The proceedings version of this paper appears as [21]. This is the full version.

Authenticated-Encryption with Associated-Data

Phillip Rogaway*

20 September 2002

Abstract

When a message is transformed into a ciphertext in a way designed to protect both its privacy and authenticity, there may be additional information, such as a packet header, that travels alongside the ciphertext (at least conceptually) and must get authenticated with it. We formalize and investigate this *authenticated-encryption with associated-data* (AEAD) problem. Though the problem has long been addressed in cryptographic practice, it was never provided a definition or even a name. We do this, and go on to look at efficient solutions for AEAD, both in general and for the authenticated-encryption scheme OCB. For the general setting we study two simple ways to turn an authenticated-encryption scheme that does not support associated-data into one that does: *nonce stealing* and *ciphertext translation*. For the case of OCB we construct an AEAD-scheme by combining OCB and the pseudorandom function PMAC, using the same key for both algorithms. We prove that, despite "interaction" between the two schemes when using a common key, the combination is sound. We also consider achieving AEAD by the generic composition of a nonce-based, privacy-only encryption scheme and a pseudorandom function.

Keywords: Associated-data problem, authenticated-encryption, block-cipher usage, key separation, modes of operation, OCB .

*Department of Computer Science, University of California at Davis, Eng. II Building, Davis, California 95616 USA; and Department of Computer Science, Faculty of Science, Chiang Mai University, Chiang Mai 50200 Thailand. Email: `rogaway@cs.ucdavis.edu` Web: `http://www.cs.ucdavis.edu/~rogaway`

Contents

1 Introduction

Security practice has long recognized the following cryptographic problem: flow a message in such a way that part of it is privacy-protected, part of it is in the clear, and all of it is authenticated. In this paper we formalize and investigate this goal, which we call *authenticated-encryption with associated-data* (AEAD).

THE GENERIC COMPOSITION APPROACH. In the past, protocol designers addressed AEAD using the generic composition paradigm (as first named and investigated by [3]), where one glues together a (privacy-only) encryption scheme and a message authentication code (MAC). One might, for example, encrypt a string M, prepend a header H, and then MAC the resulting string. Solutions like this are so natural and obvious that it seems to have escaped notice that one was addressing a cryptographic problem in its own right.

What brought about the recognition of AEAD as a distinct cryptographic problem was the development of techniques that provide privacy+authenticity *without* using the generic composition paradigm. Beginning with Jutla [15] and continuing with Gligor et al. [9] and Rogaway et al. [22] there emerged new block-cipher modes that entwined privacy and authenticity in a single, compact mode. Such "integrated" authenticated-encryption (AE) schemes promised improved efficiency compared to the generic composition of conventional mechanisms. But the schemes also had a significant (and initially unnoticed) shortcoming: an apparent inability to efficiently authenticate a string of *associated-data*, such as a message header, binding this to the ciphertext.

NAIVE SOLUTIONS. To see some of the issues, consider a protocol that flows a message $H \,\|\, C \,\|\, T$ where H is a message header and C is determined by encrypting a plaintext M under a key K_1 and T is determined by MACing the string $H \,\|\, C$ under a key K_2. Suppose, to make things faster, we wish to modify this flow to employ an AE-scheme such as OCB [22]. (1) We cannot just send an OCB-encrypted $H \,\|\, M$ because, presumably, H had to be in the clear for purposes of routing or parsing the message. (2) Nor can we OCB-encrypt just M and send this along with H, for in this case we would have done nothing to authenticate H. (3) We can not get around this problem by sending a message consisting of H, a MAC over H, and the OCB-encrypted M, because we would have done nothing to bind H to M. (4) We could send H, the OCB-encrypted M, and a MAC taken over both H and the OCB-encrypted M, but this would lengthen the transmitted message and waste time computing a MAC over information that was already authenticity-protected by OCB. (5) We could send an OCB-encrypted $H \,\|\, M$ along with H, now encrypting H only as a means to provide for its authenticity, but doing this would again lengthen the message sent. (6) We might try to erase this inefficiency by having the sender omit from the ciphertext the portion of it that corresponds to H (assuming that the ciphertext has such a structure, as it does with OCB). But such an approach does not, in general, work: an AE-scheme is not required to provide authenticity if misused in this way (and modes like [15, 22] do not provide authenticity if so misused).

CONTRIBUTIONS OF THIS PAPER. This paper singles out AEAD as a cryptographically-significant problem and provides a provable-security treatment of it. First we give a definition for the security of an AEAD-scheme. Our definition is very strong; in particular, the attack-model gives the adversary the ability to control the associated-data H, while the notion of adversarial success generalizes the notion of authenticity of ciphertexts [4, 17].

Second, we describe two ways to turn an AE-scheme into an AEAD-scheme. One method, suggested by Cam-Winget and Walker [6], we call *nonce stealing*. The method is simple and useful, but somewhat limited in its applicability as, in practice, the associated-data H can only be a few bytes. A less restrictive approach, *ciphertext translation*, works like this: we use the AE-scheme to encrypt message M under a key K, getting an intermediate ciphertext CT; we apply a hash-function $F_{K'}$ to the associated data H to get an offset Δ; and then the final ciphertext $\mathcal{C}$ is CT except that Δ is xored into its last $|\Delta|$ bits. When the associated-data is the empty string we let $\mathcal{C} = CT$ so that the AEAD-scheme will be a proper extension of the AE-scheme. Notice that if H is held fixed during a communications session then Δ may be precomputed, essentially eliminating the per-message cost of binding in H. We prove that ciphertext translation produces a secure AEAD-scheme if the underlying AE-scheme is secure and F is good either as an almost-xor-universal (AXU) hash-function or a pseudorandom function (PRF).

Third, we concretize and adjust the general ciphertext-translation solution to yield a suggestion tailored to OCB [22]. Namely, define $\dot{\text{OCB}} = (\dot{\mathcal{K}}, \dot{\mathcal{E}}, \dot{\mathcal{D}})$ by combining OCB and PMAC according to ciphertext translation—except use the same key for both primitives.[1] So to compute $\dot{\mathcal{E}}_K^{N,H}(M)$ the message M is OCB-encrypted under key K to get $\mathcal{C} = C \,\|\, T = \mathcal{E}_K^N(M)$ where C is the "ciphertext core" and T is a τ-bit "tag"; associated-data H, if nonempty, is PMAC-authenticated [5] under the same key to yield a τ-bit result $\Delta = \text{PMAC}_K(H)$ (set $\Delta = 0^\tau$ if $H = \varepsilon$); and the $\dot{\text{OCB}}$-ciphertext is $\dot{\mathcal{E}}_K^{N,H}(M) = C \,\|\, (T \oplus \Delta)$. We favor this OCB-extension because it is simple to implement (especially when the associated-data is less than one block long), retains OCB's use of a single block-cipher key, is fully parallelizable in both M and H, has near-zero per-message cost when H is fixed per session, and one recovers a parallelizable PRF $\text{MAC}_K(H)$ as $\dot{\text{OCB}}.\text{Enc}_K^{\mathbf{0},H}(\varepsilon)$.

Finally, we examine achieving AEAD by generic composition. In particular, we look at gluing together a nonce-based symmetric encryption scheme and a PRF. Our results here seem a bit different from those in [3] (both encrypt-then-mac and mac-then-encrypt work fine), but that is not surprising, because we start from somewhat different tools.

Origin of the problem. The need to handle associated-data when using an integrated AE mode was first pointed out to the author by Burt Kaliski [16]. Several more individuals soon communicated the same sentiment. Those attuned to this problem were involved in standardization efforts that needed to bind to a ciphertext some cleartext data, such as an IP address. People wanted a cheap and secure way to do this when using an AE-mode such as OCB.

Additional related work. Hawkes and Rose [13] propose a way to modify Jutla's IAPM mode [15] in order to create an AEAD-scheme. They claim a security proof and that their method works for authenticated-encryption schemes beyond IAPM. A proposal by Whiting, Housley and Ferguson [23] constructs an AEAD-scheme that entails CTR mode encryption and the CBC-MAC. A proof is offered by [14]. Somewhat further afield, recent work that considers circumstances under which a key may be safely reused across two different cryptographic mechanisms include [8, 12]. An early version of the current paper was provided to NIST and has been on their web site since Nov '01. The proceedings version of this paper appears as [21].

Remarks. (1) AE and AEAD schemes employ a nonce. They have to do this (or be stateful or probabilistic) in order to achieve semantic security [11]. It is the responsibility of the sender not to reuse any nonce. For this purpose the sender will need to maintain state (such as a counter) or use coins. The receiver can be stateless (replay-detection is not a part of the defined goal) and deterministic. (2) It is outside of the model how the associated-data H is made known to the receiver. We do not consider the associated-data to be part of the ciphertext, though the

[1] We emphasize that the reuse of cryptographic keys across two cryptographic mechanisms is, in general, a dangerous thing to do. This practice should be contemplated only when there is a proof establishing that, for the target context, key re-use does not lead to trouble. Once a key is used across two different mechanisms the combined mechanism must be thought of as a single, atomic mechanism.

receiver will need it in order to decrypt. The same comments apply the nonce N. (3) All of our solutions allow one to bind-in the associated-data H regardless of the length of the plaintext; there is no restriction such as the plaintext having some minimum number of bits. (4) Correctness of ciphertext translation using an AXU hash-function relies on the AE-scheme meeting a stronger-than-usual definition of privacy: ciphertexts should be indistinguishable from random bits (when the adversary launches a chosen-plaintext attack), which we call IND\$-CPA. This notion, used already in [22], asks more than IND-CPA, where ciphertexts must be indistinguishable from the encryption of random bits. The IND\$-CPA property also allows the direct use of an encryption scheme as a pseudorandom generator or as a PRF. (5) Given the frequency with which networking protocols need to solve the AEAD-problem—not the privacy problem, the authenticity problem, or the AE-problem—we begin to view AEAD as the "right" goal in many settings. We suggest that it is the abstract interface of an AEAD-scheme that designers of secure networking protocols should, in most instances, be writing to and thinking in terms of. Simply understanding the signature of an AEAD-scheme—what are the inputs and outputs—may make a helpful abstraction boundary.

2 Preliminaries

ADVERSARIES. An *adversary* is a program with access to an oracle. Oracle queries are tuples of strings. An adversary is *nonce-respecting* if it never repeats the first component, N, to its oracle, regardless of oracle responses. Adversaries for AE and AEAD schemes are always assumed to be nonce-respecting. We write an oracle as superscript to the adversary that uses it.

AE-SCHEMES. We follow [22] (which builds on [1,4,11]) in defining nonce-using authenticated-encryption schemes and their security. An *authenticated-encryption scheme* (an AE-scheme), or simply an *encryption scheme*, is a three-tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Associated to Π are sets $\mathsf{Nonce} = \{0,1\}^n$ and $\mathsf{Message} \subseteq \{0,1\}^*$, the latter having a linear-time membership test and satisfying $M \in \mathsf{Message} \Rightarrow M' \in \mathsf{Message}$ for any M' of the same length as M. The key space $\mathcal{K}$ is a finite nonempty set of strings. Algorithm $\mathcal{E}$ is a deterministic algorithm that takes strings $K \in \mathcal{K}$ and $N \in \mathsf{Nonce}$ and $M \in \mathsf{Message}$ and returns a string $\mathcal{C} = \mathcal{E}_K^N(M) = \mathcal{E}_K(N, M)$. Algorithm $\mathcal{D}$ is a deterministic algorithm that takes strings $K \in \mathcal{K}$ and $N \in \mathsf{Nonce}$ and $\mathcal{C} \in \{0,1\}^*$ and returns $\mathcal{D}_K^N(\mathcal{C})$, which is either a string in $\mathsf{Message}$ or else the distinguished symbol INVALID. We require that $\mathcal{D}_K^N(\mathcal{E}_K^N(M)) = M$ for all $K \in \mathcal{K}$, $N \in \mathsf{Nonce}$, and $M \in \mathsf{Message}$. We assume that $|\mathcal{E}_K^N(M)| = \ell(|M|)$ for some linear-time computable "length function" ℓ.

Let $\$(\cdot,\cdot)$ be an oracle that, on input N, M, returns a random string of length $\ell(|M|)$ where ℓ is the length function of Π. Let A be an adversary. Define $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A) = \Pr[K \xleftarrow{\$} \mathcal{K}\colon A^{\mathcal{E}_K(\cdot,\cdot)} = 1] - \Pr[A^{\$(\cdot,\cdot)} = 1]$. We call this notion IND\$-CPA: indistinguishability from random bits under a chosen-plaintext attack. It appears in [22].

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an AE-scheme. Choose $K \xleftarrow{\$} \mathcal{K}$ and run the adversary A, providing it an oracle for $\mathcal{E}_K(\cdot,\cdot)$. We say that adversary A *forges* (for this key K and on some particular run) if A outputs a pair $(N, \mathcal{C})$ where $\mathcal{D}_K^N(\mathcal{C}) \neq$ INVALID and A did not ask a query $\mathcal{E}_K(N, M)$ that resulted in a response $\mathcal{C}$. Let $\mathbf{Adv}_{\Pi}^{\mathrm{auth}}(A)$ be the probability that A forges. The probability is over the random choice of K and over the internal coin tosses, if any, of A.

We note that requiring A to be nonce-respecting does not give rise to a restrictive notion. Quite the opposite: we are allowing the adversary to choose the nonce, rather than the sender, only demanding that it does not request multiple encryptions under the same nonce. Note too that A being nonce-respecting does not forbid use of a formerly-queried nonce within A's forgery attempt.

AXU HASH-FUNCTIONS. Function families and universality conditions on them originate with Carter and Wegman [7]. A function family is a map $F\colon \mathcal{K}\times\mathcal{X} \to \{0,1\}^\tau$ where $\mathcal{K}$ has an associated distribution and $\mathcal{X} \subseteq \{0,1\}^*$. We assume that $\mathcal{X}$ has a linear-time membership test. We use a variant of the property called *almost-xor-universal* (AXU), which was first defined by [19]. For consistency with other notions we define xor-universality as a kind of adversarial advantage. For $F\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^\tau$ a function family and A an adversary, let $\mathbf{Adv}_F^{\mathrm{axu}}(A)$ be the larger of $\delta = \Pr[K \xleftarrow{\$} \mathcal{K};\ (X_1, X_2, \Delta) \leftarrow A : X_1 \neq X_2 \text{ and } F_K(X_1) \oplus F_K(X_2) = \Delta]$ and $\epsilon = \Pr[K \xleftarrow{\$} \mathcal{K};\ (X, C) \leftarrow A : F_K(X) = C]$.

PSEUDORANDOM FUNCTIONS. Pseudorandom function originate with [10]; our treatment is a concrete-security one that follows [2]. Let $F\colon \mathcal{K}\times\mathcal{X} \to \{0,1\}^\tau$ be a function family. Let $\mathrm{Rand}(\mathcal{X}, \tau)$ be the set of all functions from $\mathcal{X}$ to $\{0,1\}^\tau$. Define $\mathbf{Adv}_F^{\mathrm{prf}}(A)$ as $\Pr[K \xleftarrow{\$} \mathcal{K}\colon A^{F_K(\cdot)} = 1] - \Pr[\rho \xleftarrow{\$} \mathrm{Rand}(\mathcal{X}, \tau)\colon A^{\rho(\cdot)} = 1]$. Let $\mathrm{Perm}(n)$ be the set of all permutations from n bits to n bits.

RUNNING-TIME CONVENTIONS. When we speak of the running time of an algorithm we include its description size, relative to some fixed encoding. If $f\colon \mathcal{K}\times\mathcal{X} \to \mathcal{Y}$ then $\mathrm{Time}_f(q, \sigma)$ is the worst-case time to compute $K \xleftarrow{\$} \mathcal{K}$ plus the time to compute $f_K(X_1), \ldots, f_K(X_q)$ where $\sum |X_q| \leq \sigma$. When we write an expression for the running time of an algorithm and that expression includes an $O(\cdot)$, the constants hidden in the big-O notation are absolute constants, depending only on the details of the model of computation.

RESOURCE-PARAMETERIZED ADVANTAGE. If Π is a scheme and A is an adversary and $\mathbf{Adv}_\Pi^{\mathrm{xxx}}(A)$ is a measure of adversarial advantage already defined, then we write $\mathbf{Adv}_\Pi^{\mathrm{xxx}}(\mathcal{R})$ to mean the maximal value of $\mathbf{Adv}_\Pi^{\mathrm{xxx}}(A)$ over all adversaries A that use resources bounded by $\mathcal{R}$. Here $\mathcal{R}$ is a list of variables specifying the resources of interest for the adversary in question. The name of the variable will be enough to unambiguously indicate the resource in question. In this paper the adversarial resources to which we pay attention are: t—the running time of the adversary; q—the number of queries asked by the adversary; σ—the aggregate length of these queries; $\hat{\sigma}$—the length of the longest query; ς—the length of the adversary's output. String lengths and aggregate string lengths can be measured either in bits or in n-bit blocks (when a value n is understood); when it matters we will specify the convention. When one measure lengths in terms of n-bit blocks a string of ℓ bits contributes $\min\{1, \lceil|\ell|/n\rceil\}$ to the total. Note that an adversary's queries and its output may encode multiple strings, and we count in σ and ς the length of the *entire* string.

