Context Discovery and Commitment Attacks*
How to Break CCM, EAX, SIV, and More

Sanketh Menda, *Cornell Tech*
Julia Len, *Cornell Tech*
Paul Grubbs, *University of Michigan*
Thomas Ristenpart, *Cornell Tech*

Abstract

A line of recent work has highlighted the importance of context commitment security, which asks that authenticated encryption with associated data (AEAD) schemes will not decrypt the same adversarially-chosen ciphertext under two different, adversarially-chosen contexts (secret key, nonce, and associated data). Despite a spate of recent attacks, many open questions remain around context commitment; most obviously nothing is known about the commitment security of important schemes such as CCM, EAX, and SIV.

We resolve these open questions, and more. Our approach is to, first, introduce a new framework that helps us more granularly define context commitment security in terms of what portions of a context are adversarially controlled. We go on to formulate a new notion, called context discoverability security, which can be viewed as analogous to preimage resistance from the hashing literature. We show that unrestricted context commitment security (the adversary controls all of the two contexts) implies context discoverability security for a class of schemes encompassing most schemes used in practice. Then, we show new context discovery attacks against a wide set of AEAD schemes, including CCM, EAX, SIV, GCM, and OCB3, and, by our general result, this gives new unrestricted context commitment attacks against them.

Finally, we consider restricted context commitment security for the original SIV mode, for which no prior attack techniques work (including our context discovery based ones). We are nevertheless able to give a novel $\mathcal{O}(2^{n/3})$ attack using Wagner's k-tree algorithm for the generalized birthday problem.

1 Introduction

Designers of authenticated encryption with associated data (AEAD) have traditionally targeted security in the sense of confidentiality and ciphertext integrity, first in the context of randomized authenticated encryption [7], and then nonce-based [37] and misuse-resistant AEAD [38].

But in recent years researchers and practitioners have begun realizing that confidentiality and integrity as previously formalized prove insufficient in a variety of contexts. In particular, the community is beginning to appreciate the danger of schemes that are not key committing, meaning that an attacker can compute a ciphertext such that it can successfully decrypt under two (or more) keys. Non-key-committing AEAD was first shown to be a problem in the context of moderation in encrypted messaging [18, 26], and later in password-based encryption [31], password-based key exchange [31], key rotation schemes [2], and symmetric hybrid (or envelope) encryption [2].

Even more recently, new definitions have been proposed [5] that target committing to the key, associated data, and nonce. And while there have been proposals for new schemes [2, 5] that meet these

 The proceedings version of this paper appears at Eurocrypt 2023. This is the full version.

varying definitions, questions still remain about which current AEAD schemes are committing and in which ways. Moreover, there have been no commitment results shown for a number of important practical AEAD schemes, such as CCM [19], EAX [12], and SIV [38]. Implementing (and standardizing) new AEAD schemes takes time and so understanding which standard AEAD schemes can be securely used in which settings is a pressing issue.

This work makes four main contributions. First, we provide a new, more granular framework for commitment security, which expands on prior ones to better capture practical attack settings. Second, we show the first key commitment attack against the original SIV mode, which was previously an open question. Third, we introduce a new kind of commitment security notion for AEAD—what we call *context discoverability*—which is analogous to preimage resistance for cryptographic hash functions. Fourth, we give context discovery attacks against a range of schemes which, by a general implication, also yield new commitment attacks against those schemes. A summary of our new attacks, including comparison with prior ones, when relevant, is given in Figure 1.

Granular commitment notions. Recall that a nonce-based AEAD encryption algorithm Enc takes as input a key K, nonce N, associated data A, and a message M. It outputs a ciphertext C. Decryption Dec likewise takes in a (K, N, A) triple, which we call the decryption *context*, along with a ciphertext C, and outputs either a message M or special error symbol $\perp$.

While most prior work has focused on key commitment security, which requires commitment to only one part (the key) of the decryption context, Bellare and Hoang (BH) [5] suggest a more expansive sequence of commitment notions for nonce-based AEAD. For the first, CMT-1, an adversary wins if it efficiently computes a ciphertext C and two decryption contexts (K_1, N_1, A_1) and (K_2, N_2, A_2) such that decryption of C under either context works (does not output $\perp$) and $K_1 \neq K_2$. CMT-1 is often called key commitment.[1] CMT-3 relaxes the latter winning condition to allow a win should the decryption contexts differ in any way. We therefore refer to CMT-3 as *context commitment* and schemes that meet CMT-3 as *context committing*. These notions form a strict hierarchy, with CMT-3 being the strongest. Despite this, most prior attacks [18, 26, 31, 2] have focused solely on key commitment (CMT-1).

Our first contribution is to refine further the definitional landscape for nonce-based AEAD schemes in a way that is particularly useful for exploring context commitment attacks. In practice, attackers will often face application-specific restrictions preventing full control over the decryption context. For example, in the Dodis, Grubbs, Ristenpart, and Woodage (DGRW) [18] attacks against Facebook's message franking scheme, the adversary had to build a ciphertext that decrypts under two contexts with equivalent nonces. Their (in BH's terminology) CMT-1 attack takes on a special form, and we would like to be able to formally distinguish between attacks that achieve additional adversarial goals (e.g., different keys but equivalent nonces) and those that may not.

We therefore introduce a new, parameterized security notion that generalizes the BH notions. Our CMT[Σ] notion specifies what we call a setting $\Sigma = (\mathsf{ts}, \mathsf{S}, \mathsf{P})$ that includes a target specifier ts, a context selector S, and a predicate P. The parameters ts and S specify which parts of the context are attacker-controlled versus chosen by the game, and which of the latter are revealed to the attacker. Furthermore, the predicate P takes as input the two decryption contexts and decrypted messages, and outputs whether the pair of tuples satisfy a winning condition. An adversary wins if it outputs a ciphertext and two contexts satisfying the condition that each decrypt the ciphertext without error. The resulting family of commitment notions includes both CMT-1 and CMT-3 but also covers a landscape of further notions.

We highlight two sets of notions. The first set is composed of $\mathrm{CMT}_{\mathsf{k}}$, $\mathrm{CMT}_{\mathsf{n}}$, and $\mathrm{CMT}_{\mathsf{a}}$, which use predicates $(K_1 \neq K_2)$, $(N_1 \neq N_2)$, and $(A_1 \neq A_2)$, respectively. The first notion is equivalent to CMT-1; the latter two are new. All of them are orthogonal to each other and a scheme that meets all three simultaneously achieves CMT-3. We say these notions are *permissive* because the predicates used do not

[1]BH refer to this as CMTD-1, but for tidy AEAD schemes, CMT-1 and CMTD-1 are equivalent, so we prefer the compact term.

Scheme	$\mathsf{CDY}^*_\mathsf{a}$	$\mathsf{CDY}^*_\mathsf{n}$	$\mathsf{CMT}^*_\mathsf{a}$	$\mathsf{CMT}^*_\mathsf{k}$	CMT_k	CMT-3
GCM [21]	★✗ §4	★✗ §4	★✗ §E	☆✗ [26, 18]	☆✗ [26, 18]	☆✗ [26, 18]
SIV [39]	★✗ §4			★✗ §5	★✗ ⏩	★✗ ⏩
CCM [19]	★✗ §4				★✗ ⏩	★✗ ⏩
EAX [12]	★✗ §4	★✗ §4			★✗ ⏩	★✗ ⏩
OCB3 [30]	★✗ §4			☆✗ [2]	☆✗ [2]	☆✗ [2]
PaddingZeros	★✓ ⏩	★✓ ⏩	★✗ §E	☆✓ [2]	☆✓ [5]	★✗ ⏩
KeyHashing	★✓ ⏩	★✓ ⏩	★✗ §E	☆✓ [2]	☆✓ [2]	★✗ ⏩
CAU-C1 [5]	★✓ ⏩	★✓ ⏩		☆✓ [5]	☆✓ [5]	★✗ ⏩

Figure 1: Summary of context discovery and commitment attacks against a variety of popular AEAD schemes. Symbol ✓ indicates a proof that any attack will take at least 2^{64} time, while symbol ✗ indicates the existence of an attack that takes less than 2^{64} time; symbol ★ indicates results new to this paper and ☆ indicates prior work (citation given). CMT_k and CMT-3 are from Bellare and Hoang [5], where CMT_k was called CMT-1. The notions $\mathsf{CDY}^*_\mathsf{a}$, $\mathsf{CDY}^*_\mathsf{n}$, $\mathsf{CMT}^*_\mathsf{a}$, and $\mathsf{CMT}^*_\mathsf{k}$ are introduced in this paper, and $\mathsf{CDY}^*_\mathsf{a}$, $\mathsf{CDY}^*_\mathsf{n}$, and $\mathsf{CMT}^*_\mathsf{k}$ are implied by CMT_k. Symbol ⏩ indicates that the result is implied from one of the other columns by a reduction shown in this paper.

make any demands on other components of the context. In contrast, *restrictive* variants, which we denote via $\mathsf{CMT}^*_\mathsf{k}$, $\mathsf{CMT}^*_\mathsf{n}$, and $\mathsf{CMT}^*_\mathsf{a}$, require equality for other context components. For example, the first uses the predicate $(K_1 \neq K_2) \wedge ((N_1, A_1) = (N_2, A_2))$. These capture the types of restrictions faced in real attacks mentioned above.

Breaking the original SIV. While prior work has shown (in our terminology) $\mathsf{CMT}^*_\mathsf{k}$ attacks for GCM [26, 18], GCM-SIV [40, 31], ChaCha20/Poly1305 [26, 31], XChaCha20/Poly1305 [31], and OCB3 [2], an open question of practical interest [41] is whether there also exists a $\mathsf{CMT}^*_\mathsf{k}$ attack against Synthetic IV (SIV) mode [38]. We resolve this open question, showing an attack that works in time about 2^{53}. It requires new techniques compared to prior attacks.

SIV combines a PRF F with CTR mode encryption, encrypting by first computing a tag $T = F_K(N, A, M)$ and then applying CTR mode encryption to M, using T as the (synthetic) IV and a second key K'. The tag and CTR mode output are, together, the ciphertext. Decryption recovers the message and then recomputes the tag, rejecting the ciphertext if it does not match. Schmieg [40] and Len, Grubbs, and Ristenpart (LGR) [31] showed that when F is a universal hash-based PRF, in particular GHASH for AES-GCM-SIV, one can achieve a fast $\mathsf{CMT}^*_\mathsf{k}$ attack.

Their attack does not extend to other versions of SIV, perhaps most notably the original version that uses for F the S2V[CMAC] PRF [38]. This version has been standardized [27] and is available in popular libraries like Tink [3]. For brevity here we describe the simpler case where F is just CMAC; the body will expand on the details. At first it might seem that CMAC's well-known lack of collision resistance (for adversarially-chosen keys), should extend to allow a simple $\mathsf{CMT}^*_\mathsf{k}$ attack: find K_1, K_2 such that $T = \mathsf{CMAC}_{K_1}(N, A, M) = \mathsf{CMAC}_{K_2}(N, A, M')$ for $M \neq M'$. But the problem is that we need M, M' to also satisfy

$$M \oplus \mathsf{CTR}_{K'_1}(T) = M' \oplus \mathsf{CTR}_{K'_2}(T) \tag{1}$$

where $\mathsf{CTR}_K(T)$ denotes running counter mode with initialization vector T and block cipher key K. When using a GHASH-based PRF, the second condition "plays well" with the algebraic structure of the first condition, making it computationally easy to satisfy both simultaneously. But, here that does not work.

The core enabler for our attack is that we can recast the primary collision finding goal as a generalized birthday bound attack. For block-aligned messages, we show how the two constraints above can be rewritten as a single equation that is the xor-sum of four terms, each taking values over $\{0, 1\}^n$. Were the terms independently and uniformly random, one would immediately have an instance of a 4-sum problem, which can be solved using Wagner's k-tree algorithm [43] in time $\mathcal{O}(2^{n/3})$. But our terms are neither independent nor uniformly random. Nevertheless, our main technical lemma shows that, in the ideal cipher model, the underlying block cipher and the structure of the terms (which are dictated by the details of CMAC-SIV) allows us to analyze the distribution of these terms and show that we can still apply the k-tree algorithm and achieve the same running time. This technique may be of independent interest.

Using this, we construct a $\mathrm{CMT}_{\mathsf{k}}^{*}$ attack against S2V[CMAC]-SIV that works in time about 2^{53}, making it practical and sufficiently damaging to rule out SIV as suitable for contexts where key commitment matters.

Context discoverability. Next we introduce a new type of security notion for AEAD. The cryptographic hashing community has long realized the significance of definitions for both collision resistance and preimage resistance [14], the latter of which, roughly speaking, refers to the ability of an attacker to find some input that maps to a target output. In analyzing $\mathrm{CMT}_{\mathsf{k}}$ security for schemes, we realized that in many cases we can give very strong attacks that, given any ciphertext, can find a context that decrypts it—a sort of preimage attack against AEAD. To avoid confusion, we refer to this new security goal for AEAD as *context discoverability (CDY)*, as the adversary is tasked with efficiently computing ("discovering") a suitable context for some target ciphertext.

While we have not seen real attacks that exploit context discoverability, since CDY is to CMT what preimage resistance is to collision resistance, we believe that they are inevitable. We therefore view it beneficial to get ahead of the curve and analyze the CDY security before concrete attacks surface.

We formalize a family of CDY definitions similarly to our treatment for CMT. Our CDY[Σ] notion is parameterized by a setting $\Sigma = (\mathsf{ts}, \mathsf{S})$ that specifies a target specifier ts and a context selector S. Like for CMT[Σ], ts and S specify the parts of the context that the attacker can choose and which parts are chosen by the game and either hidden or revealed to the attacker. Unlike CMT, however, the attacker is always given a target ciphertext and needs to only produce one valid decrypting context.

Similar to $\mathrm{CMT}_{\mathsf{k}}^{*}$, $\mathrm{CMT}_{\mathsf{n}}^{*}$, $\mathrm{CMT}_{\mathsf{a}}^{*}$, we define the notions $\mathrm{CDY}_{\mathsf{k}}^{*}$, $\mathrm{CDY}_{\mathsf{n}}^{*}$, $\mathrm{CDY}_{\mathsf{a}}^{*}$. The notion $\mathrm{CDY}_{\mathsf{k}}^{*}$ captures the setting where an adversary is given arbitrary ciphertext C, nonce N, and associated data A, and must produce a key K such that C decrypts under (K, N, A). Similarly, $\mathrm{CDY}_{\mathsf{n}}^{*}$ and $\mathrm{CDY}_{\mathsf{a}}^{*}$ require the adversary to provide a nonce and associated data, respectively, given the other components chosen arbitrarily. These model restricted attack settings where parts of the context are not in the adversary's control.

We also define $\mathrm{CDY}^{*}[\mathsf{ts}]$ which generalizes this intuition to any target specifier ts. For example, in $\mathrm{CDY}^{*}[\mathsf{ts} = \{\mathsf{n}\}]$ the adversary is given arbitrary ciphertext and nonce N, and must produce a key K and associated data A such that the ciphertext decrypts under (K, N, A).

We next analyze the relations between these sets of notions. In particular, we show that if an AEAD scheme is "context compressing"—ciphertexts are decryptable under more than one context—then CMT-3 security implies CDY^{*}. This is analogous to collision resistance implying preimage resistance, though the details are different. Further, we observe that almost all deployed AEAD schemes are context compressing since they "compress" the nonce and associated data into a shorter tag. This allows us to focus on finding $\mathrm{CDY}^{*}[\Sigma]$ attacks for AEAD schemes to show that these schemes also do not meet CMT[Σ] security. Selected relationships are shown in Figure 2.

This opens up a new landscape of analysis, which we explore. We characterize a large class of AEAD schemes that use non-preimage resistant MACs and, based on this weakness, develop fast $\mathrm{CDY}_{\mathsf{a}}^{*}$ attacks. The set includes CCM, EAX, SIV, GCM, and OCB3. For EAX and CCM, this represents the first attacks of any kind for committing security. For EAX and GCM, we are also able to give $\mathrm{CDY}_{\mathsf{n}}^{*}$ attacks, which is perhaps even more surprising a priori, given that an adversary in this case only controls the nonce.

All this sheds light on the deficiencies of several popular design paradigms for AEAD, when viewed from the perspective of context commitment security. These definitions also allow us to precisely communicate attacks and threat models. For example, CDY might suffice for some applications while others might want the more computationally expensive CMT security.

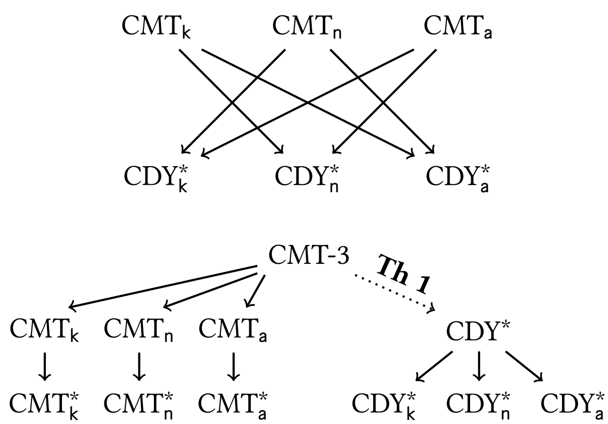


Figure 2: **(Top)** Selected relationships between permissive CMT notions and restrictive CDY notions. Solid arrows represent implications. **(Bottom)** Selected relationships between CMT-3 and the notions we introduce in this paper. Solid arrows represent implications. The dotted arrow from CMT-3 to CDY^* holds assuming "context compression" as defined in Theorem 1.

Revisiting commitment-enhancing mechanisms. Finally, in Section E we use this new framework to analyze proposed mechanisms for commitment security. First, we look at the folklore padding zeros transform which prefixes zeroes to a message before encrypting and verifies the existence of these zeroes at decryption. This transform was recommended in an early OPAQUE draft specification [29, §3.1.1] and was shown by Albertini et al. [2, §5.3] to achieve FROB security and by Bellare and Hoang [5] to achieve CMT-1 security. We show that this transform does not achieve our $\text{CMT}^*_{\mathsf{a}}$ notion (and thus CMT-3) for all AEAD schemes, ruling it out as a candidate commitment security transform. We then make similar observations about the $\mathsf{CommitKey}$ transform which appends to the ciphertext a hash commitment to the key and the nonce. Finally, we conclude by considering the practical key commitment security of the recent CAU-C1 scheme from BH [5]. While a naive adaptation of DGRW's [18] "invisible salamanders" attack to this scheme takes about 2^{81} time, we show a more optimized attack which takes a little more than 2^{64} time, showing that 64-bit key-committing security does not preclude practical attacks.

Next steps and open problems. Our results resolve a number of open problems about AEAD commitment security, and overall highlight the value of new definitional frameworks that surface different avenues for attack. That said, we leave several open problems, such as whether different flavors of context discovery or commitment attacks can be found against popular schemes—the blank entries in Figure 1. Our attack techniques do not seem to work against these schemes, but whether positive security results can be shown is unclear.

2 Background

Notation. We refer to elements of $\{0,1\}^*$ as *bitstrings*, denote the length of a bitstring x by $|x|$ and the leftmost (i.e., "most-significant") bit by $\mathsf{msb}(x)$. Given two bitstrings x and y, we denote their concatenation by $x \parallel y$, their bitwise xor by $x \oplus y$, and their bitwise and by $x \& y$. Given a number n, we denote its m-bit encoding as $\mathsf{encode}_m(n)$. For a finite set X, we use $x \leftarrow\$ X$ to denote sampling a uniform, random element from X and assigning it to x.

Sometimes, we operate in the finite field $\mathrm{GF}(2^n)$ with 2^n elements. This field is defined using an irreducible polynomial $f(\alpha)$ in $\mathrm{GF}(2)[\alpha]$ of degree n. While the choice of polynomial affects the representation and the implementation of some field operations, all finite fields of size 2^n are isomorphic, so the algorithms presented do not rely on this detail. The elements of the field are polynomials $x_0 + x_1\alpha + x_2\alpha^2 + \cdots + x_{n-1}\alpha^{n-1}$ of degree $n-1$ with binary coefficients $x_i \in \mathrm{GF}(2)$. These polynomials can be represented by the n-bit string $x_0 x_1 \cdots x_{n-1}$ of their coefficients. Both addition and subtraction of two n-bit strings, denoted $x+y$ and $x-y$, respectively, is their bitwise xor $x \oplus y$. Multiplication of two n-bit strings, denoted $x \cdot y$, corresponds to the multiplication of the corresponding polynomials x and y followed by modular reduction with the irreducible polynomial $f(\alpha)$. For concreteness, we illustrate how to double a 128-bit string with the GCM [21] polynomial $f(\alpha) = 1 + \alpha + \alpha^2 + \alpha^7 + \alpha^{128}$, denoted $2 \cdot x$, as $(x \gg 1) \oplus \Delta$ where $\Delta = 0^{128}$ if $x_{127} = 0$ and $\Delta = 111000010^{120}$ otherwise. A general method for multiplying any two 128-bit strings is given in the GCM specification [21, §6.3]. Once we have multiplication, we can implement the multiplicative inverse of a nonzero n-bit string, denoted x^{-1} as x^{2^n-2} using Lagrange's theorem.[2]

Probability. An *n-bit random variable* X is one whose value is probabilistically assigned, defined by *probability mass function* $p_X(x) := \Pr[X = x]$. We require that the probability of X over all n-bit strings sums to one, $\sum_{x \in \{0,1\}^n} p_X(x) = 1$. We say that two n-bit random variables X and Y are *independent* if, for all $x \in \{0,1\}^n$ and for all $y \in \{0,1\}^n$, it holds that $\Pr[(X = x) \wedge (Y = y)] = \Pr[X = x] \cdot \Pr[Y = y]$. The *$n$-bit uniform random variable* U is the random variable with the probability mass function $p_U(x) = \frac{1}{2^n}$ for all $x \in \{0,1\}^n$. Given two n-bit random variables X and Y, we define the *total variation distance* between them

$$\Delta(X,Y) := \max_{i \in \{0,1\}^n} \big| \Pr(X = i) - \Pr(Y = i) \big| .$$

A *random function* F from n-bit strings to m-bit strings is a collection $\{X_i : i \in \{0,1\}^n\}$ of m-bit random variables X_i, one for each n-bit input, such that for all $i \in \{0,1\}^n$, $F(i) := X_i$. A random function F from n-bit strings to m-bit strings is *uniformly random* if, for all $i \in \{0,1\}^n$, $F(i)$ is the m-bit uniform random variable. Since there is only one uniformly random function from n-bit strings to m-bit strings, we refer to it as the *uniform random function.* We say that two random functions F_1 and F_2 from n-bit strings to m-bit strings are *independent* if, for all $i \in \{0,1\}^n$ and for all $j \in \{0,1\}^n$, $F_1(i)$ and $F_2(j)$ are independent m-bit random variables.

Regularity and birthday attacks. Following Bellare and Kohno [6], we say that a function is regular if each output has the same number of preimages. More formally, a function $F : \{0,1\}^n \to \{0,1\}^m$ for $m < n$ is *regular* if $|F^{-1}(y)| = \frac{2^n}{2^m}$ for all $y \in \{0,1\}^m$, where $F^{-1}(y) := \{x \in \{0,1\}^n : F(x) = y\}$. And, if F is regular, then a *birthday attack* which randomly samples input points, succeeds in finding a collision with about $2^{m/2}$ trials. If F is not regular, then a birthday attack is expected to succeed with fewer than $2^{m/2}$ trials. In sum, regularity captures the worst-case runtime for a birthday attack.

