An earlier version of this paper appears in *Advances in Cryptology — EUROCRYPT '06*, Lecture Notes in Computer Science, vol. 4004, Springer, 2006. This is the full version of that paper.

Deterministic Authenticated-Encryption

A Provable-Security Treatment of the Key-Wrap Problem

P. ROGAWAY[*] T. SHRIMPTON[†]

Aug 20, 2007

Abstract

Standards bodies have been addressing the *key-wrap* problem, a cryptographic goal that has never received a provable-security treatment. In response, we provide one, giving definitions, constructions, and proofs. We suggest that key-wrap's goal is security in the sense of *deterministic authenticated-encryption* (DAE), a notion that we put forward. We also provide an alternative notion, a *pseudorandom injection* (PRI), which we prove to be equivalent. We provide a DAE construction, SIV, analyze its concrete security, develop a blockcipher-based instantiation of it, and suggest that the method makes a desirable alternative to the schemes of the X9.102 draft standard. The construction incorporates a method to turn a PRF that operates on a string into an equally efficient PRF that operates on a vector of strings, a problem of independent interest. Finally, we consider IV-based authenticated-encryption (AE) schemes that are maximally forgiving of repeated IVs, a goal we formalize as *misuse-resistant AE*. We show that a DAE scheme with a vector-valued header, such as SIV, directly realizes this goal.

Keywords: Authenticated encryption, cryptographic definitions, cryptographic standards, key wrapping, modes of operation, provable security, secret-key cryptography, symmetric encryption, X9.102.

[*] Dept. of Computer Science, University of California at Davis, Davis, California 95616, USA; and Dept. of Computer Science, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand.

[†] Department of Computer Science, Portland State University, Portland, Oregon 97201, USA

Contents

1 Introduction 1

2 The Draft X9.102 Standard 3

3 DAE Security 5

4 Building a DAE Scheme: The SIV Construction 6

5 Enriching a PRF to take Vectors of Strings as Input: The S2V Construction 9

6 The SIV Mode of Operation 11

7 Misuse-Resistant AE 12

8 The PRI Characterization of DAE Security 14

Acknowledgments 16

References 16

A Critique of the Draft X9.102 Standard 18

B All-in-One vs. Two-Requirement Notions for AE 22

C DAEs Achieve Semantic Security when Plaintexts Carry a Key 25

D Building a DAE Scheme: The PTE Constructions 27

E Proof of Security for S2V 31

F Key Rap 35

1 Introduction

The American Standards Committee Working Group X9F1 has proposed four *key-wrap* schemes in a draft standard known as ANS X9.102, and NIST has promulgated a request for comments on the proposal [13]. The S/MIME working group of the IEEE had earlier adopted a key-wrap scheme [17], and their discussions on this topic go back to at least 1997 [37]. NIST is considering specifying a key-wrap mechanism in their own series of recommendations [M. Dworkin, personal communications]. But despite all this, the key-wrap goal would seem to be essentially unknown to the cryptographic community. No published paper analyzes any key-wrap scheme, and there is no formal definition for key wrap in the literature, let alone any proven-secure scheme. Consequently, the goal of this paper is to put the key-wrap problem on a proper, provable-security footing. In the process, we will learn quite a bit that's new about authenticated-encryption (AE).

Before proceeding it may be useful to give a very informal description of the key-wrap goal, echoing the wording in [13, p. 1]. A key-wrap scheme is a kind of shared-key encryption scheme. It aims to provide "privacy and integrity protection for specialized data such as cryptographic keys, . . . without the use of nonces" (meaning counters or random bits). So key-wrap's raison d'être is to remove AE's reliance on a nonce or random bits. At least in the context of transporting cryptographic keys, a deterministic scheme should be just as good as a probabilistic one, anyway. Another goal of key wrap is to provide "integrity protection . . . for cleartext associated data, . . . which will typically contain control information about the wrapped key" [13, p. 1].

CONTRIBUTIONS. We begin with a critique of the X9.102 key-wrap schemes, identifying the basic characteristics of each of the four algorithms. Overall, we find the proposed mechanisms somewhat disappointing in terms of usage restrictions, efficiency, and foundations. That said, we break none of the four schemes, and we owe this work to their existence. See Section 2 and Appendix A.