3 Definition of the Goal

SYNTAX. We define an *authenticated-encryption scheme with associated-data* (an *AEAD-scheme*) as a three-tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Associated to Π are sets of strings $\mathsf{Nonce} = \{0,1\}^n$ and $\mathsf{Message} \subseteq \{0,1\}^*$, as before, and also a set $\mathsf{Header} \subseteq \{0,1\}^*$ that has a linear-time membership test. The key space $\mathcal{K}$ is as before. The encryption algorithm $\mathcal{E}$ is a deterministic algorithm that takes strings $K \in \mathcal{K}$ and $N \in \mathsf{Nonce}$ and $H \in \mathsf{Header}$ and $M \in \mathsf{Message}$. It returns a string $\mathcal{C} = \mathcal{E}_K^{N,H}(M) = \mathcal{E}_K(N, H, M)$. Decryption algorithm $\mathcal{D}$ is a deterministic algorithm that takes strings $K \in \mathcal{K}$ and $N \in \mathsf{Nonce}$ and $H \in \mathsf{Header}$ and $\mathcal{C} \in \{0,1\}^*$. It returns $\mathcal{D}_K^{N,H}(\mathcal{C})$, which is either a string in $\mathsf{Message}$ or the distinguished symbol INVALID. We require that $\mathcal{D}_K^{N,H}(\mathcal{E}_K^{N,H}(M)) = M$ for all $K \in \mathcal{K}$, $N \in \mathsf{Nonce}$, $H \in \mathsf{Header}$, and $M \in \mathsf{Message}$. As before, $|\mathcal{E}_K^N(M)| = \ell(|M|)$ for some linear-time computable length function ℓ.

SECURITY. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an AEAD-scheme with length function ℓ. Let $\$(\cdot,\cdot,\cdot)$ be an oracle that, on input (N, H, M), returns a random string of $\ell(|M|)$ bits. Let $\mathbf{Adv}_{\Pi}^{\text{PRIV}}(A) = \Pr[K \xleftarrow{\$} \mathcal{K}\colon A^{\mathcal{E}_K(\cdot,\cdot,\cdot)} = 1] - \Pr[A^{\$(\cdot,\cdot,\cdot)} = 1]$ measure the advantage of adversary A. We name this notion IND\$-CPA, as before. Note the use of capital letters ($\mathbf{Adv}^{\text{PRIV}}$) for an AEAD-scheme and the use of little letters ($\mathbf{Adv}^{\text{priv}}$) for an AE-scheme.

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an AEAD-scheme and let A be an adversary having access to an oracle $\mathcal{E}_K(\cdot,\cdot,\cdot)$ for some key K. We say that A *forges* (for this key K and on some particular run) if A outputs $(N, H, \mathcal{C})$ where $\mathcal{D}_K^{N,H}(\mathcal{C}) \neq$ Invalid and A did not ask a query $\mathcal{E}_K^{N,H}(M)$ that resulted in a response $\mathcal{C}$. Define $\mathbf{Adv}_{\Pi}^{\text{AUTH}}(A)$ as the probability that A forges, where the probability is over $K \xleftarrow{\$} \mathcal{K}$ and the random coins, if any, of A. Note the use of capital letters ($\mathbf{Adv}^{\text{AUTH}}$) for an AEAD-scheme and the use of little letters ($\mathbf{Adv}^{\text{auth}}$) for an AE-scheme.

Informally, an AEAD-scheme Π is "secure" if $\mathbf{Adv}_{\Pi}^{\text{PRIV}}(A)$ and $\mathbf{Adv}_{\Pi}^{\text{AUTH}}(A)$ are "small" for any "reasonable" adversary A. Theorems make quantitative statements about the maximum possible value of $\mathbf{Adv}_{\Pi}^{\text{PRIV}}(A)$ and $\mathbf{Adv}_{\Pi}^{\text{AUTH}}(A)$ among adversaries A with specified resources.

Remarks. Our authenticity definition is very strong: the attack model is strong insofar as the adversary is allowed to manipulate both the nonce and the associated-data (subject to the constraint that no nonce is repeated), and the adversary's goal is modest insofar as it "gets credit" even for forgeries that use bizarre nonces and associated-data values, whether new or repetitions. In a real system, the message and the nonce will primarily be controlled by the sender (for example, the nonce may be a counter) while the associated-data will primarily be chosen by the sender and/or the receiver. Still, an adversary may be able to influence these values. For example, an adversary might force a nonce to get incremented by thwarting a transmission from reaching its destination. Or an adversary might induce the sender to utilize bogus associated-data by manipulating flows in an unauthenticated handshake. Allowing the adversary to manipulate N, H, and M, and giving the adversary credit for any new $(N, H, \mathcal{C})$, leads to a robust definition.

There are of course alternatives to our definition of AEAD security. In particular, one can use IND-CPA instead of IND\$-CPA, eliminate the nonce and make encryption probabilistic, or eliminate the nonce and make encryption stateful. All of these choices result in reasonable, weaker, definitions.

4 Nonce Stealing

We now consider a first suggestion, due to Nancy Cam-Winget and Jesse Walker [6], for incorporating associated-data into an AE-scheme. We call the method *nonce stealing*.

Suppose that the nonce in an AE-scheme is n bits but the application that uses this AE-scheme is content with a nonce of $\bar{n}$ bits, where $\bar{n} < n$. In such a case associated-data may be dropped into the unused $h = n - \bar{n}$ bits. For example, the nonce for the AE-scheme may be $n = 128$ bits but the application may use an $\bar{n} = 32$ bit counter for a nonce, leaving $h = 96$ bits for associated-data.

More formally, given AE-scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ having nonce space $\mathsf{Nonce} = \{0,1\}^n$ and given a parameter $\bar{n} \in [1..n-1]$ define the AEAD-scheme $\bar{\Pi} = \Pi|\bar{n} = (\bar{\mathcal{K}}, \bar{\mathcal{E}}, \bar{\mathcal{D}})$ having nonce space $\mathsf{Nonce} = \{0,1\}^{\bar{n}}$ and a space of associated-data $\mathsf{Header} = \{0,1\}^{n-\bar{n}}$ and where $\bar{\mathcal{K}} = \mathcal{K}$ and $\bar{\mathcal{E}}_K^{N,H}(M) = \bar{\mathcal{E}}_K^{N \,\|\, H}(M)$ and $\bar{\mathcal{D}}_K^{N,H}(\mathcal{C}) = \bar{\mathcal{D}}_K^{N \,\|\, H}(\mathcal{C})$. This formalization drops the nonce in front of the associated-data but other conventions are equally acceptable.

At first glance, nonce stealing might seem of limited use, because so few bits of associated-data can be accommodated. But often a few bytes is all that one needs (e.g., the associated-data may be a 32-bit IPv4 addresses, or a pair of such addresses). Nonce stealing is simple and adds essentially no overhead. Its security is captured by following theorem.

Theorem 1 [Security of nonce stealing] *Let Π be an AE-scheme with nonce space* $\mathsf{Nonce} = \{0,1\}^n$ *and let $\bar{n} \in [1..n]$. Then*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{PRIV}}_{\Pi|\bar{n}}(t,q,\sigma) &\le \mathbf{Adv}^{\mathrm{priv}}_{\Pi}(t_1,q,\sigma)\\ \mathbf{Adv}^{\mathrm{AUTH}}_{\Pi|\bar{n}}(t,q,\sigma,\varsigma) &\le \mathbf{Adv}^{\mathrm{auth}}_{\Pi}(t_2,q,\sigma,\varsigma)\end{aligned}$$

where $t = t_1 + O(\sigma + q)$ and $t = t_2 + O(\sigma + \varsigma + q)$. $\diamondsuit$

Proof: We begin with the privacy statement. Let A be an adversary that attacks the privacy of $\bar{\Pi} = \Pi|\bar{n}$. (Recall that all AEAD-adversaries are nonce-respecting.) We construct an adversary B that attacks the privacy of Π. Algorithm B runs A. When A makes an oracle query of (N_i, H_i, M_i) adversary B asks its own oracle $(N_i \,\|\, H_i, M_i)$, returning the result to A. When A halts, outputting a bit b, adversary B outputs the same bit b.

Because A is nonce-respecting it asks queries $(N_1, H_1, M_1), \ldots, (N_q, H_q, M_q)$ with distinct N_i values. As a consequence, the $N_i \,\|\, H_i$ values are also distinct, so B is nonce-respecting. Furthermore, $\mathbf{Adv}^{\mathrm{PRIV}}_{\bar{\Pi}}(A) = \Pr[A^{\mathcal{E}_K(\cdot,\cdot,\cdot)} = 1] - \Pr[A^{\$(\cdot,\cdot,\cdot)} = 1] = \Pr[B^{\mathcal{E}_K(\cdot,\cdot)} = 1] - \Pr[B^{\$(\cdot,\cdot)} = 1] = \mathbf{Adv}^{\mathrm{PRIV}}_{\Pi}(B)$. The first inequality follows.

Authenticity is similarly straightforward. We reuse the names A and B. Let A be a nonce-respecting adversary that attacks the authenticity of $\bar{\Pi}$. We construct an adversary B that attacks the authenticity of Π. Algorithm B runs A. When A makes an oracle query of (N_i, H_i, M_i) adversary B asks its own oracle $(N_i \,\|\, H_i, M_i)$, returning the result to A. When A outputs a forgery attempt $(N, H, \mathcal{C})$ adversary B outputs the forgery attempt $(N \,\|\, H,\ \mathcal{C})$. Then $\mathbf{Adv}^{\mathrm{AUTH}}_{\bar{\Pi}}(A) = \Pr[A^{\mathcal{E}_K(\cdot,\cdot,\cdot)} \text{ forges }] = \Pr[B^{\mathcal{E}_K(\cdot,\cdot)} \text{ forges }] = \mathbf{Adv}^{\mathrm{AUTH}}_{\Pi}(B)$. The second inequality follows. ▮

The possibility of nonce stealing provides yet another reason, besides those enumerated in [22], why an AE-scheme is best designed to employ an arbitrary nonce, as opposed to a counter or random value.

5 Ciphertext Translation

We now give a solution to the AEAD-problem that permits arbitrary associated-data. In particular, we show how to transform an AE-scheme Π into an AEAD-scheme $\ddot{\Pi} = \Pi \,\ddot{\cdot}\, F$ with the help of a function family $F\colon \mathcal{K}' \times \mathsf{Header} \to \{0,1\}^\tau$. We call the technique *ciphertext translation*.

We begin with some notation. When X and Y are binary strings of possibly different lengths define $X \mathbin{\hat{\oplus}} Y$ by prepending enough 0-padding to the shorter string to make it as long as the longer string, and then xor the two strings. For example, $0101001 \mathbin{\hat{\oplus}} 111 = 0101110$.

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an AE-scheme in which the length of any ciphertext is at least τ bits, for some constant τ. Let $\mathsf{Header} \subseteq \{0,1\}^*$ be a set of strings with a linear-time membership test and let $F\colon \mathcal{K}' \times \mathsf{Header} \to \{0,1\}^\tau$ be a function family. Then we define the AEAD-scheme $\ddot{\Pi} = (\ddot{\mathcal{K}}, \ddot{\mathcal{E}}, \ddot{\mathcal{D}}) = \Pi \,\ddot{\cdot}\, F$ as follows:

- $\ddot{\mathcal{K}} = \mathcal{K} \times \mathcal{K}'$.
- $\ddot{\mathcal{E}}^{N,H}_{K\,K'}(M) = \mathcal{E}^N_K(M)$ if $H = \varepsilon$, and $\ddot{\mathcal{E}}^{N,H}_{K\,K'}(M) = \mathcal{E}^N_K(M) \mathbin{\hat{\oplus}} F_{K'}(H)$ otherwise.
- $\ddot{\mathcal{D}}^{N,H}_{K\,K'}(\mathcal{C}) = \mathcal{D}^N_K(\mathcal{C})$ if $H = \varepsilon$, and $\ddot{\mathcal{D}}^{N,H}_{K\,K'}(\mathcal{C}) = \mathcal{D}^N_K(\mathcal{C} \mathbin{\hat{\oplus}} F_{K'}(H))$ otherwise.

That is, assuming $H \ne \varepsilon$, take the associated-data H, compute from it $\Delta = F_{K'}(H)$, and encrypt M by computing its ciphertext without regards to H, and then xoring in Δ to the last τ bits.

For more concise notation we sometimes write $\ddot{\mathcal{E}}^{N,H}_{KK'}(M) = \mathcal{E}^N_K(M) \hat{\oplus} F^*_{K'}(H)$ and $\ddot{\mathcal{D}}^{N,H}_{KK'}(\mathcal{C}) = \mathcal{D}^N_K(\mathcal{C} \hat{\oplus} F^*_{K'}(H))$ where $F^*_{K'}(H) = \varepsilon$ if $H = \varepsilon$, and $F^*_{K'}(H) = F_{K'}(H)$ otherwise.

Remarks. Ciphertext translation has the following pleasant properties: (1) the method applies to any AE-scheme Π; (2) it is a proper extension of the AE-scheme in the sense that $\ddot{\mathcal{E}}^{N,\varepsilon}_{KK'}(M) = \mathcal{E}^N_K(M)$; (3) as such, no overhead is added to an AE-scheme when associated-data is *not* used; (4) if H is static over the course of a session (or even over the course of several messages) the value $\Delta = F_{K'}(H)$ may be precomputed, minimizing the per-message overhead to authenticate the associated data; (5) because F is a parameter we can instantiate it in whatever way seems most appropriate to match the characteristics of $\mathcal{E}$. Ciphertext translation also has the following unpleasant property: it uses a new key, K', different from that used by the underlying AD-scheme Π. This disadvantage will be erased in Section 6 for the specific case of $\Pi = \mathrm{OCB}$ and $F = \mathrm{PMAC}$.

It is not important which bits of the ciphertext get modified by $\Delta = F_{K'}(H)$; the last bits were chosen for concreteness and because one may wish, as in [22], to think of the last τ bits of ciphertext as an authenticity-ensuring tag.

The double-dot notation $\ddot{\Pi} = \Pi \mathbin{\ddot{\cdot}} F$ serves as a gentle reminder that two keys are used in the construction, K and K'. We will later consider the analogous transformation where a single key is used, denoting this $\dot{\Pi} = \Pi \cdot F$.

Security of ciphertext translation. Let Π is a secure AE-scheme. We give two sufficient conditions on the hash function F in order that $\Pi \mathbin{\ddot{\cdot}} F$ will be a secure AEAD-scheme. One is that F is a good AXU hash-function; the other is that F is a good PRF.

Theorem 2 [Security of ciphertext translation] *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an AE-scheme where each ciphertext is at least τ bits long. Let $F\colon \mathcal{K}' \times \mathsf{Header} \to \{0,1\}^\tau$ be a function family. Then*

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{PRIV}}_{\Pi \ddot{\cdot} F}(t, q, \sigma) &\le \mathbf{Adv}^{\mathrm{priv}}_{\Pi}(t_1, q, \sigma) \\
\mathbf{Adv}^{\mathrm{AUTH}}_{\Pi \ddot{\cdot} F}(t, q, \sigma, \hat{\sigma}, \varsigma) &\le \mathbf{Adv}^{\mathrm{auth}}_{\Pi}(t_2, q, \sigma, \varsigma) + \mathbf{Adv}^{\mathrm{priv}}_{\Pi}(t_3, q, \sigma) + \mathbf{Adv}^{\mathrm{prf}}_{F}(t_4, 2, \hat{\sigma} + \varsigma) + 2^{-\tau} \\
\mathbf{Adv}^{\mathrm{AUTH}}_{\Pi \ddot{\cdot} F}(t, q, \sigma, \hat{\sigma}, \varsigma) &\le \mathbf{Adv}^{\mathrm{auth}}_{\Pi}(t_5, q, \sigma, \varsigma) + \mathbf{Adv}^{\mathrm{priv}}_{\Pi}(t_6, q, \sigma) + \mathbf{Adv}^{\mathrm{axu}}_{F}(\hat{\sigma} + \varsigma)
\end{aligned}$$

where $t = t_1 + \mathrm{Time}_F(q, \sigma) + O(\sigma + q)$ and $t = t_2 + t_3 + t_4 + 2\ \mathrm{Time}_F(q+1, \sigma + \varsigma) + O(\sigma + \varsigma + q)$ and $t = t_5 + t_6 + 2\ \mathrm{Time}_F(q+1, \sigma + \varsigma) + O(\sigma + \varsigma + q)$. $\diamondsuit$

Proof: We begin with the privacy claim. Let A be an adversary that attacks the privacy of $\Pi \mathbin{\ddot{\cdot}} F = (\ddot{\mathcal{K}}, \ddot{\mathcal{E}}, \ddot{\mathcal{D}})$. Assume that A runs in time at most t and asks at most q queries, these totaling at most σ bits. We construct an adversary B that attacks the privacy of Π. Adversary B works as follows. First B chooses a random $K' \xleftarrow{\$} \mathcal{K}'$. Then B runs A. When A makes its ith oracle query, (N_i, H_i, M_i), adversary B makes query (N_i, M_i) to it own oracle. Adversary B receives a response $\mathcal{C}_i$, computes $\Delta_i = F^*_{K'}(H_i)$, and provides to A the ciphertext $\mathcal{C}_i \hat{\oplus} \Delta_i$. After A makes all of its oracle queries (and B makes the correspond queries) adversary A outputs a bit b. At that point adversary B outputs the same bit b. Note that B runs in time at most $t_1 = t + \mathrm{Time}_F(q, \sigma) + O(\sigma + q)$ and asks at most q queries and these total at most σ bits. Also note that B is nonce-respecting since A is. Finally, since B perfectly simulates the native environment for A we have that $\mathbf{Adv}^{\mathrm{PRIV}}_{\Pi[F]}(A) = \mathbf{Adv}^{\mathrm{priv}}_{\Pi}(B)$, establishing the first inequality.