Code-based games. To formalize security experiments, we use the *code-based games* framework of Bellare and Rogaway [11]; with refinements from Ristenpart, Shacham, and Shrimpton [36]. A *procedure* P is a sequence of code-like statements that accepts some input and produces some output. The types of variables in the code-like syntax should be clear from context and are assumed to be appropriately initialized. For example, a variable-length array T is initialized to be the empty array with subsequent operations dynamically resizing it. Procedures can also use random coins, the use of coins is usually implicit (like sampling from a discrete set) but should be clear from context. We use superscripts like P^Q to denote that procedure P calls procedure Q. An *adversary* $\mathcal{A}$ is a procedure that implements an *interface* that should be clear from context. And a *game* G is a distinguished procedure that accepts an adversary $\mathcal{A}$ with a specified interface as input, and denoted as $G(\mathcal{A})$. We use $(G \Rightarrow x)$ to denote the event that the procedure G outputs x, over the random coins of the procedure. Finally, given a game G and an adversary $\mathcal{A}$, we denote the *advantage* of $\mathcal{A}$ at G by $\mathbf{Adv}_G(\mathcal{A}) := \Pr[G(\mathcal{A}) \Rightarrow \mathsf{true}]$.

[2] Since $\mathrm{GF}(2^n)$ is a field, the set of nonzero elements x under multiplication form a cyclic group of order $2^n - 1$ so $x^{2^n-1} = 1$.

Cost of attacks. We represent cryptanalytic attacks by procedures and compute their cost using a *unit-cost RAM model*. Specifically, following [36], we use the convention that each pseudocode statement of a procedure runs in unit time. This lets us write the running time of a procedure as the maximum number of statements executed, with the maximum taken over all inputs of a given size. Similarly, we define the number of queries as the maximum number of queries executed over inputs of a given size. We recognize that this is a simplification of the real-world (e.g., see Wiener [44]), but for the attacks discussed in this paper, we nevertheless believe that it provides a good estimate.

Pseudorandom functions. A *pseudorandom function (PRF)* is a function $\mathsf{F} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ defined over a key space $\mathcal{K} \subseteq \{0,1\}^*$, message space $\mathcal{M} \subseteq \{0,1\}^*$, and output space $\mathcal{Y} \subseteq \{0,1\}^*$, that is indistinguishable from a uniform random function. More formally, we define the PRF advantage of an adversary $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(\mathcal{A}) := \left|\Pr[K \leftarrow\!\!\$\, \mathcal{K} : \mathcal{A}(\mathsf{F}(K,\cdot))] - \Pr[R \leftarrow\!\!\$\, \mathrm{Func} : \mathcal{A}(R)]\right|,$$

and say that F is a PRF if this advantage is small for all adversaries $\mathcal{A}$ that run in a feasible amount of time.

Hash functions. A *hash function* is a function $\mathsf{H} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, defined over a key space $\mathcal{K} \subseteq \{0,1\}^*$, message space $\mathcal{M} \subseteq \{0,1\}^*$, and hash space $\mathcal{Y} \subseteq \{0,1\}^*$. There are many definitions for hash function security [38], but we focus on *collision-resistance* which captures the hardness of finding distinct inputs that produce colliding outputs. We define the collision-resistance advantage of adversary $\mathcal{A}$ for H as

$$\mathbf{Adv}^{\mathrm{coll}}_{\mathsf{H}}(\mathcal{A}) := \Pr\left[K \leftarrow\!\!\$\, \mathcal{K}, (M_1, M_2) \leftarrow\!\!\$\, \mathcal{A}(K) : (M_1 \neq M_2) \text{ and } (\mathsf{H}(K, M_1) = \mathsf{H}(K, M_2))\right].$$

and say that H is a collision-resistant hash function if this advantage is small for all adversaries $\mathcal{A}$ that run in a feasible amount of time.

Block ciphers and the ideal cipher model. An n-bit *block cipher*, or a block cipher with *block length n* bits, is a function $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, where for each key $k \in \{0,1\}^n$, $E(k,\cdot)$ is a permutation on $\{0,1\}^n$. Since it is a permutation, it has an inverse which we denote by $E^{-1}(k,\cdot)$. To simplify notation, we sometimes use the shorthands $E_k(\cdot) := E(k,\cdot)$ and $E_k^{-1}(\cdot) := E^{-1}(k,\cdot)$.

An n-bit *ideal block cipher* [28] is a random map $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, such that for each key $k \in \{0,1\}^n$, $E_k(\cdot)$ is a permutation on $\{0,1\}^n$. Alternatively, we can think of an ideal block cipher as one where for each key $k \in \{0,1\}^n$, $E_k(\cdot)$ is uniformly, randomly sampled from the set of permutations on n-bits.

Authenticated encryption schemes. An *AEAD scheme* is a triple of algorithms $\mathsf{AEAD} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$, defined over a key space $\mathcal{K} \subseteq \{0,1\}^*$, nonce space $\mathcal{N} \subseteq \{0,1\}^*$, associated data space $\mathcal{A} \subseteq \{0,1\}^*$, message space $\mathcal{M} \subseteq \{0,1\}^*$, and ciphertext space $\mathcal{C} \subseteq \{0,1\}^*$.

1. $\mathsf{Kg} : \varnothing \to \mathcal{K}$ is a randomized algorithm that takes no input and returns a fresh secret key K.
2. $\mathsf{Enc} : (\mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}) \to (\mathcal{C} \cup \{\perp\})$ is a deterministic algorithm that takes a 4-tuple of a key K, nonce N, associated data A, and message M and returns a ciphertext C or an error (denoted by $\perp$).
3. $\mathsf{Dec} : (\mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C}) \to (\mathcal{M} \cup \{\perp\})$ is a deterministic algorithm that takes a 4-tuple of a key K, nonce N, associated data A, and ciphertext C and returns a plaintext M or an error (denoted by $\perp$).

We call the non-message inputs to Enc—the key, nonce, and associated data—the *encryption context* and the non-ciphertext inputs to Dec—the key, nonce, and associated data—the *decryption context*. And, for a given message, say that an encryption context is *valid* if Enc succeeds (i.e., does not output $\perp$). Similarly, for a given ciphertext, say that a decryption context is *valid* if Dec succeeds (i.e., does not output $\perp$).

For traditional AEAD correctness, we need Enc to be the inverse of Dec. In other words, for any 4-tuple $(K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$, it holds that

$$\mathsf{Dec}(K, N, A, \mathsf{Enc}(K, N, A, M)) = M.$$

$\underline{\text{CMT-1}(\mathcal{A}):}$	$\underline{\text{CMT-3}(\mathcal{A}):}$
$((K_1, N_1, A_1), (K_2, N_2, A_2), C) \leftarrow\$ \mathcal{A}$	$((K_1, N_1, A_1), (K_2, N_2, A_2), C) \leftarrow\$ \mathcal{A}$
$M_1 \leftarrow \mathsf{AEAD.Dec}(K_1, N_1, A_1, C)$	$M_1 \leftarrow \mathsf{AEAD.Dec}(K_1, N_1, A_1, C)$
$M_2 \leftarrow \mathsf{AEAD.Dec}(K_2, N_2, A_2, C)$	$M_2 \leftarrow \mathsf{AEAD.Dec}(K_2, N_2, A_2, C)$
// decryption success	// decryption success
If $M_1 = \bot$ or $M_2 = \bot$	If $M_1 = \bot$ or $M_2 = \bot$
Return false	Return false
// commitment condition	// commitment condition
If $K_1 = K_2$	If $(K_1, N_1, A_1) = (K_2, N_2, A_2)$
Return false	Return false
Return true	Return true

Figure 3: **(Left)** The CMT-1 game [5]. **(Right)** The CMT-3 game [5]. The differences are highlighted.

In addition, we impose *tidyness* [35], *ciphertext validity*, and *length uniformity* assumptions. Tidyness requires that for any 4-tuple $(K, N, A, C) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C}$, it holds that

$$\mathsf{Dec}(K, N, A, C) = M \neq \bot \implies \mathsf{Enc}(K, N, A, M) = C\,.$$

Ciphertext validity requires that for every ciphertext $C \in \mathcal{C}$ there exists at least one valid decryption context $(K, N, A) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A}$; that is $\mathsf{Dec}(K, N, A, C) \neq \bot$. Length uniformity requires that the length of a ciphertext depends only on the length of the message and the length of the associated data.

Finally, for AEAD security, we use the traditional *privacy* and *authenticity* definitions [37, §3].

Committing authenticated encryption. A number of prior notions for committing AEAD have been proposed. In Figure 3 we provide the CMT-1 and CMT-3 games from Bellare and Hoang [5]. The FROB game from Farshim, Orlandi, and Rosie [24] adapted to the AEAD setting by Grubbs, Lu, and Ristenpart [26], is the same except that the final highlighted predicate is changed to "$K_1 = K_2$ or $N_1 \neq N_2$". The FROB game asks the adversary to produce a ciphertext that decrypts under two different keys with the same nonce. The CMT-1 game is more permissive and removes the condition that the nonce be the same. The CMT-3 game is even more permissive and relaxes the different key condition to different keys, nonces, or associated data. Bellare and Hoang [5] show that CMT-3 implies CMT-1, which implies FROB. We will expand on these definitions with a more general framework next.

3 Granular Committing Encryption Definitions

We provide a more general framework for defining commitment security for encryption. As motivation, we observe that while the CMT-1 and the stronger CMT-3 notions provide good security goals for constructions, they do not precisely capture the *way* in which attacks violate security—which parts of the decryption context does the attacker need to control, which parts have been pre-selected by some other party, and which parts are known to the attacker.

These considerations are crucial for determining the exploitability of commitment vulnerabilities in practice. For instance, the vulnerability in Facebook attachment franking [22] exploited by Dodis et al. [18, §3] only works if the nonces are the same; and the key rotation attack described by Albertini et al. [2] only works with keys previously imported to the key management service. And, looking ahead, we propose a variant of the Subscribe with Google attack described by Albertini et al. [2] in which a malicious publisher provides a full decryption context only knowing the honestly published ciphertext.

We provide a more general framework for commitment security notions that more precisely captures attack settings. As we will see in subsequent sections, our definitions provide a clearer explanatory framework for vulnerabilities.

$\underline{\text{CMT}[\mathsf{ts},\mathsf{S},\mathsf{P}](\mathcal{A})\text{:}}$

$\text{cat}_c \leftarrow\!\!\$\ \mathsf{S}$
$\text{cat}_a \leftarrow\!\!\$\ \mathcal{A}(\text{Reveal}_{\mathsf{ts}}(\text{cat}_c))$
$\text{cat} \leftarrow \text{Merge}_{\mathsf{ts}}(\text{cat}_c, \text{cat}_a)$
If $\text{cat} = \bot$:
  Return false
$(C, (K_1, N_1, A_1), (K_2, N_2, A_2)) \leftarrow \text{cat}$
$M_1 \leftarrow \mathsf{AEAD.Dec}(K_1, N_1, A_1, C)$
$M_2 \leftarrow \mathsf{AEAD.Dec}(K_2, N_2, A_2, C)$
If $M_1 = \bot$ or $M_2 = \bot$:
  Return false
Return $\mathsf{P}((K_1, N_1, A_1), (K_2, N_2, A_2))$

Notion	**Predicate** P
CMT_{k}	$(K_1 \neq K_2)$
CMT_{n}	$(N_1 \neq N_2)$
CMT_{a}	$(A_1 \neq A_2)$
$\text{CMT}^*_{\mathsf{k}}$	$(K_1 \neq K_2) \wedge (N_1, A_1) = (N_2, A_2)$
$\text{CMT}^*_{\mathsf{n}}$	$(N_1 \neq N_2) \wedge (K_1, A_1) = (K_2, A_2)$
$\text{CMT}^*_{\mathsf{a}}$	$(A_1 \neq A_2) \wedge (K_1, N_1) = (K_2, N_2)$

Figure 4: **(Left)** The CMT[Σ] commitment security game, parameterized by $\Sigma = (\mathsf{ts}, \mathsf{S}, \mathsf{P})$, a target selector ts, context selector S, and predicate P. **(Right)** Predicates for the permissive notions CMT_{k}, CMT_{n}, CMT_{a} and restrictive notions $\text{CMT}^*_{\mathsf{k}}$, $\text{CMT}^*_{\mathsf{n}}$, $\text{CMT}^*_{\mathsf{a}}$, where $\mathsf{ts} = \varnothing$.

Committing security framework. We find it useful to expand the set of security notions to more granularly capture the ways in which the two decryption contexts are selected that generalizes context commitment security. In Figure 4 we detail the CMT[Σ] game, parameterized by a *setting* $\Sigma = (\mathsf{ts}, \mathsf{S}, \mathsf{P})$ that specifies a *target specifier* ts, a *context selector* S, and a *predicate* P (to be defined next.) The adversary helps compute a ciphertext and two decryption contexts $(C, (K_1, N_1, A_1), (K_2, N_2, A_2))$, what we call a *commitment attack instance (cat)*. The adversary wins if C decrypts under both decryption contexts, and the two decryption contexts satisfy the predicate P. The parameterization allows attack settings in terms of which portions of the commitment attack instance are attacker controlled versus chosen in some other way, and which of the latter are revealed to the attacker.

We now provide more details. A commitment attack instance is a tuple $(C, (K_1, N_1, A_1), (K_2, N_2, A_2))$ consisting of a ciphertext $C \in \mathcal{C}$; two keys $K_1, K_2 \in \mathcal{K}$; two nonces $N_1, N_2 \in \mathcal{N}$; and two associated data $A_1, A_2 \in \mathcal{A}$. A target specifier ts is a subset of labels $\{\mathsf{C}, \mathsf{k}_1, \mathsf{n}_1, \mathsf{a}_1, \mathsf{k}_2, \mathsf{n}_2, \mathsf{a}_2\} \times \{\cdot, \hat{\cdot}\}$. The left set labels the components of a commitment attack instance, called component labels, and the right set denotes whether the specified component is revealed to the adversary (no hat means revealed and hat means not revealed.) For example, $\mathsf{ts} = \{\mathsf{k}_1, \hat{\mathsf{k}}_2\}$ indicates the K_1 and K_2 in the context, and that K_1 is revealed to the attacker.

A context selector S is a randomized algorithm that takes no input and produces the challenger-defined elements of a commitment attack instance, denoted cat_c, as specified by the target specifier ts. The reveal function $\text{Reveal}_{\mathsf{ts}}$ parameterized by ts, takes a subset of a commitment attack instance and reveals the components that ts tells it to reveal; i.e., the specified components with no hat. The merge function $\text{Merge}_{\mathsf{ts}}(\text{cat}_c, \text{cat}_a)$ parameterized by the target specifier ts, takes two subsets of commitment attack instances cat_c (challenger-defined elements) and cat_a (adversary-defined elements) and works as follows. First, it checks for every component specified by ts that cat_c has a corresponding value. Second, it checks that for every component specified by ts, if cat_a has a value, that it matches the value in cat_c. If either of these checks fail, it outputs $\bot$. Otherwise, it returns their union $\text{cat}_c \cup \text{cat}_a$. Finally, the predicate P takes two decryption contexts output by $\text{Merge}_{\mathsf{ts}}(\text{cat}_c, \text{cat}_a)$, and outputs true if they satisfy some criteria (e.g., that $K_1 \neq K_2$), and false otherwise.

We associate to a setting $\Sigma = (\mathsf{ts}, \mathsf{S}, \mathsf{P})$, AEAD Π, and adversary $\mathcal{A}$ the CMT advantage defined as

$$\mathbf{Adv}^{\text{CMT}[\Sigma]}_{\Pi}(\mathcal{A}) := \Pr\left[\, \text{CMT}[\Sigma](\mathcal{A}) \Rightarrow \mathsf{true} \,\right].$$

Taking a concrete security approach, we will track the running time used by $\mathcal{A}$ and provide explicit advantage functions. Adapting our notions to support asymptotic definitions of security is straightforward:

in our discussions we will often say a scheme is CMT[Σ] secure as informal shorthand that no adversary can win the CMT[Σ] game with "good" probability using "reasonable" running time.

Capturing CMT-1, CMT-3, and more via predicates. To understand our definitional framework further, we can start by seeing how to instantiate it to coincide with prior notions. Let $\mathsf{ts} = \varnothing$ indicate the empty target selector, meaning that $\mathcal{A}$ chooses the ciphertext and two decryption contexts fully. Then the set of Σ settings that use the empty target selector defines a family of security goals, indexed solely by predicates, which we denote by CMT[P]. This family includes CMT-1 by setting $\mathsf{P} := (K_1 \neq K_2)$ and CMT-3 by setting $\mathsf{P} := (K_1, N_1, A_1) \neq (K_2, N_2, A_2)$. Not all instances in this family are interesting: consider, for example, when P always outputs true or false. Nevertheless, the flexibility here allows for more granular specification of adversarial ability. For instance, the predicate that requires $(K_1 \neq K_2) \wedge (N_1 = N_2)$ captures a setting like that of the Dodis et al. [18] attack against Facebook's message franking, which requires that both decryption contexts have the same nonce.

Three games of particular interest are those with predicates that focus on inequality of the three individual context components: $(K_1 \neq K_2)$, $(N_1 \neq N_2)$, and $(A_1 \neq A_2)$. For notational brevity, we let game $\mathrm{CMT}_{\mathsf{k}} := \mathrm{CMT}[\mathsf{P} = (K_1 \neq K_2)]$ and similarly $\mathrm{CMT}_{\mathsf{n}} := \mathrm{CMT}[\mathsf{P} = (N_1 \neq N_2)]$ and $\mathrm{CMT}_{\mathsf{a}} := \mathrm{CMT}[\mathsf{P} = (A_1 \neq A_2)]$. Then $\mathrm{CMT}_{\mathsf{k}}$ corresponds to CMT-1, but $\mathrm{CMT}_{\mathsf{n}}$ and $\mathrm{CMT}_{\mathsf{a}}$ are new. They are also orthogonal to CMT-1, in the sense that we can give schemes that achieve CMT-1 but not $\mathrm{CMT}_{\mathsf{a}}$ nor $\mathrm{CMT}_{\mathsf{n}}$ security (see Theorem 10). All three are, however, implied by being CMT-3 secure, and a scheme that simultaneously meets $\mathrm{CMT}_{\mathsf{k}}$, $\mathrm{CMT}_{\mathsf{n}}$, and $\mathrm{CMT}_{\mathsf{a}}$ also enjoys CMT-3 security (see Lemmas 8 and 9.)

Note that $\mathrm{CMT}_{\mathsf{k}}$, $\mathrm{CMT}_{\mathsf{n}}$, and $\mathrm{CMT}_{\mathsf{a}}$ are *permissive*: as long as the relevant component is distinct across the two contexts, it does not matter whether the other components are distinct. Also, of interest are *restrictive* versions; for example, we can consider $\mathrm{CMT}^*_{\mathsf{k}} := \mathrm{CMT}[(K_1 \neq K_2) \wedge (N_1, A_1) = (N_2, A_2)]$ which requires that the nonces and associated data are the same. Similarly, we can define restrictive notions $\mathrm{CMT}^*_{\mathsf{n}}$ and $\mathrm{CMT}^*_{\mathsf{a}}$. Restrictive versions are useful as they correspond to attacks that have limited control over the decryption context. Interestingly, these restrictive notions are not equivalent to the corresponding permissive notions, nor does a scheme that simultaneously meets $\mathrm{CMT}^*_{\mathsf{k}}$, $\mathrm{CMT}^*_{\mathsf{n}}$, and $\mathrm{CMT}^*_{\mathsf{a}}$ achieve CMT-3 security (see Theorem 11.)

Targeted attacks. Returning to settings with target specifier $\mathsf{ts} \neq \varnothing$, we can further increase the family of notions considered to capture situations where a portion of the context is pre-selected. For instance, in the key rotation example of Albertini et al. [2] mentioned earlier, we would have $\mathsf{ts} = \{\mathsf{k}_1, \mathsf{k}_2\}$ and $\mathsf{S} = \{K_1 \leftarrow_{\$} \mathcal{K}; K_2 \leftarrow_{\$} \mathcal{K}; \text{Return } (K_1, K_2)\}$ to indicate that the malicious sender has to use the two randomly generated keys.

However, not all targeted attack settings are interesting. For some target specifiers ts, we can specify a context selector S such that no adversary can achieve non-zero advantage. In particular, if we have $\mathsf{ts} = \{\mathsf{C}, \mathsf{k}_1, \mathsf{n}_1, \mathsf{a}_1\}$ and have S pick ciphertext C and context (K_1, N_1, A_1) such that $\mathsf{AEAD.Dec}(K_1, N_1, A_1, C)$ returns $\perp$, then no adversary can win the game, making the security notion trivial (all schemes achieve it.)

Hiding target components. Finally, our game considers target specifiers ts that indicate that some values chosen by S should remain hidden from $\mathcal{A}$. For example, the Subscribe with Google attack described by Albertini et al. [2] can be reframed as a meddler-in-the-middle attack as follows. A publisher creates premium content M_1 and encrypts it using a context (K_1, N_1, A_1) to get a ciphertext C. The ciphertext C is published, but the context (K_1, N_1, A_1) is hidden. A malicious third-party, only looking at the ciphertext C, tries to construct a valid decryption context (K_2, N_2, A_2) and uses that to sell fake paywall bypasses. We can formalize this setting by having the target specifier $\mathsf{ts} = \{\mathsf{C}, \hat{\mathsf{k}}_1, \hat{\mathsf{n}}_1, \hat{\mathsf{a}}_1\}$, with the context selector S as

$$K_1 \leftarrow_{\$} \mathcal{K};\, N_1 \leftarrow_{\$} \mathcal{N};\, A_1 \leftarrow_{\$} \mathcal{A};\, M_1 \leftarrow_{\$} \mathcal{M};\ \text{Return } (\mathsf{AEAD.Enc}(K_1, N_1, A_1, M_1), K_1, N_1, A_1)$$

and with $\mathsf{Reveal}_{\mathsf{ts}}(C, K_1, N_1, A_1)$ outputting C.

$\underline{\mathrm{CDY}[\mathsf{ts},\mathsf{S}](\mathcal{A})\text{:}}$
$\mathrm{dat}_c \leftarrow\!\!\$\ \mathsf{S}$
$\mathrm{dat}_a \leftarrow\!\!\$\ \mathcal{A}(\mathrm{Reveal}_{\mathsf{ts}}(t))$
$\mathrm{dat} \leftarrow \mathrm{Merge}_{\mathsf{ts}}(\mathrm{dat}_c, \mathrm{dat}_a)$
If $\mathrm{dat} = \bot$:
Return false
$(C, (K, N, A)) \leftarrow \mathrm{dat}$
$M \leftarrow \mathsf{AEAD.Dec}(K, N, A, C)$
If $M = \bot$:
Return false
Return true

$\underline{\mathrm{CDY}[\{\mathsf{k},\mathsf{n}\},\mathsf{S}](\mathcal{A})\text{:}}$
$(C, K, N) \leftarrow\!\!\$\ \mathsf{S}$
$A \leftarrow\!\!\$\ \mathcal{A}(C, K, N)$
$M \leftarrow \mathsf{AEAD.Dec}(K, N, A, C)$
If $M = \bot$:
Return false
Return true

$\underline{\mathrm{CDY}[\{\mathsf{k},\mathsf{a}\},\mathsf{S}](\mathcal{A})\text{:}}$
$(C, K, A) \leftarrow\!\!\$\ \mathsf{S}$
$N \leftarrow\!\!\$\ \mathcal{A}(C, K, A)$
$M \leftarrow \mathsf{AEAD.Dec}(K, N, A, C)$
If $M = \bot$:
Return false
Return true

Figure 5: **(Left)** The $\mathrm{CDY}[\mathsf{ts},\mathsf{S}]$ commitment security game, parameterized by a target specifier ts and a context selector S. **(Middle)** The variant of $\mathrm{CDY}[\Sigma]$ used in the definition of $\mathrm{CDY}^*_{\mathsf{a}}$. **(Right)** The variant of $\mathrm{CDY}[\Sigma]$ used in the definition of $\mathrm{CDY}^*_{\mathsf{n}}$.

