X$ |
| $C \leftarrow \mathsf{AE.Enc}(K, N, \varepsilon, M)$ ; $T \xleftarrow{\$} \{0,1\}^t$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K^*\}$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ |
| $(K', N', AD', M') \xleftarrow{\$} \mathcal{A}^{\mathrm{H}}(C, T)$ | **return** $\mathrm{T_H}[X]$ |
| $\mathrm{H}(K', N', AD')$ | |
| **return** $K \in \mathcal{Q}^*$ | |

Fig. 18: Games $\mathbf{G}_0, \mathbf{G}_1$ and $\mathbf{G}_*$ in the proof of Theorem 8, where $\mathbf{G}_0$ contains boxed content while $\mathbf{G}_1$ does not.

$\mathcal{A}$ queried the random oracle H with an input prefixed with $K$ or (2) $\mathcal{A}$ output $(K', N', AD', M')$ with $K' = K$ and then $\mathrm{H}(K', N', AD')$ was queried by the challenger. Therefore, it holds that

$$\Pr[\mathsf{bad}] = \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]. \tag{19}$$

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}$ against the one-time IND$-CPA security of the underlying $\mathsf{AE}$. $\mathcal{B}$ starts by initializing the set $\mathcal{Q}^*$ and the table $\mathrm{T_H}$ (to an empty set and empty table, respectively). Then it samples the tuple $(N, AD, M)$ uniformly at random, and calls its own encryption oracle with input $(N, \varepsilon, M)$ to derive $C$. It then samples $T$ randomly and runs $\mathcal{A}$ with input $(C, T)$. For $\mathcal{A}$'s each random oracle query $X = (K^*, *)$, $\mathcal{B}$ records $K^*$ in $\mathcal{Q}^*$ and answers the query via lazy sampling. After $\mathcal{A}$ terminates and outputs $(K', N', AD', M')$, $\mathcal{B}$ "queries" the random oracle H with the input $(K', N', AD')$ and does the following. For each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}$ computes $C_i \leftarrow \mathsf{AE.Enc}(K_i, N, \varepsilon, M)$ and checks if $C_i = C$. If this holds for any of the keys in $\mathcal{Q}^*$, $\mathcal{B}$ guesses that it was in the real world $\mathsf{REAL}_{\mathsf{AE}}^{\mathcal{B}}$ and outputs 1; otherwise, it outputs 0, guessing that it was in the ideal world $\mathsf{RAND}_{\mathsf{AE}}^{\mathcal{B}}$.

If $\mathcal{B}$ was in the real world, then it simulates $\mathcal{A}$'s view in $\mathbf{G}_*$ perfectly and hence we have $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathsf{REAL}_{\mathsf{AE}}^{\mathcal{B}} \Rightarrow 1\right]$. Here we have an inequality to capture "false positives" in the above $C_i = C$ check. On the other hand, if $\mathcal{B}$ was in the ideal world, the probability that a random ciphertext $C$ equals $\mathsf{AE.Enc}(K_i, N, \varepsilon, M)$ for some $K_i \in \mathcal{Q}^*$ is at most $|\mathcal{Q}^*|/2^{\mathsf{AE.clen}(|M|)}$. Since $|\mathcal{Q}^*| \leq q_{\mathrm{h}} + 1$, we have $\Pr\left[\mathsf{RAND}_{\mathsf{AE}}^{\mathcal{B}} \Rightarrow 1\right] \leq (q_{\mathrm{h}} + 1)/2^{\mathsf{AE.clen}(|M|)}$. Recall that $\mathbf{Adv}_{\mathsf{AE}}^{\text{ot-ind\$-cpa}}(\mathcal{B}) = \Pr\left[\mathsf{REAL}_{\mathsf{AE}}^{\mathcal{B}} \Rightarrow 1\right] - \Pr\left[\mathsf{RAND}_{\mathsf{AE}}^{\mathcal{B}} \Rightarrow 1\right]$ and $\mathsf{AE.clen}(|M|) \geq \tau$, we have

$$\mathbf{Adv}_{\mathsf{AE}}^{\text{ot-ind\$-cpa}}(\mathcal{B}) \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_{\mathrm{h}} + 1}{2^\tau}, \tag{20}$$

and it follows from equations (18)∼(20) that

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{AE}}^{\text{ot-ind\$-cpa}}(\mathcal{B}) + \frac{q_{\mathrm{h}} + 1}{2^\tau}. \tag{21}$$

Finally, we observe that in game $\mathbf{G}_1$ the committing tag $T$ in the challenge ciphertext $(C, T)$ is independent of random oracle queries and hence independent of $T'$, so in order for $\mathcal{A}$ to win one of the random oracle queries must output a committing tag equal to the independent random $T$. This probability is bounded by $(q_{\mathrm{h}} + 1)/2^t$. Therefore, we have

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \frac{q_{\mathrm{h}} + 1}{2^t}. \tag{22}$$

The proof is concluded by combining equations (21) and (22). □

### B.9 Proof of Theorem 9 (IND$-CPA Security of EtH)

*Proof.* We define three games $\mathbf{G}_0 \sim \mathbf{G}_2$ as shown in Figure 19.

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X) \wedge X = (K, *)$ : | $C \leftarrow \mathsf{E.Enc}(K, N, M)$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | $\quad \mathsf{bad} \leftarrow \mathbf{true}$ | $T \xleftarrow{\$} \{0,1\}^t$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $\boxed{T \leftarrow \mathrm{H}(K, N, AD, C)}$ |
| **return** $b'$ | **return** $\mathrm{T_H}[X]$ | **return** $(C, T)$ |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | $(K', *) \leftarrow X$ | $C \leftarrow \mathsf{E.Enc}(K, N, M)$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K'\}$ | $T \xleftarrow{\$} \{0,1\}^t$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | **return** $(C, T)$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **return** $\mathrm{T_H}[X]$ | |
| **return** $K \in \mathcal{Q}^*$ | | |

| Game $\mathbf{G}_2$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $C \xleftarrow{\$} \{0,1\}^{\mathsf{E.clen}(\lvert M \rvert)}$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **return** $\mathrm{T_H}[X]$ | $T \xleftarrow{\$} \{0,1\}^t$ |
| **return** $b'$ | | **return** $(C, T)$ |

Fig. 19: Games $\mathbf{G}_0 \sim \mathbf{G}_2$ and $\mathbf{G}_*$ in the proof of Theorem 9, where $\mathbf{G}_0$ contains boxed content while $\mathbf{G}_1$ does not.

Game $\mathbf{G}_0$ is the same as $\mathsf{REAL}^{\mathcal{A}}_{\mathsf{EtH}}$ (see Figure 4) equipped with a random oracle H. Therefore, we have $\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathsf{REAL}^{\mathcal{A}}_{\mathsf{EtH}} \Rightarrow 1\right]$.

Game $\mathbf{G}_1$ is the same as game $\mathbf{G}_0$, except that the committing tag $T$ is now sampled randomly from the set $\{0,1\}^t$ in the encryption oracle Enc. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically unless the $\mathsf{bad}$ event happens, where the adversary $\mathcal{A}$ ever queries H with an input that is prefixed with the key $K$ used in Enc. The only syntactical difference between the two games is the generation of $T$. Recall that $\mathcal{A}$ is nonce-respecting, i.e., it never repeats a nonce for its encryption queries. Therefore, if $\mathsf{bad}$ does not occur, then in both games the committing tag $T$ is a random string that is independent of $\mathcal{A}$'s view, and hence games $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical-until-$\mathsf{bad}$. By the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr[\mathsf{bad}]. \tag{23}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by considering a game $\mathbf{G}_*$ also shown in Figure 19. Here $\mathbf{G}_*$ is just defined to clarify the $\mathsf{bad}$ event, such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers $\mathsf{bad}$ in game $\mathbf{G}_0/\mathbf{G}_1$. Therefore, we have

$$\Pr[\mathsf{bad}] = \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]. \tag{24}$$

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}_1$ against the IND$-CPA security of the underlying $\mathsf{E}$. $\mathcal{B}_1$ starts by initializing the set $\mathcal{Q}^*$ and the table $\mathrm{T_H}$ (to an empty set and empty table, respectively), and then it runs $\mathcal{A}$. To answer $\mathcal{A}$'s encryption query on input $(N, AD, M)$, $\mathcal{B}_1$ calls its own encryption oracle with input $(N, M)$ and gets back $C$, then samples a random committing tag $T$ from $\{0,1\}^t$ and returns $(C, T)$ back to $\mathcal{A}$. For $\mathcal{A}$'s random oracle query $X = (K', *)$, $\mathcal{B}_1$ records $K'$ in $\mathcal{Q}^*$ and answers the query via lazy sampling. After $\mathcal{A}$ terminates, $\mathcal{B}_1$ ignores its output and queries its own encryption oracle with input $(N^*, 0^\kappa)$ and gets back $C^*$, where $N^*$ is a nonce different from those used in previous encryption queries. Then, for each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}_1$ computes $C_i \leftarrow \mathsf{E.Enc}(K_i, N^*, 0^\kappa)$ and checks if $C_i = C^*$ holds. If this holds for any of the keys in $\mathcal{Q}^*$, $\mathcal{B}_1$ guesses that it was in the real world $\mathsf{REAL}^{\mathcal{B}_1}_{\mathsf{E}}$ and outputs 1; otherwise, it outputs 0, guessing that it was in the ideal world $\mathsf{RAND}^{\mathcal{B}_1}_{\mathsf{E}}$.

If $\mathcal{B}_1$ was in the real world, then it simulates $\mathcal{A}$'s view in $\mathbf{G}_*$ perfectly and hence we have $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathsf{REAL}_{\mathsf{E}}^{\mathcal{B}_1} \Rightarrow 1\right]$. Here we have an inequality to capture "false positives" in the above $C_i = C^*$ check. On the other hand, if $\mathcal{B}_1$ was in the ideal world, the probability that a random ciphertext $C^*$ equals $\mathsf{E.Enc}(K_i, N^*, 0^\kappa)$ for some $K_i \in \mathcal{Q}^*$ is at most $|\mathcal{Q}^*|/2^{\mathsf{E.clen}(\kappa)}$. Since $|\mathcal{Q}^*| \leq q_{\mathrm{h}}$, we have $\Pr\left[\mathsf{RAND}_{\mathsf{E}}^{\mathcal{B}_1} \Rightarrow 1\right] \leq q_{\mathrm{h}}/2^{\mathsf{E.clen}(\kappa)}$. Then, recall that $\mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}_1) = \Pr\left[\mathsf{REAL}_{\mathsf{E}}^{\mathcal{B}_1} \Rightarrow 1\right] - \Pr\left[\mathsf{RAND}_{\mathsf{E}}^{\mathcal{B}_1} \Rightarrow 1\right]$ and $\mathsf{E.clen}(\kappa) \geq \kappa$, we have

$$\mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}_1) \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_{\mathrm{h}}}{2^{\mathsf{E.clen}(\kappa)}} \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_{\mathrm{h}}}{2^{\kappa}}, \tag{25}$$

and it follows from equations (23)∼(25) that

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}_1) + \frac{q_{\mathrm{h}}}{2^{\kappa}}. \tag{26}$$

We continue by making a game hop from $\mathbf{G}_1$ to $\mathbf{G}_2$. As shown in Figure 19, game $\mathbf{G}_2$ is the same as game $\mathbf{G}_1$, except that now in the encryption oracle Enc the ciphertext $C$ is also sampled uniformly at random. We bound the gap between $\mathbf{G}_1$ and $\mathbf{G}_2$ by constructing an adversary $\mathcal{B}_2$ against the IND$-CPA security of the underlying $\mathsf{E}$. $\mathcal{B}_2$ just simulates $\mathcal{A}$'s view as the challenger in $\mathbf{G}_1/\mathbf{G}_2$ does, but for $\mathcal{A}$'s encryption query, $\mathcal{B}_2$ calls its own encryption oracle to get $C$ and then finish the rest of the simulation. It is easy to see that $\mathcal{B}_2$ simulates $\mathcal{A}$'s view in $\mathbf{G}_1$ in the real world and $\mathbf{G}_2$ in the ideal world. In the end, adversary $\mathcal{B}_2$ outputs the bit that $\mathcal{A}$ outputs. Therefore, we have

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}_2). \tag{27}$$

Finally, we observe that game $\mathbf{G}_2$ is identical to $\mathsf{RAND}_{\mathsf{EtH}}^{\mathcal{A}}$ (see Figure 4) equipped with a random oracle H. This is because, in game $\mathbf{G}_2$, for each Enc query with a distinct nonce, its output is an independent uniform random string sampled from $\{0,1\}^{\mathsf{EtH.clen}(|M|)} = \{0,1\}^{\mathsf{E.clen}(|M|)+t}$.

The proof is concluded by combining equations (26) and (27). □

### B.10 Proof of Theorem 10 (INT-CTXT Security of EtH)

*Proof.* We define three games $\mathbf{G}_0 \sim \mathbf{G}_2$ as shown in Figure 20.