Guided by the proposed schemes, we offer a definition for what a key-wrap scheme should do. We call the goal *deterministic authenticated-encryption* (DAE). A thesis underlying our work is that the goal of a key-wrap scheme *is* DAE. In a DAE scheme, encryption deterministically turns a key, a header, and a message into a ciphertext. The header (which may be absent, a string, or even a vector of strings) is authenticated but not encrypted. To define security, the adversary is presented either a real encryption oracle and a real decryption oracle (both are deterministic), or else a bogus encryption oracle that just returns random bits and a bogus decryption oracle that always returns an indication of invalidity. For a good DAE scheme, the adversary should be unable to distinguish these possibilities. See Section 3.

Next we provide a DAE construction, SIV. (The acronym stands for *Synthetic IV*, where *IV* stands for *Initialization Vector*.) The construction combines a conventional IV-based encryption scheme (eg, CTR mode [27]) and a special kind of pseudorandom function (PRF)—one that takes a vector of strings as input. To encrypt, apply the PRF to the header and the message and use the result as the IV of the encryption scheme. We prove that SIV is a good DAE, assuming its components are secure. See Section 4.

In practice one would want to realize SIV from a blockcipher, and so we show how to turn a PRF f that operates on a single string into a PRF f^* that takes a vector of strings. Under our S2V construction, the cost of computing the PRF $f^* = \mathrm{S2V}[f]$ on a vector $X = (X_1, \ldots, X_n)$ is at most the total cost to compute f on each component X_i, and it can be considerably less, as the contribution from a component X_i can be precomputed if it is to be held constant. See Section 5.

For a concrete alternative to the X9.102 schemes, we suggest to instantiate SIV using modes CTR and $\mathrm{CMAC}^* = \mathrm{S2V}[\mathrm{CMAC}]$, where CTR is counter mode [27] and CMAC is an arbitrary-input-length variant of the CBC MAC [28]. The specified mechanism removes unnecessary usage restrictions, improves efficiency, and provides provable security. See Section 6.

Applications of DAEs go beyond the wrapping of keys. Many IV-based encryption schemes, such as CBC, require an adversarially unpredictable IV. Experience has shown that implementers and protocol designers often supply an incorrect IV, such as a constant or counter. In a *misuse-resistant* AE scheme the aim is to do as well as possible with whatever IV is provided. We formalize this goal and show that a DAE scheme that takes a vector-

DAE	*Deterministic authenticated-encryption*. Section 3. The main notion being investigated by this paper.
PRI	*Pseudorandom injection*. Section 8. Like a strong PRP but injective instead of bijective. DAE-security and PRI-security differ by an amount that vanishes exponentially in the *stretch* of the scheme.
MRAE	*Misuse-resistant AE,* Section 7. Strengthening of the usual definition of nonce-based AE to speak to what happens when a nonce gets reused. Easily constructed from a DAE that handles vector-valued headers.
SIV	*Synthetic IV*. Section 4. Makes a DAE by first applying a PRF to the header and message to get an IV. Also our blockcipher-based mode of operation that instantiates this using CTR mode and S2V-applied-to-CMAC.
S2V	*String to vector*. Section 5. Turns a PRF that takes a string as input into a PRF that takes a vector of strings as input. More efficient than an encoding-based approach.
PTE	*Pad-then-encipher*. Appendix D. Makes a DAE by padding the input and then enciphering. Two versions, depending on how headers are handled. Most natural approach to DAE, but less desirable than SIV.

Figure 1: Roadmap of the new notions and schemes. The first three entries are security notions; the next three entries are schemes. The section reference indicates where the definition of the notion or scheme can be found.

valued header provides an immediate solution: just regard the IV as one component of the header. Adopting this viewpoint, SIV can be regarded as an IV-based AE scheme, one as efficient with respect to blockcipher calls as conventional two-pass AE schemes like CCM [29] but more resilient to IV misuse. See Section 7.

Finally, we give an alternative characterization of DAEs. A *pseudorandom injection* (PRI) is like a blockcipher except that the ciphertext may be longer than the plaintext (also, the message space may be richer than $\{0,1\}^n$ for some fixed n, and a header may be provided). We prove PRIs equivalent to DAEs, up to a term that is negligible when the PRI is adequately length-increasing.