Context discoverability security. Dodis et al [18, §5] and Albertini et al. [2, §3.3] have pointed out that traditional CMT games are analogous to collision-resistance for hash functions, in the sense that the goal is to find two different *encryption contexts* (K_1, N_1, A_1, M_1) and (K_2, N_2, A_2, M_2) such that they produce the same ciphertext C. Under this lens, CMT with targeting (and no hiding) is like second preimage resistance, and CMT with targeting and hiding is like preimage resistance. But, the analogy to preimage resistance is not perfect, since we are not asking for *any* preimage but rather one that is not the same as the original. Further, this restriction is unnecessary. Going back to the meddler-in-the-middle example above, it suffices for an on-path attacker to produce any valid context. Thus, we find it useful to define a new preimage resistance-inspired notion of commitment security.

In Figure 5 we define the game $\mathrm{CDY}[\mathsf{ts},\mathsf{S}]$, parameterized by a setting $\Sigma = (\mathsf{ts},\mathsf{S})$ that specifies a target specifier ts and a context selector S. In more detail, a *discoverability attack instance (dat)* is a ciphertext and a decryption context $(C, (K, N, A))$. Here, a target specifier ts is a subset of $\{\mathsf{k},\mathsf{n},\mathsf{a}\} \times \{\cdot, \hat{\cdot}\}$ and a context selector S is a randomized algorithm that takes no input and produces a ciphertext and the elements of a decryption context specified by the target specifier ts. The reveal function $\mathrm{Reveal}_{\mathsf{ts}}$ and the merge function $\mathrm{Merge}_{\mathsf{ts}}(\mathrm{dat}_c, \mathrm{dat}_a)$ work similarly to their CMT counterparts. Finally, the goal of the adversary is to produce *one* valid decryption context for the target ciphertext.

We associate to a setting $\Sigma = (\mathsf{ts},\mathsf{S})$, AEAD scheme Π, and adversary $\mathcal{A}$ the CDY advantage defined as

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

Restricted CDY and its variants. To more accurately capture attack settings and to prove relations, we find it useful to define restricted variants of the $\mathrm{CDY}[\Sigma]$ game. A class of games of particular interest are ones that allow targeting under *any* context selector; we call this class *restricted CDY.* For a target specifier ts, let $\mathrm{CDY}^*[\mathsf{ts}]$ be the game where the adversary is given a ciphertext and elements of a decryption context specified by ts, all selected arbitrarily, and needs to produce the remaining elements of a decryption context such that $\mathsf{AEAD.Dec}(K, N, A, C) \neq \bot$. Formally, for an AEAD scheme Π and adversary $\mathcal{A}$, we define the CDY^* advantage as

$$\mathbf{Adv}_{\Pi}^{\mathrm{CDY}^*[\mathsf{ts}]}(\mathcal{A}) = \Pr\left[\, \text{for all } \mathsf{S},\ \mathrm{CDY}[\mathsf{ts},\mathsf{S}](\mathcal{A}) \Rightarrow \mathsf{true} \,\right].$$

In addition, we find it useful to define three specific variants of CDY^* that allow targeting two-of-three components of a decryption context. Let $\mathrm{CDY}^*_{\mathsf{a}}$ be the game where the adversary is given an arbitrary ciphertext C, key K, and nonce N, and has to produce associated data A such that $\mathsf{AEAD.Dec}(K, N, A, C) \neq \bot$. Formally, for an AEAD scheme Π and adversary $\mathcal{A}$, we define the $\mathrm{CDY}^*_{\mathsf{a}}$ advantage as

$$\mathbf{Adv}_{\Pi}^{\mathrm{CDY}^*_{\mathsf{a}}}(\mathcal{A}) = \Pr\left[\, \text{for all } \mathsf{S},\ \mathrm{CDY}[\{\mathsf{k},\mathsf{n}\},\mathsf{S}](\mathcal{A}) \Rightarrow \mathsf{true} \,\right].$$

The $\text{CDY}^*_{\mathsf{k}}$ and $\text{CDY}^*_{\mathsf{n}}$ games are defined similarly where the adversary has to produce a valid key and nonce respectively such that decryption succeeds when the remaining inputs to decryption are pre-selected. Formally, for an AEAD scheme Π and adversary $\mathcal{A}$, we define the $\text{CDY}^*_{\mathsf{k}}$ and $\text{CDY}^*_{\mathsf{n}}$ advantage as

$$\mathbf{Adv}_{\Pi}^{\text{CDY}^*_{\mathsf{k}}}(\mathcal{A}) = \Pr[\text{ for all } \mathsf{S}, \text{CDY}[\{\mathsf{n},\mathsf{a}\},\mathsf{S}](\mathcal{A}) \Rightarrow \mathsf{true}\,],$$
$$\mathbf{Adv}_{\Pi}^{\text{CDY}^*_{\mathsf{n}}}(\mathcal{A}) = \Pr[\text{ for all } \mathsf{S}, \text{CDY}[\{\mathsf{k},\mathsf{a}\},\mathsf{S}](\mathcal{A}) \Rightarrow \mathsf{true}\,].$$

Note that the context selector can only select *valid* ciphertexts, which sidesteps issues with formatting. Without this constraint, a context selector could select a ciphertext that has invalid padding for a scheme that requires valid padding, thereby making the notion trivial (all schemes achieve it.)

Furthermore, specific variants like $\text{CDY}^*_{\mathsf{a}}$ may be trivial even with this constraint. For instance, if the ciphertext embeds the nonce, then one can pick some key K, some ciphertext C embedding some nonce N_1, some other nonce N_2, then no $\text{CDY}^*_{\mathsf{a}}$ adversary can pick associated data A such that C decrypts correctly under (K, N_2, A). However, in the context of this restricted CDY notion, we think this is desired behavior and delegate capturing nuances like this to the unrestricted CDY notion (which can capture this by restricting to context selectors which ensure that the nonce embedded is the same as the nonce provided.)

With context compression, CMT-3 implies restricted CDY. A $\text{CDY}[\Sigma]$ attack does not always imply a $\text{CMT}[\Sigma]$ attack. Consider, for example, the "identity" AEAD that has $\mathsf{Enc}(K, N, A, M) \Rightarrow K \parallel N \parallel A \parallel M$ which has an immediate $\text{CDY}[\Sigma]$ attack but is $\text{CMT}[\Sigma]$ secure since a ciphertext can only be decrypted under one context.[3] However, continuing with the hash function analogy, we wonder if a "compression" assumption could make this implication hold. In Theorem 1 we show this statement for $\text{CDY}^*[\mathsf{ts} = \varnothing]$ and CMT-3. And note that this generalizes to $\text{CDY}^*[\mathsf{ts}]$ for any ts with an appropriate compression assumption. Notably, it holds for $\text{CDY}^*_{\mathsf{a}}$ if we assume compression over associated data rather than the full context.

Theorem 1. *Fix some AEAD* Π. *Then for any adversary* $\mathcal{A}$ *that wins the* $\text{CDY}^*[\mathsf{ts} = \varnothing]$ *game, we can give an adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}_{\Pi}^{\text{CDY}^*[\mathsf{ts}=\varnothing]}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}_{\Pi}^{\text{CMT-3}}(\mathcal{B}) + \text{ProbBadCtx}_{\Pi}\,, \tag{2}$$

where ProbBadCtx_{Π} *is the probability that a random decryption context, when used for encrypting a random message, is the only valid decryption context for the resulting ciphertext.*

Proof. This proof is adapted from Bellare and Rogaway [10, p.147], where they prove a similar theorem for hash functions. We construct an adversary $\mathcal{B}$ that randomly samples a context (K_1, N_1, A_1), encrypts a random message to get a ciphertext C, then asks the CDY adversary $\mathcal{A}$ to produce a decryption context for C to get (K_2, N_2, A_2). This ciphertext generation can be viewed as a valid CDY context selector S so $\mathcal{B}$ wins if the returned context is different from the one it sampled; i.e., $(K_1, N_1, A_1) \neq (K_2, N_2, A_2)$. The pseudocode for $\mathcal{B}$ and S is given in Figure 6 and the success probability is analyzed below.

Per the above discussion the advantage of $\mathcal{B}$ is

$$\mathbf{Adv}_{\Pi}^{\text{CMT-3}}(\mathcal{B}) = \Pr[(\mathcal{A}(C) \neq \bot) \wedge (\text{ctx}_1 \neq \text{ctx}_2)]\,, \tag{3}$$

where without loss of generality, we are assuming that $\mathcal{A}$ always produces a valid context or fails and produces $\bot$. But, before simplifying this equation, we need to define some terminology. First, let us define the set of valid decryption contexts for a ciphertext as

$$\Gamma(C) := \{(K, N, A) : (\Pi.\mathsf{Dec}(K, N, A, C) \neq \bot)\}\,.$$

[3] While the "identity" AEAD is not secure in the sense of privacy [37, §3], one can construct a secure counterexample by using a wide pseudorandom permutation [8].

```
B:
K_1 ←$ K; N_1 ←$ N; A_1 ←$ A
M_1 ←$ M
ctx_1 ← (K_1, N_1, A_1)
C ← Π.Enc(K_1, N_1, A_1, M_1)
ctx_2 ←$ A(C)
If ctx_2 = ⊥:
    Return ⊥
(K_1, N_2, A_2) ← ctx_2
If (K_1, N_1, A_1) = (K_2, N_2, A_2)
    Return ⊥
Return (C, (K_1, N_1, A_1), (K_2, N_2, A_2))
```

```
S:
K_1 ←$ K; N_1 ←$ N; A_1 ←$ A
M_1 ←$ M
C ← Π.Enc(K_1, N_1, A_1, M_1)
Return C
```

Figure 6: Pseudocode for the CMT-3 adversary $\mathcal{B}$ and CDY* context selector S, used in proof of Theorem 1.

Now, for a given message M, let us also define the set of "bad" decryption contexts which when used for encrypting M, remain the only valid decryption context for the resulting ciphertext

$$\text{BadCtxs}(M) := \{(K, N, A) : |\Gamma(\Pi.\mathsf{Enc}(K, N, A, M))| = 1\}.$$

Finally, let us define the probability that a random decryption context is bad

$$\text{ProbBadCtx}_{\Pi} := \Pr[(K, N, A) \in \text{BadCtxs}(M)],$$

over the choice $(K, N, A, M) \leftarrow\!\!\$\, (\mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M})$. Using this notation we can rewrite Equation 3, where the probabilities are over the choice $(K, N, A, M) \leftarrow\!\!\$\, (\mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M})$, as

$$\begin{aligned}\mathbf{Adv}_{\Pi}^{\text{CMT-3}}(\mathcal{B}) &= \Pr[(\mathcal{A}(C) \neq \perp) \wedge (\text{ctx}_1 \neq \text{ctx}_2)] \\ &\geq \Pr[(\mathcal{A}(C) \neq \perp) \wedge (\text{ctx}_1 \neq \text{ctx}_2) \wedge (\text{ctx}_1 \notin \text{BadCtxs}(M))].\end{aligned}$$

Using conditional probability, we can rewrite this term as

$$\Pr[\text{ctx}_1 \neq \text{ctx}_2 \mid (\mathcal{A}(C) \neq \perp) \wedge (\text{ctx}_1 \notin \text{BadCtxs}(m))] \cdot \Pr[(\mathcal{A}(C) \neq \perp) \wedge (\text{ctx}_1 \notin \text{BadCtxs}(m))].$$

Recall that if $\text{ctx}_1 \notin \text{BadCtxs}(m)$, then the adversary must choose one of at least two valid contexts, each of which are equally likely to be ctx_1 (even conditioned on C). Thus the probably that it picks ctx_1 is at most $1/2$, and so

$$\begin{aligned}\mathbf{Adv}_{\Pi}^{\text{CMT-3}}(\mathcal{B}) &\geq \frac{1}{2} \cdot \Pr[(\mathcal{A}(C) \neq \perp) \wedge (\text{ctx}_1 \notin \text{BadCtxs}(m))] \\ &\geq \frac{1}{2} \cdot (\Pr[\mathcal{A}(C) \neq \perp] - \Pr[\text{ctx}_1 \in \text{BadCtxs}(m)]).\end{aligned}$$

Putting it all together, we get that

$$\mathbf{Adv}_{\Pi}^{\text{CMT-3}}(\mathcal{B}) \geq \frac{1}{2} \cdot \left(\mathbf{Adv}_{\Pi}^{\text{CDY}^*[\mathsf{ts}=\varnothing]}(\mathcal{A}) - \text{ProbBadCtx}_{\Pi}\right),$$

and finally rearranging gives the desired result. □

CMT-3 implies restricted variants of CDY. We now show that if an attack against any of $\text{CDY}^*_{\mathsf{k}}$, $\text{CDY}^*_{\mathsf{n}}$, or $\text{CDY}^*_{\mathsf{a}}$ implies an attack against CMT-3. Theorem 2 shows this for $\text{CDY}^*_{\mathsf{a}}$, but it readily generalizes to $\text{CDY}^*_{\mathsf{k}}$ and $\text{CDY}^*_{\mathsf{n}}$.

$\underline{\mathcal{B}:}$

$K_1 \leftarrow\!\!\$\ \mathcal{K};\ N_1 \leftarrow\!\!\$\ \mathcal{N};\ A_1 \leftarrow\!\!\$\ \mathcal{A}$
$M_1 \leftarrow\!\!\$\ \mathcal{M}$
$C \leftarrow \Pi.\mathsf{Enc}(K_1, N_1, A_1, M_1)$
$K_2 \leftarrow K_1 + 1;\ N_2 \leftarrow N_1 + 1$
$A_2 \leftarrow\!\!\$\ \mathcal{A}(C, K_2, N_2)$
If $A_2 = \perp$
    Return $\perp$
Return $(C, (K_1, N_1, A_1), (K_2, N_2, A_2))$

$\underline{\mathsf{S}:}$

$K_1 \leftarrow\!\!\$\ \mathcal{K};\ N_1 \leftarrow\!\!\$\ \mathcal{N};\ A_1 \leftarrow\!\!\$\ \mathcal{A}$
$M_1 \leftarrow\!\!\$\ \mathcal{M}$
$C \leftarrow \Pi.\mathsf{Enc}(K_1, N_1, A_1, M_1)$
$K_2 \leftarrow K_1 + 1;\ N_2 \leftarrow N_1 + 1$
Return (C, K_2, N_2)

Figure 7: Pseudocode for the CMT-3 adversary $\mathcal{B}$ and $\mathrm{CDY}_{\mathsf{a}}^*$ context selector S, used in proof of Theorem 2.

Theorem 2. *Fix some AEAD* Π *with key space* $|\mathcal{K}| \geq 2$ *and nonce space* $|\mathcal{N}| \geq 2$*. Then for any adversary* $\mathcal{A}$ *that wins the* $\mathrm{CDY}_{\mathsf{a}}^*$ *game, we can give an adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}_{\Pi}^{\mathrm{CDY}_{\mathsf{a}}^*}(\mathcal{A}) = \mathbf{Adv}_{\Pi}^{\mathrm{CMT\text{-}3}}(\mathcal{B}),$$

and the runtime of $\mathcal{B}$ *is that of* $\mathcal{A}$*.*

Proof. We prove this by constructing $\mathcal{B}$ such that it succeeds whenever $\mathcal{A}$ succeeds. The adversary $\mathcal{B}$ randomly samples a context (K_1, N_1, A_1), encrypts a random message to get a ciphertext C, selects some other key K_2 and nonce N_2 and asks the $\mathrm{CDY}_{\mathsf{a}}^*$ adversary $\mathcal{A}$ to produce an associated data A_2 such that (K_2, N_2, A_2) can decrypt C. This ciphertext and partial context construction can be viewed as a valid context selector S. The pseudocode for the adversary $\mathcal{B}$ and the context selector S are given in Figure 7. And, notice that by construction, $\mathcal{B}$ wins whenever $\mathcal{A}$ succeeds. $\square$

This approach of constructing $\mathcal{B}$ readily generalizes to $\mathrm{CDY}_{\mathsf{n}}^*$ and $\mathrm{CDY}_{\mathsf{k}}^*$. Further, notice that the $\mathcal{B}$ constructed in Figure 7 wins $\mathrm{CMT}_{\mathsf{k}}$ and $\mathrm{CMT}_{\mathsf{n}}$; and similar relations hold for adversaries $\mathcal{B}$ constructed from $\mathrm{CDY}_{\mathsf{n}}^*$ and $\mathrm{CDY}_{\mathsf{k}}^*$ adversaries. Corollary 3 captures these implications.

Corollary 3. *Fix some AEAD* Π *with key space* $|\mathcal{K}| \geq 2$*, nonce space* $|\mathcal{N}| \geq 2$*, and associated data space* $|\mathcal{A}| \geq 2$*. Then the following three statements hold. First, for any adversary* $\mathcal{A}_1$ *that wins the* $\mathrm{CDY}_{\mathsf{a}}^*$ *game, we can give an adversary* $\mathcal{B}_1$ *such that*

$$\mathbf{Adv}_{\Pi}^{\mathrm{CDY}_{\mathsf{a}}^*}(\mathcal{A}_1) = \mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{k}}}(\mathcal{B}_1) = \mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{n}}}(\mathcal{B}_1).$$

Second, for any adversary $\mathcal{A}_2$ *that wins the* $\mathrm{CDY}_{\mathsf{n}}^*$ *game, we can give an adversary* $\mathcal{B}_2$ *such that*

$$\mathbf{Adv}_{\Pi}^{\mathrm{CDY}_{\mathsf{n}}^*}(\mathcal{A}_2) = \mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{k}}}(\mathcal{B}_2) = \mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{a}}}(\mathcal{B}_2).$$

Third, for any adversary $\mathcal{A}_3$ *that wins the* $\mathrm{CDY}_{\mathsf{k}}^*$ *game, we can give an adversary* $\mathcal{B}_3$ *such that*

$$\mathbf{Adv}_{\Pi}^{\mathrm{CDY}_{\mathsf{k}}^*}(\mathcal{A}_3) = \mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{n}}}(\mathcal{B}_3) = \mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{a}}}(\mathcal{B}_3).$$

And the runtimes of $\mathcal{B}_1$*,* $\mathcal{B}_2$*, and* $\mathcal{B}_3$ *are that of* $\mathcal{A}_1$*,* $\mathcal{A}_2$*, and* $\mathcal{A}_3$*, respectively.*

4 Context Discovery Attacks against AEAD

We show context discovery attacks on many AEAD schemes which delegate their authenticity to a non-preimage resistant MAC. Specifically, we show $\mathrm{CDY}_{\mathsf{a}}^*$ attacks on EAX [12], SIV [39], CCM [19], GCM [21], and OCB3 [30], and $\mathrm{CDY}_{\mathsf{n}}^*$ attacks on EAX [12] and GCM [21].

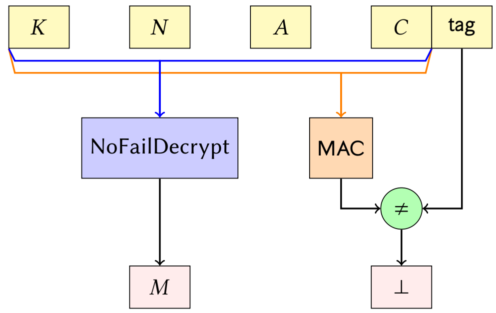


Figure 8: Decryption structure of AEAD schemes which delegate their authenticity to a MAC. Should the MAC tag comparison fail, the routine outputs an error (⊥), otherwise a message is always output by NoFailDecrypt.

We say that an AEAD delegates its authenticity to a MAC if during decryption, a message is output whenever the MAC comparison succeeds. To formalize this, we define NoFailDecrypt as a class of decryption algorithms that never fail. In other words, given a key, nonce, associated data, and ciphertext, they always produce a message. For example, ECB and CTR decryption are NoFailDecrypt algorithms since a valid ciphertext decrypts under any choice of key, nonce, and associated data. On the other hand, CBC with PKCS7 padding is not a NoFailDecrypt algorithm since there most ciphertexts do not decrypt under all choices of key, nonce, and associated data because the decrypted plaintext has incorrect padding.

With this terminology, we say that an AEAD delegates its authenticity to a MAC if it can be written as a combination of a MAC and a NoFailDecrypt algorithm such that if the MAC check fails, decryption fails; if instead the MAC check passes, then decryption outputs the result of NoFailDecrypt (which never fails). This structure is illustrated in Figure 8. As a concrete example, for EAX [12] (described in Figure 9), the MAC corresponds to checking the OMAC tag, and the NoFailDecrypt corresponds to the CTR decryption. In this section, we are particularly interested in schemes that compose this structure with a non-preimage resistant MAC like CMAC [20], GMAC [21, §6.4], or OMAC [12, Fig 1].

The $\mathrm{CDY}^*_{\mathsf{a}}$ attacks we show on these schemes have the following outline. Following the definition of the game, the challenger provides the adversary with a ciphertext $C \| \mathsf{tag}$, a target key K, and a target nonce N, and asks it to find an associated data A such that $\mathrm{Decrypt}(K, N, A, C \| \mathsf{tag}) \neq \perp$. Then, the adversary exploits the lack of preimage resistance to find an associated data A such that $\mathsf{MAC}(K, N, A, C) = \mathsf{tag}$ and returns A. Since, in these schemes, the tag check passing guarantees decryption success, we get that decryption succeeds.

For EAX [12] and GCM [21], we also show $\mathrm{CDY}^*_{\mathsf{n}}$ attacks. They proceed in a similar fashion to the $\mathrm{CDY}^*_{\mathsf{a}}$ attacks but now the adversary finds a nonce N such that $\mathsf{MAC}(K, N, A, C) = \mathsf{tag}$. But, when the nonce length is shorter than a block (which is always true with GCM, and may be true with EAX), the $\mathrm{CDY}^*_{\mathsf{n}}$ attacks are slower than the $\mathrm{CDY}^*_{\mathsf{a}}$ attacks.

The remainder of the section describes the attacks on EAX. The attacks on SIV, CCM, GCM, and OCB3 are in Appendix B.

$\mathrm{CDY}^*_{\mathsf{a}}$ and $\mathrm{CDY}^*_{\mathsf{n}}$ attacks on EAX. We consider EAX over a 128-bit block cipher as defined in Bellare, Rogaway, and Wagner [12]. For simplicity, we restrict to 128-bit tag, 128-bit nonce,[4] and block-aligned messages and associated data. We note however that this is only to make the exposition simpler and is not necessary for the attack. Pseudocode for the scheme with these parameter choices is given in Figure 9.