Game $\mathbf{G}_0$ is the same as $\mathsf{INT\text{-}CTXT}_{\mathsf{EtH}}^{\mathcal{A}}$ (see Figure 4) equipped with a random oracle H. Therefore, we have $\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] = \mathbf{Adv}_{\mathsf{EtH}}^{\text{int-ctxt}}(\mathcal{A})$.

Game $\mathbf{G}_1$ is the same as $\mathbf{G}_0$, except that the verification oracle does not check if the ciphertext $C$ can be successfully decrypted (note that the $\mathrm{T}_T$ operations can be ignored in $\mathbf{G}_1$). As a result, the winning condition in $\mathbf{G}_1$ is relaxed, i.e., if $\mathcal{A}$ wins in $\mathbf{G}_0$ then it must also win in $\mathbf{G}_1$. To see this, suppose $\mathcal{A}$ wins in $\mathbf{G}_0$ by querying $(N^*, AD^*, (C^*, T^*))$ and before this query $\mathcal{A}$ did not already win in $\mathbf{G}_1$. In this case, $\mathcal{A}$'s views in $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical. This is because before that winning query each verification query with a valid $T$ must satisfy $(N, AD, (C, T)) \in \mathcal{Q}$ (otherwise $\mathcal{A}$ wins in $\mathbf{G}_1$), which implies that $M \neq \perp$ and hence the verification oracle returns **true** in both games. Then, by the winning condition in $\mathbf{G}_1$, the above query also allows $\mathcal{A}$ to win in $\mathbf{G}_1$. Therefore, we have

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right]. \tag{28}$$

Game $\mathbf{G}_2$ is the same as $\mathbf{G}_1$, except that the committing tag is now lazily sampled from the table $\mathrm{T}_T$ in both oracles, where the table is indexed by the $(N, AD, C)$ triple. We claim that $\mathbf{G}_1$ and $\mathbf{G}_2$ behave identically unless the $\mathsf{bad}$ event happens, which occurs if the adversary $\mathcal{A}$ ever queries H with an input that is prefixed with the key $K$. The only syntactical difference between the two games are in the generation of $T$. If $\mathsf{bad}$ does not occur, in both games the committing tag $T$ is either randomly sampled for a new $(N, AD, C)$ triple or equal to the same previously sampled value if $(N, AD, C)$ is not new. Hence the two games are indeed identical-until-$\mathsf{bad}$. By the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr[\mathsf{bad}]. \tag{29}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by considering a game $\mathbf{G}_*$ also shown in Figure 20. $\mathbf{G}_*$ is defined such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers $\mathsf{bad}$ in game $\mathbf{G}_1/\mathbf{G}_2$. Therefore, we have

$$\Pr[\mathsf{bad}] = \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]. \tag{30}$$

| Game $\mathbf{G}_0$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \mathcal{K}$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X)$ | $C \leftarrow \mathsf{E.Enc}(K, N, M)$ | $T' \leftarrow \mathrm{H}(K, N, AD, C)$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | $\wedge\ X = (K, *)$ : | $T \leftarrow \mathrm{H}(K, N, AD, C)$ | **if** $T' \neq T$ : **return false** |
| $\mathcal{Q} \leftarrow \varnothing$ | $\mathsf{bad} \leftarrow$ **true** | $\mathcal{Q} \xleftarrow{\cup} \{(N, AD, (C, T))\}$ | $M \leftarrow \mathsf{E.Dec}(K, N, C)$ |
| $\mathrm{win} \leftarrow 0$ | **if** $\mathrm{T_H}[X] = \perp$ : | **return** $(C, T)$ | **if** $M \neq \perp \wedge (N, AD, (C, T)) \notin \mathcal{Q}$ : |
| $\mathcal{A}^{\mathrm{H,Enc,Ver}}$ | $\mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | | $\mathrm{win} \leftarrow 1$ |
| **return** win | **return** $\mathrm{T_H}[X]$ | | **return** $M \neq \perp$ |

| Game $\boxed{\mathbf{G}_1}$, $\mathbf{G}_2$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \mathcal{K}$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X)$ | $C \leftarrow \mathsf{E.Enc}(K, N, M)$ | **if** $\mathrm{T}_T[(N, AD, C)] = \perp$ : |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | $\wedge\ X = (K, *)$ : | **if** $\mathrm{T}_T[(N, AD, C)] = \perp$ : | $\mathrm{T}_T[(N, AD, C)] \xleftarrow{\$} \{0,1\}^t$ |
| $\mathrm{T}_T[\cdot] \leftarrow \perp$ | $\mathsf{bad} \leftarrow$ **true** | $\mathrm{T}_T[(N, AD, C)] \xleftarrow{\$} \{0,1\}^t$ | $T' \leftarrow \mathrm{T}_T[(N, AD, C)]$ |
| $\mathcal{Q} \leftarrow \varnothing$ | **if** $\mathrm{T_H}[X] = \perp$ : | $T \leftarrow \mathrm{T}_T[(N, AD, C)]$ | $\boxed{T' \leftarrow \mathrm{H}(K, N, AD, C)}$ |
| $\mathrm{win} \leftarrow 0$ | $\mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $\boxed{T \leftarrow \mathrm{H}(K, N, AD, C)}$ | **if** $T' \neq T$ : **return false** |
| $\mathcal{A}^{\mathrm{H,Enc,Ver}}$ | **return** $\mathrm{T_H}[X]$ | $\mathcal{Q} \xleftarrow{\cup} \{(N, AD, (C, T))\}$ | **if** $(N, AD, (C, T)) \notin \mathcal{Q}$ : |
| **return** win | | **return** $(C, T)$ | $\mathrm{win} \leftarrow 1$ |
| | | | **return true** |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | $(K', *) \leftarrow X$ | $C \leftarrow \mathsf{E.Enc}(K, N, M)$ | **if** $\mathrm{T}_T[(N, AD, C)] = \perp$ : |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K'\}$ | **if** $\mathrm{T}_T[(N, AD, C)] = \perp$ : | $\mathrm{T}_T[(N, AD, C)] \xleftarrow{\$} \{0,1\}^t$ |
| $\mathrm{T}_T[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp$ : | $\mathrm{T}_T[(N, AD, C)] \xleftarrow{\$} \{0,1\}^t$ | $T' \leftarrow \mathrm{T}_T[(N, AD, C)]$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | $\mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $T \leftarrow \mathrm{T}_T[(N, AD, C)]$ | **if** $T' \neq T$ : **return false** |
| $\mathcal{A}^{\mathrm{H,Enc,Ver}}$ | **return** $\mathrm{T_H}[X]$ | **return** $(C, T)$ | **return true** |
| **return** $K \in \mathcal{Q}^*$ | | | |

Fig. 20: Games $\mathbf{G}_0 \sim \mathbf{G}_2$ and $\mathbf{G}_*$ in the proof of Theorem 10, where $\mathbf{G}_1$ contains boxed content while $\mathbf{G}_2$ does not.

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}$ against the IND$-CPA security of the underlying $\mathsf{E}$. $\mathcal{B}$ starts by initializing the set $\mathcal{Q}^*$ and tables $\mathrm{T_H}, \mathrm{T}_T$ (to an empty set and empty tables, respectively), and then it runs $\mathcal{A}$. To answer $\mathcal{A}$'s encryption query on input $(N, AD, M)$, $\mathcal{B}$ calls its own encryption oracle $\mathrm{Enc}'$ with input $(N, M)$ and gets back $C$, then lazily samples $T$ with the help of table $\mathrm{T}_T$. At the end, it returns $(C, T)$ to $\mathcal{A}$. To answer $\mathcal{A}$'s verification query on input $(N, AD, (C, T))$, $\mathcal{B}$ lazily samples $T'$ with the help of the table $\mathrm{T}_T$ and checks if $T' = T$. If not, $\mathcal{B}$ returns **false**, otherwise it returns **true**. For $\mathcal{A}$'s random oracle query $X = (K', *)$, $\mathcal{B}$ records $K'$ in $\mathcal{Q}^*$ and answers the query via lazy sampling of the table $\mathrm{T_H}$. After $\mathcal{A}$ terminates, $\mathcal{B}$ picks a nonce $N^*$ that is different from those used previous encryption queries and then calls its own encryption oracle with input $(N^*, 0^\kappa)$ to get back $C^*$. Then, for each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}$ computes $C_i \leftarrow \mathsf{E.Enc}(K_i, N^*, 0^\kappa)$ and checks if $C_i = C^*$ holds. If this holds for any of the keys in $\mathcal{Q}^*$, $\mathcal{B}$ guesses that it was in the real world $\mathsf{REAL}_{\mathsf{E}}^{\mathcal{B}}$ and outputs 1; otherwise, it outputs 0, guessing that it was in the ideal world $\mathsf{RAND}_{\mathsf{E}}^{\mathcal{B}}$.

If $\mathcal{B}$ was in the real world, then it simulates $\mathcal{A}$'s view in $\mathbf{G}_*$ perfectly and hence we have $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathsf{REAL}_{\mathsf{E}}^{\mathcal{B}} \Rightarrow 1\right]$. Here we have an inequality to capture "false positives" in the above $C_i = C^*$ check. On the other hand, if $\mathcal{B}$ was in the ideal world, the probability that a random ciphertext $C^*$ equals $\mathsf{E.Enc}(K_i, N^*, 0^\kappa)$ for some $K_i \in \mathcal{Q}^*$ is at most $|\mathcal{Q}^*|/2^{\mathsf{E.clen}(\kappa)}$. Since $|\mathcal{Q}^*| \leq q_{\mathrm{h}}$, we have $\Pr\left[\mathsf{RAND}_{\mathsf{E}}^{\mathcal{B}} \Rightarrow 1\right] \leq q_{\mathrm{h}}/2^{\mathsf{E.clen}(\kappa)}$. Recalling that $\mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}) = \Pr\left[\mathsf{REAL}_{\mathsf{E}}^{\mathcal{B}} \Rightarrow 1\right] - \Pr\left[\mathsf{RAND}_{\mathsf{E}}^{\mathcal{B}} \Rightarrow 1\right]$ and $\mathsf{E.clen}(\kappa) \geq \kappa$, we have

$$\mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}) \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_{\mathrm{h}}}{2^{\mathsf{E.clen}(\kappa)}} \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_{\mathrm{h}}}{2^\kappa}, \tag{31}$$

and it follows from equations (29)∼(31) that

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}) + \frac{q_{\text{h}}}{2^{\kappa}}. \tag{32}$$

Finally, it is left to bound $\Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right]$. Suppose $\mathcal{A}$ wins by querying $\text{Ver}(N^*, AD^*, (C^*, T^*))$, then we know $(C^*, T^*)$ was not output by any $\text{Enc}(N^*, AD^*, \cdot)$ query and $\text{Ver}(N^*, AD^*, (C^*, T^*)) = 1$. The latter implies that $T^* = \text{T}_T[(N^*, AD^*, C^*)]$. If $C^*$ was output by an $\text{Enc}(N^*, AD^*, \cdot)$ query, then $T^* = \text{T}_T[(N^*, AD^*, C^*)]$ is the output committing tag and this contradicts with the above winning condition. Therefore, $\mathcal{A}$ has never learned $\text{T}_T[(N^*, AD^*, C^*)]$ by encryption queries and hence it can only guess $T^*$ when making verification queries. Since $\mathcal{A}$ can make $q_{\text{v}}$ verification queries and $T^*$ is randomly sampled from $\{0,1\}^t$, the probability that $\mathcal{A}$ wins is at most $q_{\text{v}}/2^t$. Therefore, we have

$$\Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] \leq \frac{q_{\text{v}}}{2^t}. \tag{33}$$

The proof is concluded by combining equations (28), (32) and (33). □

### B.11 Proof of Theorem 11 (CMT Security of EtH)

*Proof.* In order to win the $\mathsf{CMT}_{\mathsf{EtH}}^{\mathcal{A}}$ game (see Figure 5), $\mathcal{A}$ needs to output two tuples $(K_1, N_1, AD_1, M_1)$, $(K_2, N_2, AD_2, M_2)$ such that they are encrypted to the same ciphertext $(C, T)$ and $(K_1, N_1, AD_1) \neq (K_2, N_2, AD_2)$. By construction of EtH, this implies that the hashes of $(K_1, N_1, AD_1, C)$ and $(K_2, N_2, AD_2, C)$ collide. Therefore, we can construct $\mathcal{B}$ as follows: it first invokes $\mathcal{A}$, then extracts $(K_1, N_1, AD_1)$ and $(K_2, N_2, AD_2)$ from $\mathcal{A}$'s output. It continues by computing $C$ and outputting the tuple pair $((K_1, N_1, AD_1, C), (K_2, N_2, AD_2, C))$. □

### B.12 Proof of Theorem 12 (CDY$ Security of EtH)

*Proof.* We define two games $\mathbf{G}_0$ and $\mathbf{G}_1$ as shown in Figure 21.