Our definition of DAE merges the traditionally separate privacy and authenticity requirements of an AE scheme. It is possible to split the definition into separate privacy and authenticity goals and require both. Doing this yields an equivalent definition. Similarly, the separate privacy and authenticity requirements normally used to define AE [6, 8, 19, 20, 31, 33] can be merged into a single, unified goal. The all-in-one approach for defining AE seems to us simpler and more elegant than giving separate privacy and authenticity definitions and then asking for both. See Appendix B.

A reason for doing key wrap (DAE) instead of conventional (probabilistic) AE is the intuition that, if the plaintext carries a key, there shouldn't be any need to inject additional randomness into the encryption process. One can formalize and prove this intuition, establishing, in effect, the semantic security of DAE for the context in which keys are embedded into plaintexts. A DAE scheme cannot by itself achieve semantic security because it is deterministic—we are saying that a random enough message space compensates for this, letting you recover the equivalent of semantic security. See Appendix C.

Besides SIV we also provide a second construction, one that uses different primitives. The *pad-then-encipher* scheme, PTE, is based on an enciphering scheme (ie, a length-preserving encryption scheme, like CMC [16]). One pads the plaintext (eg, appending 0-bits) before enciphering. We prove the security of PTE. We investigate the pad-then-encipher is the paradigm because it seems to us the most natural approach to solving the key-wrap problem, as well as the approach that underlies two of the X9.102 schemes. See Appendix D.

ROADMAP. Given the number of new acronyms, definitions, and schemes introduced in this paper, the table of Figure 1 may provide a helpful summary. For the security notions, lower-case labels (eg, dae) are used as a superscript for advantage measures (eg, $\mathbf{Adv}_{\Pi}^{\mathrm{dae}}$) while their upper-case counterpart (eg, DAE) are used in English prose. Our table omits mention of notions that are standard or whose mention is confined to the appendices.

WHY THIS GOAL? There are two main reasons to prefer DAE over conventional (probabilistic or stateful) AE. First, DAE saves one from having to introduce random bits or state in contexts where these measures are infeasible or unnecessary. Relatedly, DAE saves on bandwidth, since no nonce or random value need be sent.

That said, in many contexts where one would think to use key wrap, one *can* use a conventional AE scheme, instead. This does not make studying the key-wrap problem pointless. First, it clarifies the relationship between key wrap and conventional AE. Second, DAE leads to misuse-resistant AE, and methods that achieve this aim make practical alternatives to conventional (not misuse-resistant) two-pass AE methods. Finally, practitioners have already "voted" for key-wrap by way of protocol-design and standardization efforts, and it is simply not productive to say "use a conventional AE scheme" after this option has been rejected.

THE SIGNIFICANCE OF HEADERS. We emphasize that our formalization of DAEs includes a header (also called a tweak or associated-data). For cryptographic practice, allowing a header seems to be almost essential. Network security protocols require sending packets only portions of which are encrypted, but all of which must be authenticated and bound together. Good security practice requires keys to be bound to control information such as expiration date and permitted usage, and the binding of keys to such control information has strongly informed security architecture (eg, IBM's cryptographic control vectors [25]). Regarding headers as vectors facilitates both efficiency advantages and a cleaner abstraction boundary.

FURTHER RELATED WORK. AE goals were formalized over a series of papers [6, 8, 20, 31, 33]. The idea of binding the encryption process to unencrypted strings is folklore, with recent work in this direction including [23, 31, 36]. Bellare and Rogaway [8] investigate the paradigm of adding randomness or redundancy to a plaintext and then enciphering it, an approach related to the ideas and results of Appendices C and D. Russell and Wong [35] introduce a completely different approach for dealing with the encryption of low-entropy messages, and Dodis and Smith [12] extend this entropy-based approach. Phan and Pointcheval [30] study relationships among security notions for conventional (length-preserving and headerless) ciphers. The SIV construction resembles the AE scheme EAX [9]. A less ambitious relaxation on IV requirements than that formalized as misuse-resistant encryption is given in [32]. The proceedings version of this paper was published as [34].