Let's start by contextualizing the $\mathrm{CDY}^*_{\mathsf{a}}$ game. The challenger provides us with an m-block ciphertext $C = C_1 \cdots C_m \| \mathsf{tag}$, a 128-bit target key K, and a 96-bit target nonce N. And the goal is to find a 1-block

[4]EAX [12, Figure 4] supports an arbitrary length nonce; 128 bits is the default in the popular Tink library [3], see [4].

```
OMAC(K, M):
// Compute Constants
L ← E_K(0^128)
B ← 2 · L
// split into n-bit blocks
// & xor B to the last block
Let M_1, ..., M_m ← M
M_m ← M_m ⊕ B
// CBC-MAC Evaluation
C_0 ← 0^128
For i = 1..m:
    C_i ← E_K(C_{i−1} ⊕ M_i)
Return C_m
```

```
EAX-Decrypt(K, N, A, C):
// Separate the Tag
C ‖ tag ← C
// Compute and Check Tag
𝒩 ← OMAC(K, 0^128 ‖ N)
ℋ ← OMAC(K, 0^127 1 ‖ A)
𝒞 ← OMAC(K, 0^126 10 ‖ C)
If tag ≠ (𝒩 ⊕ ℋ ⊕ 𝒞):
    Return ⊥
// CTR Decryption
r ← |C|/16      // num blocks
For i = 0..(r − 1):
    M_i ← C_i ⊕ E_K(𝒩 + i)
Return M
```

```
𝒜(C, K, N):
C ‖ tag ← C
// Compute ξ
ξ ← tag
ξ ← ξ ⊕ OMAC_K(0^128 ‖ N)
ξ ← ξ ⊕ OMAC_K(0^126 10 ‖ C)
// Reconstruct A and Return
A ← E_K^{-1}(ξ)
A ← A ⊕ E_K(0^127 1) ⊕ (2 · E_K(0^128))
Return (K, N, A)
```

Figure 9: **(Left)** Pseudocode for OMAC [12, Fig 1], used in EAX, with block-aligned inputs. **(Middle)** Pseudocode for EAX Mode [12] decryption with 128-bit tag, 128-bit nonce, and block-aligned messages and associated data. **(Right)** Pseudocode for an $\mathsf{CDY}^*_\mathsf{a}$ attack on EAX.

associated data A such that $\mathsf{EAX\text{-}Decrypt}(K, N, A, C) \neq \perp$. Notice from Figure 9 that decryption passing reduces to the tag check passing. In other words, we can rewrite the goal as finding an associated data A such that

$$\mathsf{tag} = \mathsf{OMAC}_K(0^{128} \parallel N) \oplus \mathsf{OMAC}_K(0^{126}10 \parallel C) \oplus \mathsf{OMAC}_K(0^{127}1 \parallel A)\,. \tag{4}$$

We can rearrange terms to get

$$\mathsf{OMAC}_K(0^{127}1 \parallel A) = \mathsf{tag} \oplus \mathsf{OMAC}_K(0^{128} \parallel N) \oplus \mathsf{OMAC}_K(0^{126}10 \parallel C)\,.$$

Notice that the right-hand side is composed entirely of known terms, thus we can evaluate it to some constant ξ. Using the assumption that A is 1-block, we can expand OMAC_K to get

$$E_K(E_K(0^{127}1) \oplus A \oplus (2 \cdot E_K(0^{128}))) = \xi\,.$$

Decrypting both sides under K, and solving for A gives

$$A = E_K^{-1}(\xi) \oplus E_K(0^{127}1) \oplus (2 \cdot E_K(0^{128}))\,.$$

The full pseudocode for this attack is given in Figure 9.

This attack generalizes to other parameter choices. It works as is against an arbitrary-length message, an arbitrary-length tag, and an arbitrary-length nonce. In addition, this attack can also be adapted as a $\mathsf{CDY}^*_\mathsf{n}$ attack. We start by rewriting Equation 4 as

$$\mathsf{OMAC}_K(0^{128} \parallel N) = \mathsf{tag} \oplus \mathsf{OMAC}_K(0^{126}10 \parallel C) \oplus \mathsf{OMAC}_K(0^{127}1 \parallel C)\,,$$

and solving for N as we did for A above. Since N is 1 block (128 bits), the reduction is similar, and the success probability remains one. If the nonce length was shorter, then assuming an idealized model like the ideal cipher model, the success probability reduces by a multiplicative factor of $2^{-f \cdot 128}$ where f is the fraction of bytes we do not have control over. For example, if we only had control over 14 of the 16 bytes in an encoded block, then the success probability would reduce by 2^{-16}.

This attack can also be adapted to provide partial control over the output plaintext. Notice that the output plaintext is a CTR decryption under the chosen key with the OMAC of the nonce as IV. Assuming an idealized model where the block cipher is an ideal cipher and OMAC is a random function, for every new choice of key and nonce, we get a random output plaintext. So, by trying 2^m key and nonce pairs, we can expect to control m bits of the output plaintext.

$\underline{\text{SIV-1b-Decrypt}(K, C):}$

$\mathsf{c} \leftarrow 1^{n-64}01^{31}01^{31}$

$C_1 \parallel \mathsf{tag} \leftarrow C$

$I \leftarrow \mathsf{tag}$

$K_1 \parallel K_2 \leftarrow K$

// CTR Decryption

$\mathsf{ctr} \leftarrow I \mathbin{\&} \mathsf{c}$

$M \leftarrow C_1 \oplus E_{K_2}(\mathsf{ctr})$

// IV Check

$I' \leftarrow \mathsf{CMAC}^*(K_1, M)$

If $I \neq I'$:

  Return $\perp$

Return M

$\underline{\mathsf{CMAC}^*(K, M):}$

$S \leftarrow \mathsf{CMAC}(K, 0^n)$

Return $\mathsf{CMAC}(K, S \oplus M)$

$\underline{\mathsf{CMAC}(K, X):}$

$K_s \leftarrow 2 \cdot E_K(0^n)$

Return $E_K(K_s \oplus X)$

Figure 10: **(Left)** Pseudocode for SIV Mode [39] decryption with an n-bit message and no associated data. **(Right)** Pseudocode for CMAC* [39] and CMAC [20] with an n-bit input.

5 Restrictive Commitment Attacks via k-Sum Problems

The previous section's CDY^*_a and CDY^*_n attacks against GCM, EAX, OCB3, SIV, and CCM immediately give rise to *permissive* CMT_k attacks against each scheme. This follows from our general result showing that CMT_k security implies CDY^*_a and CDY^*_n (Corollary 3). But this does not imply the ability to build restrictive CMT^*_k, CMT^*_n, or CMT^*_a attacks that require the non-adversarially controlled parts of the two decryption contexts to be identical (see Theorem 11.)

Prior work has provided (in our terminology) CMT^*_k attacks for GCM [26, 18], AES-GCM-SIV [40, 31], ChaCha20/Poly1305 [26, 31], XChaCha20/Poly1305 [31], and OCB3 [2]. An open question of practical interest [41] is whether there is a CMT^*_k attack against SIV. We resolve this open question, showing an attack that works in time about $2^{n/3}$. It requires new techniques related to the fast solution of k-sum problems, as we explain below.

Attack on 1-block SIV. We consider SIV over an n-bit block cipher (for $n \geq 64$) as defined in the draft NIST specification [39]. For ease of exposition, we restrict to the case of an n-bit message and no associated data, and describe how to generalize this to the multi-block case in Appendix D. Pseudocode for the scheme with these parameter choices is given in Figure 10.

Here, the CMT^*_k adversary seeks to produce a ciphertext $C = C_1 \parallel \mathsf{tag}$ and two $2n$-bit keys $K = K_1 \parallel K_2$ and $K' = K'_1 \parallel K'_2$ such that $\mathsf{SIV\text{-}Decrypt}(K, C) \neq \perp$ and $\mathsf{SIV\text{-}Decrypt}(K', C) \neq \perp$. Notice from Figure 10 that this reduces to two simultaneous IV checks passing which can be written as

$$\mathsf{tag} = \mathsf{CMAC}^*(K_1, C_1 \oplus E_{K_2}(\mathsf{tag} \mathbin{\&} \mathsf{c})) = \mathsf{CMAC}^*(K'_1, C_1 \oplus E_{K'_2}(\mathsf{tag} \mathbin{\&} \mathsf{c}))$$

where $\mathsf{c} = 1^{n-64}01^{31}01^{31}$ is a constant specified by the SIV standard. Our attack strategy will be to choose tag arbitrarily, so we can treat this as a constant value. Towards solving for the remaining variable C_1, we can substitute in the definition of CMAC^* to get

$$\begin{aligned}\mathsf{tag} &= E_{K_1}((2 \cdot E_{K_1}(0^n)) \oplus E_{K_1}(2 \cdot E_{K_1}(0^n)) \oplus C_1 \oplus E_{K_2}(\mathsf{tag} \mathbin{\&} \mathsf{c}))\\ &= E_{K'_1}((2 \cdot E_{K'_1}(0^n)) \oplus E_{K'_1}(2 \cdot E_{K'_1}(0^n)) \oplus C_1 \oplus E_{K'_2}(\mathsf{tag} \mathbin{\&} \mathsf{c})),\end{aligned}$$

which we can rearrange the two equalities by solving for the variable C_1, giving us the following:

$$\begin{aligned}C_1 &= E^{-1}_{K_1}(\mathsf{tag}) \oplus (2 \cdot E_{K_1}(0^n)) \oplus E_{K_1}(2 \cdot E_{K_1}(0^n)) \oplus E_{K_2}(\mathsf{tag} \mathbin{\&} \mathsf{c})\\ &= E^{-1}_{K'_1}(\mathsf{tag}) \oplus (2 \cdot E_{K'_1}(0^n)) \oplus E_{K'_1}(2 \cdot E_{K'_1}(0^n)) \oplus E_{K'_2}(\mathsf{tag} \mathbin{\&} \mathsf{c}).\end{aligned} \tag{5}$$

$\underline{\mathcal{A}():}$

$\mathsf{c} \leftarrow 1^{n-64}01^{31}01^{31}$

// Arbitrarily pick a tag

$\mathsf{tag} \leftarrow\!\!\$ \{0,1\}^n \setminus \{0^n\}$

// Define helper functions

Def $F_1(K_1) \leftarrow E^{-1}_{K_1}(\mathsf{tag}) \oplus 2 \cdot E_{K_1}(0^n) \oplus E_{K_1}(2 \cdot E_{K_1}(0^n))$

Def $F_2(K_2) \leftarrow E_{K_2}(\mathsf{tag}\ \&\ \mathsf{c})$

Def $F_3(K_1) \leftarrow E^{-1}_{K'_1}(\mathsf{tag}) \oplus 2 \cdot E_{K'_1}(0^n) \oplus E_{K'_1}(2 \cdot E_{K_1}(0^n))$

Def $F_4(K'_2) \leftarrow E_{K'_2}(\mathsf{tag}\ \&\ \mathsf{c})$

// Generate lists

For $i = 1, ..., q$:

  $x \leftarrow \mathsf{encode}_{128-2}(i)$

  // Domain separate the keys

  $K_1 \leftarrow \mathtt{00} \,\|\, x;\ K_2 \leftarrow \mathtt{01} \,\|\, x;\ K'_1 \leftarrow \mathtt{10} \,\|\, x;\ K'_2 \leftarrow \mathtt{11} \,\|\, x$

  // Query a row

  $L_1[i] \leftarrow F_1(K_1);\ L_2[i] \leftarrow F_2(K_2);\ L_3[i] \leftarrow F_3(K'_1);\ L_4[i] \leftarrow F_4(K'_2)$

// Find an 4-way collision using Wagner's k-tree algorithm [43]

$\mathsf{res} \leftarrow \mathcal{A}.\mathsf{fourWayCollision}(L_1, L_2, L_3, L_4)$

If $\mathsf{res} = \varnothing$:

  Return $\perp$

// Repackage the collision into ciphertext and keys

$(x_1, x_2, x_3, x_4) \leftarrow \mathsf{res}$

$C_1 \leftarrow F_1(x_1) \oplus F_2(x_2)$

$K_1 \leftarrow \mathtt{00} \,\|\, x_1;\ K_2 \leftarrow \mathtt{01} \,\|\, x_2;\ K'_1 \leftarrow \mathtt{10} \,\|\, x_3;\ K'_2 \leftarrow \mathtt{11} \,\|\, x_4$

Return $C_1 \,\|\, \mathsf{tag}, K_1 \,\|\, K_2, K'_1 \,\|\, K'_2$

Figure 11: Pseudocode for $\mathrm{CMT}^*_{\mathsf{k}}$ attack on SIV-1b, where fourWayCollision is defined in Figure 20.

The above implies that it suffices now to find K_1, K_2, K'_1, K'_2 that satisfy Equation 5. To ease notation, we define four helper functions, one for each term:

$$\begin{aligned} F_1(K_1) &:= E^{-1}_{K_1}(\mathsf{tag}) \oplus 2 \cdot E_{K_1}(0^n) \oplus E_{K_1}(2 \cdot E_{K_1}(0^n)), \\ F_2(K_2) &:= E_{K_2}(\mathsf{tag}\ \&\ \mathsf{c}), \\ F_3(K_1) &:= E^{-1}_{K'_1}(\mathsf{tag}) \oplus 2 \cdot E_{K'_1}(0^n) \oplus E_{K'_1}(2 \cdot E_{K_1}(0^n)), \\ F_4(K'_2) &:= E_{K'_2}(\mathsf{tag}\ \&\ \mathsf{c}), \end{aligned}$$

and recast Equation 5 as a 4-sum problem

$$F_1(K_1) \oplus F_2(K_2) \oplus F_3(K'_1) \oplus F_4(K'_2) = 0\,.$$

If these were independent random functions, then we could directly apply Wagner's k-tree algorithm [43] for finding a 4-way collision (also referred to as the generalized birthday problem). But even modeling E as an ideal cipher, the functions are neither random nor independent. For example, $F_1(x) = F_3(x)$ always.

Towards resolving this, we first ensure that the keys K_1, K_2, K'_1, and K'_2 are domain separated. This can be easily arranged: see Figure 11 for the pseudocode of our $\mathrm{CMT}^*_{\mathsf{k}}$ adversary $\mathcal{A}$ against SIV. We now turn to lower bounding $\mathcal{A}$'s advantage, which consists of two primary steps.

The first is that we argue that, in $\mathrm{CMT}^*_{\mathsf{k}}$ when running our adversary against SIV, the helper-function outputs are statistically close to uniform. Then, we show that Wagner's approach works for such values.

We observe that F_2 and F_4 trivially behave as independent random functions in the ideal cipher model for E. The analysis for F_1 and F_3 is more involved. We use the following lemma, which bounds the distinguishing advantage between a uniform n-bit string and the output of a query to either F_1 or F_3.

Lemma 4. *Let* $\mathsf{tag} \in \{0,1\}^n \setminus \{0^n\}$ *and* σ *be an n-bit random permutation with inverse* σ^{-1} *and* U *be the uniform random variable over n bit strings. Define n-bit random variables (over the choice of* σ*)*

$$A := \sigma^{-1}(\mathsf{tag}), \qquad B := 2 \cdot \sigma(0^n), \qquad C := \sigma(2 \cdot \sigma(0^n)),$$

where $\cdot$ *denotes multiplication in* $\mathrm{GF}(2^n)$*. Then no adversary that makes one query to a procedure* P *can distinguish between* $P \mapsto (U,U,U)$ *and* $P \mapsto (A,B,C)$ *with probability greater than* $6 \cdot 2^{-n}$*.*

The proof proceeds by constructing identical-until-bad games and applying the *fundamental lemma of game playing* [11] to discern the distinguishing advantage. The proof appears in Appendix C.

We combine this with the following technical statement about applying Wagner's k-tree algorithm [43] to almost-random lists.

Theorem 5. *Let* L *be a list of* ℓ *4-tuples* $x = (x_1, x_2, x_3, x_4)$*, where each entry* x *is distinguishable from an 4-tuple of independent uniformly random values with probability at most* ξ*. Let* L_1*,* L_2*,* L_3*, and* L_4 *be lists of 1-index* (x_1)*, 2-index* (x_2)*, 3-index* (x_3)*, and 4-index* (x_4) *elements of* L *respectively. Then Wagner's k-tree algorithm [43] finds a solution* $(y_1, y_2, y_3, y_4) \in L_1 \times L_2 \times L_3 \times L_4$ *such that*

$$y_1 \oplus y_2 \oplus y_3 \oplus y_4 = 0,$$

with probability at least

$$(1 - \ell \cdot \xi)\left(1 - \exp\left(-\frac{\ell^2 \cdot 2^{-n/3}}{8}\right)\right)\left(1 - \exp\left(1 - \frac{\ell^4 \cdot 2^{-4n/3}}{8} - \frac{2}{\ell^4 \cdot 2^{-4n/3}}\right)\right),$$

and time at most

$$20\ell + 4\ell^2 \cdot 2^{-n/3} + 4\mathsf{Sort}(\ell) + 2\mathsf{Sort}((1/2)\ell^2 \cdot 2^{-n/3}),$$

where $\mathsf{Sort}(k)$ *denotes the time to sort a list of* k *items.*

The proof proceeds by analyzing the algorithm step-by-step and at each step applying Chernoff bounds [25] to compute a lower bound on the success probability. The proof appears in Appendix C.

With Lemma 4 and Theorem 5, we can now prove a lower bound on the advantage of the $\mathrm{CMT}_{\mathsf{k}}^*$ adversary in Figure 11.

Theorem 6. *Let* $\mathcal{A}$ *be the* $\mathrm{CMT}_{\mathsf{k}}^*$ *adversary against SIV over an n-bit ideal cipher* E*, detailed in Figure 11. It makes* $10q$ *queries to* E *and takes at most*

$$35q + 4q^2 \cdot 2^{-n/3} + 4\mathsf{Sort}(q) + 2\mathsf{Sort}((1/2)q^2 \cdot 2^{-n/3}) + 11,$$

time, where $\mathsf{Sort}(k)$ *is the cost of sorting a list of* k *items. Then the advantage*

$$\mathbf{Adv}_{\mathrm{SIV}}^{\mathrm{CMT}_{\mathsf{k}}^*}(\mathcal{A}) \geq (1 - 8q \cdot 2^{-n})\left(1 - \exp\left(-\frac{q^2 \cdot 2^{-n/3}}{8}\right)\right)\left(1 - \exp\left(1 - \frac{q^4 \cdot 2^{-4n/3}}{8} - \frac{2}{q^4 \cdot 2^{-4n/3}}\right)\right). \quad (6)$$

Proof. By construction, the adversary $\mathcal{A}$ (Figure 11) wins whenever it finds a collision, so it suffices to lower bound this probability. First, the domain separation over the keys ensures that the two helper functions never query the ideal cipher with the same key. This, by the properties of the ideal cipher, ensures independence of the outputs. Second, F_2 and F_4 call the ideal cipher only once on a fixed output under a new key each invocation, so their outputs are indistinguishable from an n-bit uniform random value. Third, F_1 and F_3 call the ideal cipher three times under the same key each invocation. However, applying Lemma 4 gives us that their outputs are distinguishable from an n-bit uniform random value with probability at most

$6 \cdot 2^{-n}$. So, by the union bound, a row of outputs $(F_1(K_1), F_2(K_2), F_3(K_1'), F_4(K_2'))$ is distinguishable from four independent, uniformly random outputs with probability at most $8 \cdot 2^{-n}$. Then, Theorem 5 tells us that the function fourWayCollision called by $\mathcal{A}$ finds a collision with probability at least that of Equation 6.

It remains to analyze the cost of the adversary $\mathcal{A}$. First, it costs 2 operations to initialize c and tag. Second, since each loop iteration costs 15 operations, the loop costs $15q$ operations. Third, from Theorem 5, finding a 4-way collision on four lists of size q using Wagner's k-tree algorithm [43] costs at most

$$20q + 4q^2 \cdot 2^{-n/3} + 4\mathsf{Sort}(q) + 2\mathsf{Sort}((1/2)q^2 \cdot 2^{-n/3})$$

operations. Fourth, repackaging the collision and returning costs 9 operations. So, the runtime is at most

$$35q + 4q^2 \cdot 2^{-n/3} + 4\mathsf{Sort}(q) + 2\mathsf{Sort}((1/2)q^2 \cdot 2^{-n/3}) + 11\,.$$

Finally, since each loop iteration makes 10 ideal cipher queries, the algorithm makes $10q$ queries. ☐

In the following corollary, we show that when the adversary makes approximately $2^{n/3}$ queries, it can win $\mathrm{CMT}_{\mathsf{k}}^*$ against SIV with high probability, taking time approximately $2^{n/3}$.

Corollary 7. *Let $\mathcal{A}$ be the $\mathrm{CMT}_{\mathsf{k}}^*$ adversary against SIV over an n-bit ideal cipher E, detailed in Figure 11 with $q = 10 \cdot 2^{n/3}$. It makes $100 \cdot 2^{n/3}$ queries to E and takes at most*

$$750 \cdot 2^{n/3} + 4\mathsf{Sort}(10 \cdot 2^{n/3}) + 2\mathsf{Sort}(50 \cdot 2^{n/3}) + 11\,,$$

time, where $\mathsf{Sort}(n)$ is the cost of sorting a list of n items. Then

$$\mathbf{Adv}_{\mathrm{SIV}}^{\mathrm{CMT}_{\mathsf{k}}^*}(\mathcal{A}) \geq \left(1 - 80 \cdot 2^{-2n/3}\right)\left(1 - \exp\left(-12.5 \cdot 2^{n/3}\right)\right)\left(1 - \exp\left(-1249\right)\right).$$

6 Related Work

Key commitment for authenticated encryption was introduced in Farshim, Orlandi, and Rosie [24] through full robustness (FROB), which in turn was inspired by key robustness notions in the public key setting by Abdalla, Bellare, and Neven [1] and refined by Farshim et al. [23]. The FROB game asks that a ciphertext only be able to decrypt under a single key. However, the FROB game was defined for randomized authenticated encryption. Grubbs, Lu, and Ristenpart [26] adapted the FROB game to work with associated data, where they ask that a ciphertext only be able to decrypt under a single key (with no constraints on the associated data.) This notion was further generalized by Bellare and Hoang [5] to the nonce-based setting, with their committing security 1 (CMT-1) definition. The CMT-1 game asks that a ciphertext only be able to decrypt under a single key (with no constraints on the nonce nor the associated data.)

The real-world security implications of key commitment were first highlighted by Dodis et al. [18] where they exploited the lack of key commitment when encrypting attachments in Facebook Messenger's message franking protocol [22] to send abusive images that cannot be reported. Albertini et al. [2] generalized this attack from images to other file formats and called attention to more settings where lack of key commitment can be exploited to defeat integrity. While both these attacks targeted integrity, Len, Grubbs, and Ristenpart [31] introduced partitioning oracle attacks and showed how to use them for password guessing attacks by exploiting lack of key commitment to obtain large speedups over standard dictionary attacks, endangering confidentiality.

Proposals for constructing key committing ciphers also started in the Farshim, Orlandi, and Rosie paper [24] where they showed that single-key Encrypt-then-MAC, Encrypt-and-MAC, and MAC-then-Encrypt constructions produce key committing ciphers, when the MAC is collision-resistant. Grubbs, Lu, and Ristenpart [26] showed that the Encode-then-Encipher construction [9] was key committing. Dodis et al. [18] proposed a faster compression function-based key committing AEAD construction termed *encryptment*, and also discussed the closely related Duplex construction [13], which is also key committing. Albertini et al. [2] formally analyzed the folklore padding zeroes and key hashing transforms and showed that they produce key committing AEAD at a lower performance cost than prior constructions. Bellare and Hoang [5] constructed key committing variants of GCM and GCM-SIV termed CAU-C1 and CAU-SIV-C1, and generic transforms UtC and RtC that can be used to turn unique-nonce secure and nonce-reuse secure AEAD schemes respectively into key committing AEAD schemes.

The potential risk of delegating authenticity of an AEAD entirely to a non-collision-resistant MAC is folklore. Farshim, Orlandi, and Rosie [24] who introduced the notion of committing AEAD also cautioned against using non-collision-resistant MACs and CBC-MAC in particular.

On February 7, 2023, NIST announced the selection of the Ascon family for lightweight cryptography standardization [42]. The finalist version of Ascon [17] specifies two AEAD parameter sets Ascon-128 and Ascon-128a. Both parameter sets specify a 128 bit tag, which by the birthday bound, upper bounds the committing security at 64 bits. But, since the underlying algorithm is a variant of the Duplex construction with a 320-bit permutation, and the same specification specifies parameters for a hash function with 128-bit collision resistance, one can specify an AEAD with 128-bit committing security by tweaking parameters.

The Wagner paper [43] introducing the k-tree algorithm for the generalized birthday problem also specified many applications to cryptanalysis including subexponential attacks on Schnorr and Okamoto-Schnorr blind signatures over elliptic curve groups. Minder and Sinclair [34] generalized and formally analyzed the k-tree algorithm. More recently, Lyubashevsky [33] and Liu and Yu [32] have adapted the k-tree algorithm to give subexponential algorithms for variants of the Learning Parity with Noise problem.