We now prove the first authenticity claim. Reusing the name, let A be an adversary that attacks the authenticity of $\Pi \mathbin{\ddot{\cdot}} F = (\ddot{\mathcal{K}}, \ddot{\mathcal{E}}, \ddot{\mathcal{D}})$. Assume that A runs in time at most t and asks at most q queries, the longest of at most $\hat{\sigma}$ bits and the queries totaling at most σ bits, and then A outputs a string having at most ς bits. We construct an adversary $A_{\rm auth}$ that attacks the authenticity of Π and an adversary $A_{\rm priv}$ that attacks the privacy of Π and an adversary $A_{\rm prf}$ that attacks F as a PRF.

Adversary $A_{\rm auth}$ chooses a random $K' \xleftarrow{\$} \mathcal{K}'$ then runs A. When A makes its ith oracle query, (N_i, H_i, M_i), adversary $A_{\rm auth}$ makes the query (N_i, M_i) to it own oracle, getting back a response $\mathcal{C}_i$. Adversary $A_{\rm auth}$ computes $\Delta_i = F^*_{K'}(H_i)$ and provides to A the ciphertext $\ddot{\mathcal{C}}_i = \mathcal{C}_i \mathbin{\hat{\oplus}} \Delta_i$. After A makes its oracle queries (and $A_{\rm auth}$ makes the corresponding oracle queries) it outputs its forgery attempt $(N, H, \mathcal{C})$. At that point $A_{\rm auth}$ computes $\Delta = F^*_{K'}(H)$ and $\mathcal{C}^* = \mathcal{C} \mathbin{\hat{\oplus}} \Delta$. Adversary $A_{\rm auth}$ outputs its own forgery attempt of $(N, \mathcal{C}^*)$. Note that $A_{\rm auth}$ runs in $t_2 = t + \mathrm{Time}_F(q+1, \sigma+\varsigma) + O(\sigma + \varsigma + q)$ time, makes at most q queries, the longest of at most $\hat{\sigma}$ bits, and these queries total at most σ bits. Its forgery attempt has at most ς bits.

Adversary $A_{\rm auth}$ provides adversary A a perfect simulation of the environment that defines the advantage of A in attacking $\Pi \mathbin{\ddot{}} F$. Still the advantage of $A_{\rm auth}$ may be less than that of A because it is possible for A to forge (in an execution under $A_{\rm auth}$) when $A_{\rm auth}$ does not forge (in that execution). This happens iff A's forgery attempt $(N, H, \mathcal{C})$ is new for it, $(N, H, \mathcal{C}) \notin \{(N_1, H_1, \mathcal{C}_1), \ldots, (N_q, H_q, \mathcal{C}_q)\}$, but $(N, \mathcal{C}^*)$ is not new for $A_{\rm auth}$, namely, $(N, \mathcal{C} \mathbin{\hat{\oplus}} F^*_{K'}(H)) = (N_i, \mathcal{C}_i \mathbin{\hat{\oplus}} F^*_{K'}(H))$ for some $i \in [1..q]$. To analyze the situation let $\mathsf{collides}$ be the event that A makes a forgery attempt $(N, C \parallel T)$ after asking $(N_1, H_1, M_1), \ldots, (N_q, H_q, M_q)$ and getting responses $C_1 \| T_1, \ldots, C_q \| T_q$, where $N = N_i$ for some $i \in [1..q]$ and $C = C_i$ but $T \mathbin{\hat{\oplus}} F^*_{K'}(H)) = T_i \mathbin{\hat{\oplus}} F^*_{K'}(H_i))$. The last condition is the same as: $T \oplus T_i = F_{K'}(H) \oplus F_{K'}(H_i)$ if $H \neq \varepsilon$ and $H' \neq \varepsilon$; and $T \oplus T_i = F_{K'}(H)$ if $H \neq \varepsilon$ and $H_i = \varepsilon$; and $T \oplus T_i = F_{K'}(H_i)$ if $H = \varepsilon$ and $H_i \neq \varepsilon$; and $T = T_i$ if $H = \varepsilon$ and $H_i = \varepsilon$. The last possibility would mean that $(N, H, \mathcal{C})$ is not a forgery and thus it can be ignored. We have that

$$\mathbf{Adv}^{\rm AUTH}_{\Pi \mathbin{\ddot{}} F}(A) \;=\; \Pr[A^{\Pi \mathbin{\ddot{}} F} \text{ forges }] \;\leq\; \Pr[A^{\Pi}_{\rm auth} \text{ forges }] + \Pr[A^{\Pi \mathbin{\ddot{}} F} \text{ collides}]$$

By hybrid argument we bound $\Pr[A^{\Pi \mathbin{\ddot{}} F}\text{collides}]$ in terms of $\mathbf{Adv}^{\rm priv}_{\Pi}(\cdot)$ and $\mathbf{Adv}^{\rm prf}_{F}(\cdot)$ values. Note

$$\begin{aligned} \Pr[A^{\Pi \mathbin{\ddot{}} F}\text{collides}] \;=\;& (\Pr[A^{\Pi \mathbin{\ddot{}} F}\text{collides}] - \Pr[A^{\$ \mathbin{\ddot{}} F}\text{collides}]) + \\ & (\Pr[A^{\$ \mathbin{\ddot{}} F}\text{collides}] - \Pr[A^{\$ \mathbin{\ddot{}} R}\text{collides}]) + \\ & \Pr[A^{\$ \mathbin{\ddot{}} R}\text{collides}] \end{aligned} \tag{1}$$

where \$ is the oracle that, on input (N, H, M), returns $\ell(|M|)$ random bits (for ℓ the length function of the encryption scheme), and where R is selected from $\mathrm{Rand}(\{0,1\}^*, \tau)$. We now claim:

$$\Pr[A^{\Pi \mathbin{\ddot{}} F}\text{collides}] - \Pr[A^{\$ \mathbin{\ddot{}} F}\text{collides}] \leq \mathbf{Adv}^{\rm priv}_{\Pi}(t_3, q, \sigma, \hat{\sigma}) \tag{2}$$

$$\Pr[A^{\$ \mathbin{\ddot{}} F}\text{collides}] - \Pr[A^{\$ \mathbin{\ddot{}} R}\text{collides}] \leq \mathbf{Adv}^{\rm prf}_{\Pi}(t_4, 2, \hat{\sigma} + \varsigma) \tag{3}$$

$$\Pr[A^{\$ \mathbin{\ddot{}} R}\text{collides}] \leq 2^{-\tau} \tag{4}$$

for values t_3 and t_4 yet to be specified.

To justify Equation (2) and compute t_3 construct $A_{\rm priv}$ as follows. It begins by choosing $K \xleftarrow{\$} \mathcal{K}'$ and then runs adversary A. When A makes its ith oracle query, (N_i, H_i, M_i), adversary $A_{\rm priv}$ makes its own oracle call of (N_i, M_i), getting a response $\mathcal{C}_i$. Adversary $A_{\rm priv}$ then computes $\Delta_i = F^*_{K'}(H_i)$ and returns to A the value $\ddot{\mathcal{C}}_i = \mathcal{C}_i \mathbin{\hat{\oplus}} \Delta_i$. When A halts, outputting a forgery attempt $(N, H, \mathcal{C})$, adversary A_{priv} computes if event $\mathsf{collides}$ has occurred: it checks if $(N, H, \mathcal{C})$ is new for A but $(N, \mathcal{C} \mathbin{\hat{\oplus}} F^*_{K'}(H))$ is not new for A_{priv}. If so then A_{priv} outputs 1; otherwise, it outputs 0. The running time of A_{priv} is $t_3 = t + \mathrm{Time}_F(q+1, \sigma + \varsigma) + O(\sigma + \varsigma + q)$, it asks at most q queries and these total at most σ bits. Also, $\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A_{\mathrm{priv}}) = \Pr[A^{\Pi \cdot\cdot F}\mathsf{collides}] - \Pr[A^{\$ \cdot\cdot F}\mathsf{collides}]$.

To justify Equation (3) and compute t_4 construct adversary A_{prf} as follows. When A makes its ith oracle query, (N_i, H_i, M_i), adversary A_{prf} computes $\ddot{\mathcal{C}}_i \xleftarrow{\$} \{0,1\}^{\ell(|M_i|)}$ where ℓ is the length function of the encryption scheme Π. AdversaryA_{prf} then answers A's query with $\ddot{\mathcal{C}}_i$. When A outputs a forgery attempt $(N, H, \mathcal{C})$ and halts, adversary A_{prf} asks its oracle H, getting a response Δ, and then it asks its oracle H_i, getting a response Δ_i. Adversary A_{prf} then computes if event $\mathsf{collides}$ has occurred: it checks if $(N, H, \mathcal{C})$ is new for A but $(N, \mathcal{C} \mathbin{\hat{\oplus}} \Delta)$ is not new for A_{prf}. If so then A_{prf} outputs the bit 1; otherwise, it outputs 0. The running time of A_{prf} is $t_4 = t + O(\sigma + \varsigma + q)$, it asks 2 queries and these total at most $\hat{\sigma} + \varsigma$ bits. Also, by definition, $\mathbf{Adv}^{\mathrm{prf}}_{F}(A_{\mathrm{prf}}) = \Pr[A^{\$ \cdot\cdot F}\mathsf{collides}] - \Pr[A^{\$ \cdot\cdot R}\mathsf{collides}]$.

We now verify Equation (4). For A to produce a collision when interacting with a $\$ \cdot\cdot R$ oracle it must produce T_i, H_i, T, H such that $T \oplus T_i = R_{K'}(H) \oplus R_{K'}(H_i)$ if $H \neq \varepsilon$ and $H' \neq \varepsilon$; or $T \oplus T_i = R_{K'}(H)$ if $H \neq \varepsilon$ and $H_i = \varepsilon$; or $T \oplus T_i = R_{K'}(H_i)$ if $H = \varepsilon$ and $H_i \neq \varepsilon$. Equivalently, A succeeds in making a collision if, given no queries, it can predict of its oracle $R_{K'}(\cdot)$ the value of $R_{K'}(H) \oplus R_{K'}(H_i)$ for chosen and distinct H_i, H; or if it can predict of its oracle $R_{K'}(\cdot)$ the value of $R_{K'}(H)$ for a chosen H. This happens with probability $2^{-\tau}$. We have now shown the second inequality of the theorem.

For the final inequality of the theorem we proceed as above until the hybrid decomposition. There we use instead that $\Pr[A^{\Pi \cdot\cdot F}\mathsf{collides}] = (\Pr[A^{\Pi \cdot\cdot F}\mathsf{collides}] - \Pr[A^{\$ \cdot\cdot F}\mathsf{collides}]) + \Pr[A^{\$ \cdot\cdot F}\mathsf{collides}]$. The first difference is bounded as before. We have left to show $\Pr[A^{\$ \cdot\cdot F}\mathsf{collides}] \leq \mathbf{Adv}^{\mathrm{axu}}_{F}(\bar{\sigma} + \varsigma)$. As before, A, attacking $\$ \cdot\cdot F$, gets no information about F as it makes oracle queries. Still it must produce T_i, H_i, T, H such that $T \oplus T_i = F_{K'}(H) \oplus R_{K'}(H_i)$ if $H \neq \varepsilon$ and $H' \neq \varepsilon$; or $T \oplus T_i = F_{K'}(H)$ if $H \neq \varepsilon$ and $H_i = \varepsilon$; or $T \oplus T_i = F_{K'}(H_i)$ if $H = \varepsilon$ and $H_i \neq \varepsilon$. Equivalently, A succeeds in making a collision if, given no queries, it can predict of its oracle $F_{K'}(\cdot)$ the value of $F_{K'}(H) \oplus F_{K'}(H_i)$ for chosen and distinct H_i, H; or if it can predict of its oracle $F_{K'}(\cdot)$ the value of $F_{K'}(H)$ for a chosen H. From the definition, this happens with probability at most $\mathbf{Adv}^{\mathrm{axu}}_{F}(\hat{\sigma} + \varsigma)$. We have finished the proof. ∎

AXU HASH-FUNCTION VS. PRF. According to Theorem 2 the function family F used for ciphertext translation needs only to satisfy a verifiable, probabilistic criteria (the third inequality of the theorem). Still, there are some advantages to using a function family F secure according to the complexity-theoretic criterion of being a good PRF (the second inequality of the theorem). One advantage is that using a PRF for F facilitates using $\Pi \cdot\cdot F$ as a deterministic MAC: let the message M to encrypt be the empty string, let the nonce be $N = \mathbf{0}$, and let the message that one wants to MAC be the associated-data H. This addresses a question posed by Rivest, who asked if OCB can be used in some simple way to give a MAC [20]. (Note that trying to use OCB [22] or IAPM [15] as a MAC by sending only the tag block does not work.) When building an AEAD-scheme based on an AE-scheme like those in [15, 22] a more significant advantage of using a PRF for F is that it leads to a simpler algorithm than one would get by choosing any known universal hash function (e.g., one based on polynomial evaluation in a finite field). We expand on this in the following section.

6 Single-Key OCB · PMAC

When $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is an AE-scheme we defined ciphertext translation, $\ddot{\Pi} = \Pi \,\ddot{\cdot}\, F = (\ddot{\mathcal{K}}, \ddot{\mathcal{E}}, \ddot{\mathcal{D}})$, to use two different keys, K and K'. Using two keys is necessary insofar as the analogous single-key construction $\dot{\Pi} = \Pi \cdot F = (\dot{\mathcal{K}}, \dot{\mathcal{E}}, \dot{\mathcal{D}})$ where $\dot{\mathcal{K}} = \mathcal{K}$ and $\dot{\mathcal{E}}_K^{N,H}(M) = \mathcal{E}_K^N(M) \,\hat{\oplus}\, F_K^*(H)$ and $\dot{\mathcal{D}}_K^{N,H}(\mathcal{C}) = \mathcal{D}_K^N(\mathcal{C}) \,\hat{\oplus}\, F_K^*(H)$ certainly will not, in general, work; it is easy to exhibit a counterexample to demonstrate this. Nonetheless, we single out a useful case where the single-key definition *does* work: when coupling $\Pi = \text{OCB}$ [22] with $F = \text{PMAC}$ [5].

Throughout this section the following holds. The block length n is fixed, as is an underlying block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and tag length $\tau \in [1..n]$. When combining OCB and PMAC we use the same block cipher E and tag length τ for both algorithms. We write $\dot{\text{OCB}} = (\dot{\mathcal{K}}, \dot{\mathcal{E}}, \dot{\mathcal{D}}) = \text{OCB} \cdot \text{PMAC}$ and $\ddot{\text{OCB}} = (\ddot{\mathcal{K}}, \ddot{\mathcal{E}}, \ddot{\mathcal{D}}) = \text{OCB} \,\ddot{\cdot}\, \text{PMAC}$. String lengths are measured in n-bit blocks. For convenience, we recall the definitions of OCB and PMAC in Appendix A.

To be explicit, we are constructing the AEAD-scheme $\dot{\text{OCB}} = \text{OCB} \cdot \text{PMAC} = (\dot{\mathcal{K}}, \dot{\mathcal{E}}, \dot{\mathcal{D}})$ from $\text{OCB} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and $\text{PMAC}\colon \mathcal{K} \times \{0,1\}^* \to \{0,1\}^\tau$ in the following way. The nonce space for $\dot{\text{OCB}}$ remains $\mathsf{Nonce} = \{0,1\}^n$. The key space remains $\dot{\mathcal{K}} = \mathcal{K}$. The space of associated-data is $\mathsf{Header} = \{0,1\}^*$. Encryption is defined by $\dot{\mathcal{E}}_K^{N,H}(M) = \mathcal{E}_K^N(M) \,\hat{\oplus}\, \text{PMAC}_K^*(H)$. Decryption is done according to $\dot{\mathcal{D}}_K^{N,H}(\mathcal{C}) = \mathcal{D}_K^N(\mathcal{C}) \,\hat{\oplus}\, \text{PMAC}_K^*(H)$. Recall that under the notation introduced already, $\text{PMAC}_K^*(H)$ is the empty string if $H = \varepsilon$ and it is $\text{PMAC}_K(H)$ otherwise.

Interference. To prove the security of $\dot{\text{OCB}}$ one might hope to establish that there is no significant interaction between what goes on in OCB and in PMAC when the two algorithms use a common key K. One would aim to show that all the internal values generated by the algorithms will, almost certainly, be distinct. But an inspection of OCB and PMAC reveals that this is simply not true. First there is the common definition of the variable $L = E_K(0^n)$ used by the algorithms. Worse still is the fact that OCB defines an internal variable $R = E_K(N \oplus L)$ while PMAC defines an internal variable $Y[1] = E_K(M[1] \oplus L)$. Both N and $M[1]$ are under the adversary's control, and so it can force OCB's R and PMAC's $Y[1]$ to take on identical values. Though it is not clear how an adversary can exploit such interaction, its possibility would normally spell serious trouble: either the joint scheme really will be breakable, or it won't be breakable but the prospects for a reasonable proof will be dim.

We prove that, despite the interactions described above, OCB · PMAC is secure. We manage to prove this without opening up the (already complex) proofs for OCB and PMAC—something that might have seemed unavoidable in the presence of such cross-scheme interactions.