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\text{H}(X)$ |
|---|---|
| $\text{T}_{\text{H}}[\cdot] \leftarrow \perp$ ; $K \xleftarrow{\$} \{0,1\}^{\kappa}$ ; $N \xleftarrow{\$} \mathcal{N}$ ; $AD \xleftarrow{\$} \mathcal{AD}$ ; $M \xleftarrow{\$} \mathcal{M}$ | **if** $X = (K, *)$ : |
| $C \leftarrow \mathsf{E.Enc}(K, N, M)$ ; $T \xleftarrow{\$} \{0,1\}^t$ | $\mathsf{bad} \leftarrow \mathbf{true}$ |
| $\boxed{\text{T}_{\text{H}}[(K, N, AD, C)] \leftarrow T}$ | **if** $\text{T}_{\text{H}}[X] = \perp : \text{T}_{\text{H}}[X] \xleftarrow{\$} \{0,1\}^t$ |
| $(K', N', AD', M') \xleftarrow{\$} \mathcal{A}^{\text{H}}(C, T)$ | **return** $\text{T}_{\text{H}}[X]$ |
| $C' \leftarrow \mathsf{E.Enc}(K', N', M')$ ; $T' \leftarrow \text{H}(K', N', AD', C')$ | |
| **return** $(C', T') = (C, T)$ | |

| Game $\mathbf{G}_*$ | Oracle $\text{H}(X)$ |
|---|---|
| $\text{T}_{\text{H}}[\cdot] \leftarrow \perp$ ; $K \xleftarrow{\$} \{0,1\}^{\kappa}$ ; $N \xleftarrow{\$} \mathcal{N}$ ; $AD \xleftarrow{\$} \mathcal{AD}$ ; $M \xleftarrow{\$} \mathcal{M}$ | $(K^*, *) \leftarrow X$ |
| $C \leftarrow \mathsf{E.Enc}(K, N, M)$ ; $T \xleftarrow{\$} \{0,1\}^t$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K^*\}$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | **if** $\text{T}_{\text{H}}[X] = \perp : \text{T}_{\text{H}}[X] \xleftarrow{\$} \{0,1\}^t$ |
| $(K', N', AD', M') \xleftarrow{\$} \mathcal{A}^{\text{H}}(C, T)$ | **return** $\text{T}_{\text{H}}[X]$ |
| $C' \leftarrow \mathsf{E.Enc}(K', N', M')$ ; $\text{H}(K', N', AD', C')$ | |
| **return** $K \in \mathcal{Q}^*$ | |

Fig. 21: Games $\mathbf{G}_0, \mathbf{G}_1$ and $\mathbf{G}_*$ in the proof of Theorem 12, where $\mathbf{G}_0$ contains boxed content while $\mathbf{G}_1$ does not.

Game $\mathbf{G}_0$ is the same as $\mathsf{CDY}_{\mathsf{EtH},\mathsf{S}_\$}^{\mathcal{A}}$ (see Figure 6) equipped with a random oracle H. Therefore, we have $\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] = \mathbf{Adv}_{\mathsf{EtH},\mathsf{S}_\$}^{\text{cdy}}(\mathcal{A}) = \mathbf{Adv}_{\mathsf{EtH}}^{\text{cdy\$}}(\mathcal{A})$.

Game $\mathbf{G}_1$ is the same as game $\mathbf{G}_0$, except that the committing tag $T$, calculated by the challenger, is now independent of queries to the random oracle H. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically

unless the $\mathsf{bad}$ event happens, where the random oracle H is ever queried with an input that is prefixed with the key $K$ sampled by the challenger. This is because, if $\mathsf{bad}$ does not occur, then $T$ in $\mathbf{G}_0$ is also independent of random oracle queries made by $\mathcal{A}$ and by the challenger (to compute $T'$). Therefore, by the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr[\mathsf{bad}]. \tag{34}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by constructing a game $\mathbf{G}_*$ also shown in Figure 21. $\mathbf{G}_*$ is defined such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers $\mathsf{bad}$ in game $\mathbf{G}_0/\mathbf{G}_1$ when (1) $\mathcal{A}$ queried the random oracle H with an input prefixed with $K$ or (2) $\mathcal{A}$ output $(K', N', AD', M')$ with $K' = K$ and then $\mathrm{H}(K', N', AD', C')$ was queried by the challenger. Therefore, we have

$$\Pr[\mathsf{bad}] = \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]. \tag{35}$$

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}$ against the IND$-CPA security of the underlying encryption scheme $\mathsf{E}$. $\mathcal{B}$ starts by initializing the set $\mathcal{Q}^*$ and the table $\mathrm{T_H}$ (to an empty set and empty table, respectively). Then it samples the tuple $(N, AD, M)$ uniformly at random, and calls its own encryption oracle with input $(N, M)$ to derive $C$. It then samples $T$ randomly and runs $\mathcal{A}$ with input $(C, T)$. For $\mathcal{A}$'s each random oracle query $X = (K^*, *)$, $\mathcal{B}$ records $K^*$ in $\mathcal{Q}^*$ and answers the query via lazy sampling. After $\mathcal{A}$ terminates and outputs $(K', N', AD', M')$, $\mathcal{B}$ takes a nonce $N^* \neq N$, and queries its own encryption oracle with the input $(N^*, 0^\kappa)$ to get back $C'$. Then it "queries" the random oracle H with the input $(K', N', AD', C')$ and continues as follows. For each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}$ computes $C_i \leftarrow \mathsf{E.Enc}(K_i, N^*, 0^\kappa)$ and checks if $C_i = C$. If this holds for any of the keys in $\mathcal{Q}^*$, $\mathcal{B}$ guesses that it was in the real world $\mathsf{REAL}_{\mathsf{E}}^{\mathcal{B}}$ and outputs 1; otherwise, it outputs 0, guessing that it was in the ideal world $\mathsf{RAND}_{\mathsf{E}}^{\mathcal{B}}$.

If $\mathcal{B}$ was in the real world, then it simulates $\mathcal{A}$'s view in $\mathbf{G}_*$ perfectly and hence we have $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathsf{REAL}_{\mathsf{E}}^{\mathcal{B}} \Rightarrow 1\right]$. Here we have an inequality to capture "false positives" in the above $C_i = C$ check. On the other hand, if $\mathcal{B}$ was in the ideal world, the probability that a random ciphertext $C$ equals $\mathsf{E.Enc}(K_i, N^*, 0^\kappa)$ for some $K_i \in \mathcal{Q}^*$ is at most $|\mathcal{Q}^*|/2^\kappa$. Since $|\mathcal{Q}^*| \leq q_{\mathrm{h}} + 1$, we have $\Pr\left[\mathsf{RAND}_{\mathsf{E}}^{\mathcal{B}} \Rightarrow 1\right] \leq (q_{\mathrm{h}} + 1)/2^\kappa$. Recalling that $\mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}) = \Pr\left[\mathsf{REAL}_{\mathsf{E}}^{\mathcal{B}} \Rightarrow 1\right] - \Pr\left[\mathsf{RAND}_{\mathsf{E}}^{\mathcal{B}} \Rightarrow 1\right]$, we have

$$\mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}) \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_{\mathrm{h}} + 1}{2^\kappa}, \tag{36}$$

and it follows from equations (34)∼(36) that

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}) + \frac{q_{\mathrm{h}} + 1}{2^\kappa}. \tag{37}$$

Finally, we observe that in game $\mathbf{G}_1$ the committing tag $T$ in the challenge ciphertext $(C, T)$ is independent of random oracle queries and hence independent of $T'$, so in order for $\mathcal{A}$ to win one of the random oracle queries must output a committing tag equal to the independent random $T$. This probability is bounded by $(q_{\mathrm{h}} + 1)/2^t$. Therefore, we have

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \frac{q_{\mathrm{h}} + 1}{2^t}. \tag{38}$$

The proof is concluded by combining equations (37) and (38). □

### B.13 Proof of Theorem 13 (MR-IND$-CPA Security of AEtH)

*Proof.* We define four games $\mathbf{G}_0 \sim \mathbf{G}_3$ as shown in Figure 22.

Game $\mathbf{G}_0$ is the same as $\mathsf{REAL}_{\mathsf{AEtH}}^{\mathcal{A}}$ (see Figure 4) equipped with a random oracle H. Therefore, we have $\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[\mathsf{REAL}_{\mathsf{AEtH}}^{\mathcal{A}} \Rightarrow 1\right]$.

Game $\mathbf{G}_1$ is the same as $\mathbf{G}_0$, except that the committing tag is now lazily sampled from the table $\mathrm{T}_T$ in Enc oracle, where the table is indexed by the queried $(N, AD, C|_\tau)$ triple. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically unless the $\mathsf{bad}$ event happens, which occurs if the adversary $\mathcal{A}$ ever queries H with an input that is prefixed with the key $K$. The only syntactical difference between the two games is in the generation of $T$. If $\mathsf{bad}$ does not occur, in both games the committing tag $T$ is either randomly

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | **if** $\mathcal{A}$ queries $\mathrm{H} \wedge X = (K, *)$ : | $C \leftarrow \mathsf{AE.Enc}(K, N, AD, M)$ |
| $\mathrm{T}_\mathrm{H}[\cdot] \leftarrow \perp$ ; $\mathrm{T}_T[\cdot] \leftarrow \perp$ | $\quad \mathsf{bad} \leftarrow \mathbf{true}$ | **if** $\mathrm{T}_T[(N, AD, C\|_\tau)] = \perp$ : |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H},\mathrm{Enc}}$ | **if** $\mathrm{T}_\mathrm{H}[X] = \perp : \mathrm{T}_\mathrm{H}[X] \xleftarrow{\$} \{0,1\}^t$ | $\quad \mathrm{T}_T[(N, AD, C\|_\tau)] \xleftarrow{\$} \{0,1\}^t$ |
| **return** $b'$ | **return** $\mathrm{T}_\mathrm{H}[X]$ | $T \leftarrow \mathrm{T}_T[(N, AD, C\|_\tau)]$ |
| | | $\boxed{T \leftarrow \mathrm{H}(K, N, AD, C\|_\tau)}$ |
| | | **return** $(C, T)$ |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | $(K', *) \leftarrow X$ | $C \leftarrow \mathsf{AE.Enc}(K, N, AD, M)$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K'\}$ | **if** $\mathrm{T}_T[(N, AD, C\|_\tau)] = \perp$ : |
| $\mathrm{T}_\mathrm{H}[\cdot] \leftarrow \perp$ ; $\mathrm{T}_T[\cdot] \leftarrow \perp$ | **if** $\mathrm{T}_\mathrm{H}[X] = \perp : \mathrm{T}_\mathrm{H}[X] \xleftarrow{\$} \{0,1\}^t$ | $\quad \mathrm{T}_T[(N, AD, C\|_\tau)] \xleftarrow{\$} \{0,1\}^t$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H},\mathrm{Enc}}$ | **return** $\mathrm{T}_\mathrm{H}[X]$ | $T \leftarrow \mathrm{T}_T[(N, AD, C\|_\tau)]$ |
| **return** $K \in \mathcal{Q}^*$ | | **return** $(C, T)$ |

| Game $\mathbf{G}_2$, $\boxed{\mathbf{G}_3}$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $\mathrm{T}_\mathrm{H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T}_\mathrm{H}[X] = \perp : \mathrm{T}_\mathrm{H}[X] \xleftarrow{\$} \{0,1\}^t$ | $C \xleftarrow{\$} \{0,1\}^{\mathsf{AE.clen}(\|M\|)}$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H},\mathrm{Enc}}$ | **return** $\mathrm{T}_\mathrm{H}[X]$ | **if** $\mathrm{T}_T[(N, AD, C\|_\tau)] = \perp$ : |
| **return** $b'$ | | $\quad \mathrm{T}_T[(N, AD, C\|_\tau)] \xleftarrow{\$} \{0,1\}^t$ |
| | | $T \leftarrow \mathrm{T}_T[(N, AD, C\|_\tau)]$ |
| | | $\boxed{T \xleftarrow{\$} \{0,1\}^t}$ |
| | | **return** $(C, T)$ |

Fig. 22: Games $\mathbf{G}_0 \sim \mathbf{G}_3$ and $\mathbf{G}_*$ in the proof of Theorem 13, where $\mathbf{G}_0$ and $\mathbf{G}_3$ contain boxed content while $\mathbf{G}_1$ and $\mathbf{G}_2$ do not.

sampled for new $(N, AD, C|_\tau)$ triple, or gets the same previously sampled value if $(N, AD, C|_\tau)$ is not new. Hence the two games are indeed identical-until-$\mathsf{bad}$. By the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr[\mathsf{bad}]. \tag{39}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by considering a game $\mathbf{G}_*$ also shown in Figure 22. Here $\mathbf{G}_*$ is just defined to clarify the $\mathsf{bad}$ event, such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers $\mathsf{bad}$ in game $\mathbf{G}_0/\mathbf{G}_1$. Therefore, we have

$$\Pr[\mathsf{bad}] = \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]. \tag{40}$$

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}_1$ against the MR-IND$-CPA security of the underlying $\mathsf{AE}$. $\mathcal{B}_1$ starts by initializing the set $\mathcal{Q}^*$ and tables $\mathrm{T}_\mathrm{H}, \mathrm{T}_T$ (to an empty set and empty tables, respectively), and then it runs $\mathcal{A}$. To answer $\mathcal{A}$'s encryption query on input $(N, AD, M)$, $\mathcal{B}_1$ calls its own encryption oracle with input $(N, AD, M)$ and gets back $C$, then lazily samples $T$ with the help of table $\mathrm{T}_T$. At the end, it returns $(C, T)$. For $\mathcal{A}$'s random oracle query $X = (K', *)$, $\mathcal{B}_1$ records $K'$ in $\mathcal{Q}^*$ and answers the query via lazy sampling. After $\mathcal{A}$ terminates, $\mathcal{B}_1$ ignores its output and queries its own encryption oracle with input $(N^*, AD^*, 0^\kappa)$ and gets back $C^*$, where $N^*$ is a nonce different from those used in previous encryption queries and $AD^*$ is any valid additional data.Then, for each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}_1$ computes $C_i \leftarrow \mathsf{AE.Enc}(K_i, N^*, AD^*, 0^\kappa)$ and checks if $C_i = C^*$ holds. If this holds for any of the keys in $\mathcal{Q}^*$, $\mathcal{B}_1$ guesses that it was in the real world $\mathsf{REAL}_{\mathsf{AE}}^{\mathcal{B}_1}$ and outputs 1; otherwise, it outputs 0, guessing that it was in the ideal world $\mathsf{RAND}_{\mathsf{AE}}^{\mathcal{B}_1}$.