2 The Draft X9.102 Standard

Four key-wrap schemes are defined in the draft ANS X9.102 standard [13]. The schemes are called AESKW and TDKW (which are essentially the same scheme, the former using AES and the later using triple-DES), AKW1, and AKW2. Scheme AESKW is based on a six-round, non-standard Feistel network. It was first proposed by NIST and is pictured and specified in Figure 7. Scheme AKW1 involves two layers of CBC encryption and one application of SHA1. It was developed by the S/MIME working group of the IETF [17] and is pictured in Figure 8. Scheme AKW2 involves a CBC encryption layer and a CBC MAC layer. It was designed to accommodate legacy financial-services devices and is pictured in Figure 9.

EXPLANATION OF SUMMARY TABLE. Here and in Figure 2 we summarize basic properties of the X9.102 schemes [13]; see Appendix A for further discussion. The columns represent three of the four schemes (TDKW is omitted because of its similarity to AESKW). Let us explain the meaning of the table's rows. **Goal**: This is our understanding of the mechanism's aim. Schemes AESKW and TDKW seem intended to achieve DAE, the focus of this paper. We don't know if the schemes actually achieve this goal (we expect that they do). Scheme AKW1 is described as a probabilistic scheme that aims to achieve (probabilistic) AE, the original notion of AE put forward in [6, 8, 20]. But we explain in Appendix A why we view AKW1 as a peculiar approach for trying to achieve probabilistic AE. Scheme AKW1 seems to reflect no single and ascertainable cryptographic goal (this sometimes happens in committee-based design). Scheme AKW2 achieves only a specialized (not of general interest) notion of deterministic privacy, along with a deterministic version of authenticity-of-ciphertexts. The combination of these aims is substantially weaker than the goal of being a DAE. **Message space:** The message space over which the scheme is defined. **Header space:** The space of headers (also called tweaks or associated data) over which the scheme is defined. **But:** Specifies any technical requirement about the relationship between

	AESKW	**AKW1**	**AKW2**
Goal	DAE	see text	see text
Message space	$X \in \{0,1\}^{[0..2^{38}-64]}$	$X \in (\text{BYTE}^8)^{[1..2^{16}]}$	$X \in (\text{BYTE}^8)^{[2..2^{16}]}$
Header space	$H \in \text{BYTE}^{\le 255}$	not supported	$H \in (\text{BYTE}^8)^{+}$
But	$H \neq \varepsilon$ or $X \neq \varepsilon$	nothing	nothing
Ciphertext bits	$64\lceil(\lvert H\rvert + \lvert X\rvert)/64\rceil + 64$	$\lvert X\rvert + 128$	$\lvert X\rvert + Tlen$ where $Tlen \in [32..64]$
Expansion	$\lvert H\rvert + 64$ to $\lvert H\rvert + 127$	128	32–64
Blockcipher	AES (any key length)	TDEA (two-key or three-key)	TDEA (two-key or three-key)
Forging prob	2^{-63}	2^{-64}	2^{-Tlen} with $Tlen \in [32..64]$
Maxqueries	2^{48}	2^{32}	2^{32}
Overhead	$12\times$	$2\times + 3$ (plus SHA1 overhead)	$2\times$ on message, $1\times$ on header
Key usage	one blockcipher key	one blockcipher key	two blockcipher keys
Parallelizable?	no	no	no
Preprocess header?	no	not applicable	yes
Expedited auth?	no	no	yes
Provably secure?	no	no	no

Figure 2: Basic characteristics of key-wrap mechanisms AESKW, AKW1, and AKW2 from the X9.102 draft. We omit TDKW because of its similarity to AESKW. Explanations of rows are given in the body.

the message space and the header space. **Ciphertext bits:** Ciphertext length as a function of already-named values. **Expansion:** How much longer the ciphertext is than the plaintext. **Blockcipher:** The underlying blockcipher. **Forging prob:** The target forging probability asserted by the spec. We are not asserting that the scheme actually achieves this value. **Maxqueries:** The maximum number of plaintext values that may be encrypted in an implementation compliant with the spec. We are not asserting that the scheme is actually secure up to this value. **Overhead:** The computational overhead, measured in blockcipher calls per block of data (message or header). Scheme AKW1 incurs additional overhead for applying SHA1 to the message. **Key usage:** The number of blockcipher keys that key the blockcipher calls. **Parallelizable?** Can the computation-time of the mechanism be arbitrarily sped up by adding additional hardware? **Preprocess header?** Can a fixed header be cryptographically processed just once, as opposed to dealing with it for each and every message? **Expedited auth?** Is it faster to see if a ciphertext is inauthentic than to fully decrypt it? **Provably secure?** Does the mechanism enjoy any provable-security guarantee? That is, has a proof of security been offered, under standard or reasonable assumptions, that the mechanism achieves some well-defined and desirable goal?