Security of $\dot{\text{OCB}}$. The main result of this section is a quantitative bound on the security (privacy and authenticity) of $\dot{\text{OCB}} = \text{OCB} \cdot \text{PMAC}$.

Theorem 3 [Security of $\dot{\text{OCB}}$]

$$\begin{aligned}
\mathbf{Adv}^{\text{PRIV}}_{\dot{\text{OCB}}[\text{Perm}(n),\tau]}(q,\sigma) &\leq 9.5\,\sigma_1^2 \,/\, 2^n \\
\mathbf{Adv}^{\text{AUTH}}_{\dot{\text{OCB}}[\text{Perm}(n),\tau]}(q,\sigma,\varsigma) &\leq 13\,\sigma_2^2 \,/\, 2^n + 2^{-\tau}
\end{aligned}$$

where $\sigma_1 = \sigma + q + 3$ *and* $\sigma_2 = \sigma + q + 5\varsigma + 11$. ◇

We have stated the theorem for the information-theoretic setting, where one is using a random permutation instead of a "real" block cipher. Passing to the complexity-theoretic setting is standard. The needed complexity-theoretic assumption is a pseudorandom permutation for privacy, and a strong pseudorandom permutation for authenticity. Since our target is Theorem 3 we henceforth understand OCB, O$\dot{\text{C}}$B, O$\ddot{\text{C}}$B, and PMAC to all be taken over the block cipher $\text{Perm}(n)$, sometimes omitting this from the notation.

PROOF IDEA. From [5, 22] and Theorem 2 we know right off that $\text{O}\ddot{\text{C}}\text{B} = \text{OCB} \mathbin{\ddot{\cdot}} \text{PMAC} = (\ddot{\mathcal{K}}, \ddot{\mathcal{E}}, \ddot{\mathcal{D}})$ is secure. We would like to show that $\text{O}\dot{\text{C}}\text{B} = \text{OCB} \cdot \text{PMAC} = (\dot{\mathcal{K}}, \dot{\mathcal{E}}, \dot{\mathcal{D}})$ is secure, too. So, at least for privacy, it would be enough to show that reasonable adversaries can't do a good job at distinguishing an oracle for $\dot{\mathcal{E}}_\pi$ (for $\pi \stackrel{\$}{\leftarrow} \text{Perm}(n)$) from an oracle for $\ddot{\mathcal{E}}_{\pi,\pi'}$ (for $\pi, \pi' \stackrel{\$}{\leftarrow} \text{Perm}(n)$). We show this by carefully expanding the adversary's capabilities when attacking $\dot{\mathcal{E}}$ or $\ddot{\mathcal{E}}$. We define two oracles, $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$. The oracles begin by choosing random permutations π and (π, π'), respectively. Each oracle then accepts ten types of queries. The oracles have been designed so that, using $\dot{\mathcal{O}}$ an adversary can compute $\dot{\mathcal{E}}_\pi$ for a random π; and using $\ddot{\mathcal{O}}$ an adversary can compute $\dot{\mathcal{E}}_{\pi,\pi'}$ for a random π, π'. Then we show that oracles $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$ are themselves adversarially indistinguishable.

Some subtleties arise when trying to work out this approach. One is that the oracles $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$ must enable the adversary to compute $\dot{\mathcal{D}}$ and $\ddot{\mathcal{D}}$ as well as $\dot{\mathcal{E}}$ and $\ddot{\mathcal{E}}$. This is necessary to ensure that indistinguishability of $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$ implies authenticity of O$\dot{\text{C}}$B. The oracles themselves must be made simple enough to reason about, powerful enough that an adversary can compute what is needed, but not so powerful that the oracles become distinguishable.

ORACLES $\dot{\mathcal{O}}$ AND $\ddot{\mathcal{O}}$, AND VALID QUERIES TO THEM. Oracle $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$ are defined in Figure 1. The description there omits checks on the validity of oracle queries, which we now explain. An oracle query $Q = (ty, N, i, M)$ that follows a sequence of oracle queries $\mathcal{Q} = (Q_1, \ldots, Q_r)$ with responses $\mathcal{Z} = (Z_1, \ldots, Z_r)$ is said to be *valid* if all of the following hold:

(V0) $ty \in [0..9]$ and $N \in \{0,1\}^n$ and $i \in [1..2^{n-1} - 1]$ and $M \in \{0,1\}^*$.

(V1) If $ty \in [1..4]$ then $(ty, N, i, M) \notin \mathcal{Q}$; if $ty = 5$ then $(5, \cdot, i, M) \notin \mathcal{Q}$ (for any value filling in the dot, language that we henceforth omit); if $ty = 6$ then $(6, \cdot, \cdot, M) \notin \mathcal{Q}$; if $ty = 7$ then $(7, \cdot, \cdot, M) \notin \mathcal{Q}$; if $ty = 8$ then $(8, N, \cdot, M) \notin \mathcal{Q}$; and if $ty = 9$ then $(9, N, \cdot, M) \notin \mathcal{Q}$.

(V2) If $ty = 1$ then there is no $Q_s = (4, N, i, \cdot) \in \mathcal{Q}$ that returned $Z_s = M$; and if $ty = 4$ then there was no $Q_s = (1, N, i, \cdot) \in \mathcal{Q}$ that returned $Z_s = M$.

(V3) If $ty = 1$ then $(3, N, i, \cdot) \notin \mathcal{Q}$ and if $ty = 3$ then $(1, N, i, \cdot) \notin \mathcal{Q}$.

(V4) If $ty = 5$ then $i \neq 1$.

(V5) If $ty = 6$ then $M \neq 0^n$.

(V6) There is no $(0, \cdot, \cdot, \cdot) \notin \mathcal{Q}$, and if $ty = 0$ then there is some $(\cdot, N, \cdot, \cdot) \in \mathcal{Q}$.

We say that a sequence of queries $\mathcal{Q} = (Q_1, \ldots, Q_q)$ and their responses $\mathcal{Z} = (Z_1, \ldots, Z_q)$ is valid if each query Q_s is valid given the earlier queries $(Q_1, \ldots, Q_{s-1})$ and their responses $(Z_1, \ldots, Z_{s-1})$. We also demand that the last query $Q_q = (0, \cdot, \cdot, \cdot)$ be of type 0.

A query Q following $(\mathcal{Q}, \mathcal{Z})$ is invalid if it is not valid. We define $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$ to return a random n-bit string in response to any invalid query. Since the validity condition is easily checked by an adversary one can assume without loss of generality that adversaries do not ask invalid queries.

Let us sketch the meaning of the validity conditions. Condition V0 says not to consider ill-formed queries or queries with $i = 0$. It also guarantees that the first bit of i (when regarded as an n-bit string) is 0. Conditions V1 and V2 demand that an adversary not ask a query that it already knows the answer to. Condition V3 prohibits an adversary from asking both $\pi(iL \oplus R)] \oplus iL \oplus R$ and $\pi(iL \oplus R)$. Condition V4 keeps the adversary from trivially learning an R-value, while condition V5 keeps the adversary from trivially learning L. Query type 0 lets the adversary add $iL \oplus R$ to a value of its choice, but condition V6 says that it can only do this once and it must have already asked a query that used the N that gave rise to this R.

Oracle $\dot{\mathcal{O}}$	Oracle $\ddot{\mathcal{O}}$
Initialization	**Initialization**
$\pi \xleftarrow{\$} \mathrm{Perm}(n)$; $L \leftarrow \pi(0^n)$	$\pi \xleftarrow{\$} \mathrm{Perm}(n)$; $L \leftarrow \pi(0^n)$
	$\pi' \xleftarrow{\$} \mathrm{Perm}(n)$; $L' \leftarrow \pi(0^n)$
To respond to query $\dot{\mathcal{O}}(ty, N, i, M)$	**To respond to query** $\ddot{\mathcal{O}}(ty, N, i, M)$
$R \leftarrow \pi(N \oplus L)$	$R \leftarrow \pi(N \oplus L)$; $R' \leftarrow \pi(N \oplus L')$
case ty **of**	**case** ty **of**
0: **return** $M \oplus iL \oplus R$	0: **return** $M \oplus iL \oplus R$
1: **return** $\pi(M \oplus iL \oplus R) \oplus iL \oplus R$	1: **return** $\pi(M \oplus iL \oplus R) \oplus iL \oplus R$
2: **return** $\pi(M \oplus iL \oplus R \oplus L \cdot \mathtt{x}^{-1})$	2: **return** $\pi(M \oplus iL \oplus R \oplus L \cdot \mathtt{x}^{-1})$
3: **return** $\pi(M \oplus iL \oplus R)$	3: **return** $\pi(M \oplus iL \oplus R)$
4: **return** $\pi^{-1}(C \oplus iL \oplus R) \oplus iL \oplus R$	4: **return** $\pi^{-1}(C \oplus iL \oplus R) \oplus iL \oplus R$
5: **return** $\pi(M \oplus iL)$ $\quad //i \neq 1$	5: **return** $\pi'(M \oplus iL')$ $\quad //i \neq 1$
6: **return** $\pi(M)$ $\quad //M \neq 0^n$	6: **return** $\pi'(M)$ $\quad //M \neq 0^n$
7: **return** $\pi(M \oplus L \cdot \mathtt{x}^{-1})$	7: **return** $\pi'(M \oplus L' \cdot \mathtt{x}^{-1})$
8: **return** $\pi(M \oplus R)$	8: **return** $\pi'(M \oplus R')$
9: **return** $\pi(M \oplus R \oplus L \cdot \mathtt{x}^{-1})$	9: **return** $\pi'(M \oplus R' \oplus L' \cdot \mathtt{x}^{-1})$

Figure 1: Oracles $\dot{\mathcal{O}}$ (left) and $\ddot{\mathcal{O}}$ (right).

Closeness of $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$. For oracles $\mathcal{X}$ and $\mathcal{Y}$ define $\mathbf{Adv}^{\mathrm{dist}}_{\mathcal{X},\mathcal{Y}}(A) = \left|\Pr[A^{\mathcal{X}} = 1] - \Pr[A^{\mathcal{Y}} = 1]\right|$. This immediately gives the corresponding resource-bounded notion, as explained at the end of Section 2. The main technical lemma that we need can now be stated. It's proof is given in Appendix B. Recall that n has been fixed and oracles $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$ silently depend on this parameter.

Lemma 4 [Indistinguishability of $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$] $\quad \mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{O}},\ddot{\mathcal{O}}}(q) \;\le\; 8q^2/2^n$ $\quad\diamondsuit$

Relating $\dot{\mathrm{OCB}}$ and $\ddot{\mathrm{OCB}}$ security. We have defined $\dot{\mathcal{O}}$ in such a way that having an oracle for $\dot{\mathcal{O}}_\pi$ lets one compute $\dot{\mathcal{E}}_\pi$ on any number of points and lets one compute $\dot{\mathcal{D}}_\pi$ on one point. Similarly, we have defined $\ddot{\mathcal{O}}$ in such a way that having an oracle for $\ddot{\mathcal{O}}_{\pi,\pi'}$ lets one compute $\ddot{\mathcal{E}}_{\pi,\pi'}$ on any number of points and lets one compute $\ddot{\mathcal{D}}_{\pi,\pi'}$ on one point. We use this to relate the security of $\dot{\mathrm{OCB}}$ to the security of $\ddot{\mathrm{OCB}}$ and the distinguishability of $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$. We have the following result.

Lemma 5 [Relating $\dot{\mathrm{OCB}}$ to $\ddot{\mathrm{OCB}}$ and the distinguishability of $\dot{\mathcal{O}}$, $\ddot{\mathcal{O}}$]

$$\mathbf{Adv}^{\mathrm{PRIV}}_{\dot{\mathrm{OCB}}}(q,\sigma) \;\le\; \mathbf{Adv}^{\mathrm{PRIV}}_{\ddot{\mathrm{OCB}}}(q,\sigma) + \mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{O}},\ddot{\mathcal{O}}}(\sigma)$$

$$\mathbf{Adv}^{\mathrm{AUTH}}_{\dot{\mathrm{OCB}}}(q,\sigma,\varsigma) \;\le\; \mathbf{Adv}^{\mathrm{AUTH}}_{\ddot{\mathrm{OCB}}}(q,\sigma,\varsigma) + \mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{O}},\ddot{\mathcal{O}}}(\sigma+\varsigma)$$

Proof of Lemma 5: We start with the privacy claim. Let A be an adversary that asks at most q queries, these totaling at most σ blocks, and suppose that A distinguishes $\dot{\mathcal{E}}$ and $\ddot{\mathcal{E}}$ with advantage $\delta = \mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{E}},\ddot{\mathcal{E}}}(A)$. Then there exists another adversary, B, that asks at most σ queries and distinguishes $\dot{\mathcal{O}}$ and $\ddot{\mathcal{O}}$ with identical advantage δ. This adversary is constructed in Figure 2. We leave it for the reader to check that B with an $\dot{\mathcal{O}}$-oracle perfectly simulates for A an $\dot{\mathcal{E}}$-oracle, and B with an $\ddot{\mathcal{O}}$-oracle perfectly simulates for A an $\ddot{\mathcal{E}}$-oracle. From this it follows that $\mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{E}},\ddot{\mathcal{E}}}(q,\sigma) \le \mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{O}},\ddot{\mathcal{O}}}(\sigma)$.

Let \$ be the oracle that, on input (N, H, M), returns $|M| + \tau$ random bits. Then $\mathbf{Adv}^{\mathrm{PRIV}}_{\dot{\mathrm{OCB}}}(q,\sigma) = \mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{E}},\$}(q,\sigma) \le \mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{E}},\ddot{\mathcal{E}}}(q,\sigma) + \mathbf{Adv}^{\mathrm{dist}}_{\ddot{\mathcal{E}},\$}(q,\sigma) \le \mathbf{Adv}^{\mathrm{PRIV}}_{\ddot{\mathrm{OCB}}}(q,\sigma) + \mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{E}},\ddot{\mathcal{E}}}(q,\sigma) \le \mathbf{Adv}^{\mathrm{PRIV}}_{\ddot{\mathrm{OCB}}}(q,\sigma) +$

Define $g(ty, N, i, M)$ as follows:

If there has already been a call of $g(ty, N, i, M)$, return the value previously returned.

If $ty = 4$ and there has already been a call of $g(1, N, i, M')$ that returned M, return M'.

If $ty = 1$ and there has already been a call of $g(1, N, i, C)$ that returned M, return M.

Otherwise return $\mathcal{O}(ty, N, i, M)$.

$\mathsf{i} = 1$; $\mathsf{N} = 0^n$ $\leftarrow$ *Dummy values, for readability, for when this argument is not used by* $\mathcal{O}$

Run adversary A

When A makes an oracle query, (N, H, M), do the following:

Partition M into $M[1] \cdots M[m]$ and partition H into $H[1] \cdots H[h]$

for $i \leftarrow 1$ **to** $m - 1$ **do** $C_i \leftarrow g(1, N, i, M[i])$

$Y[m] \leftarrow g(2, N, m, \mathrm{len}(M[m]))$; $C[m] \leftarrow Y[m] \oplus M[m]$

Checksum $\leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\, 0^* \oplus Y[m]$

$\mathit{Tag} \leftarrow g(3, N, m, \text{Checksum})$; $T \leftarrow \mathit{Tag}\,[\text{first } \tau \text{ bits}]$

for $i \leftarrow 2$ **to** $h - 1$ **do** $Y[i] \leftarrow g(5, \mathsf{N}, i, H[i])$

if $|H| < n$ **then** $\mathit{Tag} \leftarrow g(6, \mathsf{N}, \mathsf{i}, \mathsf{pad}(H))$

else if $H = n$ **then** $\mathit{Tag} \leftarrow g(7, \mathsf{N}, \mathsf{i}, H)$

else if $|H_m| < n$ **then** $\mathit{Tag} \leftarrow g(8, H[1], \mathsf{i}, Y[2] \oplus \cdots \oplus Y[h-1] \oplus \mathsf{pad}(H[h]))$

else if $|H_m| = n$ **then** $\mathit{Tag} \leftarrow g(9, H[1], \mathsf{i}, Y[2] \oplus \cdots \oplus Y[h-1] \oplus H[h])$

$T = \mathit{Tag}\,[\text{first } \tau \text{ bits}]$; $\mathcal{C} \leftarrow C \,\|\, (T \oplus \Delta)$

Answer A's query with $\mathcal{C}$

When A outputs a bit, b $\leftarrow$ *for a privacy-attacking adversary*

return b

When A outputs a forgery attempt $(N, C \,\|\, T, H)$ $\leftarrow$ *for an authenticity-attacking adversary*

Partition $\mathcal{C}$ into $C[1] \cdots C[\varsigma]\, T$ and partition H into $H[1] \cdots H[h]$

for $i \leftarrow 1$ **to** $\varsigma - 1$ **do** $M_i \leftarrow g(4, N, i, C[i])$

$Y[\varsigma] \leftarrow g(2, N, \varsigma, \mathrm{len}(C[\varsigma]))$; Checksum $\leftarrow M[1] \oplus \cdots \oplus M[\varsigma - 1] \oplus C[\varsigma]\, 0^* \oplus Y[\varsigma]$

for $i \leftarrow 2$ **to** $h - 1$ **do** $Y[i] \leftarrow g(5, \mathsf{N}, i, H[i])$

if $|H| < n$ **then** $\Delta \leftarrow g(6, \mathsf{N}, \mathsf{i}, \mathsf{pad}(H))$

else if $H = n$ **then** $\Delta \leftarrow g(7, \mathsf{N}, \mathsf{i}, H)$

else if $|H_\varsigma| < n$ **then** $\Delta \leftarrow g(8, H[1], \mathsf{i}, Y[2] \oplus \cdots \oplus Y[h-1] \oplus \mathsf{pad}(H[h]))$

else if $|H_M| = n$ **then** $\Delta \leftarrow g(9, H[1], \mathsf{i}, Y[2] \oplus \cdots \oplus Y[h-1] \oplus H[h])$

$\mathit{Tag}^* \leftarrow g(1, N, \varsigma, \text{Checksum})$, $\mathit{Tag} \leftarrow g(0, N, \varsigma, \mathit{Tag}^*)$, $T' \leftarrow \mathit{Tag}\,[\text{first } \tau \text{ bits}]$

$\Delta' = \Delta\,[\text{first } \tau \text{ bits}]$; **if** $T' \oplus \Delta' = T$ **then return** 1 **else return** 0

Figure 2: Construction for the proof of Lemma 5.