If $\mathcal{B}_1$ was in the real world, then it simulates $\mathcal{A}$'s view in $\mathbf{G}_*$ perfectly and hence we have $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathsf{REAL}_{\mathsf{AE}}^{\mathcal{B}_1} \Rightarrow 1\right]$. Here we have an inequality to capture "false positives" in the above $C_i = C^*$ check. On the other hand, if $\mathcal{B}_1$ was in the ideal world, the probability that a random ciphertext $C^*$ equals $\mathsf{AE.Enc}(K_i, N^*, AD^*, 0^\kappa)$ for some $K_i \in \mathcal{Q}^*$ is at most $|\mathcal{Q}^*|/2^{\mathsf{AE.clen}(\kappa)}$. Since

$|\mathcal{Q}^*| \leq q_{\mathrm{h}}$, we have $\Pr\left[\mathsf{RAND}_{\mathsf{AE}}^{\mathcal{B}_1} \Rightarrow 1\right] \leq q_{\mathrm{h}}/2^{\mathsf{AE.clen}(\kappa)}$. Then, recalling that $\mathbf{Adv}_{\mathsf{AE}}^{\text{mr-ind\$-cpa}}(\mathcal{B}_1) = \Pr\left[\mathsf{REAL}_{\mathsf{AE}}^{\mathcal{B}_1} \Rightarrow 1\right] - \Pr\left[\mathsf{RAND}_{\mathsf{AE}}^{\mathcal{B}_1} \Rightarrow 1\right]$ and $\mathsf{AE.clen}(\kappa) \geq \kappa$, we have

$$\mathbf{Adv}_{\mathsf{AE}}^{\text{mr-ind\$-cpa}}(\mathcal{B}_1) \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_{\mathrm{h}}}{2^{\mathsf{AE.clen}(\kappa)}} \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_{\mathrm{h}}}{2^{\kappa}}, \tag{41}$$

and it follows from equations (39)∼(41) that

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{AE}}^{\text{mr-ind\$-cpa}}(\mathcal{B}_1) + \frac{q_{\mathrm{h}}}{2^{\kappa}}. \tag{42}$$

We continue by making a game hop from $\mathbf{G}_1$ to $\mathbf{G}_2$. As shown in Figure 22, game $\mathbf{G}_2$ is the same as game $\mathbf{G}_1$, except that now in the encryption oracle Enc the ciphertext $C$ is sampled uniformly at random. We bound the gap between $\mathbf{G}_1$ and $\mathbf{G}_2$ by constructing an adversary $\mathcal{B}_2$ against the MR-IND\$-CPA security of the underlying $\mathsf{AE}$. $\mathcal{B}_2$ just simulates $\mathcal{A}$'s view as the challenger in $\mathbf{G}_1/\mathbf{G}_2$ does, but for $\mathcal{A}$'s encryption query, $\mathcal{B}_2$ calls its own encryption oracle to get $C$ and then it finishes the rest of the simulation. It is easy to see that $\mathcal{B}_2$ simulates $\mathcal{A}$'s view in $\mathbf{G}_1$ in the real world and $\mathbf{G}_2$ in the ideal world. In the end, adversary $\mathcal{B}_2$ outputs the bit that $\mathcal{A}$ outputs. Therefore, we have

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{AE}}^{\text{mr-ind\$-cpa}}(\mathcal{B}_2). \tag{43}$$

We make the final game hop from $\mathbf{G}_2$ to $\mathbf{G}_3$ and for this we define a bad event $\mathsf{bad}'$. We say that $\mathsf{bad}'$ occurs if $\mathcal{A}$ queried the encryption oracle with input $(N, AD, M)$ where the table $\mathrm{T}_T$ was already defined for an entry $(N, AD, C|_\tau)$. If $\mathsf{bad}'$ does not happen, games $\mathbf{G}_2$ and $\mathbf{G}_3$ behave identically, i.e. the committing tag $T$ will be a random string for $\mathcal{A}$'s every query. The games $\mathbf{G}_2$ and $\mathbf{G}_3$ behave identically if it does not happen for some encryption query $(N, AD, M)$ that the table $\mathrm{T}_T$ was already set at point $(N, AD, C|_\tau)$. By the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_3^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathsf{bad}'\right] = \binom{q_{\mathrm{e}}}{2} \frac{1}{2^{\tau}} \leq \frac{q_{\mathrm{e}}^2}{2^{\tau+1}}, \tag{44}$$

where we bound the probability of $\mathsf{bad}'$ happening by the probability that a collision in $C|_\tau$ value for two different queries occurs.

Finally, we observe that game $\mathbf{G}_3$ is identical to $\mathsf{RAND}_{\mathsf{AEtH}}^{\mathcal{A}}$ (see Figure 4) equipped with a random oracle H. This is because, in game $\mathbf{G}_3$, for each Enc query, its output is an independent uniform random string sampled from $\{0,1\}^{\mathsf{AEtH.clen}(|M|)} = \{0,1\}^{\mathsf{AE.clen}(|M|)+t}$.

The proof is concluded by combining equations (42)∼(44). □

### B.14 Proof of Theorem 14 (MR-INT-CTXT Security of AEtH)

*Proof.* We define two games $\mathbf{G}_0$ and $\mathbf{G}_1$ as shown in Figure 23.

Game $\mathbf{G}_0$ is the same as $\mathsf{INT\text{-}CTXT}_{\mathsf{AEtH}}$ (see Figure 4) equipped with a random oracle H. Therefore, we have $\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] = \mathbf{Adv}_{\mathsf{AEtH}}^{\text{int-ctxt}}(\mathcal{A})$.

Game $\mathbf{G}_1$ is the same as $\mathbf{G}_0$, except that the committing tag is now lazily sampled from the table $\mathrm{T}_T$ in both oracles, where the table is indexed by the $(N, AD, C|_\tau)$ triple. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically unless the $\mathsf{bad}$ event happens, which occurs if the adversary $\mathcal{A}$ ever queries H with an input that is prefixed with the key $K$. The two syntactical difference between the two games are in the generation of $T$. If $\mathsf{bad}$ does not occur, in both games the committing tag $T$ is either randomly sampled for new $(N, AD, C|_\tau)$ triple, or gets the same previously sampled value if $(N, AD, C|_\tau)$ is not new. Hence the two games are indeed identical-until-$\mathsf{bad}$. By the Difference Lemma of the game-based proof technique [31], we have

$$\left|\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \Pr[\mathsf{bad}]. \tag{45}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by considering a game $\mathbf{G}_*$ also shown in Figure 23. $\mathbf{G}_*$ is defined such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers $\mathsf{bad}$ in $\mathbf{G}_0/\mathbf{G}_1$. Therefore, we have

$$\Pr[\mathsf{bad}] = \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]. \tag{46}$$

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}_1$ against the INT-CTXT security of the underlying $\mathsf{AE}$. $\mathcal{B}_1$ starts by initializing the set $\mathcal{Q}^*$ and tables $\mathrm{T}_{\mathrm{H}}, \mathrm{T}_T$ (to an empty set and empty tables,

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \mathcal{K}$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X)$ | $C \leftarrow \mathsf{AE.Enc}(K, N, AD, M)$ | **if** $\mathrm{T}_T[(N, AD, C\|_\tau)] = \bot$ : |
| $\mathrm{T}_\mathrm{H}[\cdot] \leftarrow \bot$ | $\wedge\, X = (K, *)$ : | **if** $\mathrm{T}_T[(N, AD, C\|_\tau)] = \bot$ : | $\mathrm{T}_T[(N, AD, C\|_\tau)] \xleftarrow{\$} \{0,1\}^t$ |
| $\mathrm{T}_T[\cdot] \leftarrow \bot$ | $\mathsf{bad} \leftarrow \mathbf{true}$ | $\mathrm{T}_T[(N, AD, C\|_\tau)] \xleftarrow{\$} \{0,1\}^t$ | $T' \leftarrow \mathrm{T}_T[(N, AD, C\|_\tau)]$ |
| $\mathcal{Q} \leftarrow \varnothing$ | **if** $\mathrm{T}_\mathrm{H}[X] = \bot$ : | $T \leftarrow \mathrm{T}_T[(N, AD, C\|_\tau)]$ | $\boxed{T' \leftarrow \mathrm{H}(K, N, AD, C\|_t)}$ |
| $\mathrm{win} \leftarrow 0$ | $\mathrm{T}_\mathrm{H}[X] \xleftarrow{\$} \{0,1\}^t$ | $\boxed{T \leftarrow \mathrm{H}(K, N, AD, C\|_\tau)}$ | **if** $T' \neq T$ : **return false** |
| $\mathcal{A}^{\mathrm{H,Enc,Ver}}$ | **return** $\mathrm{T}_\mathrm{H}[X]$ | $\mathcal{Q} \xleftarrow{\cup} \{(N, AD, (C, T))\}$ | $M \leftarrow \mathsf{AE.Dec}(N, AD, C)$ |
| **return** win | | **return** $(C, T)$ | **if** $M \neq \bot \wedge (N, AD, (C, T)) \notin \mathcal{Q}$ : |
| | | | $\mathrm{win} \leftarrow 1$ |
| | | | **return** $M \neq \bot$ |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | $(K', *) \leftarrow X$ | $C \leftarrow \mathsf{AE.Enc}(K, AD, \varepsilon, M)$ | **if** $\mathrm{T}_T[(N, AD, C\|_\tau)] = \bot$ : |
| $\mathrm{T}_\mathrm{H}[\cdot] \leftarrow \bot$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K'\}$ | **if** $\mathrm{T}_T[(N, AD, C\|_\tau)] = \bot$ : | $\mathrm{T}_T[(N, AD, C\|_\tau)] \xleftarrow{\$} \{0,1\}^t$ |
| $\mathrm{T}_T[\cdot] \leftarrow \bot$ | **if** $\mathrm{T}_\mathrm{H}[X] = \bot$ : | $\mathrm{T}_T[(N, AD, C\|_\tau)] \xleftarrow{\$} \{0,1\}^t$ | $T' \leftarrow \mathrm{T}_T[(N, AD, C\|_\tau)]$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | $\mathrm{T}_\mathrm{H}[X] \xleftarrow{\$} \{0,1\}^t$ | $T \leftarrow \mathrm{T}_T[(N, AD, C\|_\tau)]$ | **if** $T \neq T'$ : **return false** |
| $\mathcal{A}^{\mathrm{H,Enc,Ver}}$ | **return** $\mathrm{T}_\mathrm{H}[X]$ | **return** $(C, T)$ | $M \leftarrow \mathsf{AE.Dec}(N, AD, C)$ |
| **return** $K \in \mathcal{Q}^*$ | | | **return** $M \neq \bot$ |

Fig. 23: Games $\mathbf{G}_0$, $\mathbf{G}_1$ and $\mathbf{G}_*$ in the proof of Theorem 10, where $\mathbf{G}_0$ contains boxed content while $\mathbf{G}_1$ does not.

respectively), and then it runs $\mathcal{A}$. To answer $\mathcal{A}$'s encryption query on input $(N, AD, M)$, $\mathcal{B}_1$ calls its own encryption oracle with input $(N, AD, M)$ and gets back $C$, then lazily samples $T$ with the help of table $\mathrm{T}_T$. At the end, it returns $(C, T)$ back to $\mathcal{A}$. To answer $\mathcal{A}$'s verification query on input $(N, AD, (C, T))$, $\mathcal{B}_1$ lazily samples $T'$ with the help of the table $\mathrm{T}_T$ and checks if $T'$ equals $T$. If not, $\mathcal{B}_1$ returns **false**, otherwise it calls its own verification oracle with input $(N, AD, C)$ and forwards the answer back to $\mathcal{A}$. For $\mathcal{A}$'s random oracle query $X = (K', *)$, $\mathcal{B}_1$ records $K'$ in $\mathcal{Q}^*$ and answers the query via lazy sampling using the table $\mathrm{T}_\mathrm{H}$. After $\mathcal{A}$ terminates, $\mathcal{B}_1$ picks a nonce $N^*$ that is different from those used by $\mathcal{A}$ in encryption oracle queries; then, for each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}$ computes $C_i \leftarrow \mathsf{AE.Enc}(K_i, N^*, AD, 0^\kappa)$ and queries its own verification oracle with $(N^*, AD, C_i)$.