Concurrent work. In independent and concurrent work made public very recently, Chan and Rogaway [15] introduced a new definitional framework for committing AE. Their goal is to capture multiple different types of commitment attacks—what they call *misattributions*, or an adversary being able to construct distinct pairs (K, N, A, M) and (K', N', A', M') that both "explain" a single ciphertext C—in a unified way. Their main definition only captures commitment to an entire (K, N, A, M) tuple; but in [15, Appendix A], they briefly describe an extension to only require commitments to a subset of the values.

The extended version of their framework is similar to our CMT[Σ] definition. While both frameworks aim to capture granular win conditions beyond CMT-3, they are orthogonal. Their framework models the multi-key setting with many randomly chosen unknown-to-the-adversary, known-to-the-adversary, and chosen-by-the-adversary keys. While our CMT[Σ] captures the distinction between *permissive* and *restrictive* notions, and settings that impose restrictions on the nonce and associated data. We also introduce the notion of context discoverability and describe its relation to CMT[Σ].

Chan and Rogaway [15] also independently observed that AEAD with non-preimage resistant MACs are vulnerable to commitment attacks and show attacks on GCM and OCB3 similar to the ones we give in Section 4.

Acknowledgments

We thank the anonymous reviewers of Eurocrypt 2023 for their feedback. Sanketh thanks Giacomo Pope for helpful discussions. This work was supported in part by NSF grant CNS #2120651, and the NSF Graduate Research Fellowship under Grant No. DGE-2139899.

References

[1] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland,*

February 9-11, 2010. Proceedings, volume 5978 of *Lecture Notes in Computer Science*, pages 480–497. Springer, 2010. doi:10.1007/978-3-642-11799-2_28.

[2] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. USENIX Security 2022, 2022. URL: https://ia.cr/2020/1456.

[3] Moreno Ambrosin et al. Tink, 2021. URL: https://github.com/google/tink/releases/tag/v1.6.1.

[4] Moreno Ambrosin et al. Tink EAX key manager, 2021. URL: https://github.com/google/tink/blob/v1.6.1/java_src/src/main/java/com/google/crypto/tink/aead/AesEaxKeyManager.java#L115-L116.

[5] Mihir Bellare and Viet Tung Hoang. Efficient schemes for committing authenticated encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 845–875. Springer, 2022. doi:10.1007/978-3-031-07085-3_29.

[6] Mihir Bellare and Tadayoshi Kohno. Hash function balance and its impact on birthday attacks. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 401–418. Springer, 2004. doi:10.1007/978-3-540-24676-3_24.

[7] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptol.*, 21(4):469–491, 2008. doi:10.1007/s00145-008-9026-x.

[8] Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers. Format-preserving encryption. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2009. doi:10.1007/978-3-642-05445-7_19.

[9] Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330. Springer, 2000. URL: https://cseweb.ucsd.edu/~mihir/papers/ee.pdf, doi:10.1007/3-540-44448-3_24.

[10] Mihir Bellare and Phillip Rogaway. *Introduction to Modern Cryptography*. 2005. URL: https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf.

[11] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006. doi:10.1007/11761679_25.

[12] Mihir Bellare, Phillip Rogaway, and David A. Wagner. The EAX mode of operation. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 2004. URL: https://web.cs.ucdavis.edu/~rogaway/papers/eax.pdf, doi:10.1007/978-3-540-25937-4_25.

[13] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011. doi:10.1007/978-3-642-28496-0_19.

[14] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2002. doi:10.1007/3-540-45708-9_21.

[15] John Chan and Phillip Rogaway. On committing authenticated-encryption. In *European Symposium on Research in Computer Security*, pages 275–294. Springer, 2022. URL: https://ia.cr/2022/1260.

[16] Frank Denis et al. AEAD constructions, Robustness, 2022. URL: https://doc.libsodium.org/secret-key_cryptography/aead#robustness.

[17] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Submission to nist, may 2021. URL: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/ascon-spec-final.pdf.

[18] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 155–186. Springer, 2018. doi:10.1007/978-3-319-96884-1_6.

[19] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. NIST Special Publication 800-38C, 2004. doi:10.6028/NIST.SP.800-38C.

[20] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST Special Publication 800-38B, 2005. doi:10.6028/NIST.SP.800-38B.

[21] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, 2017. doi:10.6028/NIST.SP.800-38D.

[22] Facebook. Messenger secret conversations: Technical whitepaper, 2017. URL: https://about.fb.com/wp-content/uploads/2016/07/messenger-secret-conversations-technical-whitepaper.pdf.

[23] Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Robust encryption, revisited. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*, pages 352–368. Springer, 2013. doi:10.1007/978-3-642-36362-7_22.

[24] Pooya Farshim, Claudio Orlandi, and Razvan Rosie. Security of symmetric primitives under incorrect usage of keys. *IACR Trans. Symmetric Cryptol.*, 2017(1):449–473, 2017. doi:10.13154/tosc.v2017.i1.449-473.

[25] Michel Goemans. Chernoff bounds, and some applications, 2015. URL: https://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf.

[26] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 66–97. Springer, 2017. doi:10.1007/978-3-319-63697-9_3.

[27] Dan Harkins. Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES). Request for Comments - Informational, 2008. URL: https://datatracker.ietf.org/doc/rfc5297/.

[28] Joe Kilian and Phillip Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *J. Cryptol.*, 14(1):17–35, 2001. doi:10.1007/s001450010015.

[29] Hugo Krawczyk. The OPAQUE Asymmetric PAKE Protocol. Technical report, October 2019. URL: https://datatracker.ietf.org/doc/draft-krawczyk-cfrg-opaque/03/.

[30] Ted Krovetz and Phillip Rogaway. The OCB Authenticated-Encryption Algorithm. Request for Comments - Informational, 2014. URL: https://datatracker.ietf.org/doc/rfc7253/.

[31] Julia Len, Paul Grubbs, and Thomas Ristenpart. Partitioning oracle attacks. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 195–212. USENIX Association, 2021. URL: https://ia.cr/2020/1491.

[32] Hanlin Liu and Yu Yu. A non-heuristic approach to time-space tradeoffs and optimizations for BKW. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III*, volume 13793 of *Lecture Notes in Computer Science*, pages 741–770. Springer, 2022. doi:10.1007/978-3-031-22969-5_25.

[33] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2005. doi:10.1007/11538462_32.

[34] Lorenz Minder and Alistair Sinclair. The extended k-tree algorithm. *J. Cryptol.*, 25(2):349–382, 2012. doi:10.1007/s00145-011-9097-y.

[35] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 257–274. Springer, 2014. doi:10.1007/978-3-642-55220-5_15.

[36] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of indifferentiability and universal composability. *IACR Cryptol. ePrint Arch.*, page 339, 2011. URL: http://ia.cr/2011/339.

[37] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 98–107. ACM, 2002. doi:10.1145/586110.586125.

[38] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. 4004:373–390, 2006. doi:10.1007/11761679_23.

[39] Phillip Rogaway and Thomas Shrimpton. The SIV Mode of Operation for Deterministic Authenticated-Encryption (Key Wrap) and Misuse-Resistant Nonce-Based Authenticated-Encryption, 2007. Draft 0.32. URL: https://web.cs.ucdavis.edu/~rogaway/papers/siv.pdf.

[40] Sophie Schmieg. Invisible Salamanders in AES-GCM-SIV, 2020. URL: https://keymaterial.net/2020/09/07/invisible-salamanders-in-aes-gcm-siv/.

[41] Sophie, indistinguishable from random noise (@SchmiegSophie), 2020. via Twitter. URL: https://web.archive.org/web/20200909134511/https://twitter.com/SchmiegSophie/status/1303690812933382148.

[42] NIST Lightweight Cryptography Team. NIST announces the selection of the Ascon family for lightweight cryptography standardization, feb 2023. URL: https://www.nist.gov/news-events/news/2023/02/lightweight-cryptography-standardization-process-nist-selects-ascon.

[43] David A. Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002. URL: https://people.eecs.berkeley.edu/~daw/papers/genbday.html.

[44] Michael J. Wiener. The full cost of cryptanalytic attacks. *J. Cryptol.*, 17(2):105–124, 2004. doi:10.1007/s00145-003-0213-5.

A1-Encrypt(K, N, A, M):	A1-Decrypt(K, N, A, C):	$\mathcal{A}$:

$\tau_1 \leftarrow K$
$\tau_2 \leftarrow N \oplus A$
$C_{\mathsf{inner}} \leftarrow M$
Return $\tau_1 \parallel \tau_2 \parallel C_{\mathsf{inner}}$
	$\tau_1 \parallel \tau_2 \parallel C_{\mathsf{inner}} \leftarrow C$
	If $\tau_1 \neq K$:
	    Return $\perp$
	If $\tau_2 \neq (N \oplus A)$:
	    Return $\perp$
	Return C_{inner}
		$(K_1, N_1, A_1) \leftarrow (0^n, 1^n, 0^n)$
		$(K_2, N_2, A_2) \leftarrow (0^n, 0^n, 1^n)$
		$M \leftarrow 0^n$
		$C \leftarrow$ A1-Encrypt(K_1, N_1, A_1, M)
		Return $(C, (K_1, N_1, A_1), (K_2, N_2, A_2))$

Figure 12: Pseudocode for A1 encryption and decryption, used in proof of Theorem 10.

A Selected Relations Between Granular Notions

In this section, we give proofs for selected relations between our granular commitment notions, introduced in Section 3.

Permissive notions and CMT-3. Lemmas 8 and 9 show that CMT-3 implies the permissive notions, and that satisfying all the permissive notions implies CMT-3.

Lemma 8. *Fix some AEAD* Π. *Then for adversaries* $\mathcal{A}_1$, $\mathcal{A}_2$, *and* $\mathcal{A}_3$ *for* $\mathrm{CMT}_{\mathsf{k}}$, $\mathrm{CMT}_{\mathsf{n}}$, *and* $\mathrm{CMT}_{\mathsf{a}}$ *respectively, we can construct adversaries* $\mathcal{B}_1$, $\mathcal{B}_2$, *and* $\mathcal{B}_3$ *for* CMT-3 *such that* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{k}}}(\mathcal{A}_1) \leq \mathbf{Adv}_{\Pi}^{\mathrm{CMT\text{-}3}}(\mathcal{B}_1)$, $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{n}}}(\mathcal{A}_2) \leq \mathbf{Adv}_{\Pi}^{\mathrm{CMT\text{-}3}}(\mathcal{B}_2)$, *and* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{a}}}(\mathcal{A}_3) \leq \mathbf{Adv}_{\Pi}^{\mathrm{CMT\text{-}3}}(\mathcal{B}_3)$. *The runtime of* $\mathcal{B}_1$, $\mathcal{B}_2$, *and* $\mathcal{B}_3$ *is that of* $\mathcal{A}_1$, $\mathcal{A}_2$, *and* $\mathcal{A}_3$, *respectively.*

Proof. The CMT-3 predicate $((K_1 \neq K_2) \vee (N_1 \neq N_2) \vee (A_1 \neq A_2))$ is logically implied by each of the $\mathrm{CMT}_{\mathsf{k}}$, $\mathrm{CMT}_{\mathsf{n}}$, and $\mathrm{CMT}_{\mathsf{a}}$ predicates. Thus, a $\mathrm{CMT}_{\mathsf{k}}$, $\mathrm{CMT}_{\mathsf{n}}$, or $\mathrm{CMT}_{\mathsf{a}}$ adversary succeeds at the CMT-3 games with at least as much probability as the $\mathrm{CMT}_{\mathsf{k}}$, $\mathrm{CMT}_{\mathsf{n}}$, or $\mathrm{CMT}_{\mathsf{a}}$ game respectively. So, statement holds by setting $\mathcal{B}_1 = \mathcal{A}_1$, $\mathcal{B}_2 = \mathcal{A}_2$, and $\mathcal{B}_3 = \mathcal{A}_3$. ☐

Lemma 9. *Fix some AEAD* Π. *Let* ξ_1, ξ_2, *and* ξ_3 *be constants such that for all adversaries* $\mathcal{A}_1$, $\mathcal{A}_2$, *and* $\mathcal{A}_3$ *for* $\mathrm{CMT}_{\mathsf{k}}$, $\mathrm{CMT}_{\mathsf{n}}$, *and* $\mathrm{CMT}_{\mathsf{a}}$ *respectively, it holds that* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{k}}}(\mathcal{A}_1) \leq \xi_1$, $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{n}}}(\mathcal{A}_2) \leq \xi_2$, *and* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{a}}}(\mathcal{A}_3) \leq \xi_3$. *Then for all adversaries* $\mathcal{B}$ *for* CMT-3, *it holds that* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT\text{-}3}}(\mathcal{B}) \leq 3 \cdot \max(\xi_1, \xi_2, \xi_3)$.

Proof. Suppose towards contradiction that there exists an adversary $\mathcal{B}$ for CMT-3 with $\mathbf{Adv}_{\Pi}^{\mathrm{CMT\text{-}3}}(\mathcal{B}) > 3 \cdot \max(\xi_1, \xi_2, \xi_3)$. Then, on success, $\mathcal{B}$ produces a ciphertext and two contexts that satisfy the CMT-3 predicate $((K_1 \neq K_2) \vee (N_1 \neq N_2) \vee (A_1 \neq A_2))$. So, we can find and fix an inequality $(K_1 \neq K_2)$, $(N_1 \neq N_2)$, or $(A_1 \neq A_2)$ that holds in at least one-third of the successes. If it is $(K_1 \neq K_2)$, then $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{k}}}(\mathcal{B}) \geq \max(\xi_1, \xi_2, \xi_3)$, violating our assumptions. Similarly, if it was $(N_1 \neq N_2)$ or $(A_1 \neq A_2)$, we get a violation of our assumptions. ☐

Permissive notions are orthogonal. Theorem 10 shows that $\mathrm{CMT}_{\mathsf{k}}$, $\mathrm{CMT}_{\mathsf{n}}$, and $\mathrm{CMT}_{\mathsf{a}}$ capture orthogonal security goals by constructing an AEAD which is maximally secure under one but maximally insecure under the remaining two. For ease of exposition, we use a minimal counterexample which is not a secure AEAD (in the sense of privacy [37, §3]), but, the underlying idea can be used to construct a secure counterexample.

Theorem 10. *We define an AEAD* Π *and construct a* $\mathcal{O}(1)$*-time adversary* $\mathcal{A}$ *such that* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{n}}}(\mathcal{A}) = 1$ *and* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{a}}}(\mathcal{A}) = 1$, *and for all adversaries* $\mathcal{B}$, $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{k}}}(\mathcal{B}) = 0$.

Proof. We prove this by constructing an AEAD $A1$, which adapts the identity encryption scheme to include, in the ciphertext, the key and the xor of the nonce and associated data, and checking these values during decryption. Pseudocode for this scheme is given in Figure 12.

A2-Encrypt(K, N, A, M):
$\tau \leftarrow K \oplus N \oplus A$
$C_{\mathsf{inner}} \leftarrow M$
Return $\tau \parallel C_{\mathsf{inner}}$

A2-Decrypt(K, N, A, C):
$\tau \parallel C_{\mathsf{inner}} \leftarrow C$
If $\tau \neq (K \oplus N \oplus A)$:
Return $\perp$
Return C_{inner}

$\mathcal{A}$:
$(K_1, N_1, A_1) \leftarrow (0^n, 1^n, 0^n)$
$(K_2, N_2, A_2) \leftarrow (1^n, 0^n, 0^n)$
$M \leftarrow 0^n$
$C \leftarrow$ A1-Encrypt(K_1, N_1, A_1, M)
Return $(C, (K_1, N_1, A_1), (K_2, N_2, A_2))$

Figure 13: Pseudocode for A2 encryption and decryption, used in proof of Theorem 11

First, we construct the adversary $\mathcal{A}$, as defined in Figure 12, such that it produces two contexts which have the same key and same xor of the nonce and associated data. And since decryption passes whenever this holds, we get that $\mathbf{Adv}_{A11}^{\mathrm{CMT}_{\mathsf{n}}}(\mathcal{A}) = \mathbf{Adv}_{A11}^{\mathrm{CMT}_{\mathsf{a}}}(\mathcal{A}) = 1$.

Now, we argue that any adversary $\mathcal{B}$ for the $\mathrm{CMT}_{\mathsf{k}}$ game over AEAD $A1$ always fails. Recall that the winning condition for the $\mathrm{CMT}_{\mathsf{k}}$ game is to produce a ciphertext C and two contexts (K_1, N_1, A_1) and (K_2, N_2, A_2) such that $K_1 \neq K_2$, and decryption succeeds under both contexts. For $A1$, by construction, the latter condition implies that $K_1 = K_2$, while the former condition implies that $K_1 \neq K_2$, leading to a contradiction. □

Restrictive notions and CMT-3. Theorem 11 shows that, unlike with permissive notions (see Lemma 9), satisfying all restrictive notions does not imply CMT-3. For ease of exposition, we use a minimal counterexample which is not a secure AEAD (in the sense of privacy [37, §3]), but, the underlying idea can be used to construct a secure counterexample.

Theorem 11. *We define an AEAD* Π *and a* $\mathcal{O}(1)$*-time adversary* $\mathcal{A}$ *such that* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{k}}}(\mathcal{A}) = 1$*, and for all adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$*, and* $\mathcal{B}_3$*,* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{k}}^*}(\mathcal{B}_1) = 0$*,* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{n}}^*}(\mathcal{B}_2) = 0$*, and* $\mathbf{Adv}_{\Pi}^{\mathrm{CMT}_{\mathsf{a}}^*}(\mathcal{B}_3) = 0$*.*

Proof. We prove this by constructing an AEAD $A2$, which adapts the identity encryption scheme to include, in the ciphertext, the xor of the key, nonce, and associated data, and check this value during decryption. Pseudocode for this scheme is given in Figure 13.

First, we construct the adversary $\mathcal{A}$, as defined in Figure 12, such that it produces two contexts which have the same xor of the key, nonce, and associated data. And since decryption passes whenever this holds, we get that $\mathbf{Adv}_{A1}^{\mathrm{CMT}_{\mathsf{k}}}(\mathcal{A}) = 1$.

Now, we argue that any adversary $\mathcal{B}_1$ for the $\mathrm{CMT}_{\mathsf{k}}^*$ game over AEAD $A1$ always fails. Recall that the winning condition for the $\mathrm{CMT}_{\mathsf{k}}^*$ game is to produce a ciphertext C and two contexts (K_1, N_1, A_1) and (K_2, N_2, A_2) such that $K_1 \neq K_2$, $(N_1, A_1) = (N_2, A_2)$, and decryption succeeds under both contexts. For $A2$, by construction, the last condition implies that $(K_1 \oplus N_1 \oplus A_1) = (K_2 \oplus N_2 \oplus A_2)$, while the first two conditions imply that $(K_1 \oplus N_1 \oplus A_1) \neq (K_2 \oplus N_2 \oplus A_2)$, leading to a contradiction. The arguments for $\mathcal{B}_2$ and $\mathcal{B}_3$ are similar. □

$\underline{\mathsf{CMAC}(K, X):}$

$K_s \leftarrow 2 \cdot E_K(0^{128})$
Return $E_K(K_s \oplus X)$

$\underline{\mathsf{CMAC}^*(K, A, M):}$

$S \leftarrow \mathsf{CMAC}(K, 0^{128})$
$S \leftarrow (2 \cdot S) \oplus \mathsf{CMAC}(K, A)$
Return $\mathsf{CMAC}_K(S \oplus M)$

$\underline{\mathsf{SIV\text{-}Decrypt}(K, A, C):}$

$C_1 \parallel \mathsf{tag} \leftarrow C$
$I \leftarrow \mathsf{tag}$
$K_1 \parallel K_2 \leftarrow K$
// CTR Decryption
$\mathsf{ctr} \leftarrow I \ \&\ 1^{64}01^{31}01^{31}$
$M \leftarrow C_1 \oplus E_{K_2}(\mathsf{ctr})$
// IV Check
$I' \leftarrow \mathsf{CMAC}^*(K_1, A, M)$
If $I \neq I'$:
    Return $\perp$
Return M

$\underline{\mathcal{A}(C, K):}$

$C_1 \parallel \mathsf{tag} \leftarrow C$
// Compute ξ
$K_s \leftarrow 2 \cdot E_K(0^{128})$
$\xi \leftarrow M \oplus (2 \cdot \mathsf{CMAC}_{K_1}(0^{128}))$
$\xi \leftarrow \xi \oplus E_{K_1}^{-1}(\mathsf{tag})$
$\xi \leftarrow \xi \oplus (2 \cdot E_{K_1}(0^{128}))$
// Reconstruct A and Return
$A \leftarrow E_{K_1}^{-1}(\xi) \oplus (2 \cdot E_{K_1}(0^{128}))$
Return (K, A)

Figure 14: **(Left/Top)** Pseudocode for CMAC [20] with 128-bit inputs. **(Left/Bottom)** Pseudocode for CMAC* [39] with a 128-bit message and a 128-bit associated data. **(Middle)** Pseudocode for SIV Mode [39] decryption with a 128-bit message and a 128-bit associated data. **(Right)** Pseudocode for an $\mathrm{CDY}_{\mathsf{a}}^*$ attack on SIV.

B Context Discovery Attacks on More Schemes

Continuing from Section 4, in this section we describe context discovery attacks on SIV [39], CCM [19], GCM [21], and OCB3 [30].

$\mathrm{CDY}_{\mathsf{a}}^*$ attack on SIV. We consider SIV over a 128-bit block cipher (like AES-128) as defined in the draft NIST specification [39]. We restrict to the case of a 128-bit message and 128-bit associated data, and pseudocode for the scheme with these parameter choices is given in Figure 14.

In more detail, we consider the setting where the challenger provides the adversary with a 1-block ciphertext $C = C_1 \parallel \mathsf{tag}$ and a 256-bit target key K. And the goal is to find an 1-block associated data A such that $\mathsf{SIV\text{-}Decrypt}(K, A, C) \neq \perp$. Notice from Figure 14 that decryption passing reduces to the IV check passing. In other words, we can rewrite the goal as finding an associated data A such that

$$\mathsf{tag} = \mathsf{CMAC}_{K_1}(M \oplus \mathsf{CMAC}_{K_1}(A) \oplus (2 \cdot \mathsf{CMAC}_{K_1}(0^{128}))),$$

where $K_1 \parallel K_2 \leftarrow K$ and $M \leftarrow C_1 \oplus E_{K_2}(\mathsf{ctr})$. Expand the outermost CMAC to get

$$\mathsf{tag} = E_{K_1}((M \oplus \mathsf{CMAC}_{K_1}(A) \oplus (2 \cdot \mathsf{CMAC}_{K_1}(0^{128}))) \oplus (2 \cdot E_{K_1}(0^{128}))),$$

Decrypt both sides under K_1 and rearrange to get

$$\mathsf{CMAC}_{K_1}(A) = M \oplus (2 \cdot \mathsf{CMAC}_{K_1}(0^{128})) \oplus E_{K_1}^{-1}(\mathsf{tag}) \oplus (2 \cdot E_{K_1}(0^{128})).$$

Notice that the right-hand side is composed entirely of known terms, thus we can evaluate it to some constant ξ. Since A is 1-block, we can expand the CMAC to get

$$E_{K_1}(A \oplus (2 \cdot E_{K_1}(0^{128}))) = \xi$$

Decrypt both sides under K_1 and solve for A to get

$$A = E_{K_1}^{-1}(\xi) \oplus (2 \cdot E_{K_1}(0^{128})).$$

The full pseudocode for this attack is given in Figure 14.