$\mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{O}},\ddot{\mathcal{O}}}(q,\sigma)$ where the last inequality is the result from the prior paragraph. We have shown the first equation of the lemma.

Carrying on, let A be a (new) adversary that asks at most q queries, these totaling at most σ blocks, and suppose that A outputs a message having at most ς blocks. Adversary A's forgery is valid with probability $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{O\dot{C}B}}(A)$ if A interacts with an $\dot{\mathcal{E}}$-oracle, and it is valid with probability $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{O\ddot{C}B}}(A)$ if A interacts with an $\ddot{\mathcal{E}}$-oracle. From A we construct a (new) adversary B as (again) specified in Figure 2. When interacting with an $\dot{\mathcal{O}}$ oracle, adversary B asks at most $\sigma+\varsigma$ queries and outputs 1 with probability $\Pr[B^{\dot{\mathcal{O}}}=1]=\mathbf{Adv}^{\mathrm{AUTH}}_{\mathrm{O\dot{C}B}}(A)$. When interacting with an $\ddot{\mathcal{O}}$ oracle it asks the same number of queries and outputs 1 with probability $\Pr[B^{\ddot{\mathcal{O}}}=1]=\mathbf{Adv}^{\mathrm{AUTH}}_{\mathrm{O\ddot{C}B}}(A)$. We leave it for the reader to check those facts, which follow because B with an $\dot{\mathcal{O}}$-oracle provides A the same environment that A would see if A had an $\dot{\mathcal{E}}$ oracle, while B with an $\ddot{\mathcal{O}}$-oracle provides A the same environment that A would see if A had an $\ddot{\mathcal{E}}$ oracle. (In addition, A's forgery attempt is correctly decrypted with respect to the same oracle, and it outputs 1 if and only if the forgery was valid.) Thus $\mathbf{Adv}^{\mathrm{AUTH}}_{\mathrm{O\dot{C}B}}(A)-\mathbf{Adv}_{\mathrm{O\ddot{C}B}}(A)=\Pr[B^{\dot{\mathcal{O}}}=1]-\Pr[B^{\ddot{\mathcal{O}}}=1]$, so $\mathbf{Adv}^{\mathrm{AUTH}}_{\mathrm{O\dot{C}B}}(A)\leq\mathbf{Adv}^{\mathrm{AUTH}}_{\mathrm{O\ddot{C}B}}(A)+\mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{O}},\ddot{\mathcal{O}}}(B)$. Passing to the resource-based statement we have that $\mathbf{Adv}^{\mathrm{AUTH}}_{\mathrm{O\dot{C}B}}(q,\sigma,\varsigma)\leq\mathbf{Adv}_{\mathrm{O\ddot{C}B}}(q,\sigma,\varsigma)+\mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{O}},\ddot{\mathcal{O}}}(q,\sigma+\varsigma)$, which is the second equation of the lemma. ▮

Proof of Theorem 3: We can now complete the proof of Theorem 3. We will use the following results from [5, 22]:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB[Perm}(n),\tau]}(q,\sigma) &\leq 1.5\,\sigma_1^2/2^n\\
\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB[Perm}(n),\tau]}(q,\sigma,\varsigma) &\leq 1.5\,\sigma_2^2/2^n+2^{-\tau}\\
\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{PMAC[Perm}(n),\tau]}(q,\sigma) &\leq 2\,\sigma_3^2/2^n
\end{aligned}$$

where $\sigma_1=\sigma+q+3$ and $\sigma_2=\sigma+q+5\varsigma+11$ and $\sigma_3=\sigma+1$. Combining the equations above with Theorem 2 we have that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{PRIV}}_{\mathrm{OCB}\,\cdot\cdot\,\mathrm{PMAC[Perm}(n),\tau]}(q,\sigma) &\leq 1.5\,\sigma_1^2/2^n\\
\mathbf{Adv}^{\mathrm{AUTH}}_{\mathrm{OCB}\,\cdot\cdot\,\mathrm{PMAC[Perm}(n),\tau]}(q,\sigma,\varsigma) &\leq 5\,\sigma_2^2/2^n+2^{-\tau}
\end{aligned}$$

Combining Lemma 4, Lemma 5, and the two equations above we get:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{PRIV}}_{\mathrm{O\dot{C}B}}(q,\sigma) &\leq 9.5\sigma_1^2/2^n\\
\mathbf{Adv}^{\mathrm{AUTH}}_{\mathrm{O\dot{C}B}}(q,\sigma,\varsigma) &\leq 13\,\sigma_2^2/2^n+2^{-\tau}
\end{aligned}$$

This completes the proof of Theorem 3. ▮

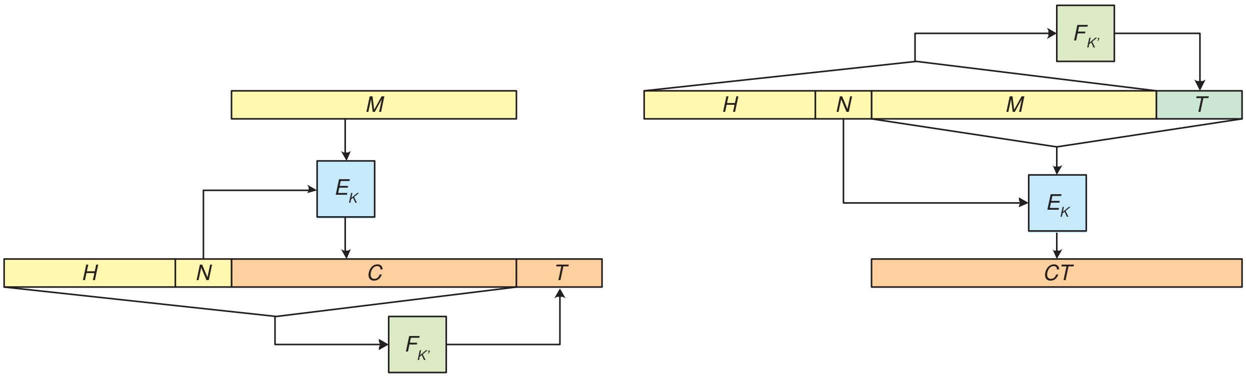


Figure 3: Two methods of generic composition for making an AEAD-scheme: (nonce-based, AD-using) encrypt-then-mac (left) and (nonce-based, AD-using) mac-then-encrypt (right).

7 Generic Composition

So far we have focused on making an AEAD-scheme $\bar{\Pi}$ out of an AE-scheme Π and a function family F. However, it makes just as much sense to construct $\bar{\Pi}$ from a (privacy-only) encryption scheme Π and a function family F. Following [3], we call the construction of an AE/AEAD-scheme from a privacy-only encryption scheme and a MAC the *generic composition approach*.

We consider two different methods for generic composition: (1) nonce-based, AD-using, *encrypt-then-mac*, where, using a nonce N, one encrypts message M to a ciphertext C and then MACs the associated-data H along with N and C to get a tag T to accompany C; and (2) nonce-based, AD-using *mac-then-encrypt*, where one MACs the associated-data H and the message M and a nonce N to make a tag T, and then encrypts, using N, the message M and the tag T. See Figure 3.

Our viewpoints in this section are strongly motivated by [3]. But we have already made definitional choices different from theirs, and this does impact our findings.

Encrypt-then-MAC. Let $\langle X, Y\rangle$ denote a string that encodes strings X and Y. In particular, X and Y should be recoverable, and in linear time, from $\langle X, Y\rangle$. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. For simplicity, assume Π has a message space of $\mathsf{Message} = \{0,1\}^*$. Let its nonce space be $\mathsf{Nonce} = \{0,1\}^n$. Let $F\colon \mathcal{K}' \times \{0,1\}^* \to \{0,1\}^\tau$ be a function family. Let $\mathsf{Header} \subseteq \{0,1\}^*$. Given all this, we define the AEAD-scheme $[\Pi, F] = (\bar{\mathcal{K}}, \bar{\mathcal{E}}, \bar{\mathcal{D}})$ as follows:

- $\bar{\mathcal{K}} = \mathcal{K} \times \mathcal{K}'$.
- $\bar{\mathcal{E}}^{N,H}_{K,K'}(M) = C \parallel T$ where $C = \mathcal{E}^N_K(M)$ and $T = F_{K'}(\langle N, H, C\rangle)$.
- $\bar{\mathcal{D}}^{N,H}_{K,K'}(\mathfrak{C}) = \mathcal{D}^N_K(C)$ if $|\mathfrak{C}| \geq \tau$ and $\mathfrak{C} = C \parallel T$ where $|T| = \tau$ and $F_{K'}(\langle N, H, C\rangle) = T$; and $\bar{\mathcal{D}}^{N,H}_{K,K'}(\mathfrak{C}) = \text{Invalid}$ otherwise.

Note that in the construction above, to match our nonce-based treatment of AEAD-schemes, we have assumed a symmetric encryption scheme Π that explicitly surfaces its nonce (as opposed to its being a probabilistic or stateful encryption encryption scheme).

As one might expect, if Π is a secure encryption scheme (in the $\mathbf{Adv}^{\mathrm{priv}}_{\Pi}$ sense) and F is a good pseudorandom function (in the $\mathbf{Adv}^{\mathrm{prf}}_{F}$-sense) then $[\Pi, F]$ is a good AEAD-scheme. The quantitative result is as follows.

Theorem 6 [Security of encrypt-then-mac] *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be an encryption scheme with message space* $\{0,1\}^*$ *and let* $F\colon \mathcal{K}' \times \{0,1\}^* \to \{0,1\}^\tau$ *be a function family. Then*

$$\mathbf{Adv}^{\mathrm{PRIV}}_{[\Pi,F]}(t,q,\sigma) \;\leq\; \mathbf{Adv}^{\mathrm{priv}}_{\Pi}(t_1,q,\sigma) + \mathbf{Adv}^{\mathrm{prf}}_{F}(t_2,q,\sigma)$$

$$\mathbf{Adv}^{\mathrm{AUTH}}_{[\Pi,F]}(t,q,\sigma,\varsigma) \;\leq\; \mathbf{Adv}^{\mathrm{prf}}_{F}(t_3,q+1,\sigma+\varsigma) + 2^{-\tau}$$

where $t = t_1 + t_2 + \mathrm{Time}_F(q,\sigma) + O(\sigma+q)$ *and* $t = t_3 + \mathrm{Time}_{\mathcal{E}}(q,\sigma+\varsigma) + O(\sigma+\varsigma+q)$. $\diamondsuit$

Proof: Let A be an adversary that attacks the privacy of $\bar{\Pi} = [\Pi, F] = (\bar{\mathcal{K}}, \bar{\mathcal{E}}, \bar{\mathcal{D}})$. Suppose A runs in time t and asks at most q queries, these totaling at most σ bits. We construct an adversary A_Π that attacks the privacy of Π and an adversary A_F that attacks F.

Adversary A_Π works as follows. It chooses $K' \xleftarrow{\$} \mathcal{K}'$. Then it runs adversary A. When A makes an oracle call (N,H,M) have A_Π make an oracle call of (N,M), getting back a ciphertext C. Then A_Π returns to A the string $C \,\|\, F_{K'}(\langle N,H,C\rangle)$. When A outputs a bit b and halts, have A_Π output the identical bit b and halt.

Adversary A_F works as follows. It runs adversary A. When A makes an oracle call (N,H,M) have A_F choose a random $C \xleftarrow{\$} \{0,1\}^{\ell(|M|)}$ where ℓ is the length function of Π. Then A_F calls its oracle on $\langle N,H,C\rangle$, getting a return value T. Have A_F return to A the string $C \,\|\, T$. When A outputs a bit b and halts, have A_F output the identical bit b and halt.

We now analyze the construction. Let $\mathcal{E}_K F_{K'}$ be a synonym for $\bar{\mathcal{E}}_{K,K'}$. Let $\$\ F_{K'}$ be the oracle that, on input N,H,M, chooses a random $C \xleftarrow{\$} \{0,1\}^{\ell(|M|)}$ and returns $C \,\|\, F_{K'}(\langle N,H,C\rangle)$. Let $\$\ \$$ be the oracle that, on input (N,H,M), returns a random string of $\ell(|M|)+\tau$ bits. Now note that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{PRIV}}_{\bar{\Pi}}(A) &= \Pr[A^{\mathcal{E}_K F_{K'}} = 1] - \Pr[A^{\$\ \$} = 1] \\
&= \Pr[A^{\mathcal{E}_K F_{K'}} = 1] - \Pr[A^{\$\ F_{K'}} = 1] + \Pr[A^{\$\ F_{K'}} = 1] - \Pr[A^{\$\ \$} = 1] \\
&= \Pr[A_\Pi^{\mathcal{E}_K} = 1] - \Pr[A_\Pi^{\$} = 1] + \Pr[A_F^{F_K} = 1] - \Pr[A_F^{\$} = 1] \\
&= \mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A_\Pi) + \mathbf{Adv}^{\mathrm{prf}}_{F}(A_F)
\end{aligned}$$

The running time of A_Π is $t + \mathrm{Time}_F(q,\sigma) + O(\sigma+q)$. The running time of A_F is $t + O(\sigma+q)$. The first equation follows.

Let B be an adversary that attacks the authenticity of $\bar{\Pi} = [\Pi,F] = (\bar{\mathcal{K}}, \bar{\mathcal{E}}, \bar{\mathcal{D}})$. Suppose B runs in time t and asks at most q queries, these totaling at most σ bits, and then B produces an output of ς bits. We construct an adversary B_F that attacks F.

Adversary B_F works as follows. It chooses a random key K. Then it runs adversary B. When B makes an oracle query (N,H,M) have B_F compute $C \leftarrow \mathcal{E}_K^N(M)$. Then B_F calls its own oracle on the string $\langle N,H,C\rangle$, receiving a value T. Then have B_F answer B's oracle query with $C \,\|\, T$. When B halts, outputting a forgery attempt (N^*, H^*, C^*T^*), where $|T^*| = \tau$, let B_F call its oracle on $\langle N^*, H^*, C^*\rangle$, obtaining a return value T. If $T = T^*$ and (N^*, H^*, C^*T^*) is new then have B_F output 1; otherwise, have B_F output 0.

Note that the running time of B_F is $t + \mathrm{Time}_{\mathcal{E}}(q,\sigma) + O(\sigma+\varsigma+q)$ and B_F asks $q+1$ oracle queries and these queries total at most $\sigma+\varsigma$ bits.

If the oracle to B_F is a valid PRF-oracle $F_{K'}$ then B, when being run by B_F, forges with probability $\mathbf{Adv}^{\mathrm{AUTH}}_{\bar{\Pi}}(B)$. If the oracle to B_F is a random function ρ then B, when being run by B_F, forges with probability $2^{-\tau}$. The second equation in the theorem now follows. ∎

MAC-THEN-ENCRYPT. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, $F\colon \mathcal{K}' \times \{0,1\}^* \to \{0,1\}^\tau$, $\mathsf{Header} \subseteq \{0,1\}^*$, $\mathsf{Message} = \{0,1\}^*$ and $\mathsf{Nonce} = \{0,1\}^n$ all be as before. Then we define the AEAD-scheme $[F,\Pi] = (\bar{\mathcal{K}}, \bar{\mathcal{E}}, \bar{\mathcal{D}})$ as follows:

- $\bar{\mathcal{K}} = \mathcal{K} \times \mathcal{K}'$.
- $\bar{\mathcal{E}}^{N,H}_{K,K'}(M) = \mathcal{E}^N_K(\langle M, T\rangle)$ where $T = F_{K'}(\langle N, H, M\rangle)$.
- $\bar{\mathcal{D}}^{N,H}_{K,K'}(\mathcal{C}) = M$ if $\langle M, T\rangle = \mathcal{D}^N_K(\mathcal{C})$ where $|T| = \tau$ and $T = F_{K'}(\langle N, H, M\rangle)$; and $\bar{\mathcal{D}}^{N,H}_{K,K'}(\mathcal{C}) =$ INVALID otherwise.

Following [3] one might expect that $[F,\Pi]$ may be insecure even when F is secure as a pseudorandom function and Π achieves privacy. But the definitions we are using differ from those in [3] and $[F,\Pi]$ is in fact secure under natural assumptions. The essential difference lies in the nonce-based treatment of encryption we have used; this forces there to be only one possible ciphertext for a plaintext (once the nonce is pinned down), eliminating the kind of counter-example demonstrated by [3].