The above adversary $\mathcal{B}_1$ correctly simulates the $\mathcal{A}$'s view in $\mathbf{G}_*$, and we claim that, if $\mathcal{A}$ wins in $\mathbf{G}_*$, then $\mathcal{B}_1$ wins in its $\mathsf{INT\text{-}CTXT}^{\mathcal{B}_1}_{\mathsf{AE}}$ game. If $\mathcal{A}$ wins, then the $\mathsf{AE}$ key $K$ (sampled by $\mathcal{B}_1$'s challenger) must be included in the set $\mathcal{Q}^*$. Since $\mathcal{B}_1$ enumerated all keys in $\mathcal{Q}^*$ to compute ciphertexts with a new nonce, the ciphertext derived from the correct key $K$ must be a valid forgery for $\mathcal{B}_1$ to win in its game. Therefore, it holds that

$$\Pr\left[\mathbf{G}^{\mathcal{A}}_* \Rightarrow 1\right] \leq \mathbf{Adv}^{\text{mr-int-ctxt}}_{\mathsf{AE}}(\mathcal{B}_1). \tag{47}$$

Now we bound $\Pr\left[\mathbf{G}^{\mathcal{A}}_1 \Rightarrow 1\right]$ by constructing another MR-INT-CTXT adversary $\mathcal{B}_2$ against the underlying $\mathsf{AE}$. Adversary $\mathcal{B}_2$ is the same as adversary $\mathcal{B}_1$, except that $\mathcal{B}_2$ terminates immediately after $\mathcal{A}$ terminates, i.e., $\mathcal{B}_2$ does not record keys in $\mathcal{Q}^*$ or make the final verification queries after $\mathcal{A}$ terminates. Clearly, $\mathcal{B}_2$ correctly simulates $\mathcal{A}$'s view in $\mathbf{G}_1$. We continue by showing that if $\mathcal{A}$ wins, then adversary $\mathcal{B}_2$ also wins in its $\mathsf{INT\text{-}CTXT}^{\mathcal{B}_2}_{\mathsf{AE}}$ game. Suppose $\mathcal{A}$ wins by querying $\mathrm{Ver}(N^*, AD^*, (C^*, T^*))$. By definition of the winning condition of $\mathcal{A}$, we know $(C^*, T^*)$ was not output by any $\mathrm{Enc}(N^*, AD^*, \cdot)$ query and $\mathrm{Ver}(N^*, AD^*, (C^*, T^*)) = 1$. By construction of $\mathcal{B}_2$ , the latter implies that $T^* = \mathrm{T}_T[(N^*, AD^*, C^*|_\tau)]$ and $\mathrm{Ver}'(N^*, AD^*, C^*) = 1$, so it suffices to show that $C^*$ was not output by any $\mathrm{Enc}'(N^*, AD^*, \cdot)$ query made by $\mathcal{B}_2$. Suppose otherwise, that $C^*$ was output by some query to $\mathrm{Enc}'(N^*, AD^*, \cdot)$. In that case, $(C^*, T^*)$ would have been returned by $\mathsf{Enc}$ oracle to $\mathcal{A}$ since we know $T^* = \mathrm{T}_T[(N^*, AD^*, C^*|_\tau)]$. That would be a contradiction. Therefore, $\mathcal{B}_2$ also wins and it holds that

$$\Pr\left[\mathbf{G}^{\mathcal{A}}_1 \Rightarrow 1\right] \leq \mathbf{Adv}^{\text{mr-int-ctxt}}_{\mathsf{AE}}(\mathcal{B}_2). \tag{48}$$

The proof is concluded by combining equations (45)$\sim$(48). □

### B.15 Proof of Theorem 15 (CMT Security of AEtH)

*Proof.* In order to win the $\mathsf{CMT}^{\mathcal{A}}_{\mathsf{AEtH}}$ game (see Figure 5), $\mathcal{A}$ needs to output two tuples $(K_1, N_1, AD_1, M_1)$, $(K_2, N_2, AD_2, M_2)$ such that they are encrypted to the same ciphertext and $(K_1, N_1, AD_1) \neq (K_2, N_2, AD_2)$. By construction of AEtH, this implies that the hashes of $(K_1, N_1, AD_1, C\,|_\tau)$ and $(K_2, N_2, AD_2, C\,|_\tau)$ collide, which breaks collision resistance of H. Therefore, we can construct $\mathcal{B}$ as follows: it first invokes $\mathcal{A}$, then extracts $(K_1, N_1, AD_1)$ and $(K_2, N_2, AD_2)$ from $\mathcal{A}$'s output. It continues by computing $C$ and outputting the tuple pair $((K_1, N_1, AD_1, C|_\tau), (K_2, N_2, AD_2, C|_\tau))$. □

### B.16 Proof of Theorem 16 (CDY$ Security of AEtH)

*Proof.* We define two games $\mathbf{G}_0$ and $\mathbf{G}_1$ as shown in Figure 24.

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\mathrm{H}(X)$ |
|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ ; $K \xleftarrow{\$} \{0,1\}^\kappa$ ; $N \xleftarrow{\$} \mathcal{N}$ ; $AD \xleftarrow{\$} \mathcal{AD}$ ; $M \xleftarrow{\$} \mathcal{M}$ | **if** $X = (K, *)$ : |
| $C \leftarrow \mathsf{AE.Enc}(K, N, AD, M)$ ; $T \xleftarrow{\$} \{0,1\}^t$ | $\mathsf{bad} \leftarrow \mathbf{true}$ |
| $\boxed{\mathrm{T_H}[(K, N, AD, C\|_\tau)] \leftarrow T}$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ |
| $(K', N', AD', M') \xleftarrow{\$} \mathcal{A}^{\mathrm{H}}(C, T)$ | **return** $\mathrm{T_H}[X]$ |
| $C' \leftarrow \mathsf{AE.Enc}(K', N', AD', M')$ ; $T' \leftarrow \mathrm{H}(K', N', AD', C'\|_\tau)$ | |
| **return** $(C', T') = (C, T)$ | |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ |
|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ ; $K \xleftarrow{\$} \{0,1\}^\kappa$ ; $N \xleftarrow{\$} \mathcal{N}$ ; $AD \xleftarrow{\$} \mathcal{AD}$ ; $M \xleftarrow{\$} \mathcal{M}$ | $(K^*, *) \leftarrow X$ |
| $C \leftarrow \mathsf{AE.Enc}(K, N, AD, M)$ ; $T \xleftarrow{\$} \{0,1\}^t$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K^*\}$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ |
| $(K', N', AD', M') \xleftarrow{\$} \mathcal{A}^{\mathrm{H}}(C, T)$ | **return** $\mathrm{T_H}[X]$ |
| $C' \leftarrow \mathsf{AE.Enc}(K', N', AD', M')$ ; $\mathrm{H}(K', N', AD', C'\|_\tau)$ | |
| **return** $K \in \mathcal{Q}^*$ | |

Fig. 24: Games $\mathbf{G}_0, \mathbf{G}_1$ and $\mathbf{G}_*$ in the proof of Theorem 16, where $\mathbf{G}_0$ contains boxed content while $\mathbf{G}_1$ does not.

Game $\mathbf{G}_0$ is the same as $\mathsf{CDY}^{\mathcal{A}}_{\mathsf{AEtH},\mathsf{S\$}}$ (see Figure 6) equipped with a random oracle H. Therefore, we have $\Pr\big[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\big] = \mathbf{Adv}^{\mathrm{cdy}}_{\mathsf{AEtH},\mathsf{S\$}}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{cdy\$}}_{\mathsf{AEtH}}(\mathcal{A})$.

Game $\mathbf{G}_1$ is the same as game $\mathbf{G}_0$, except that the committing tag $T$, calculated by the challenger, is now independent of queries to the random oracle H. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically unless the bad event happens, where the random oracle H is ever queried with an input that is prefixed with the key $K$ sampled by the challenger. This is because, if bad does not occur, then $T$ in $\mathbf{G}_0$ is also independent of random oracle queries made by $\mathcal{A}$ and by the challenger (to compute $T'$). Therefore, by the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\big[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\big] - \Pr\big[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\big] \leq \Pr[\mathsf{bad}]. \tag{49}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by constructing a game $\mathbf{G}_*$ also shown in Figure 24. $\mathbf{G}_*$ is defined such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers bad in game $\mathbf{G}_0/\mathbf{G}_1$ when (1) $\mathcal{A}$ queried the random oracle H with an input prefixed with $K$ or (2) $\mathcal{A}$ output $(K', N', AD', M')$ with $K' = K$ and then $\mathrm{H}(K', N', AD', C'|_\tau)$ was queried by the challenger, where $C' \leftarrow \mathsf{AE.Enc}(K', N', AD', M')$. Therefore, it holds that

$$\Pr[\mathsf{bad}] = \Pr\big[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\big]. \tag{50}$$

Then, we bound $\Pr\big[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\big]$ by constructing an adversary $\mathcal{B}$ against the one-time IND$-CPA security of the underlying AE. $\mathcal{B}$ starts by initializing the set $\mathcal{Q}^*$ and the table $\mathrm{T_H}$ (to an empty set and empty table, respectively). Then it samples the tuple $(N, AD, M)$ uniformly at random, and calls its own encryption oracle with input $(N, AD, M)$ to derive $C$. It then samples $T$ randomly and runs $\mathcal{A}$ with

input $(C, T)$. For $\mathcal{A}$'s each random oracle query $X = (K^*, *)$, $\mathcal{B}$ records $K^*$ in $\mathcal{Q}^*$ and answers the query via lazy sampling. After $\mathcal{A}$ terminates and outputs $(K', N', AD', M')$, $\mathcal{B}$ "queries" the random oracle H with the input $(K', N', AD', C'|_\tau)$, where $C' \leftarrow \mathsf{AE.Enc}(K', N', AD', M')$, and does the following. For each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}$ computes $C_i \leftarrow \mathsf{AE.Enc}(K_i, N, AD, M)$ and checks if $C_i = C$. If this holds for any of the keys in $\mathcal{Q}^*$, $\mathcal{B}$ guesses that it was in the real world $\mathsf{REAL}^{\mathcal{B}}_{\mathsf{AE}}$ and outputs 1; otherwise, it outputs 0, guessing that it was in the ideal world $\mathsf{RAND}^{\mathcal{B}}_{\mathsf{AE}}$.

If $\mathcal{B}$ was in the real world, then it simulates $\mathcal{A}$'s view in $\mathbf{G}_*$ perfectly and hence we have $\Pr\big[\mathbf{G}^{\mathcal{A}}_* \Rightarrow 1\big] \leq \Pr\Big[\mathsf{REAL}^{\mathcal{B}}_{\mathsf{AE}} \Rightarrow 1\Big]$. Here we have an inequality to capture "false positives" in the above $C_i = C$ check. On the other hand, if $\mathcal{B}$ was in the ideal world, the probability that a random ciphertext $C$ equals $\mathsf{AE.Enc}(K_i, N, AD, M)$ for some $K_i \in \mathcal{Q}^*$ is at most $|\mathcal{Q}^*|/2^{\mathsf{AE.clen}(|M|)}$. Since $|\mathcal{Q}^*| \leq q_{\mathrm{h}} + 1$, we have $\Pr\Big[\mathsf{RAND}^{\mathcal{B}}_{\mathsf{AE}} \Rightarrow 1\Big] \leq (q_{\mathrm{h}} + 1)/2^{\mathsf{AE.clen}(|M|)}$. Recalling that $\mathbf{Adv}^{\text{ot-ind\$-cpa}}_{\mathsf{AE}}(\mathcal{B}) = \Pr\Big[\mathsf{REAL}^{\mathcal{B}}_{\mathsf{AE}} \Rightarrow 1\Big] - \Pr\Big[\mathsf{RAND}^{\mathcal{B}}_{\mathsf{AE}} \Rightarrow 1\Big]$ and $\mathsf{AE.clen}(|M|) \geq \tau$, we have

$$\mathbf{Adv}^{\text{ot-ind\$-cpa}}_{\mathsf{AE}}(\mathcal{B}) \geq \Pr\big[\mathbf{G}^{\mathcal{A}}_* \Rightarrow 1\big] - \frac{q_{\mathrm{h}} + 1}{2^\tau}, \tag{51}$$

and it follows from equations (49)∼(51) that

$$\Pr\big[\mathbf{G}^{\mathcal{A}}_0 \Rightarrow 1\big] - \Pr\big[\mathbf{G}^{\mathcal{A}}_1 \Rightarrow 1\big] \leq \mathbf{Adv}^{\text{ot-ind\$-cpa}}_{\mathsf{AE}}(\mathcal{B}) + \frac{q_{\mathrm{h}} + 1}{2^\tau}. \tag{52}$$

Finally, we observe that in game $\mathbf{G}_1$ the committing tag $T$ in the challenge ciphertext $(C, T)$ is independent of random oracle queries and hence independent of $T'$, so in order for $\mathcal{A}$ to win one of the random oracle queries must output a committing tag equal to the independent random $T$. This probability is bounded by $(q_{\mathrm{h}} + 1)/2^t$. Therefore, we have

$$\Pr\big[\mathbf{G}^{\mathcal{A}}_1 \Rightarrow 1\big] \leq \frac{q_{\mathrm{h}} + 1}{2^t}. \tag{53}$$

The proof is concluded by combining equations (52) and (53). □

### B.17 Proof of Theorem 17 (MR-IND$-CPA Security of chaSIV)

*Proof.* We start by defining three games $\mathbf{G}_0 \sim \mathbf{G}_2$ as shown in Figure 25.