$\mathrm{CDY}_{\mathsf{a}}^*$ attack on CCM. We consider CCM over a 128-bit block cipher (like AES-128) as defined in NIST SP 800-38C [19], with a nonce size of 12 bytes and tag size of 16 bytes. For ease of exposition, we restrict

```
CCM Constants:
// adapted from §A.1 of [19]
t ← 16
n ← 12
q ← 3     // picked such that q + n = 15
encT ← encode_3((t − 2)/2)
encQ ← encode_3(q − 1)
CtrFlags ← 00000 ‖ encQ
```

```
EncodeMessage(A, N, M):
// adapted from §A.2 of [19]
Adata ← (1 if |A| > 0, 0 otherwise)
a ← (A length in bytes)
p ← (M length in bytes)
mlen ← encode_24(p)
EncFlags ← 0 ‖ Adata ‖ encT ‖ encQ
B_0 ← EncFlags ‖ N ‖ mlen
// Encode ad length, ad, and message
Y ← encode_16(a) ‖ A ‖ M
// split into 16-byte blocks
B_1, ..., B_r ← Y
Return B_0, ..., B_r
```

```
CCM-Decrypt(K, A, N, C):
// CTR decryption
J ← CtrFlags ‖ N ‖ encode_24(0)
tag ← C_0 ⊕ E_K(J)
r ← ⌈|C|/16⌉ − 1     // num msg blocks
For i = 1..r:
    M_i ← C_i ⊕ E_K(J + i)
// CBC-MAC evaluation
B_0, ..., B_r ← EncodeMessage(A, N, M)
Y_0 ← E_K(B_0)
For i = 1..r:
    Y_i ← E_K(Y_{i−1} ⊕ B_i)
If tag ≠ Y_r:
    Return ⊥
Return M
```

```
A(C, K, N):
C_0 ‖ C_1 ← C
// CTR decrypt
J ← CtrFlags ‖ N ‖ encode_24(0)
M ← C_1 ⊕ E_K(J + 1)
// Compute Known Parts of Encode
EncFlags ← 0 ‖ 1 ‖ encT ‖ encQ
mlen ← encode_24(16)
B_0 ← EncFlags ‖ N ‖ mlen
B_1 ← encode_16(30) ‖ 0^{14*8}
B_3 ← M
// Compute B_2
B_2 ← E_K^{−1}(B_3 ⊕ E_K^{−1}(E_K(J) ⊕ C_0))
B_2 ← B_2 ⊕ E_K(E_K(B_0) ⊕ B_1)
// Reconstruct A and Return
A ← 0^{14*8} ‖ B_2
Return (K, N, A)
```

Figure 15: **(Bottom/Left)** Pseudocode for CCM Mode [19] decryption with 12 byte nonce and 16 byte tag, for associated data of at most 2^{16} bytes, and block-aligned messages and associated data of length $14+16\cdot m$ bytes for some $m \geq 0$. **(Bottom/Right)** Pseudocode for an $\mathsf{CDY}^*_{\mathsf{a}}$ attack on CCM.

the message to be 16 bytes, and the associated data to be of length $14 + 16 \cdot m$ bytes for some $m \geq 0$, but we note that this is only for exposition and the attack generalizes. Pseudocode for the scheme with these parameter choices is given in Figure 15.

In more detail, we consider the setting where the challenger provides the adversary with a 1-block ciphertext $C = C_0 \parallel C_1$, a 128-bit target key K, and a 96-bit target nonce. And the goal is to find an associated data A such that $\text{CCM-Decrypt}(K, A, N, C) \neq \perp$. Notice from Figure 15 that decryption passing reduces to the tag check passing which lets us rewrite the goal as finding an associated data A such that

$$C_0 = E_K(J) \oplus Y_r$$

where Y_r is the CBC-MAC of $\text{EncodeMessage}(A, N, M)$ as defined in Figure 15. Notice that for a 30-byte associated data A, 12-byte nonce N, and 16-byte message M, EncodeMessage works as follows. It produces four 16-byte blocks (B_0, B_1, B_2, B_3) where B_0 is flags and the nonce N, $B_1 = \mathsf{encode}_{16}(30) \parallel A[:14]$, $B_2 = A[14:30]$, and $B_3 = m$. Using this, we can expand Y_r to get

$$C_0 = E_K(J) \oplus E_K(E_K(E_K(E_K(B_0) \oplus B_1) \oplus B_2) \oplus B_3)$$

```
GHASH(H, X):
// Split into 16-byte blocks
X_1, ..., X_m ← X
// Compute X_1 · H^m + ··· + X_m · H
Y_0 ← 0^128
For i = 1 to m:
    Y_i ← (Y_{i-1} ⊕ X_1) · H
Return Y_m
```

```
GCM-Decrypt(K, N, A, C):
C ‖ tag ← C
J_0 = N ‖ 0^31 ‖ 1
// GHASH Evaluation
H ← E_K(0^128)
lens ← encode_64(|A|) ‖ encode_64(|C|)
S ← GHASH(H, A ‖ C ‖ lens)
If tag ≠ (S ⊕ E_K(J_0)):
    Return ⊥
// CTR Decryption
clen ← |C|/128
For i ← 1 to clen:
    M[i] ← E_K(J_0 + i) ⊕ C[i]
Return M
```

Figure 16: **(Left)** Pseudocode for GHASH [21, §6.4]. **(Right)** Pseudocode for GCM Mode [21, §7] decryption with a 96-bit nonce, a 128-bit tag, and block-aligned messages and associated data.

Rearranging, decrypting both sides under K, and solving for B_2 we get

$$B_2 = E_K^{-1}(E_K^{-1}(C_0 \oplus E_K(J)) \oplus B_3) \oplus E_K(E_K(B_0) \oplus B_1)$$

Notice that we know all the terms on the right-hand side, so setting B_2 to this value provides the desired tag collision. The full pseudocode for this attack is given in Figure 15.

Finally, we turn to the statement at the onset that this attack generalizes to other parameter choices. First, it generalizes to any block-aligned message, we'd just need to do more arithmetic to solve for B_2. Second, the associated data we choose can be arbitrary except for an aligned 16-byte block. Third, if we only have partial control over an aligned block of associated data, then assuming an idealized model like the ideal cipher model, the success probability reduces by a multiplicative factor of $2^{-f \cdot 128}$ where f is the fraction of bytes we don't have control over.

$\mathbf{CDY}_\mathsf{a}^*$ and $\mathbf{CDY}_\mathsf{n}^*$ Attacks on GCM. We consider GCM over a 128-bit block cipher (like AES-128) as defined in NIST SP 800-38D [21]. For simplicity, we restrict to a 96-bit nonce, a 128-bit tag, and block-aligned messages and associated data. We note however that this is to make the exposition easier, and the attack generalizes to the case without these constraints. Pseudocode for the scheme with these parameter choices is given in Figure 16.

Let's start by contextualizing the $\mathrm{CDY}_\mathsf{a}^*$ game. The challenger provides us with an m-block ciphertext $C = C_1 \cdots C_m \parallel \mathsf{tag}$, a 128-bit target key K, and a 96-bit target nonce N. And the goal is to find an 1-block associated data A such that $\text{GCM-Decrypt}(K, N, A, C) \neq \perp$. Notice from Figure 16 that decryption passing reduces to the tag check passing. In other words, we can rewrite the goal as finding an associated data A such that

$$\mathsf{tag} = \mathsf{GHASH}(H, A \parallel C \parallel \mathsf{lens}) \oplus E_K(J_0),$$

where $H = E_K(0^{128})$, $\mathsf{lens} = \mathsf{encode}_{64}(1) \parallel \mathsf{encode}_{64}(|C|)$, and $J_0 = N \parallel 0^{31} \parallel 1$ are as defined in Figure 16. We can rearrange terms to get

$$\mathsf{GHASH}(H, A \parallel C \parallel \mathsf{lens}) = \mathsf{tag} \oplus E_K(J_0),$$

We can expand the GHASH as a polynomial over GF(2^{128}) [21, §6.4], to get

$$A \cdot H^{m+2} + C_1 \cdot H^{m+1} + \cdots + C_m \cdot H^2 + \mathsf{lens} \cdot H = \mathsf{tag} \oplus E_K(J_0). \tag{7}$$

Since everything except A is fixed, we can solve for A as

$$A = H^{-(m+2)} \left(\mathsf{tag} \oplus E_K(J_0) + C_1 \cdot H^{m+1} + \cdots + C_m \cdot H^2 + \mathsf{lens} \cdot H\right).$$

$\underline{\mathcal{A}(C, K, N)\text{:}}$

$C \parallel \mathsf{tag} \leftarrow C$

// Initialize Constants

$H \leftarrow E_K(0^{128})$

$\mathsf{lens} = \mathsf{encode}_{64}(1) \parallel \mathsf{encode}_{64}(|C|)$

$J_0 = N \parallel 0^{31} \parallel 1$

// Reconstruct A and Return

$A = H^{-(m+2)}\left(\mathsf{tag} \oplus E_K(J_0) + C_1 \cdot H^{m+1} + \cdots + C_m \cdot H^2 + \mathsf{lens} \cdot H\right)$

Return (K, N, A)

Figure 17: Pseudocode for a $\mathsf{CDY}^*_{\mathsf{a}}$ attack on GCM.

The full pseudocode for this attack is given in Figure 17.

Now, we turn to the statement at the onset that this attack generalizes to other parameter choices. First, as-is it works against GCM with a shorter tag (which is just a truncation). Second, it readily generalizes to shorter nonces, which requires us to update the generation J_0 in the pseudocode. Third, it readily generalizes to non-block aligned messages, which requires us to update the construction of lens and add trailing zeroes to the final block. Fourth, it generalizes to the setting with any block-aligned associated data, as long as the attacker controls one block of associated data. If we have partial control over a block, then assuming an idealized model like the ideal cipher model, the success probability reduces by a multiplicative factor of $2^{-f \cdot 128}$ where f is the fraction of bytes we don't have control over. For example, if we only had control over 14 of the 16 bytes in an encoded block, then the success probability would reduce by 2^{-16}.

Finally, this attack can also be adapted as a $\mathsf{CDY}^*_{\mathsf{n}}$ attack. Let's start by rewriting Equation 7 as

$$\mathsf{tag} \oplus E_K(J_0) = \sum_{i=1}^{\ell} A_i \cdot H^{(a-i+1)+m+1} + \sum_{i=1}^{m} C_i \cdot H^{(m-i+1)+1} + \mathsf{lens} \cdot H,$$

for an m-block ciphertext $C = C_1 \cdots C_m \parallel \mathsf{tag}$ and ℓ-block associated data $A = A_1 \cdots A_\ell$; and $H = E_K(0^{128})$, $\mathsf{lens} = \mathsf{encode}_{64}(|A|) \parallel \mathsf{encode}_{64}(|C|)$, and $J_0 = N \parallel 0^{31} \parallel 1$ as defined in Figure 16. Then rearranging we get

$$\begin{aligned} J_0 = E_K^{-1}\big(&\mathsf{tag} + A_1 \cdot H^{\ell+m+1} + \cdots + A_a \cdot H^{m+2} \\ &+ C_1 \cdot H^{m+1} + \cdots + C_m \cdot H^2 + \mathsf{lens} \cdot H\big). \end{aligned} \tag{8}$$

But, recall that $J_0 = N \parallel 0^{31} \parallel 1$, so control over the nonce N only give us control over the first 96 bits, and the remaining 32 are constant. So, this attack doesn't always work. But, in an idealized model like the ideal cipher model, we can lower bound the success at about 2^{-32}.

$\mathsf{CDY}^*_{\mathsf{a}}$ Attack on OCB3. We consider OCB3 over a 128-bit block cipher as defined in IRTF RFC 7253 [30]. For simplicity, we restrict to the variant with a 96-bit nonce, 128-bit tag and block-aligned messages and associated data. Pseudocode for the scheme with these parameter choices is given in Figure 18.

Let's start by contextualizing the $\mathsf{CDY}^*_{\mathsf{a}}$ game. The challenger provides us with an m-block ciphertext $C = C_1 \cdots C_m \parallel \mathsf{tag}$, a 128-bit target key K, and a 96-bit target nonce N. And the goal is to find an 1-block associated data A such that OCB3-Decrypt$(K, N, A, C) \neq \perp$. Notice from Figure 18 that decryption passing reduces to the tag check passing. In other words, we can rewrite the goal as finding an associated data A such that

$$\mathsf{tag} = E_K(\mathsf{Checksum}_m \oplus \Delta_m \oplus L_\$) \oplus \text{OCB3-Hash}(K, A)$$

where $\mathsf{Checksum}_m$, Δ_m, $L_\$$ are defined as in OCB3-Decrypt in Figure 18 using the input (K, N, C). We can rearrange terms to get

$$\text{OCB3-Hash}(K, A) = E_K(\mathsf{Checksum}_m \oplus \Delta_m \oplus L_\$) \oplus \mathsf{tag}.$$

$\underline{\text{OCB3-Setup}(K)\text{:}}$

$L_* \leftarrow E_K(0^{128})$
$L_\$ \leftarrow 2 \cdot E_K(0^{128})$
For $i \geq 0$:
    $L[i] \leftarrow 2^{2+i} \cdot E_K(0^{128})$
def ntz(i):
    Return number of trailing zeroes
    in the binary representation of i
def str2num(s):
    Return number represented by s

$\underline{\text{OCB3-Hash}(K, A)\text{:}}$

$A_1, \ldots, A_m \leftarrow A$
$\mathsf{sum}_0 \leftarrow 0^{128}$
$\Phi_0 \leftarrow 0^{128}$
For $i \leftarrow 1$ to m:
    $\Phi_i \leftarrow \Phi_{i-1} \oplus L[\mathsf{ntz}(i)]$
    $\mathsf{sum}_i \leftarrow \mathsf{sum}_{i-1} \oplus E_K(A_i \oplus \Phi_i)$
Return sum_m

$\underline{\text{OCB3-Decrypt}(K, N, A, C)\text{:}}$

$C_1, \ldots, C_m \,\|\, \mathsf{tag} \leftarrow C$
$L_*, L_\$, L \leftarrow$ OCB3-Setup(K)
// Per-Decryption Constants
$\mathsf{nonce} \leftarrow 0^{31} \,\|\, 1 \,\|\, N$
$\mathsf{bottom} \leftarrow \mathsf{str2num}(\mathsf{nonce}[123..128])$
$\mathsf{Ktop} \leftarrow E_K(\mathsf{nonce}[1..122] \,\|\, 0^6)$
$\mathsf{Stretch} \leftarrow \mathsf{Ktop} \,\|\, (\mathsf{Ktop}[1..64] \oplus \mathsf{Ktop}[9..72])$
$\Delta_0 \leftarrow \mathsf{Stretch}[(1 + \mathsf{bottom})..(128 + \mathsf{bottom})]$
$\mathsf{Checksum}_0 \leftarrow 0^{128}$
// Decryption
For $i \leftarrow 0$ to m:
    $\Delta_i \leftarrow \Delta_{i-1} \oplus L[\mathsf{ntz}(i)]$
    $M_i \leftarrow \Delta_i \oplus E_K(C_i \oplus \Delta_i)$
    $\mathsf{Checksum}_i \leftarrow \mathsf{Checksum}_{i-1} \oplus M_i$
$\mathsf{tag}' \leftarrow E_K(\mathsf{Checksum}_m \oplus \Delta_m \oplus L_\$)$
$\mathsf{tag}' \leftarrow \mathsf{tag}' \oplus$ OCB3-Hash(K, A)
If $\mathsf{tag}' \neq \mathsf{tag}$:
    Return $\perp$
Return M

$\underline{\mathcal{A}(C, K, N)\text{:}}$

$C_1, \ldots, C_m \,\|\, \mathsf{tag} \leftarrow C$
// Setup
$L_* \leftarrow E_K(0^{128})$
$L_\$ \leftarrow 2 \cdot E_K(0^{128})$
For $i \geq 0$:
    $L[i] \leftarrow 2^{2+i} \cdot E_K(0^{128})$
// Per-Decryption Constants
$\mathsf{nonce} \leftarrow 0^{31} \,\|\, 1 \,\|\, N$
$\mathsf{bottom} \leftarrow \mathsf{str2num}(\mathsf{nonce}[123..128])$
$\mathsf{Ktop} \leftarrow E_K(\mathsf{nonce}[1..122] \,\|\, 0^6)$
$\mathsf{Stretch} \leftarrow \mathsf{Ktop} \,\|\, (\mathsf{Ktop}[1..64] \oplus \mathsf{Ktop}[9..72])$
$\Delta_0 \leftarrow \mathsf{Stretch}[(1 + \mathsf{bottom})..(128 + \mathsf{bottom})]$
$\mathsf{Checksum}_0 \leftarrow 0^{128}$
// Compute Checksum and Offsets
For $i \leftarrow 0$ to m:
    $\Delta_i \leftarrow \Delta_{i-1} \oplus L[\mathsf{ntz}(i)]$
    $M_i \leftarrow \Delta_i \oplus E_K(C_i \oplus \Delta_i)$
    $\mathsf{Checksum}_i \leftarrow \mathsf{Checksum}_{i-1} \oplus M_i$
// Reconstruct A and Return
$\xi \leftarrow E_K(\mathsf{Checksum}_m \oplus \Delta_m \oplus L_\$) \oplus \mathsf{tag}$
$A = E_K^{-1}(\xi) \oplus 4 \cdot E_K(0^{128})$
Return (K, N, A)

Figure 18: **(Left/Top)** Setup to generate key-dependent constants and define helper functions [30]. **(Left/Bottom)** Hash for processing associated data [30, §4.1], with block-aligned messages. **(Middle)** Pseudocode for OCB3 mode [30, §4.2-§4.3] decryption, with block-aligned messages and associated data, 128-bit tag and a 96-bit nonce. **(Right)** Pseudocode for a $\mathsf{CDY}^*_\mathsf{a}$ attack on OCB3.

Notice that the right-hand side is composed entirely of known terms, so we can evaluate it to some constant ξ. This allows us to simplify the equation to

$$\text{OCB3-Hash}(K, A) = \xi.$$

Using the assumption that A is 1-block, we can expand OCB3-Hash(K, A) to

$$E_K(A \oplus L[\mathsf{ntz}(1)]) = \xi.$$

Recall from the definition of ntz in Figure 18 that $\mathsf{ntz}(1) - 0$, and that $L[1] = 4 \cdot E_K(0^{128})$. Using this, we can simplify to

$$E_K(A \oplus 4 \cdot E_{K'}(0^{128})) = \xi.$$

Then decrypt both sides

$$A \oplus 4 \cdot E_{K'}(0^{128}) = E_K^{-1}(\xi),$$

and solve for A to get

$$A = E_K^{-1}(\xi) \oplus 4 \cdot E_{K'}(0^{128}).$$

The full pseudocode for this attack is given in Figure 18.

C Four Sum Attacks on Block Cipher Outputs

<u>Procedure P:</u>

$\pi \leftarrow \{\}$ // empty mapping — Game 0
$A \leftarrow\!\!\$\ \{0,1\}$ // $\sigma^{-1}(\mathsf{tag})$ — Game 1
$\pi[A] = \mathsf{tag}$
$B \leftarrow\!\!\$\ \{0,1\}^n$ // $2 \cdot \sigma(0^n)$
If $A = 0^n$:
  $\mathsf{bad}_0 \leftarrow \mathsf{true}$
  $B \leftarrow 2 \cdot \mathsf{tag}$
If $A \neq 0^n$ and $B = 2 \cdot \mathsf{tag}$:
  $\mathsf{bad}_1 \leftarrow \mathsf{true}$
  $B \leftarrow\!\!\$\ \{0,1\}^n \setminus \{2 \cdot \mathsf{tag}\}$
$\pi[0^n] = 2^{-1}B$
$C \leftarrow\!\!\$\ \{0,1\}^n$ // $\sigma(2 \cdot \sigma(0^n)) = \sigma(B)$
If $B = 0^n$:
  $\mathsf{bad}_2 \leftarrow \mathsf{true}$
  $C \leftarrow 2^{-1}B$
If $B = A$:
  $\mathsf{bad}_3 \leftarrow \mathsf{true}$
  $C \leftarrow \mathsf{tag}$
If $B \neq 0^n$ and $C = 2^{-1}B$:
  $\mathsf{bad}_4 \leftarrow \mathsf{true}$
  $C \leftarrow\!\!\$\ \{0,1\}^n \setminus \{2^{-1}B\}$
If $B \neq A$ and $C = \mathsf{tag}$:
  $\mathsf{bad}_5 \leftarrow \mathsf{true}$
  $C \leftarrow\!\!\$\ \{0,1\}^n \setminus \{\mathsf{tag}\}$
$\pi[B] = C$
Return (A, B, C)

Figure 19: Two games corresponding to computing procedure P. Game 0, which models the "ideal" world, does not include the highlighted statements, Game 1, which models the "real" world, includes highlighted statements.

Randomness of three specific block cipher outputs. We start by showing that the three block cipher outputs that arise in the SIV attack in Section 5, in the random cipher model, are indistinguishable from three outputs of a uniform random function.

Lemma 4 (From §5). *Let* $\mathsf{tag} \in \{0,1\}^n \setminus \{0^n\}$ *and* σ *be an n-bit random permutation with inverse* σ^{-1} *and* U *be the uniform random variable over n bit strings. Define n-bit random variables (over the choice of* σ*)*

$$A := \sigma^{-1}(\mathsf{tag}), \qquad B := 2 \cdot \sigma(0^n), \qquad C := \sigma(2 \cdot \sigma(0^n)),$$

where $\cdot$ *denotes multiplication in* $\mathrm{GF}(2^n)$*. Then no adversary that makes one query to a procedure* P *can distinguish between* $P \mapsto (U,U,U)$ *and* $P \mapsto (A,B,C)$ *with probability greater than* $6 \cdot 2^{-n}$.

Proof. We start by constructing two identical-until-bad games Game 0 and Game 1 corresponding to $P \mapsto (U,U,U)$ and $P \mapsto (A,B,C)$ respectively. In Game 0, the output of the procedure P is three independently uniformly random n-bit strings. In Game 1, the output of the procedure P is a sampling of A, B, and C which are parameterized by a random permutation. We emulate this random permutation *lazily* by independently uniformly randomly sampling mappings and setting a bad bit if the sampled mapping is inconsistent with a previously sampled mapping, using the variable π to keep track of previously sampled mappings. The constructed games are shown in Figure 19.

Since these games are identical-until-bad the *fundamental lemma of game playing* [11] gives us that the adversary's distinguishing advantage is upper bounded by the probability that any of the bad bits are set.

The bits bad_0 and bad_2 are set when a uniformly randomly sampled n-bit value (A and B) equals a fixed n-bit value (0^n and 0^n). Similarly, bad_3 is set when a uniformly randomly sampled n-bit value (B) equals a previously independently uniformly randomly sampled n-bit value (A). Hence, the probability of these bits being set is at most 2^{-n} each.

The bits bad_1 and bad_5 can only be set when a uniformly randomly sampled n-bit value (B and C) equals a fixed n-bit value ($2 \cdot \mathsf{tag}$ and 0^n). Similarly, bad_4 can only be set when a uniformly randomly sampled n-bit value (C) equals half of a previously independently uniformly randomly sampled n-bit value (B). Hence, the probability that these bits are set is also at most 2^{-n} each.

Applying the union bound, the probability that any of the bad bits are set is at most $6 \cdot 2^{-n}$. □

Solving the 4-sum problem with almost-random lists. Next, we lower bound the success probability of solving the 4-sum problem with lists consisting of entries which are indistinguishable from random, using Wagner's k-tree algorithm [43]. The proof uses Chernoff bounds which we recall below.

Lemma 12 (Chernoff bounds [25])**.** *Let $X_1, \ldots, X_n$ be independent, 0/1-valued random variables taking 1 with probability p and taking 0 with probability $1-p$. Let the sum $X := \sum_i X_i$ and its expectation $\mu := \mathbb{E}[X] = np$. Then,*

1. *(lower tail)* $\Pr[X \leq (1-\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu}{2}\right)$ *for all* $0 \leq \delta < 1$, *and*
2. *(upper tail)* $\Pr[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu}{2+\delta}\right)$ *for all* $0 \leq \delta$.