Theorem 7 [Security of mac-then-encrypt] *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme with message space $\{0,1\}^*$ and let $F\colon \mathcal{K}' \times \{0,1\}^* \to \{0,1\}^\tau$ be a function family. Then*

$$\mathbf{Adv}^{\text{PRIV}}_{[F,\Pi]}(t,q,\sigma) \;\leq\; \mathbf{Adv}^{\text{priv}}_{\Pi}(t_1,q,\sigma)$$

$$\mathbf{Adv}^{\text{AUTH}}_{[F,\Pi]}(t,q,\sigma,\varsigma) \;\leq\; \mathbf{Adv}^{\text{prf}}_{F}(t_2,q+1,\sigma+\varsigma) + 2^{-\tau}$$

where $t = t_1 + \mathrm{Time}_F(q,\sigma) + O(\sigma+q)$ and $t = t_2 + \mathrm{Time}_{\mathcal{E}}(q,\sigma) + \mathrm{Time}_{\mathcal{D}}(1,\varsigma) + O(\sigma+\varsigma+q)$. $\diamondsuit$

Proof: Let A be an adversary that attacks the privacy of $\bar{\Pi} = [F,\Pi] = (\bar{\mathcal{K}}, \bar{\mathcal{E}}, \bar{\mathcal{D}})$. Suppose A runs in time t and asks at most q queries, these totaling at most σ bits. We construct an adversary A_Π that attacks the privacy of Π.

Adversary A_Π works as follows. It chooses $K' \xleftarrow{\$} \mathcal{K}'$. Then it runs adversary A. When A makes an oracle call (N,H,M) have A_Π compute $T = F_{K'}(\langle N,H,M\rangle)$ and then have A_Π make its own oracle call of $(N, \langle M, T\rangle)$, getting back a ciphertext C. Then A_Π returns to A the string $C \| F_{K'}(\langle N,H,C\rangle)$. When A outputs a bit b and halts, have A_Π output the identical bit b and halt.

We have that $\mathbf{Adv}^{\text{PRIV}}_{\bar{\Pi}}(A) = \Pr[A^{\bar{\mathcal{E}}_K} = 1] - \Pr[A^{\$} = 1] = \Pr[A_\Pi^{\mathcal{E}_K} = 1] - \Pr[A_\Pi^{\$} = 1] = \mathbf{Adv}^{\text{priv}}_{\Pi}(A_\Pi)$. The running time of A_Π is $t + \mathrm{Time}_F(q,\sigma) + O(\sigma+q)$. The first equation follows.

Now let B be an adversary that attacks the authenticity of $\bar{\Pi} = [\Pi, F] = (\bar{\mathcal{K}}, \bar{\mathcal{E}}, \bar{\mathcal{D}})$. Suppose B runs in time t and asks at most q queries, these totaling at most σ bits, and then B produces an output of at most ς bits. We construct an adversary B_F that attacks F.

Adversary B_F works as follows. It chooses a random key K. Then it runs adversary B. When B makes an oracle query (N,H,M) have B_F make its own oracle call of $\langle N,H,M\rangle$ getting a return value of T. Then let B_F compute $\mathcal{C} \leftarrow \mathcal{E}^N_K(\langle M,T\rangle)$. Then B_F returns to B value $\mathcal{C}$. When B halts, outputting a forgery attempt $(N^*, H^*, \mathcal{C}^*)$ let B_F compute $M^* \parallel T^* = \mathcal{D}^{N^*}_K(\mathcal{C})$ where $|T^*| = \tau$. (Adversary B fails and returns 0 if $|\mathcal{D}^{N^*}_K(\mathcal{C})| < \tau$.) Then B_F calls its oracle on $\langle H^*, N^*, M^*\rangle$, obtaining a return value T. If $T = T^*$ and $(N^*, H^*, \mathcal{C}^*)$ is new then B_F outputs 1; otherwise it outputs 0.

The running time of B_F is $t + \mathrm{Time}_{\mathcal{E}}(q,\sigma) + \mathrm{Time}_{\mathcal{D}}(1,\varsigma) + O(\sigma+\varsigma+q)$ and B_F asks $q+1$ oracle queries and these queries total at most $\sigma+\varsigma$ bits.

If the oracle to B_F is a valid PRF-oracle $F_{K'}$ then B, when being run under B_F, forges with probability $\mathbf{Adv}^{\text{AUTH}}_{\bar{\Pi}}(B)$. This is because B forges only when $(N^*, H^*, \mathcal{C}^*)$ is new, which means that N^* is new or H^* is new or M^* has not yet been queried along with N^* and H^*. If the oracle to B_F is a random function ρ then B, when being run by B_F, forges with probability $2^{-\tau}$. The second equation in the theorem now follows. ▮

REMARKS. The two constructions of this section use a nonce-based symmetric encryption scheme that is secure in the $\mathbf{Adv}^{\mathrm{priv}}$ sense. It needs to be emphasized that conventional modes of operation, like CBC mode with the IV as the nonce, do not achieve good security in this sense. Thus one can not expect to achieve a secure AEAD-scheme, in the sense we have defined it, by using the constructions of this section and encrypting under CBC mode, with the IV as the nonce. There are simple ways to achieve the notion of privacy defined here, starting from a block cipher, but an investigation of this topic would take us too far afield.

8 Acknowledgments

Burt Kaliski first described the AEAD problem to me and inspired me to work on it. I was also motivated by NIST's modes-of-operation effort and a workshop they organized; thanks to Elaine Barker, William Burr, and Morris Dworkin. I received useful feedback from Mihir Bellare, John Black, Nancy Cam-Winget, Robert Moskowitz, Ron Rivest, Jesse Walker, and some anonymous referees. Work on this paper was funded by NSF CCR-0208842 and a gift from CISCO Systems. Special thanks to CISCO's Dave McGrew, who has followed and championed my work.

References

[1] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. *Proceedings of 38th Annual Symposium on Foundations of Computer Science* (FOCS 97), IEEE, 1997. `www.cs.ucdavis.edu/~rogaway/`

[2] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, vol. 61, no. 3, 2000. (Earlier version in *Advances in Cryptology – CRYPTO '94*.) `www.cs.ucdavis.edu/~rogaway/`)

[3] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology – ASIACRYPT '00*. Lecture Notes in Computer Science, vol. 1976, T. Okamoto., ed., Springer-Verlag, 2000. `www-cse.ucsd.edu/users/mihir/`

[4] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient encryption. *Advances in Cryptology – ASIACRYPT '00*. Lecture Notes in Computer Science, vol. 1976, T. Okamoto., ed., Springer-Verlag, 2000. `www.cs.ucdavis.edu/~rogaway/`

[5] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. *Advances in Cryptology – EUROCRYPT 2002*. Lecture Notes in Computer Science, vol. 2332, Springer-Verlag, 2002. `www.cs.ucdavis.edu/~rogaway`

[6] N. Cam-Winget and J. Walker. Personal communications, June 2001.

[7] L. Carter and M. Wegman. Universal hash functions. *J. of Computer and System Sciences*, vol. 18, pp. 143–154, 1979.

[8] J. Coron, M. Joye, D. Naccache, and P. Paillier. Universal padding schemes for RSA. *Advances in Cryptology – CRYPTO '02*, Lecture Notes in Computer Science, vol. 2442, pp. 226–241, 2002.

[9] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. *Fast Software Encryption*, Lecture Notes in Computer Science, Springer-Verlag, 2001.

[10] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM,* vol. 33, no. 4, pp. 210–217, 1986.

[11] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, vol. 28, April 1984, pp. 270–299.

[12] S. Haber and B. Pinkas. Securely combining public-key cryptosystems. *Proceedings of the 8th ACM Conference on Computer and Communications Security* (CCS-8), ACM Press, pp. 215–224, 2001.

[13] P. Hawkes and G. Rose. A mode of operation with partial encryption and message integrity (PEMI). Manuscript, 2002.

[14] J. Jonsson. On the security of CTR + CBC MAC. *Selected Areas in Cryptography, Ninth Annual Workshop* (SAC 2002), Lecture Notes in Computer Science, Springer-Verlag, 2002.

[15] C. Jutla. Encryption modes with almost free message integrity. *Advances in Cryptology – EUROCRYPT 2001.* Lecture Notes in Computer Science, vol. 2045, Springer-Verlag, 2001.

[16] B. Kaliski. Personal communication, May 2001.

[17] J. Katz and M. Yung. Unforgeable encryption and adaptively secure modes of operation. *Fast Software Encryption '00.* Lecture Notes in Computer Science, B. Schneier, ed., 2000.

[18] J. Kilian and P. Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *J. of Cryptology*, vol. 14, no. 1, pp. 17–35, 2001. Earlier version in CRYPTO '96.

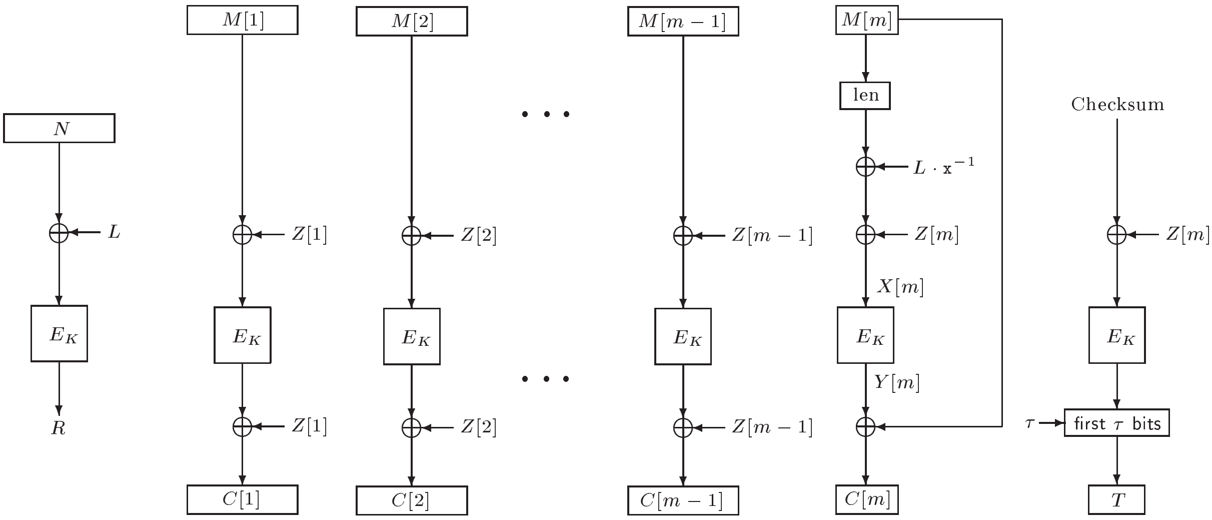
[19] H. Krawczyk. LFSR-based hashing and authentication. *Advances in Cryptology – CRYPTO '94.* Lecture Notes in Computer Science, vol. 839, Springer-Verlag, pp. 129–139, 1994.

[20] R. Rivest. Personal communications, Aug 2001.

[21] P. Rogaway. Authenticated-encryption with associated-data. *Ninth ACM Conference on Computer and Communications Security* (CCS-9). ACM Press, 2002. Proceedings version of this paper.

[22] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. *Eighth ACM Conference on Computer and Communications Security* (CCS-8). ACM Press, 2001. `www.cs.ucdavis.edu/~rogaway`

[23] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). Submission to NIST, June 2002. `csrc.nist.gov/encryption/modes/`

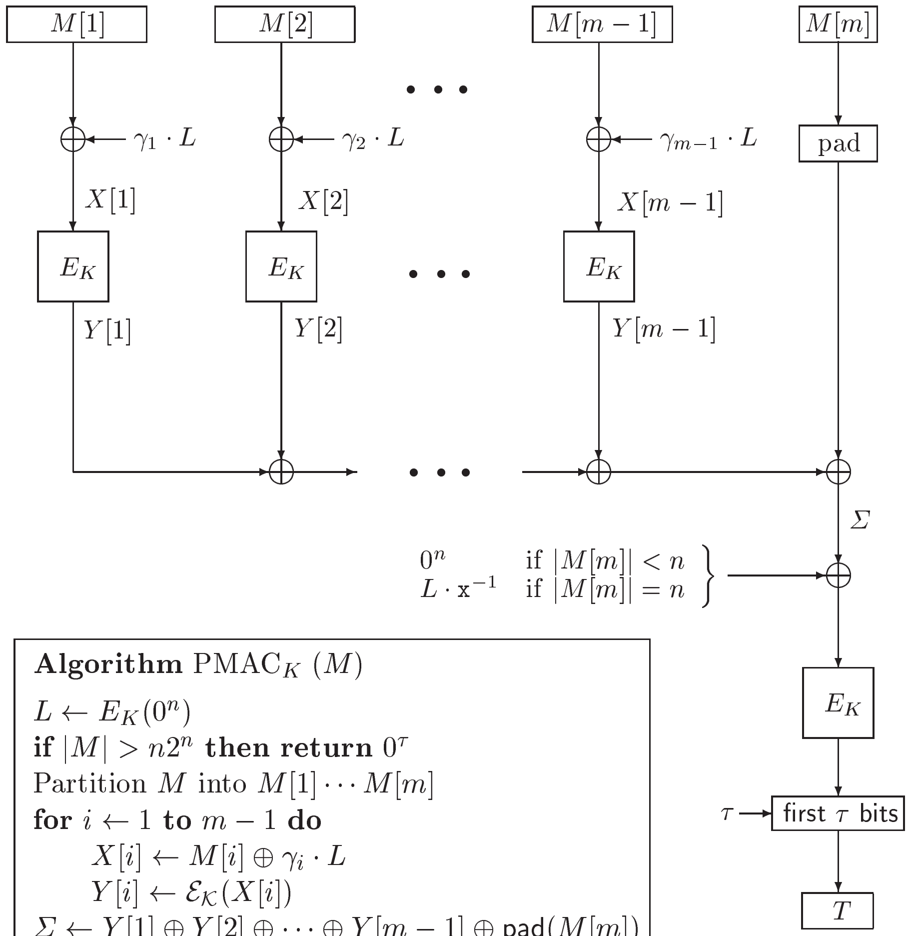


Algorithm $\mathcal{E}_K^N(M)$	Algorithm $\mathcal{D}_K^N(\mathcal{C})$
Partition M into $M[1]\cdots M[m]$	Partition $\mathcal{C}$ into $C[1]\cdots C[m]\,T$
$L \leftarrow E_K(0^n)$; $\quad R \leftarrow E_K(N \oplus L)$	$L \leftarrow E_K(0^n)$; $\quad R \leftarrow E_K(N \oplus L)$
for $i \leftarrow 1$ **to** m **do** $Z[i] = \gamma_i \cdot L \oplus R$	**for** $i \leftarrow 1$ **to** m **do** $Z[i] = \gamma_i \cdot L \oplus R$
for $i \leftarrow 1$ **to** $m-1$ **do**	**for** $i \leftarrow 1$ **to** $m-1$ **do**
$\quad C[i] \leftarrow E_K(M[i] \oplus Z[i]) \ \oplus\ Z[i]$	$\quad M[i] \leftarrow E_K^{-1}(C[i] \oplus Z[i]) \ \oplus\ Z[i]$
$X[m] \leftarrow \text{len}(M[m]) \oplus L \cdot \mathtt{x}^{-1} \oplus Z[m]$	$X[m] \leftarrow \text{len}(C[m]) \oplus L \cdot \mathtt{x}^{-1} \oplus Z[m]$
$Y[m] \leftarrow E_K(X[m])$	$Y[m] \leftarrow E_K(X[m])$
$C[m] \leftarrow Y[m] \oplus M[m]$	$M[m] \leftarrow Y[m] \oplus C[m]$
$C \leftarrow C[1]\cdots C[m]$	$M \leftarrow M[1]\cdots M[m]$
Checksum $\leftarrow$	Checksum $\leftarrow$
$\quad M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$	$\quad M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
$T \leftarrow E_K(\text{Checksum} \oplus Z[m])$ [first τ bits]	$T' \leftarrow E_K(\text{Checksum} \oplus Z[m])$ [first τ bits]
return $\mathcal{C} \leftarrow C \,\\|\, T$	**if** $T = T'$ **then return** M
	$\quad$ **else return** INVALID

Figure 4: OCB. The plaintext is M, the key is K, the nonce is N, the key space is $\mathcal{K}$. Each $Z[i] = \gamma_i \cdot L \oplus R$.