Game $\mathbf{G}_0$ is the same as $\mathsf{REAL}^{\mathcal{A}}_{\mathsf{chaSIV}}$ (see Figure 4) equipped with a random oracle H. Therefore, we have $\Pr\big[\mathbf{G}^{\mathcal{A}}_0 \Rightarrow 1\big] = \Pr\Big[\mathsf{REAL}^{\mathcal{A}}_{\mathsf{chaSIV}} \Rightarrow 1\Big]$.

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X) \wedge X = (K, *)$ : | $K_e \leftarrow \mathsf{F}(K, N)$ |
| $\mathrm{T_H}[\cdot] \leftarrow \bot$ | $\quad \mathsf{bad} \leftarrow \mathbf{true}$ | $T \leftarrow \mathsf{H}(K, N, AD, M)$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **if** $\mathrm{T_H}[X] = \bot : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $\boxed{T \leftarrow \{0,1\}^t}$ |
| **return** $b'$ | **return** $\mathrm{T_H}[X]$ | $N_e \leftarrow T|_r$ |
| | | $C \leftarrow \mathsf{E.Enc}(K_e, N_e, M)$ |
| | | **return** $(C, T)$ |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | $(K', *) \leftarrow X$ | $K_e \leftarrow \mathsf{F}(K, N)$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | $\mathcal{Q}^* \xleftarrow{\sqcup} \{K'\}$ | $T \leftarrow \{0,1\}^t$ |
| $\mathrm{T_H}[\cdot] \leftarrow \bot$ | **if** $\mathrm{T_H}[X] = \bot : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $N_e \leftarrow T|_r$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **return** $\mathrm{T_H}[X]$ | $C \leftarrow \mathsf{E.Enc}(K_e, AD, M)$ |
| **return** $K \in \mathcal{Q}^*$ | | **return** $(C, T)$ |

Fig. 25: Games $\mathbf{G}_0, \mathbf{G}_1$ and $\mathbf{G}_*$ in the proof of Theorem 17, where $\mathbf{G}_0$ contains boxed content while $\mathbf{G}_1$ does not.

Game $\mathbf{G}_1$ is the same as $\mathbf{G}_0$, except that the committing tag is now sampled randomly from the set $\{0,1\}^t$ in the encryption oracle Enc. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically unless the $\mathsf{bad}$ event happens, which occurs if the adversary $\mathcal{A}$ ever queries H with an input that is prefixed with the key $K$. The only syntactical difference between the two games is in the generation of $T$. Recall that $\mathcal{A}$ will never repeat $(N, AD, M)$ input, therefore, if $\mathsf{bad}$ does not occur, in both games the committing tag $T$ is random string that is independent of $\mathcal{A}$'s view. Hence, games $\mathbf{G}_0$ and $\mathbf{G}_1$ are indeed identical-until-$\mathsf{bad}$. By the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr[\mathsf{bad}]. \tag{54}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by considering a game $\mathbf{G}_*$ also shown in Figure 25. Here $\mathbf{G}_*$ is just defined to clarify the $\mathsf{bad}$ event, such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers $\mathsf{bad}$ in game $\mathbf{G}_0/\mathbf{G}_1$. Therefore, we have

$$\Pr[\mathsf{bad}] = \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]. \tag{55}$$

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{C}_1$ against the PRF security of the underlying $\mathsf{F}$. $\mathcal{C}_1$ starts by initializing the set $\mathcal{Q}^*$ and table $\mathrm{T_H}$ (to an empty set and empty table, respectively), and then it runs $\mathcal{A}$. To answer $\mathcal{A}$'s encryption query on input $(N, AD, M)$, $\mathcal{C}_1$ calls its own PRF oracle with input $N$ to get $K_e$ back. Then it samples $T$ at random and calculates $C \leftarrow \mathsf{E.Enc}(K_e, T|_r, M)$, returning $(C, T)$ back to $\mathcal{A}$. For $\mathcal{A}$'s random oracle query $X = (K', *)$, $\mathcal{C}_1$ records $K'$ in $\mathcal{Q}^*$ and answers the query via lazy sampling. After $\mathcal{A}$ terminates, $\mathcal{C}_1$ ignores its output and queries its own PRF oracle with input $N^*$ and gets back $K_e^*$, where $N^*$ is a nonce different from those used in previous PRF queries. Then, for each key $K_i \in \mathcal{Q}^*$, $\mathcal{C}_1$ computes $K_{e,i} \leftarrow \mathsf{F}(K_i, N^*)$ and checks if $K_{e,i} = K_e^*$ holds. If this holds for any of the keys in $\mathcal{Q}^*$, $\mathcal{C}_1$ guesses that it was interacting with the real PRF $\mathsf{F}$ and outputs 1; otherwise, it outputs 0, guessing that it was interacting with a truly random function.

If $\mathcal{C}_1$ had access to real $\mathsf{F}$, then it simulates $\mathcal{A}$'s view in $\mathbf{G}_*$ perfectly and hence we have $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathcal{C}_1^{\mathsf{F}_K} \Rightarrow 1\right]$. Here we have an inequality to capture "false positives" in the above $K_{e,i} = K_e^*$ check. On the other hand, if $\mathcal{C}_1$ was in the ideal world, the probability that a random output $K_e^*$ equals $\mathsf{F}(K_i, N^*)$ for some $K_i \in \mathcal{Q}^*$ is at most $|\mathcal{Q}^*|/2^\kappa$. Since $|\mathcal{Q}^*| \leq q_\mathrm{h}$, we have $\Pr\left[\mathcal{C}_1^{\mathsf{f}} \Rightarrow 1\right] \leq q_\mathrm{h}/2^\kappa$. Then, recalling that $\mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{C}_1) = \Pr\left[\mathcal{C}_1^{\mathsf{F}_K} \Rightarrow 1\right] - \Pr\left[\mathcal{C}_1^{\mathsf{f}} \Rightarrow 1\right]$ we have

$$\mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{C}_1) \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_\mathrm{h}}{2^\kappa}, \tag{56}$$

and it follows from equations (54)∼(56) that

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{C}_1) + \frac{q_\mathrm{h}}{2^\kappa}. \tag{57}$$

We continue by making a game hop from $\mathbf{G}_1$ to $\mathbf{G}_2$ by replacing $\mathsf{F}$ invocations with invocations to a random function $\mathsf{f}$. This a standard game hop where a PRF adversary $\mathcal{C}_2$ simulates $\mathbf{G}_1$ / $\mathbf{G}_2$ while having access to its own PRF oracle. Therefore, it holds

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{C}_2). \tag{58}$$

Next, we define two more games $\mathbf{G}_3$ and $\mathbf{G}_4$ as shown in Figure 26. In game $\mathbf{G}_3$ we define a bad event $\mathsf{bad}'$ which captures the event that there is a collision in derived nonce $N_e$ for some nonce $N$ queried by $\mathcal{A}$. Since the underlying $\mathsf{E}$ is secure only in a nonce-respecting case, we need to make sure there will be no collisions in derived nonce for different instances of $\mathsf{E}$, where these instances of $\mathsf{E}$ directly correspond to nonces that are queried by $\mathcal{A}$. In order to define the bad event $\mathsf{bad}'$, we also introduce the table $\mathrm{T}_N$ in game $\mathbf{G}_3$, where the elements of the table will be sets containing derived nonces of a $\mathsf{E}$ instance indexed by nonce that was queried by $\mathcal{A}$. Games $\mathbf{G}_2$ and $\mathbf{G}_3$ are identical-until-$\mathsf{bad}$ and so it holds

$$\Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_3^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathsf{bad}'\right] \leq \frac{R_1^2}{2^r} + \cdots + \frac{R_{Q_\mathrm{e}}^2}{2^r}, \tag{59}$$

where the individual summands represent the standard collision birthday bound.

Game $\mathbf{G}_4$ is the same as game $\mathbf{G}_3$, except that now $C$ in encryption oracle is always sampled uniformly at random. To bound the gap between $\mathbf{G}_3$ and $\mathbf{G}_4$, we apply a hybrid argument as follows. Consider a sequence of hybrid games $\mathbf{H}_0, \mathbf{H}_1, \ldots, \mathbf{H}_{Q_\mathrm{e}}$ such that $\mathbf{G}_3 = \mathbf{H}_0$ and $\mathbf{G}_4 = \mathbf{H}_{Q_\mathrm{e}}$. Now, let $\mathcal{Q}_N =$

| Game $\mathbf{G}_2$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $K_e \leftarrow \mathsf{f}(N)$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **return** $\mathrm{T_H}[X]$ | $T \leftarrow \{0,1\}^t$ |
| **return** $b'$ | | $N_e \leftarrow T\vert_r$ |
| | | $C \leftarrow \mathsf{E.Enc}(K_e, N_e, M)$ |
| | | **return** $(C, T)$ |

| Game $\mathbf{G}_3$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $K_e \leftarrow \mathsf{f}(N)$ |
| $\mathrm{T}_N[\cdot] \leftarrow \perp$ | **return** $\mathrm{T_H}[X]$ | $T \leftarrow \{0,1\}^t$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | | $N_e \leftarrow T\vert_r$ |
| **return** $b'$ | | **if** $N_e \in \mathrm{T}_N[N]$ : |
| | | $\quad \mathsf{bad}' \leftarrow$ **true** |
| | | $\quad$ **abort** |
| | | $\mathrm{T}_N[N] \xleftarrow{\sqcup} \{N_e\}$ |
| | | $C \leftarrow \mathsf{E.Enc}(K_e, N_e, M)$ |
| | | **return** $(C, T)$ |

| Game $\mathbf{G}_4$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ |
|---|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $T \leftarrow \{0,1\}^t$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **return** $\mathrm{T_H}[X]$ | $C \leftarrow \{0,1\}^{\vert M\vert}$ |
| **return** $b'$ | | **return** $(C, T)$ |

Fig. 26: Games $\mathbf{G}_2 \sim \mathbf{G}_4$ in the proof of Theorem 17.