Theorem 5 (From §5). *Let L be a list of ℓ 4-tuples $x = (x_1, x_2, x_3, x_4)$, where each entry x is distinguishable from an 4-tuple of independent uniformly random values with probability at most ξ. Let L_1, L_2, L_3, and L_4 be lists of 1-index (x_1), 2-index (x_2), 3-index (x_3), and 4-index (x_4) elements of L respectively. Then Wagner's k-tree algorithm [43] finds a solution $(y_1, y_2, y_3, y_4) \in L_1 \times L_2 \times L_3 \times L_4$ such that*

$$y_1 \oplus y_2 \oplus y_3 \oplus y_4 = 0\,,$$

with probability at least

$$(1-\ell\cdot\xi)\left(1-\exp\left(-\frac{\ell^2\cdot 2^{-n/3}}{8}\right)\right)\left(1-\exp\left(1-\frac{\ell^4\cdot 2^{-4n/3}}{8}-\frac{2}{\ell^4\cdot 2^{-4n/3}}\right)\right),$$

and time at most

$$20\ell + 4\ell^2 \cdot 2^{-n/3} + 4\mathsf{Sort}(\ell) + 2\mathsf{Sort}((1/2)\ell^2 \cdot 2^{-n/3})\,.$$

where $\mathsf{Sort}(k)$ denotes the time to sort a list of k items.

Proof. We start by conditioning on none of the entries x of L being distinguishable from a 4-tuple of independent uniformly random values. Since we are given that each entry is distinguishable with probability at most ξ, the probability that *any* of the ℓ entries are distinguishable is at most $\ell \cdot \xi$, by the union bound. So, this condition that none of the entries are distinguishable holds with probability $(1-\ell\cdot\xi)$.

Wagner's k-tree algorithm (see Figure 20) can only find 4-sum solutions with certain structure. To capture this we are going to define the intermediate lists L_{12} and L_{34} and bound their size, then define colls and bound its size.

Following the pseudocode, define

$$\begin{aligned}L_{12} &:= \{(L_1[i_1] \oplus L_2[i_2], (i_1, i_2)) \\ &\qquad : \mathsf{low}_{n/3}(L_1[i_1] \oplus L_2[i_2]) = 0^{n/3}\ ,\ i_1, i_2 \in \{1, \ldots, \ell\}\}\,,\end{aligned}$$

Since we assumed that each entry of L_1 and L_2 is independently and uniformly sampled, each unique pair $(z_1, z_2) \in L_1 \times L_2$ has an independent $2^{-n/3}$ chance of satisfying $\mathsf{low}_{n/3}(z_1 \oplus z_2) = 0^{n/3}$. Using the Chernoff lower tail bound (Lemma 12) with $\delta = 1/2$ we get that L_{12} has size at least $(1/2)\ell^2 \cdot 2^{-n/3}$ with probability

$$1 - \exp\left(-\frac{\ell^2 \cdot 2^{-n/3}}{8}\right).$$

Using the same argument, we get the same bound for L_{34}.

Next, following the pseudocode, define

$$\mathsf{colls} := \{(L_{12}[i_1] \oplus L_{34}[i_2], (i_1, i_2)) : L_{12}[i_1] \oplus L_{34}[i_2] = 0^n\ ,\ i_1, i_2 \in \{1, \ldots, m\}\}\,,$$

where m is the size of L_{12} and L_{34}. Since we assumed that entries of L_1, L_2, L_3, and L_4 were independently and uniformly sampled, and the $\mathsf{lowBitMerge}$ subroutine did not touch the high $2n/3$ bits, we can view the high $2n/3$ bits of L_{12} and L_{34} as independently and uniformly sampled. Then, each unique pair $(z_{12}, z_{34}) \in L_{12} \times L_{34}$ has an independent $2^{-2n/3}$ chance of satisfying $z_{12} \oplus z_{34} = 0^n$. Using the Chernoff lower tail bound (Lemma 12) with $\delta = 1 - \mu^{-1}$ we get that colls has size at least 2 with probability

$$1 - \exp\left(-\frac{\mu}{2}\left(1 - \frac{1}{\mu}\right)^2\right) = 1 - \exp\left(1 - \frac{\mu}{2} - \frac{1}{2\mu}\right),$$

where $\mu = m^2 \cdot 2^{-2n/3}$. Simplifying and plugging in $m = (1/2)\ell^2 \cdot 2^{-n/3}$, we get the probability

$$1 - \exp\left(1 - \frac{\ell^4 \cdot 2^{-4n/3}}{8} - \frac{2}{\ell^4 \cdot 2^{-4n/3}}\right).$$

Unwinding the stack, the probability that Wagner's k-tree algorithm (Figure 20) finds a solution is lower bounded by the probability that the lists are indistinguishable from random, L_{12} and L_{34} have size at least $(1/2)\ell^2 \cdot 2^{-n/3}$, and colls has size at least 1, which, from the above discussion, is at least

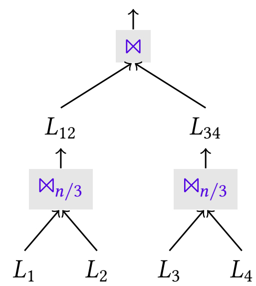
$$(1 - \ell \cdot \xi)\left(1 - \exp\left(-\frac{\ell^2 \cdot 2^{-n/3}}{8}\right)\right)\left(1 - \exp\left(1 - \frac{\ell^4 \cdot 2^{-4n/3}}{8} - \frac{2}{\ell^4 \cdot 2^{-4n/3}}\right)\right).$$

Now we turn to analyzing the runtime. We first analyze the subroutines merge and $\mathsf{lowBitMerge}$, then analyze the full $\mathsf{fourWayCollision}$ routine. First, the merge subroutine on lists of size m: the two sort operations cost $2\mathsf{Sort}(m)$, and the body of the loop is run $2m$ times with at most 4 statements. So, the total cost is $8m + 2\mathsf{Sort}(m)$. Second, the $\mathsf{lowBitMerge}$ subroutine on lists of size m: computing the intermediary lists costs $2m$ operations, the two sort operations cost $2\mathsf{Sort}(m)$, and the body of the loop is run $2m$ times with at most 4 statements. So, the total cost is $10m + 2\mathsf{Sort}(m)$.

Lastly, the $\mathsf{fourWayCollision}$ routine calls the $\mathsf{lowBitMerge}$ subroutine two times on lists of size ℓ and the merge subroutine on lists of size $(1/2)\ell^2 \cdot 2^{-n/3}$. Plugging in the above computed costs for the routines,

$$\begin{aligned}&2(10\ell + 2\mathsf{Sort}(\ell)) + 8 \cdot ((1/2)\ell^2 \cdot 2^{-n/3}) + 2\mathsf{Sort}((1/2)\ell^2 \cdot 2^{-n/3}) \\ &= 20\ell + 4\ell^2 \cdot 2^{-n/3} + 4\mathsf{Sort}(\ell) + 2\mathsf{Sort}((1/2)\ell^2 \cdot 2^{-n/3})\,.\end{aligned}$$

This completes the proof. □




```
fourWayCollision(L1, L2, L3, L4):
L12 ← lowBitMerge(n/3, L1, L2)
L34 ← lowBitMerge(n/3, L3, L4)
colls ← merge(L12, L34)
Return colls

merge(L1, L2):
m ← |L1| = |L2|
// Compute the intermediary lists
For i = 1, ..., m:
    H1.append(L1[i], i)
    H2.append(L2[i], i)
// Sort the intermediary lists
Sort(H1); Sort(H2)
// Look for collisions
colls ← []
j1 ← 0; j2 ← 0
While j1 < m or j2 < m:
    If H1[j1] == H2[j2]:
        i1 ← H1[j1]
        i2 ← H2[j2]
        colls.append(L1[i1] ⊕ L2[i2], (i1, i2))
    Else If H1[j1] < H2[j2]:
        j1 ← j1 + 1
    Else:
        j2 ← j2 + 1
Return colls
```

```
lowBitMerge(s, L1, L2):
m ← |L1| = |L2|
// Compute the intermediary lists
For i = 1, ..., m:
    // low_s denotes the low-s bits
    H1.append(low_s(L1[i]), i)
    H2.append(low_s(L2[i]), i)
// Sort the intermediary lists
Sort(H1); Sort(H2)
// Look for low-bit collisions
lbcolls ← []
j1 ← 0; j2 ← 0
While j1 < m or j2 < m:
    If H1[j1] == H2[j2]:
        i1 ← H1[j1]
        i2 ← H2[j2]
        lbcolls.append(L1[i1] ⊕ L2[i2], (i1, i2))
    Else If H1[j1] < H2[j2]:
        j1 ← j1 + 1
    Else:
        j2 ← j2 + 1
Return lbcolls
```

Figure 20: **(Top)** A visualization of Wagner's k-tree algorithm [43] for finding a 4-way collision, where $\bowtie_s$ and $\bowtie$ denote the lowBitMerge and merge subroutines, respectively. **(Left/Top)** Pseudocode for Wagner's k-tree algorithm [43] for finding a 4-way collision. **(Left/Bottom)** The merge subroutine which merges two lists. **(Right)** The lowBitMerge subroutine [43] which merges two lists on their lower s bits.

$\underline{\text{SIV-Decrypt}(K, A, C)\text{:}}$

$\mathsf{c} \leftarrow 1^{n-64}01^{31}01^{31}$
$C_1, \ldots, C_m \,\|\, \mathsf{tag} \leftarrow C$
$I \leftarrow \mathsf{tag}$
$K_1 \,\|\, K_2 \leftarrow K$
// CTR Decryption
$\mathsf{ctr} \leftarrow I \,\&\, \mathsf{c}$
For $i = 1..m$:
  $M_i \leftarrow C_i \oplus E_{K_2}(\mathsf{ctr} + i - 1)$
// IV Check
$I' \leftarrow \mathsf{CMAC}^*(K_1, A, M_1 \,\|\, \cdots \,\|\, M_m)$
If $I \neq I'$:
  Return $\perp$
Return M

$\underline{\mathsf{CMAC}^*(K, A, M)\text{:}}$

$S \leftarrow \mathsf{CMAC}(K, 0^n)$
$S \leftarrow (2 \cdot S) \oplus \mathsf{CMAC}(K, A)$
Return $\mathsf{CMAC}(K, M_1 \,\|\, \cdots \,\|\, M_{m-1} \,\|\, (S \oplus M_m))$

$\underline{\mathsf{CMAC}(K, X)\text{:}}$

$K_s \leftarrow 2 \cdot E_K(0^n)$
$X_1, \ldots, X_m \leftarrow X$
$X_m \leftarrow X_m \oplus K_s$
$\xi_0 \leftarrow 0^n$
For $i = 1..m$:
  $\xi_i \leftarrow E_K(\xi_{i-1} \oplus X_i)$
Return ξ_n

Figure 21: **(Left)** Pseudocode for SIV Mode [39] decryption with block-aligned message and associated data. **(Right)** Pseudocode for CMAC* [39] with two block-aligned inputs, and CMAC [20] with block-aligned input.

D Commitment Attack on Block-Aligned SIV

Continuing from Section 5, in this section we describe how to extend the commitment attack on one-block SIV (in §5) to SIV [39] with block-aligned message and associated data. The pseudocode for this scheme is given in Figure 21.

As in the one block case, the $\mathrm{CMT}_{\mathsf{k}}^*$ adversary seeks to produce a ciphertext $C = C_1 \,\|\, \cdots \,\|\, C_m \,\|\, \mathsf{tag}$ and two $2n$-bit keys $K = K_1 \| K_2$ and $K' = K_1' \| K_2'$ such that $\text{SIV-Decrypt}(K, C) \neq \perp$ and $\text{SIV-Decrypt}(K', C) \neq \perp$. Notice from Figure 21 that this reduces to two simultaneous IV checks passing which can be written as

$$\mathsf{tag} = \mathsf{CMAC}^*(K_1, A, M) = \mathsf{CMAC}^*(K_1', A, M')$$

where M and M' are CTR decryptions of C under K_2 and K_2' respectively. Our attack strategy will be to choose the tag tag, the associated data A, and the first $(m-1)$-blocks of the ciphertext $C_1, \ldots, C_{m-1}$ arbitrarily, so we can treat them as constants. Towards solving for the remaining variable, the last block of ciphertext, C_m, we can substitute in the definition of CMAC^*

$$\begin{aligned} \mathsf{tag} &= \mathsf{CMAC}\big(K_1, M_1 \,\|\, \cdots \,\|\, M_{m-1} \,\|\, (M_m \oplus (2 \cdot E_{K_1}(2 \cdot E_{K_1}(0^n))) \oplus \mathsf{CMAC}(K_1, A))\big) \\ &= \mathsf{CMAC}\big(K_1', M_1' \,\|\, \cdots \,\|\, M_{m-1}' \,\|\, (M_m' \oplus (2 \cdot E_{K_1'}(2 \cdot E_{K_1'}(0^n))) \oplus \mathsf{CMAC}(K_1', A))\big), \end{aligned}$$

then substituting the definition of CMAC

$$\begin{aligned} \mathsf{tag} &= E_{K_1}\big(\xi_{m-1} \oplus M_m \oplus (2 \cdot E_{K_1}(2 \cdot E_{K_1}(0^n))) \oplus \mathsf{CMAC}(K_1, A)\big) \\ &= E_{K_1'}\big(\xi_{m-1}' \oplus M_m' \oplus (2 \cdot E_{K_1'}(2 \cdot E_{K_1'}(0^n))) \oplus \mathsf{CMAC}(K_1', A)\big), \end{aligned}$$

where ξ_{m-1} and ξ_{m-1}' are intermediate values in the CMAC computation (see Figure 21). Rewriting M_m and M_m' in terms of C_m,

$$\begin{aligned} \mathsf{tag} &= E_{K_1}\big(\xi_{m-1} \oplus C_m \oplus E_{K_2}((\mathsf{tag} \,\&\, \mathsf{c}) + m - 1) \oplus (2 \cdot E_{K_1}(2 \cdot E_{K_1}(0^n))) \oplus \mathsf{CMAC}(K_1, A)\big) \\ &= E_{K_1'}\big(\xi_{m-1}' \oplus C_m \oplus E_{K_2'}((\mathsf{tag} \,\&\, \mathsf{c}) + m - 1) \oplus (2 \cdot E_{K_1'}(2 \cdot E_{K_1'}(0^n))) \oplus \mathsf{CMAC}(K_1', A)\big), \end{aligned}$$

where $\mathsf{c} = 1^{n-64}01^{31}01^{31}$. We can rearrange these equalities solving for the variable C_m, to get

$$\begin{aligned} C_m &= E_{K_1}^{-1}(\mathsf{tag}) \oplus \xi_{m-1} \oplus E_{K_2}((\mathsf{tag} \,\&\, \mathsf{c}) + m - 1) \oplus (2 \cdot E_{K_1}(2 \cdot E_{K_1}(0^n))) \oplus \mathsf{CMAC}(K_1, A) \\ &= E_{K_1'}^{-1}(\mathsf{tag}) \oplus \xi_{m-1}' \oplus E_{K_2'}((\mathsf{tag} \,\&\, \mathsf{c}) + m - 1) \oplus (2 \cdot E_{K_1'}(2 \cdot E_{K_1'}(0^n))) \oplus \mathsf{CMAC}(K_1', A) \end{aligned} \tag{9}$$

The above implies that it suffices now to find K_1, K_2, K_1', K_2' that satisfy Equation 9. To ease notation, we define four helper functions, one for each term:

$$
\begin{aligned}
F_1(K_1) &:= E^{-1}_{K_1}(\mathsf{tag}) \oplus \xi_{m-1} \oplus (2 \cdot E_{K_1}(2 \cdot E_{K_1}(0^n))) \oplus \mathsf{CMAC}(K_1, A)\,, \\
F_2(K_2) &:= E_{K_2}((\mathsf{tag} \,\&\, \mathsf{c}) + m - 1)\,, \\
F_3(K_1) &:= E^{-1}_{K_1'}(\mathsf{tag}) \oplus \xi'_{m-1} \oplus (2 \cdot E_{K_1'}(2 \cdot E_{K_1'}(0^n))) \oplus \mathsf{CMAC}(K_1', A)\,, \\
F_4(K_2') &:= E_{K_2'}((\mathsf{tag} \,\&\, \mathsf{c}) + m - 1)\,,
\end{aligned}
$$

and recast Equation 9 as a 4-sum problem

$$
F_1(K_1) \oplus F_2(K_2) \oplus F_3(K_1') \oplus F_4(K_2') = 0\,.
$$

Now, we can use techniques similar to the ones used in the one-block case to solve this in time about $2^{n/3}$.

```
P0-GCM[E].Dec(K, N, A, C):
C ∥ tag ← C
J_0 ← N ∥ 0^{n−ℓ−1} ∥ 1
// Tag check
H ← E_K(0^n)
lens ← encode_(n/2)(|A|) ∥ encode_(n/2)(|C|)
S ← GHASH(H, A ∥ C ∥ lens)
If tag ≠ (S ⊕ E_K(J_0)):
    Return ⊥
// CTR decryption
clen ← |C|/128
For i ← 1 to clen:
    M[i] ← E_K(J_0 + i) ⊕ C[i]
// Check padding zeroes
If M[1] ∥ M[2] ≠ 0^{2n}:
    Return ⊥
Return M[2..]

GHASH(H, X):
// Split into 16-byte blocks
X_1, ..., X_m ← X
// Compute X_1 · H^m + ··· + X_m · H
Y_0 ← 0^128
For i = 1 to m:
    Y_i ← (Y_{i−1} ⊕ X_1) · H
Return Y_m
```

```
A():
// Initialize with arbitrary constants
N ← 0^ℓ; K ← 0^n
J_0 ← N ∥ 0^{n−ℓ−1} ∥ 1
C ← (0^n ⊕ E_K(J_0 + 1)) ∥ (0^n ⊕ E_K(J_0 + 2))
// Compute colliding associated data
α_1 ← 0^n; α_2 ← 1^n
β_1 ← 0^n
H ← E_K(0^n)
β_2 ← H^{−4} · (α_1 H^5 + α_2 H^5 + β_1 H^4)
A_1 ← α_1 ∥ β_1
A_2 ← α_2 ∥ β_2
// Repackage into a context collision
lens ← encode_(n/2)(|A_1|) ∥ encode_(n/2)(|C|)
tag ← GHASH(H, A_1 ∥ C ∥ lens) ⊕ E_K(J_0)
Return (C ∥ tag), (K, N, A_1), (K, N, A_2)
```

Figure 22: **(Left/Top)** Pseudocode for GCM Mode [21, §7] decryption, over a n-bit block cipher, adapted to check for a block of padding zeroes, with a ℓ-bit nonce, a n-bit tag, and block-aligned messages and associated data. **(Left/Bottom)** Pseudocode for GHASH [21, §6.4]. **(Right)** Pseudocode for $\mathrm{CMT}_{\mathsf{a}}^{*}$ attack on $\mathsf{P0\text{-}GCM}[E]$.

E Revisiting Commitment-Enhancing Transforms

First, we look at three folklore, generic commitment enhancing transforms—padding zeroes [2, 5], key hashing [2], and libsodium's recommendation [16]—that have been shown to achieve $\mathrm{CMT}_{\mathsf{k}}$, and show that they do not achieve $\mathrm{CMT}_{\mathsf{a}}^{*}$ and thus do not achieve CMT-3. Then, we turn to Bellare and Hoang's [5] CAU-C1 scheme, and show a practically relevant key commitment attack that takes about 2^{64} time.

Padding zeroes transform. *Padding zeroes* is a folklore transform for turning a CTR-based encryption scheme into a key committing encryption scheme. It calls for prefixing a string of zeroes to the beginning of the message before encryption, and checking that the string remains intact during decryption. An early draft of the OPAQUE protocol [29, §3.1.1] recommended it to make GCM usable for envelope encryption. Albertini et al. [2, §5.3] formally analyzed this transform and showed that it achieves the CROB definition of Farshim et al. [24], which is equivalent to FROB [24], when used with GCM and ChaCha20/Poly1305. Bellare and Hoang [5, Appendix G] futher analyzed the scheme and showed that it is $\mathrm{CMT}_{\mathsf{k}}$ secure, when used with GCM and ChaCha20/Poly1305.

We focus on the transform applied to GCM with an n-bit ideal cipher E, which we call $\mathsf{P0\text{-}GCM}[E]$ (pictured in Figure 22). We show that it does not achieve our restrictive $\mathrm{CMT}_{\mathsf{a}}^{*}$ notion. This means it also does not achieve CMT-3 security. We present the following attack and note that it also generalizes to ChaCha20/Poly1305.

Recall, from Figure 3 that to defeat restrictive $\mathrm{CMT}_{\mathsf{a}}^{*}$ it suffices to produce a ciphertext C, a nonce N, a key K, and different associated data $A_1 \neq A_2$ such that $\mathsf{Dec}(K, N, A_1, C) \neq \perp$ and $\mathsf{Dec}(K, N, A_2, C) \neq \perp$. This reduces to two constraints: the tag checks should pass, and the padding zeroes checks should pass.

First, let us arbitrarily fix the nonce $N \leftarrow 0^\ell$ and key $K \leftarrow 0^n$. Then to pass the padding zeroes check constraint, we can construct the ciphertext as the CTR encryption of 0^{2n} under (K, N) which is

$$C = (0^n \oplus E_K(J_0 + 1)) \,\|\, (0^n \oplus E_K(J_0 + 2))$$

where $J_0 \leftarrow N \,\|\, 0^{n-\ell-1} \,\|\, 1$ as defined in Figure 22.

Now, it remains to produce a tag tag that collides with this ciphertext C under different associated data $A_1 \neq A_2$. Let us construct the associated data as

$$A_1 \leftarrow \alpha_1 \,\|\, \beta_1 \quad \text{and} \quad A_2 \leftarrow \alpha_2 \,\|\, \beta_2 \,,$$

with some constants $\alpha_1 \neq \alpha_2$ (to get $A_1 \neq A_2$) and *sacrificial blocks*[5] β_1 and β_2 to be computed later.

Let tag_1 and tag_2 be the tags obtained with A_1 and A_2 respectively, then

$$\begin{aligned}\mathsf{tag}_1 &\leftarrow \mathsf{GHASH}(H, A_1 \,\|\, C \,\|\, \mathsf{lens}) \oplus E_K(J_0),\\ \mathsf{tag}_2 &\leftarrow \mathsf{GHASH}(H, A_2 \,\|\, C \,\|\, \mathsf{lens}) \oplus E_K(J_0)\,,\end{aligned}$$

where C and K were fixed earlier and $H \leftarrow E_K(0^n)$, $J_0 \leftarrow N_1 \| 0^{n-\ell-1} \| 1$, and lens is an encoding of the lengths of the ciphertext and associated data. So, for tag_1 to equal tag_2 it suffices to have

$$\mathsf{GHASH}(H, A_1 \,\|\, C \,\|\, \mathsf{lens}) = \mathsf{GHASH}(H, A_2 \,\|\, C \,\|\, \mathsf{lens})\,.$$

Expanding GHASH as a polynomial

$$\begin{aligned}&\alpha_1 \cdot H^5 + \beta_1 \cdot H^4 + E_K(J_0 + 1) \cdot H^3 + E_K(J_0 + 2) \cdot H^2 + \mathsf{lens} \cdot H\\ &= \alpha_2 \cdot H^5 + \beta_2 \cdot H^4 + E_K(J_0 + 1) \cdot H^3 + E_K(J_0 + 2) \cdot H^2 + \mathsf{lens} \cdot H\,,\end{aligned}$$

and simplifying we get

$$\beta_1 H^4 + \beta_2 H^4 = \alpha_1 H^5 + \alpha_2 H^5.$$

We can then simply choose an arbitrary value for β_1 and solve for β_2 to complete the attack. Pseudocode for the attack is given in Figure 22.