INTERPRETATION. Given Figure 2 and the associated discussion in Appendix A, our conclusion is that none of the X9.102 algorithms are mature. Most severely, none has been proven secure—and, prior to this paper, there was not even a clear target for a security proof. Each scheme has multiple problems from among the following: a restricted message space; an inability to handle an associated header; a restricted header space; ciphertext lengths that grow with the header length (even though the header is only authenticated); a large number of blockcipher calls; mysterious aspects of the construction (eg, the byte-reversals or xoring-in counters); and use of cryptographic primitives beyond a blockcipher. For a modern encryption scheme one might reasonably hope for a formally defined and provably achieved security goal, an aesthetic construction coming out of an enunciated paradigm, message headers being supported and the message space and header space being large and natural sets, message expansion of some fixed value, one or two blockcipher calls per block, and further efficiency characteristics (like being able to cheaply handle static headers).

That said, we do not mean to be overly negative about the X9.102 schemes. We have not broken any of them, and the six-round Feistel-network of AESKW/TDKW could have beyond-birthday-phenomenon security. The standardization effort has engendered our own work, and it is very hard to design a correct key-wrap scheme prior to having supporting definitions and results. Finally, it is hard to design a correct key-wrap scheme if the abstraction boundary one is thinking in terms of is a blockcipher, too low-level a tool to make a convenient conceptual starting point.

3 DAE Security

NOTATION. For a distribution $\mathcal{S}$ let $S \xleftarrow{\$} \mathcal{S}$ mean that S is selected randomly from $\mathcal{S}$ (if $\mathcal{S}$ is a finite set the assumed distribution is uniform). All strings are binary strings. When X and Y are strings we write $X \| Y$ for their concatenation. When $X \in \{0,1\}^*$ is a string $|X|$ is its length and, if $1 \le i \le j \le |X|$, then $X[i..j]$ is the substring running from its i^{th} to j^{th} characters, or the empty string ε otherwise. By a vector we mean a sequence of zero or more strings, and we write $\{0,1\}^{**}$ for the space of all vectors. We write a vector as $X = (X_1, \ldots, X_n)$ where $n = |X|$ is its number of components. If $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_m)$ are vectors then X, Y is the vector $(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$. In pseudocode, Boolean variables are silently initialized to `false`, sets are initialized to the empty set, and partial functions are initialized to everywhere undefined (set to `undef`). An *adversary* is an algorithm with access to one or more oracles, which we write as superscripts. By $A^{\mathcal{O}} \Rightarrow 1$ we mean the event that adversary A, running with its oracle $\mathcal{O}$, outputs 1. When an adversary has an oracle with an expressed domain D we understand that the oracle returns the distinguished value $\perp$, read as *invalid*, if the adversary asks a query outside of D.

SYNTAX. A scheme for *deterministic authenticated-encryption*, or DAE, is a tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The *key space* $\mathcal{K}$ is a set of strings or infinite strings endowed with a distribution. For a practical scheme there must be a probabilistic algorithm that samples from $\mathcal{K}$, and we identify this algorithm with the distribution it induces. The *encryption algorithm* $\mathcal{E}$ and *decryption algorithm* $\mathcal{D}$ are deterministic algorithms that take an input in $\mathcal{K} \times \{0,1\}^{**} \times \{0,1\}^*$ and return either a string or the distinguished value $\perp$. We write $\mathcal{E}_K^H(X)$ or $\mathcal{E}_K(H, X)$ for $\mathcal{E}(K, H, X)$ and $\mathcal{D}_K^H(Y)$ or $\mathcal{D}_K(H, Y)$ for $\mathcal{D}(K, H, Y)$. We assume there are sets $\mathcal{H} \subseteq \{0,1\}^{**}$, the *header space*, and $\mathcal{X} \subseteq \{0,1\}^*$, the *message space*, such that $\mathcal{E}_K^H(X) \in \{0,1\}^*$ iff $H \in \mathcal{H}$ and $X \in \mathcal{X}$. We assume that $X \in \mathcal{X} \Rightarrow \{0,1\}^{|X|} \subseteq \mathcal{X}$. The *ciphertext space* is $\mathcal{Y} = \{\mathcal{E}_K^H(X)\colon\ K \in \mathcal{K},\ H \in \mathcal{H},\ X \in \mathcal{X}\}$. We require $\mathcal{D}_K^H(Y) = X$ if $\mathcal{E}_K^H(X) = Y$, and $\mathcal{D}_K^H(Y) = \perp$ if there is no such X. It will be our convention that $\mathcal{E}_K^H(\perp) = \mathcal{D}_K^H(\perp) = \perp$ for all $K \in \mathcal{K}$ and $H \in \mathcal{H}$. For any $K \in \mathcal{K}$, $H \in \mathcal{H}$, and $X \in \mathcal{X}$, we assume that $|\mathcal{E}_K^H(X)| = |X| + e(H, X)$ for a function $e\colon \{0,1\}^{**} \times \{0,1\}^* \to \mathbb{N}$ where $e(H, X)$ depends only on the number of components of H, the length of each of these components, and the length of X. The function e is called the *expansion function* of the DAE scheme. Often we are concerned with the minimum expansion that might arise, and so define the number $s = \min_{H \in \mathcal{H}, X \in \mathcal{X}}\{e(H, X)\}$ as the *stretch* of the scheme.