$\{N_1, ..., N_{Q_e}\}$ be the list of nonces queried by $\mathcal{A}$. Game $\mathbf{H}_i$ is the same as game $\mathbf{G}_3$, except that the output of all queries that have been made (to $\mathsf{E}$) with a nonce $N_j$, with $j \leq i$, are replaced by independent random strings as in $\mathbf{G}_4$. By construction, consecutive games $\mathbf{H}_{i-1}$ and $\mathbf{H}_i$ are the same except for the queries made to $\mathsf{E}$ with nonce $N_i$. We can construct an adversary $\mathcal{B}_i$ against the IND$-CPA security of the underlying $\mathsf{E}$ such that it simulates $\mathcal{A}$'s view in $\mathbf{H}_{i-1}$ in the real world and $\mathbf{H}_i$ in the ideal world. To do so, $\mathcal{B}_i$ simulates $\mathcal{A}$'s views in $\mathbf{H}_{i-1}$ and $\mathbf{H}_i$ in the same way following their game procedures, except that, for Enc queries made with nonce $N_i$, $\mathcal{B}_i$ samples a random $T$ and calls its own encryption oracle $\mathrm{Enc}'$ to simulate the $\mathsf{E}$ ciphertext $C$. Finally, $\mathcal{B}_i$ outputs the bit that $\mathcal{A}$ outputs. By construction, we have $\Pr\big[\mathbf{H}_{i-1}^{\mathcal{A}} \Rightarrow 1\big] - \Pr\big[\mathbf{H}_i^{\mathcal{A}} \Rightarrow 1\big] = \mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}_i)$. With a hybrid argument, there exists an efficient adversary $\mathcal{B}$ such that

$$\begin{aligned}\Pr\big[\mathbf{G}_3^{\mathcal{A}} \Rightarrow 1\big] - \Pr\big[\mathbf{G}_4^{\mathcal{A}} \Rightarrow 1\big] &= \Pr\big[\mathbf{H}_0^{\mathcal{A}} \Rightarrow 1\big] - \Pr\big[\mathbf{H}_k^{\mathcal{A}} \Rightarrow 1\big] \\ &= \sum_{i=1}^{Q_e} \big(\Pr\big[\mathbf{H}_{i-1}^{\mathcal{A}} \Rightarrow 1\big] - \Pr\big[\mathbf{H}_i^{\mathcal{A}} \Rightarrow 1\big]\big) \\ &\leq Q_e \mathbf{Adv}_{\mathsf{E}}^{\text{ind\$-cpa}}(\mathcal{B}).\end{aligned} \tag{60}$$

Finally, we observe that game $\mathbf{G}_4$ is identical to $\mathsf{RAND}_{\mathsf{chaSIV}}^{\mathcal{A}}$ (see Figure 4) equipped with a random oracle H. This is because, in game $\mathbf{G}_4$, for each Enc query, its output is an independent uniform random string sampled from $\{0,1\}^{|M|+t}$. Therefore, we have

$$\mathbf{Adv}_{\mathsf{chaSIV}}^{\text{mr-ind\$-cpa}}(\mathcal{A}) = \Pr\big[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\big] - \Pr\big[\mathbf{G}_4^{\mathcal{A}} \Rightarrow 1\big]. \tag{61}$$

The proof is concluded by combining equations (57)∼(61). □

### B.18 Proof of Theorem 18 (MR-INT-CTXT Security of chaSIV)

*Proof.* We start by defining three games $\mathbf{G}_0 \sim \mathbf{G}_2$ as shown in Figure 27.

| Game $\mathbf{G}_0$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X)$ | $K_e \leftarrow \mathsf{F}(K, N)$ | $K_e \leftarrow \mathsf{F}(K, N)$ ; $N_e \leftarrow T\|_r$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | $\quad \wedge\, X = (K, *)$ : | $T \leftarrow \mathrm{H}(K, N, AD, M)$ | $M \leftarrow \mathsf{E.Dec}(K_e, N_e, C)$ |
| $\mathcal{Q} \leftarrow \varnothing$ | $\quad \mathsf{bad} \leftarrow \mathbf{true}$ | $N_e \leftarrow T\|_r$ | $T' \leftarrow \mathrm{H}(K, N, AD, M)$ |
| $\mathrm{win} \leftarrow 0$ | **if** $\mathrm{T_H}[X] = \perp$ : | $C \leftarrow \mathsf{E.Enc}(K_e, N_e, M)$ | **if** $T \neq T'$ : **return false** |
| $\mathcal{A}^{\mathrm{H,Enc,Ver}}$ | $\quad \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $\mathcal{Q} \xleftarrow{\cup} \{(N, AD, (C, T))\}$ | **if** $M \neq \perp \wedge (N, AD, (C, T)) \notin \mathcal{Q}$ : |
| **return** win | **return** $\mathrm{T_H}[X]$ | **return** $(C, T)$ | $\quad \mathrm{win} \leftarrow 1$ |
| | | | **return** $M \neq \perp$ |

| Game $\boxed{\mathbf{G}_1}, \mathbf{G}_2$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | **if** $\mathcal{A}$ queries $\mathrm{H}(X)$ | $K_e \leftarrow \mathsf{F}(K, N)$ | $K_e \leftarrow \mathsf{F}(K, N)$ ; $N_e \leftarrow T\|_r$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | $\quad \wedge\, X = (K, *)$ : | **if** $\mathrm{T}_T[(N, AD, M)] = \perp$ : | $M \leftarrow \mathsf{E.Dec}(K_e, N_e, C)$ |
| $\mathrm{T}_T[\cdot] \leftarrow \perp$ | $\quad \mathsf{bad} \leftarrow \mathbf{true}$ | $\quad \mathrm{T}_T[(N, AD, M)] \xleftarrow{\$} \{0,1\}^t$ | **if** $\mathrm{T}_T[(N, AD, M)] = \perp$ : |
| $\mathcal{Q} \leftarrow \varnothing$ | **if** $\mathrm{T_H}[X] = \perp$ : | $T \leftarrow \mathrm{T}_T[(N, AD, M)]$ | $\quad \mathrm{T}_T[(N, AD, M)] \xleftarrow{\$} \{0,1\}^t$ |
| $\mathrm{win} \leftarrow 0$ | $\quad \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $\boxed{T \leftarrow \mathrm{H}(K, N, AD, M)}$ | $T' \leftarrow \mathrm{T}_T[(N, AD, M)]$ |
| $\mathcal{A}^{\mathrm{H,Enc,Ver}}$ | **return** $\mathrm{T_H}[X]$ | $N_e \leftarrow T\|_r$ | $\boxed{T' \leftarrow \mathrm{H}(K, N, AD, M)}$ |
| **return** win | | $C \leftarrow \mathsf{E.Enc}(K_e, N_e, M)$ | **if** $T \neq T'$ : **return false** |
| | | $\mathcal{Q} \xleftarrow{\cup} \{(N, AD, (C, T))\}$ | **if** $(N, AD, (C, T)) \notin \mathcal{Q}$ : |
| | | **return** $(C, T)$ | $\quad \mathrm{win} \leftarrow 1$ |
| | | | **return true** |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ | Oracle $\mathrm{Enc}(N, AD, M)$ | Oracle $\mathrm{Ver}(N, AD, (C, T))$ |
|---|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^\kappa$ | $(K', *) \leftarrow X$ | $K_e \leftarrow \mathsf{F}(K, N)$ | $K_e \leftarrow \mathsf{F}(K, N)$ ; $N_e \leftarrow T\|_r$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K'\}$ | **if** $\mathrm{T}_T[(N, AD, M)] = \perp$ : | $M \leftarrow \mathsf{E.Dec}(K_e, N_e, C)$ |
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ | **if** $\mathrm{T_H}[X] = \perp$ : | $\quad \mathrm{T}_T[(N, AD, M)] \xleftarrow{\$} \{0,1\}^t$ | **if** $\mathrm{T}_T[(N, AD, M)] = \perp$ : |
| $\mathrm{T}_T[\cdot] \leftarrow \perp$ | $\quad \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ | $T \leftarrow \mathrm{T}_T[(N, AD, M)]$ | $\quad \mathrm{T}_T[(N, AD, M)] \xleftarrow{\$} \{0,1\}^t$ |
| $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{H,Enc}}$ | **return** $\mathrm{T_H}[X]$ | $N_e \leftarrow T\|_r$ | $T' \leftarrow \mathrm{T}_T[(N, AD, M)]$ |
| **return** $K \in \mathcal{Q}^*$ | | $C \leftarrow \mathsf{E.Enc}(K_e, AD, M)$ | **if** $T \neq T'$ : **return false** |
| | | **return** $(C, T)$ | **return true** |

Fig. 27: Games $\mathbf{G}_0 \sim \mathbf{G}_2$ and $\mathbf{G}_*$ in the proof of Theorem 18, where $\mathbf{G}_1$ contains boxed content while $\mathbf{G}_2$ does not.

Game $\mathbf{G}_0$ is the same as $\mathsf{INT\text{-}CTXT}^{\mathcal{A}}_{\mathsf{chaSIV}}$ (see Figure 4) equipped with a random oracle H. Therefore, we have $\Pr\big[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\big] = \Pr\Big[\mathsf{INT\text{-}CTXT}^{\mathcal{A}}_{\mathsf{chaSIV}} \Rightarrow 1\Big]$.

Game $\mathbf{G}_1$ is the same as $\mathbf{G}_0$, except that the verification oracle does not check if the ciphertext $C$ can be successfully decrypted (note that the $\mathrm{T}_T$ operations can be ignored in $\mathbf{G}_1$). As a result, the winning condition in $\mathbf{G}_1$ is relaxed, i.e., if $\mathcal{A}$ wins in $\mathbf{G}_0$ then it must also win in $\mathbf{G}_1$. To see this, suppose $\mathcal{A}$ wins in $\mathbf{G}_0$ by querying $(N^*, AD^*, (C^*, T^*))$ and before this query $\mathcal{A}$ did not already win in $\mathbf{G}_1$. In this case, $\mathcal{A}$'s views in $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical. This is because before that winning query each verification query with a valid $T$ must satisfy $(N, AD, (C, T)) \in \mathcal{Q}$ (otherwise $\mathcal{A}$ wins in $\mathbf{G}_1$), which implies that $M \neq \perp$ and hence the verification oracle returns **true** in both games. Then, by the winning condition in $\mathbf{G}_1$, the above query also allows $\mathcal{A}$ to win in $\mathbf{G}_1$. Therefore, we have

$$\Pr\big[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\big] \leq \Pr\big[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\big]. \tag{62}$$

Game $\mathbf{G}_2$ is the same as $\mathbf{G}_1$, except that the committing tag is now lazily sampled from the table $\mathrm{T}_T$ in both oracles, where the table is indexed by the $(N, AD, M)$ triple. We claim that $\mathbf{G}_1$ and $\mathbf{G}_2$ behave identically unless the $\mathsf{bad}$ event happens, which occurs if the adversary $\mathcal{A}$ ever queries H with an input that is prefixed with the key $K$. The only syntactical difference between the two games is in the generation of $T$. If $\mathsf{bad}$ does not occur, in both games the committing tag $T$ is either randomly sampled for a new $(N, AD, M)$ triple or equal to the same previously sampled value if $(N, AD, M)$ is not new.

Hence the two games are indeed identical-until-bad. By the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr[\mathsf{bad}]. \tag{63}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by considering a game $\mathbf{G}_*$ also shown in Figure 27. Here $\mathbf{G}_*$ is just defined to clarify the bad event, such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers bad in game $\mathbf{G}_1/\mathbf{G}_2$. Therefore, we have

$$\Pr[\mathsf{bad}] = \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]. \tag{64}$$

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}$ against the PRF security of the underlying $\mathsf{F}$. $\mathcal{B}$ starts by initializing the set $\mathcal{Q}^*$ and table $\mathrm{T_H}$ (to an empty set and empty table, respectively), and then it runs $\mathcal{A}$. During the simulation of encryption and verification oracles for $\mathcal{A}$, the adversary $\mathcal{B}$ will keep track of nonces queried by $\mathcal{A}$. If $\mathcal{A}$ makes a query with a new nonce $N$, $\mathcal{B}$ will query its own PRF oracle to get $K_e$ back and save the result $(N, K_e)$ internally. Then, if $\mathcal{A}$ makes a query with a nonce that already "appeared" before, $\mathcal{B}$ will just fetch the previous result $K_e$ that corresponds to the queried nonce.

To answer $\mathcal{A}$'s encryption query on input $(N, AD, M)$, $\mathcal{B}$ calls its own PRF oracle with input $N$, or fetches a previous result if the nonce is not new, to get $K_e$. Then it lazily samples $T$ with the help of table $\mathrm{T}_T$ and calculates $C \leftarrow \mathsf{E.Enc}(K_e, T|_r, M)$, returning $(C, T)$ back to $\mathcal{A}$. To answer $\mathcal{A}$'s verification query on input $(N, AD, (C, T))$, $\mathcal{B}$ calls its own PRF oracle with input $N$, or fetches a previous result if the nonce is not new, to get $K_e$. Then it calculates $M \leftarrow \mathsf{E.Dec}(K_e, T'|_r, C)$. It continues by lazily sampling $T'$ with the help of table $\mathrm{T}_T$ and checking if $T' = T$. If not, $\mathcal{B}$ returns **false** to $\mathcal{A}$; otherwise, it returns **true** to $\mathcal{A}$. For $\mathcal{A}$'s random oracle query $X = (K', *)$, $\mathcal{B}$ records $K'$ in $\mathcal{Q}^*$ and answers the query via lazy sampling. After $\mathcal{A}$ terminates, $\mathcal{B}$ ignores its output and queries its own PRF oracle with input $N^*$ and gets back $K_e^*$, where $N^*$ is a nonce different from those used in previous PRF queries. Then, for each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}$ computes $K_{e,i} \leftarrow \mathsf{F}(K_i, N^*)$ and checks if $K_{e,i} = K_e^*$ holds. If this holds for any of the keys in $\mathcal{Q}^*$, $\mathcal{B}$ guesses that it was interacting with the real PRF $\mathsf{F}$ and outputs 1; otherwise, it outputs 0, guessing that it was interacting with a truly random function.