A Definitions of OCB and PMAC

The following material is taken from [5, 22]. Throughout, fix integer $n > 0$. Let $\mathsf{ntz}(i)$ (where $i \geq 1$) be the number of trailing 0-bits in the binary representation of integer i. If $X \in \{0,1\}^*$ then $\text{len}(X) = \max\{1, \lceil |X|/n \rceil\}$. For $X \in \{0,1\}^*$ and $|X| \leq n$ let $X\,0^* = X\,0^{n-|X|}$. Let $\mathsf{pad}(X)$ be X if $|X| = n$ and $X10^{n-|X|-1}$ if $|X| < n$. Let $\text{GF}(2^n)$ be the field with 2^n points. We think of a point a in $\text{GF}(2^n)$ as an n-bit string $a_{n-1}\ldots a_1 a_0 \in \{0,1\}^n$ or as the polynomial $a(\mathtt{x}) = a_{n-1}\mathtt{x}^{n-1} + \cdots + a_1\mathtt{x} + a_0$. Addition and multiplication is defined in the usual way where, for multiplication, we fix an irreducible degree n polynomial. Define $\gamma = \gamma^n$ by $\gamma^1 = (0\ 1)$ and, for $\ell > 0$, $\gamma^{\ell+1} = (0\gamma_0^\ell\ 0\gamma_1^\ell \cdots 0\gamma_{2\ell-2}^\ell\ 0\gamma_{2\ell-1}^\ell\ 1\gamma_{2\ell-1}^\ell\ 1\gamma_{2\ell-2}^\ell \cdots 1\gamma_1^\ell\ 1\gamma_0^\ell)$. We write "Partition M into $M[1]\cdots M[m]$" for "Let $m = \text{len}(M)$ and let $M[1],\ldots,M[m]$ be strings such that $M[1]\cdots M[m] = M$ and $|M[i]| = n$ for $1 \leq i < m$." We write "Partition $\mathcal{C}$ into $C[1]\cdots C[m]\,T$" for "if $|\mathcal{C}| < \tau$ then



Algorithm $\mathrm{PMAC}_K\ (M)$

$L \leftarrow E_K(0^n)$
if $|M| > n2^n$ **then return** 0^τ
Partition M into $M[1] \cdots M[m]$
for $i \leftarrow 1$ **to** $m - 1$ **do**
    $X[i] \leftarrow M[i] \oplus \gamma_i \cdot L$
    $Y[i] \leftarrow \mathcal{E}_\mathcal{K}(X[i])$
$\Sigma \leftarrow Y[1] \oplus Y[2] \oplus \cdots \oplus Y[m-1] \oplus \mathsf{pad}(M[m])$
if $|M[m]| = n$ **then** $X[m] = \Sigma \oplus L \cdot \mathtt{x}^{-1}$
            else $X[m] \leftarrow \Sigma$
$T = E_K(X[m])$ [first τ bits]
return T

Figure 5: PMAC. The message to MAC is M and the key is K. Value $L = E_K(0^n)$ is derived from K.

return Invalid. Otherwise let $C = \mathcal{C}$ [first $|\mathcal{C}| - \tau$ bits], $T = \mathcal{C}$[last τ bits], and $m = \mathrm{len}(C)$, and let $C[1], \ldots, C[m]$ be strings such that $C[1] \cdots C[m] = C$ and $|C[i]| = n$ for $1 \le i < m$."

To use OCB or PMAC one must specify a block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. One must also specify a tag length $\tau \in [1..n]$. We let OCB$[E, \tau]$ and PMAC$[E, \tau]$ be the algorithms with the indicated parameters. Encryption and decryption under OCB depend on an n-bit nonce N. Encryption and decryption under OCB is defined in Figure 4, while PMAC is defined in Figure 5

B Proof of Lemma 4

Let \$ denote the oracle that responds to every query by returning n-random bits. To show $\mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{O}},\ddot{\mathcal{O}}}(q) \le 8q^2/2^n$ we use the triangle inequality after establishing that

$$\mathbf{Adv}^{\mathrm{dist}}_{\dot{\mathcal{O}},\$}(q) \quad \le \quad 4q^2/2^n \quad \text{and} \tag{5}$$

$$\mathbf{Adv}^{\mathrm{dist}}_{\ddot{\mathcal{O}},\$}(q) \quad \le \quad 4q^2/2^n \tag{6}$$

We now prove Equation (5). The proof for Equation (6) is analogous and therefore omitted.

We use the game-playing approach, as in works like [18]. First we define, in Figure 6, a game that we denote game $\mathcal{O}$. Though not shown in the pseudocode, game $\mathcal{O}$ (like game $\dot{\mathcal{O}}$) returns n random bits in response to any invalid query (invalid queries are defined in Section 6). An inspection of game $\mathcal{O}$ makes clear that it and game $\dot{\mathcal{O}}$ engender identical views to any adversary: $\Pr[A^{\dot{\mathcal{O}}} = 1] = \Pr[A^{\mathcal{O}} = 1]$ for any A.

Next we modify game $\mathcal{O}$, as is standard, by dropping each statement that immediately follows the setting of the variable *bad* to `true`. The revised game, which we call game $\mathcal{R}$, is show in Figure 7. As before, random bits are returned in response to any invalid query.

We claim that game $\mathcal{R}$ provides an adversary with a view identical to game \$; that is, in response to any query made, game $\mathcal{R}$ returns n random bits. To see this, first observe that for any query Q of type $ty \in \{1,2,3,5,6,7,8,9\}$ we set $Y \xleftarrow{\$} \{0,1\}^n$ and return Y. For $ty = 4$ we set $X \xleftarrow{\$} \{0,1\}^n$ and return $X \oplus iL \oplus R$ where iL and R are independent of X. This has the effect of returning a random string in $\{0,1\}^n$. Finally, there may be a single query of type $ty = 0$. We claim that the response to this one query is also uniform. For if the adversary asks a query $(0, N, i, M)$ where $N \notin \mathcal{N}$ then the response Z is uniform because a fresh random value R will be used in computing $M \oplus iL \oplus R$. On the other hand, if the adversary selects an N-value that was used already, then the game already selected a random R. That earlier R value, however, had no influence on the $Z_1, \ldots, Z_r$ values that were returned to the adversary before its $ty = 0$ query. The selected R-value is therefore independent of the values M and i that the adversary chooses. The value R was chosen independent of L. Thus the R that is used in computing the response $Z = M \oplus iL \oplus R$ is a uniform random variable that is independent of the random variable $M \oplus iL$, and so Z is uniform.

Since games $\mathcal{O}$ and $\mathcal{R}$ are syntactically identical apart from what happens after the flag *bad* is set to `true`, using the customary technique from the game-playing approach we have that for any adversary A,

$$\mathbf{Adv}^{\mathrm{dist}}_{\mathcal{O},\mathcal{R}}(A) \quad \le \quad \Pr[A^{\mathcal{R}} \text{ sets flag } bad]$$

Our goal, then, is to show that

$$\Pr[A^{\mathcal{R}} \text{ sets flag } bad] \quad \le \quad 4q^2/2^n \tag{7}$$

if A makes q or fewer queries. Lemma 4 will then follows.

We now expunge the many rounds of interaction present in game $\mathcal{R}$, simplifying to an adversary getting one vector of strings and providing another. To make this simplification remember that in game $\mathcal{R}$ the first $q-1$ responses returned to the adversary are the random strings $Z_1, \ldots, Z_{q-1}$. The strings don't depend on the adversary's queries and they don't depend on any calculations carried out during the game. (For the latter claim one needs first to rewrite the code associated to queries of type 4: choose Z at random instead of X and then let $X \leftarrow Z \oplus iL \oplus R$.) As a result of this, we may choose $Z_1, \ldots, Z_{q-1} \xleftarrow{\$} \{0,1\}^n$ at the very beginning of the game instead of one Z_s

Initialization
$\pi(x) \leftarrow \mathsf{undef}$ for all $x \in \{0,1\}^n$; $\mathcal{N} \leftarrow \emptyset$; $bad \leftarrow \mathsf{false}$;
$L \stackrel{\$}{\leftarrow} \{0,1\}^n$; $\pi(0^n) \leftarrow L$

To respond to query $\mathcal{O}(ty, N, i, M)$
if $N \in \mathcal{N}$ **then** $R \leftarrow \pi(N \oplus L)$
else { $\mathcal{N} \leftarrow \mathcal{N} \cup \{N\}$; $R \stackrel{\$}{\leftarrow} \{0,1\}^n$
  if $R \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $R \stackrel{\$}{\leftarrow} \overline{\text{Range}}(\pi)$
  if $N \oplus L \in \text{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $R \leftarrow \pi(N \oplus L)$
  $\pi(N \oplus L) \leftarrow R$ }
case ty **of**
0: $Z \leftarrow M \oplus iL \oplus R$
1: $X \leftarrow M \oplus iL \oplus R$; $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$
  if $Y \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \stackrel{\$}{\leftarrow} \overline{\text{Range}}(\pi)$
  if $X \in \text{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \leftarrow \pi(X)$
  $\pi(X) \leftarrow Y$; $Z \leftarrow Y \oplus iL \oplus R$
2: $X \leftarrow M \oplus iL \oplus R \oplus L \cdot \mathtt{x}^{-1}$; $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$
  if $Y \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \stackrel{\$}{\leftarrow} \overline{\text{Range}}(\pi)$
  if $X \in \text{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \leftarrow \pi(X)$
  $Z \leftarrow \pi(X) \leftarrow Y$
3: $X \leftarrow M \oplus iL \oplus R$; $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$
  if $Y \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \stackrel{\$}{\leftarrow} \overline{\text{Range}}(\pi)$
  if $X \in \text{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \leftarrow \pi(X)$
  $Z \leftarrow \pi(X) \leftarrow Y$
4: $C \leftarrow M$; $Y \leftarrow C \oplus iL \oplus R$; $X \stackrel{\$}{\leftarrow} \{0,1\}^n$
  if $X \in \text{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $X \stackrel{\$}{\leftarrow} \overline{\text{Dom}}(\pi)$
  if $Y \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $X \leftarrow \pi^{-1}(Y)$
  $\pi(X) \leftarrow Y$; $Z \leftarrow X \oplus iL \oplus R$
5: $X \leftarrow M \oplus iL$; $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$ $//i \neq 1$
  if $Y \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \stackrel{\$}{\leftarrow} \overline{\text{Range}}(\pi)$
  if $X \in \text{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \leftarrow \pi(X)$
  $Z \leftarrow \pi(X) \leftarrow Y$
6: $X \leftarrow M$; $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$ $//M \neq 0^n$
  if $Y \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \stackrel{\$}{\leftarrow} \overline{\text{Range}}(\pi)$
  if $X \in \text{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \leftarrow \pi(X)$
  $Z \leftarrow \pi(X) \leftarrow Y$
7: $X \leftarrow M \oplus L \cdot \mathtt{x}^{-1}$; $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$
  if $Y \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \stackrel{\$}{\leftarrow} \overline{\text{Range}}(\pi)$
  if $X \in \text{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \leftarrow \pi(X)$
  $Z \leftarrow \pi(X) \leftarrow Y$
8: $X \leftarrow M \oplus R$; $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$
  if $Y \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \stackrel{\$}{\leftarrow} \overline{\text{Range}}(\pi)$
  if $X \in \text{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \leftarrow \pi(X)$
  $Z \leftarrow \pi(X) \leftarrow Y$
9: $X \leftarrow M \oplus R \oplus L \cdot \mathtt{x}^{-1}$; $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$
  if $Y \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \stackrel{\$}{\leftarrow} \overline{\text{Range}}(\pi)$
  if $X \in \text{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $Y \leftarrow \pi(X)$
  $Z \leftarrow \pi(X) \leftarrow Y$
return Z

Figure 6: Oracle $\mathcal{O}$. This oracle is adversarially equivalent to oracle $\dot{\mathcal{O}}$.

Initialization
$\pi(x) \leftarrow \mathsf{undef}$ for all $x \in \{0,1\}^n$; $\mathcal{N} \leftarrow \emptyset$; $bad \leftarrow \mathsf{false}$
$L \xleftarrow{\$} \{0,1\}^n$; $\pi(0^n) \leftarrow L$

To respond to query $\mathcal{R}(ty, N, i, M)$
if $N \in \mathcal{N}$ **then** $R \leftarrow \pi(N \oplus L)$
else { $\mathcal{N} \leftarrow \mathcal{N} \cup \{N\}$; $R \xleftarrow{\$} \{0,1\}^n$
  if $R \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $N \oplus L \in \mathrm{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $\pi(N \oplus L) \leftarrow R$ }
case ty **of**
 0: $Z \leftarrow M \oplus iL \oplus R$
 1: $X \leftarrow M \oplus iL \oplus R$; $Y \xleftarrow{\$} \{0,1\}^n$
  if $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $X \in \mathrm{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $\pi(X) \leftarrow Y$; $Z \leftarrow Y \oplus iL \oplus R$
 2: $X \leftarrow M \oplus iL \oplus R \oplus L \cdot \mathtt{x}^{-1}$; $Y \xleftarrow{\$} \{0,1\}^n$
  if $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $X \in \mathrm{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $Z \leftarrow \pi(X) \leftarrow Y$
 3: $X \leftarrow M \oplus iL \oplus R$; $Y \xleftarrow{\$} \{0,1\}^n$
  if $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $X \in \mathrm{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $Z \leftarrow \pi(X) \leftarrow Y$
 4: $C \leftarrow M$; $Y \leftarrow C \oplus iL \oplus R$; $X \xleftarrow{\$} \{0,1\}^n$
  if $X \in \mathrm{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $\pi(X) \leftarrow Y$; $Z \leftarrow X \oplus iL \oplus R$
 5: $X \leftarrow M \oplus iL$; $Y \xleftarrow{\$} \{0,1\}^n$ $// i \neq 1$
  if $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $X \in \mathrm{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $Z \leftarrow \pi(X) \leftarrow Y$
 6: $X \leftarrow M$; $Y \xleftarrow{\$} \{0,1\}^n$ $// M \neq 0^n$
  if $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $X \in \mathrm{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $Z \leftarrow \pi(X) \leftarrow Y$
 7: $X \leftarrow M \oplus L \cdot \mathtt{x}^{-1}$; $Y \xleftarrow{\$} \{0,1\}^n$
  if $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $X \in \mathrm{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $Z \leftarrow \pi(X) \leftarrow Y$
 8: $X \leftarrow M \oplus R$; $Y \xleftarrow{\$} \{0,1\}^n$
  if $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $X \in \mathrm{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $Z \leftarrow \pi(X) \leftarrow Y$
 9: $X \leftarrow M \oplus R \oplus L \cdot \mathtt{x}^{-1}$; $Y \xleftarrow{\$} \{0,1\}^n$
  if $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $X \in \mathrm{Dom}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $Z \leftarrow \pi(X) \leftarrow Y$
return Z

Figure 7: Oracle $\mathcal{R}$ omits statements following the setting of bad to true. In doing so it returns n random bits in response to each query.

per query. Then, instead of giving the adversary the Z_s-values one-by-one, we give the adversary the vector $\mathcal{Z} = (Z_1, \ldots, Z_{q-1})$ at the very beginning. The adversary now has a potentially easier task of finding valid queries $\mathcal{Q} = (Q_1, \ldots, Q_{q-1})$ that will cause event *bad* to get set to `true`. We call this new game $\mathcal{S}$. It is described in Figure 8.

Initialization

$Z_1, \ldots, Z_{q-1} \xleftarrow{\$} \{0,1\}^n$
Provide the adversary query answers $\mathcal{Z} = (Z_1, \cdots, Z_{q-1})$

Finalization

The adversary provides queries $\mathcal{Q} = (Q_1, \ldots, Q_q)$ where $Q_s = (ty_s, N_s, i_s, M_s)$
$L \xleftarrow{\$} \{0,1\}^n$; $\mathcal{N} \leftarrow \{N_1, \ldots, N_q\}$; $R(N) \xleftarrow{\$} \{0,1\}^n$ for each $N \in \mathcal{N}$
for $s \leftarrow 1$ **to** $q-1$ **do case** ty_s **of**

1:	$X_s \leftarrow M_s \oplus i_s L \oplus R(N_s)$,	$Y_s \leftarrow Z_s \oplus i_s L \oplus R(N_s)$
2:	$X_s \leftarrow M_s \oplus i_s L \oplus R(N_s) \oplus L \cdot \mathtt{x}^{-1}$,	$Y_s \leftarrow Z_s$
3:	$X_s \leftarrow M_s \oplus i_s L \oplus R(N_s)$,	$Y_s \leftarrow Z_s$
4:	$X_s \leftarrow Z_s \oplus i_s L \oplus R(N_s)$,	$Y_s \leftarrow M_s \oplus i_s L \oplus R(N_s)$
5:	$X_s \leftarrow M_s \oplus i_s L$,	$Y_s \leftarrow Z_s$
6:	$X_s \leftarrow M_s$,	$Y_s \leftarrow Z_s$
7:	$X_s \leftarrow M_s \oplus L \cdot \mathtt{x}^{-1}$,	$Y_s \leftarrow Z_s$
8:	$X_s \leftarrow M_s \oplus R(N_s)$,	$Y_s \leftarrow Z_s$
9:	$X_s \leftarrow M_s \oplus R(N_s) \oplus L \cdot \mathtt{x}^{-1}$,	$Y_s \leftarrow Z_s$

$\mathcal{X} \leftarrow \{\!\{0^n\}\!\} \cup \{\!\{N \oplus L : N \in \mathcal{N}\}\!\} \cup \{\!\{X_1, \ldots, X_{q-1}\}\!\}$
$\mathcal{Y} \leftarrow \{\!\{L\}\!\} \cup \{\!\{R(N) : N \in \mathcal{N}\}\!\} \cup \{\!\{Y_1, \ldots, Y_{q-1}\}\!\}$
bad $\leftarrow$ (there is a repetition in $\mathcal{X}$) or (there is a repetition in $\mathcal{Y}$)

Figure 8: Definition of game $\mathcal{S}$.