Key hashing transform. Albertini et al. [2, §5.4] proposed a generic transform for converting any AEAD scheme into one that ensures key commitment, or more formally FROB security. The scheme, which we call $\mathsf{CommitKey}$, uses independent collision-resistant PRFs F_{com} and F_{enc} to derive a commitment string and AEAD encryption key, respectively, from the secret key and a nonce. While they provide four variants of this scheme that either use a nonce or do not in the evaluation of F_{com} and F_{enc}, we will specifically focus on their Type IV variant which uses a nonce for each PRF evaluation. Our results can be easily extended to the other variants. Pseudocode for this scheme is given in Figure 23.

Since the ciphertext includes a collision-resistant commitment to the key, it achieves $\mathrm{CMT}_{\mathsf{k}}$ security. Here, we show that it does not achieve our restrictive $\mathrm{CMT}^*_{\mathsf{a}}$ definition for all AEAD schemes, meaning it does not meet CMT-3 security. We show that $\mathsf{CommitKey}$ cannot be used as a generic transform for any type of context commitment. In particular, when it is used with GCM, we can provide an adversary that breaks the restrictive $\mathrm{CMT}^*_{\mathsf{a}}$ security of the scheme. This attack is similar to the $\mathsf{P0\text{-}GCM}$ one.

First, we fix the nonce $N \leftarrow (0^\ell, 0^\ell) = (N_0, N_1)$ and key $K \leftarrow 0^n$. We also fix a one-block ciphertext $C \leftarrow 0^n$ and compute the key commitment string as $K_{\mathrm{enc}} \leftarrow F_{\mathrm{enc}}(K, N_0)$. Now, it remains to produce a tag

[5]This is terminology from Schmieg [40], and refers to blocks the adversary must control.

$\underline{\mathsf{CommitKey.Dec}(K, N, A, C)}$:
$(N_0, N_1) \leftarrow N$; $C_{\mathsf{inner}} \,\|\, K'_{\mathrm{com}} \leftarrow C$
$K_{\mathrm{enc}} \leftarrow F_{\mathrm{enc}}(K, N_0)$
$K_{\mathrm{com}} \leftarrow F_{\mathrm{com}}(K, N_0)$
If $K'_{\mathrm{com}} \neq K_{\mathrm{com}}$:
Return $\perp$
$M \leftarrow \mathsf{AEAD.Dec}(K_{\mathrm{enc}}, N_1, A, C_{\mathsf{inner}})$
Return M

$\underline{\mathcal{A}()}$:
// Initialize with arbitrary constants
$N_0 \leftarrow 0^{\ell}$; $N_1 \leftarrow 0^{\ell}$; $K \leftarrow 0^n$
$C \leftarrow 0^n$
$K_{\mathrm{enc}} \leftarrow F_{\mathrm{enc}}(K, N_0)$
$K_{\mathrm{com}} \leftarrow F_{\mathrm{com}}(K, N_0)$
$J_0 \leftarrow N \,\|\, 0^{n-\ell-1} \,\|\, 1$
// Compute colliding associated data
$\alpha_1 \leftarrow 0^n$; $\alpha_2 \leftarrow 1^n$
$\beta_1 \leftarrow 0^n$
$H \leftarrow E_{K_{\mathrm{enc}}}(0^n)$
$\beta_2 \leftarrow H^{-3} \cdot (\alpha_1 H^4 + \alpha_2 H^4 + \beta_1 H^3)$
$A_1 \leftarrow \alpha_1 \,\|\, \beta_1$
$A_2 \leftarrow \alpha_2 \,\|\, \beta_2$
// Repackage into a context collision
$\mathsf{lens} \leftarrow \mathsf{encode}_{(n/2)}(\lvert A_1 \rvert) \,\|\, \mathsf{encode}_{(n/2)}(\lvert C \rvert)$
$\mathsf{tag} \leftarrow \mathsf{GHASH}(H, A_1 \,\|\, C \,\|\, \mathsf{lens}) \oplus E_{K_{\mathrm{enc}}}(J_0)$
Return $(C \| \mathsf{tag} \| K_{\mathrm{com}}), (K, N, A_1), (K, N, A_2)$

Figure 23: **(Left)** The decryption algorithm for the generic transform scheme CommitKey proposed by Albertini et al. [2, §5.4]. It transforms an AEAD scheme into one that is key-committing. **(Right)** Pseudocode for $\mathrm{CMT}_{\mathsf{a}}^{*}$ attack on $\mathsf{CommitKey}[E]$.

T that collides with this ciphertext C under different associated data $A_1 \neq A_2$. As before, we construct the associated data as $A_1 \leftarrow \alpha_1 \,\|\, \beta_1$ and $A_2 \leftarrow \alpha_2 \,\|\, \beta_2$ with constants $\alpha_1 \neq \alpha_2$ (to get $A_1 \neq A_2$) and sacrificial blocks β_1 and β_2 to be chosen later.

Let T_1 and T_2 be the tags obtained with A_1 and A_2 respectively, then

$$T_1 \leftarrow \mathsf{GHASH}(H, A_1 \,\|\, C \,\|\, \mathsf{lens}) \oplus E_K(J_0)\,,$$
$$T_2 \leftarrow \mathsf{GHASH}(H, A_2 \,\|\, C \,\|\, \mathsf{lens}) \oplus E_K(J_0)\,,$$

where $H \leftarrow E_K(0^n)$, $J_0 \leftarrow N_1 \| 0^{n-\ell-1} \| 1$, and lens is an encoding of the lengths of the ciphertext and associated data. So, for T_1 to equal T_2 it suffices to have $\mathsf{GHASH}(H, A_1 \,\|\, C) = \mathsf{GHASH}(H, A_2 \,\|\, C)$. Expanding GHASH as a polynomial

$$\alpha_1 H^4 + \beta_1 H^3 + C H^2 + \mathsf{lens} \cdot H = \alpha_2 H^4 + \beta_2 H^3 + C H^2 + \mathsf{lens} \cdot H\,,$$

and simplifying to get

$$\beta_1 \cdot H^3 + \beta_2 \cdot H^3 = \alpha_1 \cdot H^4 + \alpha_2 \cdot H^4\,.$$

We can then simply choose an arbitrary value for β_1 and solve for β_2 to complete the attack. Pseudocode for the attack is given in Figure 23.

The Libsodium approach. Libsodium [16] proposed a generic transform for converting an AEAD scheme into one with key commitment. The transform suggests replacing the AEAD's tag T with a cryptographic hash of the tag T, the key K, and the nonce N. This can be seen as a strengthening of the CommitKey discussed above, thus gets the $\mathrm{CMT}_{\mathsf{k}}$ security but unfortunately also suffers from a similar restrictive $\mathrm{CMT}_{\mathsf{a}}^{*}$ attack. We show that the libsodium approach cannot be used as a generic transform over GCM, we provide an adversary that breaks the restrictive $\mathrm{CMT}_{\mathsf{a}}^{*}$ security of the resulting scheme. This attack works similarly to that for CommitKey.

First, we fix the nonce $N \leftarrow 0^{\ell}$, key $K \leftarrow 0^n$, and inner tag $T \leftarrow 0^n$. We also fix a one-block ciphertext $C' \leftarrow 0^n$, and compute the wrapper tag as $\mathsf{tag} \leftarrow F_{\mathrm{enc}}(K, N, t)$. Now, it remains to produce different associated data $A_1 \neq A_2$ such that the inner tag T verifies the ciphertext C. As before, we construct the

libsodium.Dec(K, N, A, C):
$C_{\mathsf{inner}} \parallel \mathsf{tag} \leftarrow C$ $T \leftarrow \text{ComputeTag}(K, N, A, C_{\mathsf{inner}})$ If $\mathsf{H}(T, K, N) \neq \mathsf{tag}$:   Return $\perp$ $M \leftarrow \mathsf{AEAD.Dec}(K, N, A, C_{\mathsf{inner}} \parallel T)$ Return M

$\mathcal{A}()$:

// Initialize with arbitrary constants
$N \leftarrow 0^\ell; K \leftarrow 0^n$
$T \leftarrow 0^n; C \leftarrow 0^n$
$J_0 \leftarrow N \parallel 0^{n-\ell-1} \parallel 1$
$\mathsf{lens} \leftarrow \mathsf{encode}_{(n/2)}(\lvert A_1 \rvert) \parallel \mathsf{encode}_{(n/2)}(\lvert C \rvert)$
// Compute colliding associated data
$\alpha_1 \leftarrow 0^n; \alpha_2 \leftarrow 1^n$
$\beta_1 \leftarrow 0^n$
$H \leftarrow E_K(0^n)$
$\beta_1 \leftarrow H^{-3}\left(E_K(J_0) + \alpha_1 \cdot H^4 + C \cdot H^2 + \mathsf{lens} \cdot H\right)$
$\beta_2 \leftarrow H^{-3}\left(E_K(J_0) + \alpha_2 \cdot H^4 + C \cdot H^2 + \mathsf{lens} \cdot H\right)$
$A_1 \leftarrow \alpha_1 \parallel \beta_1$
$A_2 \leftarrow \alpha_2 \parallel \beta_2$
// Repackage into a context collision
$\mathsf{tag} \leftarrow \mathsf{H}(T, K, N)$
Return $(C \parallel \mathsf{tag}), (K, N, A_1), (K, N, A_2)$

Figure 24: **(Left)** The decryption algorithm for libsodium's approach [16]. It transforms a tag-based AEAD scheme into one that is key-committing. **(Right)** Pseudocode for $\mathrm{CMT}_{\mathsf{a}}^*$ attack on libsodium's approach.

associated data as $A_1 \leftarrow \alpha_1 \parallel \beta_1$ and $A_2 \leftarrow \alpha_2 \parallel \beta_2$ with constants $\alpha_1 \neq \alpha_2$ (to get $A_1 \neq A_2$) and sacrificial blocks β_1 and β_2 to be chosen later.

Then we can write the tag check condition as

$$T = \mathsf{GHASH}(H, A_1 \parallel C \parallel \mathsf{lens}) \oplus E_K(J_0)\,,$$
$$T = \mathsf{GHASH}(H, A_2 \parallel C \parallel \mathsf{lens}) \oplus E_K(J_0)\,,$$

where $H \leftarrow E_K(0^n)$, $J_0 \leftarrow N_1 \| 0^{n-\ell-1} \| 1$, and lens is an encoding of the lengths of the ciphertext and associated data. Since we set the inner tag $T = 0^n$ earlier, we write this condition as

$$E_K(J_0) = \mathsf{GHASH}(H, A_1 \parallel C \parallel \mathsf{lens}) = \mathsf{GHASH}(H, A_2 \parallel C \parallel \mathsf{lens})\,.$$

Expanding GHASH as a polynomial we get two equations

$$E_K(J_0) = \alpha_1 \cdot H^4 + \beta_1 \cdot H^3 + C \cdot H^2 + \mathsf{lens} \cdot H\,,$$
$$E_K(J_0) = \alpha_2 \cdot H^4 + \beta_2 \cdot H^3 + C \cdot H^2 + \mathsf{lens} \cdot H\,,$$

which can be rewritten as

$$\beta_1 = H^{-3}\left(E_K(J_0) + \alpha_1 \cdot H^4 + C \cdot H^2 + \mathsf{lens} \cdot H\right),$$
$$\beta_2 = H^{-3}\left(E_K(J_0) + \alpha_2 \cdot H^4 + C \cdot H^2 + \mathsf{lens} \cdot H\right),$$

Pseudocode for the attack is given in Figure 24.

Faster salamanders against Bellare and Hoang's CAU-C1. Bellare and Hoang [5, §5] proposed tweaking GCM tag generation using Davies-Meyer-inspired ideas to produce a key committing cipher CAU-C1. Pseudocode for the tag generation portion of GCM and CAU-C1 are given in Figure 25. Both schemes use a nonce of length m and block cipher E with block length n. In practice, typically GCM will be used with AES-128 and so $n = 128$. In the untruncated tag setting where CAU-C1 produces a 128-bit tag, Bellare and Hoang prove that it provides 2^{64} key committing security. But, 2^{64} is on the edge of practicality, and it is unclear if attacks like the *invisible salamanders attack* of Dodis et al. [18] are practical.

GCM-Tag(K, N, A, C):	CAU-C1-Tag(K, N, A, C):
$H \leftarrow E_K(0^n)$	$H \leftarrow E_K(0^n)$
$R \leftarrow \mathsf{GHASH}(H, A \,\|\, C)$	$R \leftarrow \mathsf{GHASH}(H, A \,\|\, C)$
// postprocess GHASH value	// postprocess GHASH value
$Y \leftarrow N \,\|\, 0^{n-\ell-1} \,\|\, 1$	$Y \leftarrow N \,\|\, 0^{n-\ell-1} \,\|\, 1$
$S \leftarrow E_K(Y) \oplus R$	$V \leftarrow Y \oplus R$
Return S	$S \leftarrow E_K(V) \oplus V$
	Return S

Figure 25: **(Left)** Tag computation in GCM [21]. **(Right)** Tag computation in CAU-C1 [5]. The differences are highlighted in blue.

Indeed, a straightforward adaptation of the invisible salamanders attack [18, §3.2] against Facebook's message franking scheme would require time about 2^{81}. To explain, recall that the DGRW attack works as follows against GCM. At a high level, it relies on a sender Alice sending a ciphertext twice with the same nonce N, but with different keys K_1 and K_2 each time. Alice constructs the ciphertext so that under K_1 and N the ciphertext decrypts to an innocuous BMP image file and under K_2 and N the ciphertext decrypts to an abusive JPEG image. When the recipient Bob tries to report the ciphertext corresponding to the abusive image, only the first innocuous image will be seen by the platform.

In the simple version of the attack, Alice constructs the ciphertext by encrypting the abusive image under K_2 and N and then solving a linear equation to compute the last ciphertext block needed. The resulting ciphertext outputs to the same GHASH authentication tag under both (K_1, N) and (K_2, N). However, this would mean that under (K_1, N) the ciphertext would decrypt to junk bytes. To create meaningful plaintexts, the DGRW attack exploits the structures of JPEG and BMP images: when decrypting under (K_1, N) to produce the harmless BMP image, the vital JPEG data is in junk bytes at the end of the file that the BMP parser ignores, and when decrypting under (K_2, N) to produce the abusive JPEG image, the vital BMP data is contained in a JPEG comment before the JPEG data that is ignored. Still, the attack relies on the first 4 bytes of the ciphertext to decrypt to plaintext bytes that are semantically meaningful for BMP and JPEG parsers. The attack resolves this by fixing two keys and brute-force searching for a nonce N that collides the desired plaintext bytes to the same ciphertext bytes under each key, which requires time about 2^{32}. DGRW suggest that this can be sped up by instead fixing a nonce and doing a birthday attack on the keys to produce a collision, which should take time about 2^{17}.

CAU-C1 prevents this attack from working directly, because the attack now depends on colliding the authentication tag S computed using a Davies-Meyer hash shown in Figure 25 (right) rather than solving a simple linear equation that exploits the structure of GHASH. An attacker could compensate by running a birthday-style attack to find two keys such that they collide $E_K(V) \oplus V$, for different keys, to the same value. For a block cipher E with block length 128, this requires time about 2^{64}. However, the two keys would also need to collide on the 4 bytes of plaintext, which now adds a multiplicative factor of 2^{17}. This results in a total time of about 2^{81}.

We show a new invisible salamanders attack against CAU-C1 that takes about $2^{64} + 2^{32}$ time. This brings the attack back into the feasible region for well-resourced adversaries. The key insight is that we can essentially "separately" solve the problem of finding key collisions against the Davies-Meyer tag (in time about 2^{64}) and finding ciphertexts for those keys that conform to the plaintext format requirements for targeted file formats. For the latter, we focus on the file formats chosen in DGRW, namely a JPEG and BMP. But our attack can readily be extended using the techniques of Albertini et al. [2, §4] to work against more than 250 file format combinations.

Towards building up the attack, we work backwards from the adversary's goal: computing two tuples

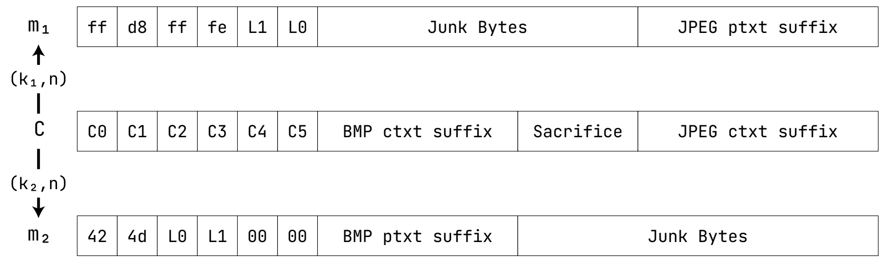


Figure 26: A slightly simplified construction of JPEG/BMP salamanders, adapted from Dodis et al. [18]. The ciphertext C (middle) is decrypted with (K_1, n) and (K_2, n) to get M_1 (top) and M_2 (bottom) respectively.

(K_1, N, A, M_1) and (K_2, N, A, M_2) such that

$$C := \mathsf{CAU\text{-}C1.Enc}(K_1, N, A, M_1) = \mathsf{CAU\text{-}C1.Enc}(K_2, N, A, M_2) \ ,$$

where M_1 is a valid JPEG image and M_2 is a valid BMP image. This in turn implies that the tags

$$\mathsf{CAU\text{-}C1\text{-}Tag}(K_1, N, A, C) = \mathsf{CAU\text{-}C1\text{-}Tag}(K_2, N, A, C)$$

collide, which, substituting in the tag generation definitions gives that

$$E_{K_1}(V_1) \oplus V_1 = E_{K_2}(V_2) \oplus V_2 \ . \tag{10}$$

We choose $V_1 = V_2 = 0^n$, and we will see later how the attack arranges for this to be true. Doing so simplifies Equation 10 to $E_{K_1}(0^n) = E_{K_2}(0^n)$. We then can use a birthday attack to find K_1, K_2 satisfying this equation in time about 2^{64} since E has blocksize $n = 128$.

The attacker must then ensure that $0^n = Y_1 \oplus R_1$ and $0^n = Y_2 \oplus R_2$. Notice that Y_1 and Y_2 are just the nonce N with padding, so the attacker ends up needing to ensure that $R_1 = R_2$ which means finding a GHASH collision

$$\mathsf{GHASH}(H_1, A \,\|\, C \,\|\, \mathsf{lens}) = \mathsf{GHASH}(H_2, A \,\|\, C \,\|\, \mathsf{lens}) \ . \tag{11}$$

where lens encodes the lengths of the ciphertext C and associated data A.

Before we can solve this equation, we need to fix the ciphertext. We want to construct a η-block ciphertext $C = \zeta_1 \cdots \zeta_\eta$ that decrypts under different key-nonce pairs (K_1, N) and (K_2, N) to a JPEG and a BMP, respectively. The ciphertext starts with some leading bytes (to be fixed later) corresponding to metadata; followed by an encryption of BMP data under (K_2, N); followed by the two sacrificial blocks ζ_j and ζ_k to be fixed later; followed by an encryption of JPEG data under (K_1, N). This construction is illustrated in Figure 26.

We defer to Dodis et al. [18, §3.2] for technical details of this construction and note that this step can be swapped out in favor of a different file format combination in which case we defer to Albertini et al. [2, §4]. But, for the sake of runtime analysis we will mention a few details.

First, the two leading bytes in M_1 and M_2 (`ff d8` and `42 4d`) correspond to file headers and need to be encoded precisely. Second, the `ff fe` in M_1 and `00 00` in M_2 correspond to the JPEG "comment header" and the start of "BMP data" and need to be encoded precisely as well. Third, the `L0` and `L1` in M_1 corresponds to length of the "JPEG comment" parsed as $\ell_{\text{JPEGComment}} := \texttt{L0} + 256 \cdot \texttt{L1}$. Similarly, the `L0` and `L1` in M_2 corresponds to length of the "BMP data" parsed as $\ell_{\text{BMPData}} := \texttt{L0} + 256 \cdot \texttt{L1}$. Informally, since we are putting the "BMP data" inside the "JPEG comment" we require that $\ell_{\text{JPEGComment}} > \ell_{\text{BMPData}}$, and that ℓ_{BMPData} is greater than the size of our BMP file.

While this is not the most efficient approach, for simplicity, we follow Dodis et al. [18, §3.2] and ask that the first 4 bytes be encoded precisely and they allow flexibility in the last two bytes. We start by picking the kitten BMP file used in Dodis et al. [18, §3.2] which has size of 9502 bytes and fixing $\ell_{\text{BMPData}} = 9502$ (i.e., `L0=1e` and `L1=25`). We then enumerate nonces until we can satisfy

$$\begin{aligned}\mathsf{Dec}((K_1, N), C_0C_1C_2C_3) &= \texttt{ff d8 ff fe}\,, \quad \text{and} \\ \mathsf{Dec}((K_2, N), C_0C_1C_2C_3) &= \texttt{42 4d 1e 25}\,.\end{aligned}$$

Put differently, we want

$$\mathsf{Enc}((K_1, N), \texttt{ff d8 ff fe}) = \mathsf{Enc}((K_2, N), \texttt{42 4d 1e 25})\,.$$

Using a birthday attack on the nonce, we can achieve this in about 2^{32} time in the average case. Once we have this, we can set

$$C_4C_5 := \mathsf{Enc}((K_2, N), \texttt{00 00})$$

and, with high probability, we will get that

$$\texttt{L0 L1} := \mathsf{Dec}((K_1, N), C_4C_5)$$

will satisfy

$$\ell_{\text{JPEGComment}} = \texttt{L0} + 256 \cdot \texttt{L1} > 9502 = \ell_{\text{BMPData}}\,,$$

which is the sole remaining condition. In summary, with about 2^{32} effort, we can find a nonce N and construct a ciphertext C such that C decrypts under (K_1, N) and (K_2, N) to a JPEG and a BMP respectively.

With the newly constructed ciphertext C and fixed nonce N in hand, we can go back to Equation 11, writing it as two equations

$$\begin{aligned}Y &= \mathsf{GHASH}(H_1, A \,\|\, C \,\|\, \mathsf{lens})\,, \\ Y &= \mathsf{GHASH}(H_2, A \,\|\, C \,\|\, \mathsf{lens})\,.\end{aligned}$$

Split A and C into blocks as $A = \alpha_1 \cdots \alpha_\nu$ and $C = \zeta_1 \cdots \zeta_j \zeta_k \cdots \zeta_\eta$ with sacrificial blocks ζ_j and ζ_k to be fixed later. Then expanding GHASH as a polynomial gives

$$\begin{aligned}Y &= \sum_{i=1}^{\nu} \left(\alpha_i \cdot H_1^{i+\eta+1}\right) + \sum_{i=1}^{\eta} \left(\zeta_i \cdot H_1^{i+1}\right) + \mathsf{lens} \cdot H_1\,, \ \text{and} \\ Y &= \sum_{i=1}^{\nu} \left(\alpha_i \cdot H_2^{i+\eta+1}\right) + \sum_{i=1}^{\eta} \left(\zeta_i \cdot H_2^{i+1}\right) + \mathsf{lens} \cdot H_2\,.\end{aligned}$$

Rearranging terms, we get that

$$\begin{aligned}\zeta_j \cdot H_1^{j+1} + \zeta_k \cdot H_1^{k+1} &= Y + \sum_{i=1}^{\nu} \left(\alpha_i \cdot H_1^{i+\eta+1}\right) + \sum_{i=1}^{\eta} \left(\zeta_i \cdot H_1^{i+1}\right) + \mathsf{lens} \cdot H_1\,, \ \text{and} \\ \zeta_j \cdot H_2^{j+1} + \zeta_k \cdot H_2^{k+1} &= Y + \sum_{i=1}^{\nu} \left(\alpha_i \cdot H_2^{i+\eta+1}\right) + \sum_{i=1}^{\eta} \left(\zeta_i \cdot H_2^{i+1}\right) + \mathsf{lens} \cdot H_2\,.\end{aligned}$$

We can view these two equations as a system of 2 linear equations in 2 variables (ζ_j and ζ_k). Since we are operating over the finite field GF(2^{128}), we can solve this system via matrix inversion to find the sacrificial blocks ζ_j and ζ_k.