If $\mathcal{B}$ had access to real $\mathsf{F}$, then it simulates $\mathcal{A}$'s view in $\mathbf{G}_*$ perfectly and hence we have $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathcal{B}^{\mathsf{F}_K} \Rightarrow 1\right]$. Here we have an inequality to capture "false positives" in the above $K_{e,i} = K_e^*$ check. On the other hand, if $\mathcal{B}$ was in the ideal world, the probability that a random output $K_e^*$ equals $\mathsf{F}(K_i, N^*)$ for some $K_i \in \mathcal{Q}^*$ is at most $|\mathcal{Q}^*|/2^\kappa$. Since $|\mathcal{Q}^*| \leq q_\mathrm{h}$, we have $\Pr\left[\mathcal{B}^{\mathsf{f}} \Rightarrow 1\right] \leq q_\mathrm{h}/2^\kappa$. Then, recalling that $\mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{B}) = \Pr\left[\mathcal{B}^{\mathsf{F}_K} \Rightarrow 1\right] - \Pr\left[\mathcal{B}^{\mathsf{f}} \Rightarrow 1\right]$ we have

$$\mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{B}) \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_\mathrm{h}}{2^\kappa}, \tag{65}$$

and it follows from equations (63)∼(65) that

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{B}) + \frac{q_\mathrm{h}}{2^\kappa}. \tag{66}$$

Now it is left to bound $\Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right]$. Suppose $\mathcal{A}$ wins by querying $\mathrm{Ver}(N^*, AD^*, (C^*, T^*))$ and that the winning tuple decrypts to $M^*$. We know $\mathcal{A}$ did not query $(N^*, AD^*, M^*)$ previously to the encryption oracle because by the tidyness of $\mathsf{E}$, if $\mathcal{A}$ did make such a query, result of $\mathsf{E.Enc}(K_e, N^*, M^*)$ would be exactly $C^*$. Then, since $T^* = \mathrm{T}_T[(N^*, AD^*, M^*)]$, that previous encryption oracle would have returned $(C^*, T^*)$ which is a contradiction with $(C^*, T^*) \notin \mathcal{Q}$ part of the winning condition. Therefore, the only way $\mathcal{A}$ can win is by querying Ver and trying to "guess" $T^*$ such that $T^* = \mathrm{T}_T[(N^*, AD^*, M^*)]$.

The probability that $\mathcal{A}$ wins with its first verification query is $\frac{1}{2^t}$ since we know $\mathrm{T}_T[(N^*, AD^*, M^*)]$ would be sampled after $\mathcal{A}$ has made its guess $T^*$. For its second verification query, assuming the first was unsuccessful, $\mathcal{A}$ can either query new $(N^*, AD^*)$ pair, leading again to the winning probability $\frac{1}{2^t}$. Otherwise, it can try to guess $\mathrm{T}_T[(N^*, AD^*, M^*)]$ from its first verification query. The probability that it wins is $(1 - \frac{1}{2^t})\frac{1}{2^t - 1} = \frac{1}{2^t}$, where the first multiplicative term is the probability that $\mathcal{A}$ did not win in its first query. For the subsequent verification queries the adversary can only follow the same strategy, either trying to guess the tag for new $(N^*, AD^*, M^*)$ tuple or trying to guess the tag for one of the "old" tuples. In both cases, the probability that it succeeds is $\frac{1}{2^t}$. Therefore, in total,

$$\Pr\left[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1\right] \leq \frac{q_\mathrm{v}}{2^t}. \tag{67}$$

The proof is concluded by combining equations (62), (66) and (67). □

### B.19 Proof of Theorem 19 (CMT Security of chaSIV)

*Proof.* In order to win the $\mathsf{CMT}^{\mathcal{A}}_{\mathsf{chaSIV}}$ game (see Figure 5), $\mathcal{A}$ needs to output two tuples $(K_1, N_1, AD_1, M_1)$, $(K_2, N_2, AD_2, M_2)$ such that they are encrypted to the same ciphertext and $(K_1, N_1, AD_1) \neq (K_2, N_2, AD_2)$. By construction of chaSIV, this implies that the hashes of $(K_1, N_1, AD_1, M_1)$ and $(K_2, N_2, AD_2, M_2)$ collide, which breaks collision resistance of $\mathsf{H}$. Therefore, we can construct $\mathcal{B}$ as follows: it first invokes $\mathcal{A}$, then extracts $(K_1, N_1, AD_1)$ and $(K_2, N_2, AD_2)$ from $\mathcal{A}$'s output. At the end, it outputs the tuple pair $((K_1, N_1, AD_1, M_1), (K_2, N_2, AD_2, M_2))$. □

### B.20 Proof of Theorem 20 (CDY$ Security of chaSIV)

*Proof.* We define two games $\mathbf{G}_0$ and $\mathbf{G}_1$ as shown in Figure 28.

| Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$ | Oracle $\mathrm{H}(X)$ |
|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ ; $K \xleftarrow{\$} \{0,1\}^\kappa$ ; $N \xleftarrow{\$} \mathcal{N}$ ; $AD \xleftarrow{\$} \mathcal{AD}$ ; $M \xleftarrow{\$} \mathcal{M}$ | **if** $X = (K, *)$ : |
| $K_e \leftarrow \mathsf{F}(K, N)$ ; $T \xleftarrow{\$} \{0,1\}^t$ | $\mathsf{bad} \leftarrow \mathbf{true}$ |
| $\boxed{\mathrm{T_H}[(K, N, AD, M)] \leftarrow T}$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ |
| $N_e \leftarrow T\|_r$ ; $C \leftarrow \mathsf{E.Enc}(K_e, N_e, M)$ | **return** $\mathrm{T_H}[X]$ |
| $(K', N', AD', M') \xleftarrow{\$} \mathcal{A}^{\mathrm{H}}(C, T)$ | |
| $K'_e \leftarrow \mathsf{F}(K', N')$ ; $T' \leftarrow \mathrm{H}(K', N', AD', M')$ | |
| $N_e \leftarrow T\|_r$ ; $C' \leftarrow \mathsf{E.Enc}(K'_e, N'_e, M')$ | |
| **return** $(C', T') = (C, T)$ | |

| Game $\mathbf{G}_*$ | Oracle $\mathrm{H}(X)$ |
|---|---|
| $\mathrm{T_H}[\cdot] \leftarrow \perp$ ; $K \xleftarrow{\$} \{0,1\}^\kappa$ ; $N \xleftarrow{\$} \mathcal{N}$ ; $AD \xleftarrow{\$} \mathcal{AD}$ ; $M \xleftarrow{\$} \mathcal{M}$ | $(K^*, *) \leftarrow X$ |
| $K_e \leftarrow \mathsf{F}(K, N)$ ; $T \xleftarrow{\$} \{0,1\}^t$ | $\mathcal{Q}^* \xleftarrow{\cup} \{K^*\}$ |
| $N_e \leftarrow T\|_r$ ; $C \leftarrow \mathsf{E.Enc}(K_e, N_e, M)$ | **if** $\mathrm{T_H}[X] = \perp : \mathrm{T_H}[X] \xleftarrow{\$} \{0,1\}^t$ |
| $\mathcal{Q}^* \leftarrow \varnothing$ | **return** $\mathrm{T_H}[X]$ |
| $(K', N', AD', M') \xleftarrow{\$} \mathcal{A}^{\mathrm{H}}(C, T)$ | |
| $\mathrm{H}(K', N', AD', M')$ | |
| **return** $K \in \mathcal{Q}^*$ | |

Fig. 28: Games $\mathbf{G}_0, \mathbf{G}_1$ and $\mathbf{G}_*$ in the proof of Theorem 20, where $\mathbf{G}_0$ contains boxed content while $\mathbf{G}_1$ does not.

Game $\mathbf{G}_0$ is the same as $\mathsf{CDY}^{\mathcal{A}}_{\mathsf{chaSIV},\mathsf{S}_\$}$ (see Figure 6) equipped with a random oracle H. Therefore, we have $\Pr\big[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\big] = \mathbf{Adv}^{\mathrm{cdy}}_{\mathsf{chaSIV},\mathsf{S}_\$}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{cdy\$}}_{\mathsf{chaSIV}}(\mathcal{A})$.

Game $\mathbf{G}_1$ is the same as game $\mathbf{G}_0$, except that the committing tag $T$, calculated by the challenger, is now independent of queries to the random oracle H. We claim that $\mathbf{G}_0$ and $\mathbf{G}_1$ behave identically unless the $\mathsf{bad}$ event happens, where the random oracle H is ever queried with an input that is prefixed with the key $K$ sampled by the challenger. This is because, if $\mathsf{bad}$ does not occur, then $T$ in $\mathbf{G}_0$ is also independent of random oracle queries made by $\mathcal{A}$ and by the challenger (to compute $T'$). Therefore, by the Difference Lemma of the game-based proof technique [31], we have

$$\Pr\big[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\big] - \Pr\big[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\big] \leq \Pr[\mathsf{bad}]. \tag{68}$$

Next, similar to the proof of Theorem 5, we bound $\Pr[\mathsf{bad}]$ by constructing a game $\mathbf{G}_*$ also shown in Figure 28. $\mathbf{G}_*$ is defined such that $\mathcal{A}$ wins in $\mathbf{G}_*$ if and only if $\mathcal{A}$ triggers $\mathsf{bad}$ in game $\mathbf{G}_0/\mathbf{G}_1$ when (1) $\mathcal{A}$ queried the random oracle H with an input prefixed with $K$ or (2) $\mathcal{A}$ output $(K', N', AD', M')$ with $K' = K$ and then $\mathrm{H}(K', N', AD', C')$ was queried by the challenger. Therefore, we have

$$\Pr[\mathsf{bad}] = \Pr\big[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\big]. \tag{69}$$

Then, we bound $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right]$ by constructing an adversary $\mathcal{B}$ against the PRF security of the underlying $\mathsf{F}$. $\mathcal{B}$ starts by initializing the set $\mathcal{Q}^*$ and table $\mathrm{T_H}$ (to an empty set and empty table, respectively), and then it runs $\mathcal{A}$. Then it samples the tuple $(N, AD, M)$ uniformly at random, and calls its own PRF oracle with input $N$ to derive $K_e$. It continues by sampling $T$, setting $N_e \leftarrow T|_r$, calculating $C \leftarrow \mathsf{E.Enc}(K_e, N_e, M)$ and running $\mathcal{A}$ with $(C, T)$. For $\mathcal{A}$'s each random oracle query $X = (K^*, *)$, $\mathcal{B}$ records $K^*$ in $\mathcal{Q}^*$ and answers the query via lazy sampling. After $\mathcal{A}$ terminates and outputs $(K', N', AD', M')$, $\mathcal{B}$ "queries" the random oracle H with the input $(K', N', AD', M')$ and continues as follows. For each key $K_i \in \mathcal{Q}^*$, $\mathcal{B}$ computes $K_{e,i} \leftarrow \mathsf{F}(K_i, N)$ and checks if $K_{e,i} = K_e$. If this holds for any of the keys in $\mathcal{Q}^*$, $\mathcal{B}$ guesses that it was interacting with the real PRF $\mathsf{F}$ and outputs 1; otherwise, it outputs 0, guessing that it was interacting with a truly random function.

If $\mathcal{B}$ had access to real PRF, then it simulates $\mathcal{A}$'s view in $\mathbf{G}_*$ perfectly and hence we have $\Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] \leq \Pr\left[\mathcal{B}^{\mathsf{F}_K} \Rightarrow 1\right]$. Here we have an inequality to capture "false positives" in the above $K_{e,i} = K_e$ check. On the other hand, if $\mathcal{B}$ was in the ideal world, the probability that a random output $K_e$ equals $\mathsf{F}(K_i, N)$ for some $K_i \in \mathcal{Q}^*$ is at most $|\mathcal{Q}^*|/2^\kappa$. Since $|\mathcal{Q}^*| \leq q_\mathrm{h} + 1$, we have $\Pr\left[\mathcal{B}^{\mathsf{f}} \Rightarrow 1\right] \leq (q_\mathrm{h} + 1)/2^\kappa$. Then, recalling that $\mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{B}) = \Pr\left[\mathcal{B}^{\mathsf{F}_K} \Rightarrow 1\right] - \Pr\left[\mathcal{B}^{\mathsf{f}} \Rightarrow 1\right]$ we have

$$\mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{B}) \geq \Pr\left[\mathbf{G}_*^{\mathcal{A}} \Rightarrow 1\right] - \frac{q_\mathrm{h} + 1}{2^\kappa}, \tag{70}$$

and it follows from equations (68)∼(70) that

$$\Pr\left[\mathbf{G}_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \mathbf{Adv}_{\mathsf{F}}^{\mathrm{prf}}(\mathcal{B}) + \frac{q_\mathrm{h} + 1}{2^\kappa}. \tag{71}$$

Finally, we observe that in game $\mathbf{G}_1$ the committing tag $T$ in the challenge ciphertext $(C, T)$ is independent of random oracle queries and hence independent of $T'$, so in order for $\mathcal{A}$ to win one of the random oracle queries must output a committing tag equal to the independent random $T$. This probability is bounded by $(q_\mathrm{h} + 1)/2^t$. Therefore, we have

$$\Pr\left[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1\right] \leq \frac{q_\mathrm{h} + 1}{2^t}. \tag{72}$$

The proof is concluded by combining equations (71) and (72). □