$L \xleftarrow{\$} \{0,1\}^n$; $\mathcal{N} \leftarrow \{N_1, \ldots, N_q\}$; $R(N) \xleftarrow{\$} \{0,1\}^n$ for each $N \in \mathcal{N}$
for $s \leftarrow 1$ **to** $q-1$ **do case** ty_s **of**

1:	$X_s \leftarrow M_s \oplus i_s L \oplus R(N_s)$,	$Y_s \leftarrow Z_s \oplus i_s L \oplus R(N_s)$
2:	$X_s \leftarrow M_s \oplus i_s L \oplus R(N_s) \oplus L \cdot \mathtt{x}^{-1}$,	$Y_s \leftarrow Z_s$
3:	$X_s \leftarrow M_s \oplus i_s L \oplus R(N_s)$,	$Y_s \leftarrow Z_s$
4:	$X_s \leftarrow Z_s \oplus i_s L \oplus R(N_s)$,	$Y_s \leftarrow M_s \oplus i_s L \oplus R(N_s)$
5:	$X_s \leftarrow M_s \oplus i_s L$,	$Y_s \leftarrow Z_s$
6:	$X_s \leftarrow M_s$,	$Y_s \leftarrow Z_s$
7:	$X_s \leftarrow M_s \oplus L \cdot \mathtt{x}^{-1}$,	$Y_s \leftarrow Z_s$
8:	$X_s \leftarrow M_s \oplus R(N_s)$,	$Y_s \leftarrow Z_s$
9:	$X_s \leftarrow M_s \oplus R(N_s) \oplus L \cdot \mathtt{x}^{-1}$,	$Y_s \leftarrow Z_s$

$\mathcal{X} \leftarrow \{\!\{0^n\}\!\} \cup \{\!\{N \oplus L : N \in \mathcal{N}\}\!\} \cup \{\!\{X_1, \ldots, X_{q-1}\}\!\}$
$\mathcal{Y} \leftarrow \{\!\{L\}\!\} \cup \{\!\{R(N) : N \in \mathcal{N}\}\!\} \cup \{\!\{Y_1, \ldots, Y_{q-1}\}\!\}$
bad $\leftarrow$ (there is a repetition in $\mathcal{X}$) or (there is a repetition in $\mathcal{Y}$)

Figure 9: Definition of game $\mathcal{T}$. Values $Z_1, \ldots, Z_{q-1} \in \{0,1\}^n$ are fixed, distinct constants, and valid queries $\mathcal{Q} = (Q_1, \ldots, Q_q)$, where $Q_s = (ty_s, N_s, i_s, M_s)$, are also fixed.

In game $\mathcal{S}$ we have recast the computing of *bad*. We no longer maintain π, which, in game $\mathcal{R}$, holds the growing sets Dom(π) and Range(π) and keeps track of the correspondence between points in the former and points in the latter. We recognize that in game $\mathcal{R}$ the association of domain points to range points was never actually used—all that was needed was to know *what* points had been admitted into the domain and what points had been admitted into the range. Thus in game $\mathcal{S}$ we keep these as multisets, $\mathcal{X}$ and $\mathcal{Y}$, and dispense with π. For $s \in [1..q-1]$, query $Q_s = (ty_s, N_s, i_s, M_s)$ and its response Z_s brings in one or two points into $\mathcal{X}$ (depending on whether or not the nonce N_s is new) and it brings in one or two point into into $\mathcal{Y}$ (again depending on whether or not the nonce N_s is new). There is one additional point, 0^n in $\mathcal{X}$, and there is one additional point, L, in $\mathcal{Y}$. The flag *bad* is set to `true` exactly when there is a repetition in $\mathcal{X}$ or a repetition in $\mathcal{Y}$—exactly what would have happened in game $\mathcal{R}$. To emphasize that a set is a multiset we put it in double braces.

To show Equation (7) we now aim to show that

$$\Pr[A^{\mathcal{S}} \text{ sets flag } bad] \quad \leq \quad 4q^2/2^n \tag{8}$$

The probability is over the internal coin tosses of A, the random choice of $Z_1, \ldots, Z_{q-1} \xleftarrow{\$} \{0,1\}^n$, and the random values L and $R(N)$, for each $N \in \mathcal{N}$. Since the adversary A is computationally unbounded one may assume it to be deterministic, eliminating that first set of coins. We now eliminate the second set of coins as well.

Consider two ways for flag *bad* to get set to `true` in game $\mathcal{S}$: in a *Z-collision* there is a collision between two Z_s-values, and in a *!Z-collision* flag *bad* gets set to `true` even though there is no collision among Z_s-values. The probability of a Z-collision is $\binom{q-1}{2}2^{-n}$. We may therefore modify game $\mathcal{S}$ to a game $\mathcal{S}'$ where $Z_1, \ldots, Z_s$ are selected as random but distinct points in $\{0,1\}^n$ knowing that the probability *bad* getting set to `true` in game $\mathcal{S}$ exceeds the probability of *bad* getting set to `true` in game $\mathcal{S}'$ by at most $\binom{q-1}{2}2^{-n}$. It is from this game, $\mathcal{S}'$, that we eliminate the coins $Z[]1, \ldots, Z_{q-1}$. Instead of showing that $\Pr[A^{\mathcal{S}'} \text{ sets } bad\ ] \leq \delta$ where the probability is over random distinct $\mathcal{Z} = (Z_1, \ldots, Z_{q-1})$, L and $R(N)$, for each $N \in \mathcal{N}$, we make the stronger assertion that for *any* fixed, distinct $\mathcal{Z} = (Z_1, \ldots, Z_{q-1})$ and resulting valid queries $(Q_1, \ldots, Q_q)$, the probability that *bad* gets set to `true` is at most δ. We rewrite this game in Figure 9 and call it game $\mathcal{T}$.

We now proceed with a somewhat tedious case analysis to see what is the probability that *bad* gets set to `true` in game $\mathcal{T}$. A point X is put into $\mathcal{X}$ due to one of 11 different causes. We identify these causes with the "type" of the point X. Referring to Figure 8 we see that X may be placed into $\mathcal{X}$ because:

- type 0: it is 0^n.
- type 1–9: it is a point X_s that resulted from a query of type $ty_s \in [1..9]$.
- type 10: it is $N \oplus L$ for some $N \in \mathcal{N}$.

Similarly, a point Y in is put into $\mathcal{Y}$ due to one of 11 different causes:

- type 0: it is L.
- type 1–9: it is a point Y_s that resulted from a query of type $ty_s \in [1..9]$.
- type 10: it is $R(N)$ for some $N \in \mathcal{N}$.

Each type-j point X, for $j \in [0..10]$, might collide with a type-j'-point X', for any $j' \in [0..J]$. There are therefore $1+2+\cdots+11 = 66$ possible types of collisions —pairings of types of points (X, X')—to consider. There are additional collisions in $\mathcal{Y}$ that we will have to account for as well.

We claim that each type of collision occurs with probability at most 2^{-n}.

We begin with (X, X') collisions. The X-value has associated to it constants (N, i, M, Z) and random variables L and $R = R(N)$. The X'-value has associated to it constants (N', i', M', Z') and random variables L and $R' = R(N')$. The validity conditions imply various restrictions on (N, i, M, Z) and (N', i', M', Z') that we shall need.

We enumerate the cases for $\mathcal{X}$-collisions (X, X'). With respect to $\mathcal{X}$-collisions, type-1 points and type-3 points are treated the same way. We can thus ignore type-3 points apart from their colliding with type-1 points.

(0,0) This type of collision is impossible because there is only one type-0 point.

(1,0) $\Pr[M \oplus iL \oplus R = 0^n] = 2^{-n}$ because L is random and $i \neq 0$.

(1,1) $\Pr[M \oplus iL \oplus R = M' \oplus i'L \oplus R'] = \Pr[(i \oplus i')L = (M \oplus M') \oplus (R \oplus R')] = 2^{-n}$ because either $i \neq i'$, in which case we rely on the randomness of L, or $N \neq N'$, in which case we count on the randomness of R, or $i = i'$ and $N = N'$ and $M \neq M'$ (by V1) and the probability of a collision is 0.

(2,0) $\Pr[M \oplus iL \oplus R \oplus L\mathtt{x}^{-1} = 0^n] = 2^{-n}$ because L is random.

(2,1) $\Pr[M \oplus iL \oplus R \oplus L\mathtt{x}^{-1} = M' \oplus i'L \oplus R' \oplus L] = \Pr[(M \oplus M') \oplus (R \oplus R') = (i \oplus i' \oplus \mathtt{x}^{-1}L)] \leq 2^{-n}$ because $i \oplus i' \oplus x^{-1} \neq 0$ by condition V0 (namely, $i \oplus i'$ begins with a zero-bit because $i < 2^{n-1}$ while and x^{-1} begins with a one-bit).

(2,2) $\Pr[M \oplus iL \oplus R \oplus L\mathtt{x}^{-1} = M' \oplus i'L \oplus R' \oplus L\mathtt{x}^{-1}] = \Pr[(i \oplus i')L = (M \oplus M') \oplus (R \oplus R')] \leq 2^{-n}$ as in case $(1,1)$.

(3,1) $\Pr[M \oplus iL \oplus R = M' \oplus i'L \oplus R'] = \Pr[(i \oplus i')L = (M \oplus M') \oplus R \oplus R'] \leq 2^{-n}$ because, by condition V3, $N \neq N'$ (so R and R' are independent) or $i \neq i'$.

(4,0) $\Pr[Z \oplus iL \oplus R = 0^n] = 2^{-n}$ because L is random.

(4,1) $\Pr[Z \oplus iL \oplus R = M' \oplus i'L \oplus R'] = \Pr[(Z \oplus M') \oplus (i \oplus i')L = R \oplus R'] \leq 2^{-n}$ because $N \neq N'$ or $i \neq i'$ or ($N = N'$ and $i = i'$ and $Z \neq M'$) by validity condition V2.

(4,2) $\Pr[Z \oplus iL \oplus R = M' \oplus i'L \oplus R' \oplus L\mathtt{x}^{-1}] = \Pr[(Z \oplus M) \oplus (i \oplus i' \oplus x^{-1})L = R \oplus R'] \leq 2^{-n}$ because $i \oplus i' \oplus x^{-1} \neq 0$ by condition V0.

(4,4) $\Pr[Z \oplus iL \oplus R = Z' \oplus i'L \oplus R'] = \Pr[(Z \oplus Z') \oplus (i \oplus i')L = R \oplus R'] \leq 2^{-n}$ because of condition V1.

(5,0) $\Pr[M \oplus iL = 0^n] = 2^{-n}$ because L is random and $i \neq 0$.

(5,1) $\Pr[M \oplus iL = M' \oplus i'L \oplus R] = 2^{-n}$ by the randomness of R. Similarly for cases (5,2) and (5,4).

(5,5) $\Pr[M \oplus iL = M' \oplus i'L] = \Pr[(M \oplus M') = (i \oplus i')L] \leq 2^{-n}$ because, by condition V1, $i \neq i'$ (in which case we get 2^{-n}) or $i = i'$ and $M \neq M'$ (in which case we get 0).

(6,0) $\Pr[M = 0^n] = 0$ by condition V5.

(6,1) $\Pr[M = M' \oplus i'L \oplus R'] = 2^{-n}$ by the randomness of R. Similarly for cases (6,2) and (6,4).

(6,5) $\Pr[M = M' \oplus i'L] = 2^{-n}$ by the randomness of L.

(6,6) $\Pr[M = M'] = 0$ by condition $V1$.

(7,0) $\Pr[M \oplus L \cdot \mathtt{x}^{-1} = 0^n] = 2^{-n}$ by the randomness of L.

(7,1) $\Pr[M \oplus L \cdot \mathtt{x}^{-1} = M' \oplus i'L \oplus R] = 2^{-n}$ by the randomness of R. Similarly for cases (7,2) and (7,4).

(7,5) $\Pr[M \oplus L \cdot \mathtt{x}^{-1} = M' \oplus i'L] = \Pr[(M \oplus M') = (i' \oplus \mathtt{x}^{-1})L] \leq 2^{-n}$ because $i \neq \mathtt{x}^{-1}$.

(7,6) $\Pr[M \oplus L \cdot \mathtt{x}^{-1} = M'] \leq 2^{-n}$ by the randomness of L.

(7,7) $\Pr[M \oplus L \cdot \mathtt{x}^{-1} = M' \oplus L \cdot \mathtt{x}^{-1}] = 0$ by condition V1.

(8,0) $\Pr[M \oplus R = 0^n] = 2^{-n}$ by the randomness of R.

(8,1) $\Pr[M \oplus R = M' \oplus i'L \oplus R'] = 2^{-n}$ by the randomness of L. Similarly for (8,2), (8,4), (8,5)

(8,6) $\Pr[M \oplus R = M'] = 2^{-n}$ by the randomness of R. Similarly for (8,7).

(8,8) $\Pr[M \oplus R = M' \oplus R'] = \Pr[(M \oplus M') = (R \oplus R')] \leq 2^{-n}$ since, by condition V1, $N \neq N'$ (in which case we get 2^{-n}) or $N = N'$ and $M \neq M'$ (in which case we get 0).

(9,0) $\Pr[M \oplus R \oplus L\mathtt{x}^{-1} = 0^n] = 2^{-n}$ by the randomness of R.

(9,1) $\Pr[M \oplus R \oplus L\mathtt{x}^{-1} = M' \oplus i'L \oplus R'] = \Pr[(M \oplus M') \oplus (i' \oplus \mathtt{x}^{-1})L = (R \oplus R')] \leq 2^{-n}$ because $i' \neq \mathtt{x}^{-1}$ and L is random.

(9,2) $\Pr[M \oplus R \oplus L\mathtt{x}^{-1} = M' \oplus i'L \oplus R' \oplus L\mathtt{x}^{-1}] = \Pr[(M \oplus M') \oplus (R \oplus R') = i'L] \le 2^{-n}$ by the randomness of L.

(9,4) $\Pr[M \oplus R \oplus L\mathtt{x}^{-1} = Z' \oplus i'L \oplus R'] = \Pr[(M \oplus Z') \oplus (R \oplus R') = (\mathtt{x}^{-1} \oplus i')L] \le 2^{-n}$ since $i' \ne \mathtt{x}^{-1}$.

(9,5) $\Pr[M \oplus R \oplus L\mathtt{x}^{-1} = i'M' \oplus i'L] \le 2^{-n}$ by the randomness of R. Similarly for (9,6) and (9,7).

(9,8) $\Pr[M \oplus R \oplus L\mathtt{x}^{-1} = M' \oplus R'] \le 2^{-n}$ by the randomness of L

(9,9) $\Pr[M \oplus R \oplus L\mathtt{x}^{-1} = M' \oplus R' \oplus L\mathtt{x}^{-1}] = \Pr[M \oplus M' = R \oplus R'] \le 2^{-n}$ by condition V1.

(10,0) $\Pr[N \oplus L = 0^n] = 2^{-n}$ by the randomness of L.

(10,1) $\Pr[N \oplus L = M' \oplus i'L \oplus R] = 2^{-n}$ by the randomness of R. Similarly for cases (10,2) and (10,4).

(10,5) $\Pr[N \oplus L = M' \oplus i'L] = \Pr[N \oplus M' = (i' \oplus 1)L] = 2^{-n}$ because $i \ne 1$ by condition V4.

(10,6) $\Pr[N \oplus L = M'] = 2^{-n}$ by the randomness of L.

(10,7) $\Pr[N \oplus L = M' \oplus L\mathtt{x}^{-1}] = \Pr[N \oplus M' = (1 \oplus \mathtt{x}^{-1})L] = 2^{-n}$ by the randomness of L.

(10,8) $\Pr[N \oplus L = M' \oplus R'] = 2^{-n}$ by the randomness of R'. Similarly for case (10,9).

(10,10) $\Pr[N \oplus L = N' \oplus L] = 0$ because $N \ne N'$.

Since there are at most $1 + (q-1) + (q-1) = 2q - 1$ points in the multiset $\mathcal{X}$ we conclude that the probability of an $\mathcal{X}$ collision is at most $\binom{2q-1}{2} 2^{-n} \le 2q^2/2^n$.

Now we look at collisions in $\mathcal{Y}$. These are fewer cases because the assignments for queries of 1 and 4 are identical (let 1 be the representative), as are the assignments for queries of types 2, 3, 5, 6, 7, 8, 9 are identical (let 2 be the representative). Recall that we earlier assumed that $Z_1, \ldots, Z_{q-1}$ values are distinct, but we have still to account for the associated collision probability.

(0,0) This type of collision is impossible because there is only one is impossible because there is only one type-0 point.

(1,0) $\Pr[Z \oplus iL \oplus R = L] = 2^{-n}$ by the randomness of R.

(1,1) $\Pr[Z \oplus iL \oplus R = Z' \oplus i'L \oplus R'] = \Pr[(Z \oplus Z') \oplus (i \oplus i')L = R \oplus R'] \le 2^{-n}$ because either $i \ne i'$ or $N \ne N'$ or $Z \ne Z'$.

(2,0) $\Pr[Z = L] = 2^{-n}$ because L is random.

(2,1) $\Pr[Z = Z' \oplus i'L \oplus R'] = 2^{-n}$ by the randomness of R.

(2,2) $\Pr[Z \oplus Z'] = 2^{-n}$ (where these collisions are considered before fixing Z-values).

(10,0) $\Pr[R = L] = 2^{-n}$.

(10,1) $\Pr[R = M' \oplus i'L \oplus R'$ by the randomness of L.

(10,2) $\Pr[R = Z] = 2^{-n}$ by the randomness of R.

(10,10) $\Pr[R = R'] = 2^{-n}$ since R and R' correspond to distinct N and N'.

Overall, we get a probability of a $\mathcal{Y}$ collision (including the initial assignment to $\mathcal{Z}$) of at most $\binom{2q-1}{2} 2^{-n} \le 2q^2/2^n$. The probability that *bad* gets set in game $\mathcal{S}$ is thus at most $4q^2/2^n$, completing the proof of Lemma 4.