Among what is formalized above: (1) encryption and decryption are given by algorithms, not just functions; (2) trying to encrypt something outside of the header space or message space returns $\perp$; (3) trying to decrypt something that isn't the encryption of anything returns $\perp$; (4) if you can encrypt a string of some length you can encrypt all strings of that length; and (5) the length of a ciphertext exceeds the length of the plaintext by an amount that depends on, at most, the length of the plaintext and the length of the components of the header.

A DAE is *length-preserving* if $e(H, X) = 0$ for all $H \in \mathcal{H}$, $X \in \mathcal{X}$. An *enciphering scheme* is a length-preserving DAE. A *tweakable blockcipher* is an enciphering scheme where the plaintext space is $\mathcal{X} = \{0,1\}^n$ for some $n \ge 1$. A *blockcipher* is a tweakable blockcipher where the header space $\mathcal{H} = \{\varepsilon\}$ is a singleton set; as such, we omit mention of it and write $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$.

SECURITY. We now give our formalization for DAE security.

Definition 1 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a DAE scheme with header space $\mathcal{H}$, message space $\mathcal{X}$, and expansion function e. The **DAE-advantage** of adversary A in breaking Π is defined as

$$\mathbf{Adv}_{\Pi}^{\text{dae}}(A) \;=\; \Pr\left[K \xleftarrow{\$} \mathcal{K}:\; A^{\mathcal{E}_K(\cdot,\cdot),\, \mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right]. \qquad \blacksquare$$

On query $H \in \mathcal{H}$, $X \in \mathcal{X}$, the adversary's *random-bits* oracle $\$(\cdot, \cdot)$ returns a random string of length $|X| + e(H, X)$. As always, oracle queries outside the specified domain return $\perp$. The $\perp(\cdot, \cdot)$ oracle returns $\perp$ on every input. We assume that the adversary does not ask (H, Y) of its right (ie, second) oracle if some previous left (ie, first) oracle query (H, X) returned Y; does not ask (H, X) of its left oracle if some previous right-oracle query (H, Y) returned X; does not ask left queries outside of $\mathcal{H} \times \mathcal{X}$; and does not repeat a query. The last two assumptions are without loss of generality, as an adversary that violated any of these constraints could be replaced by a more efficient and equally effective adversary (in the $\mathbf{Adv}_{\Pi}^{\mathrm{dae}}$-sense) that did not. The first two assumptions are to prevent trivial wins.

DISCUSSION. The DAE-notion of security directly captures the amalgamation of privacy and authenticity. Assume that $\mathbf{Adv}_{\Pi}^{\mathrm{dae}}(A)$ is insignificantly small for any reasonable adversary. Then, for privacy, we know that any sequence of distinct $\mathcal{E}_K$-queries results in a distribution on outputs resembling a distribution on outputs that depends only on the length of each query (in fact, the outputs look like random strings of the appropriate lengths). For authenticity we have that, despite the ability to perform a chosen-plaintext attack (as provided by the $\mathcal{E}_K$ oracle), we are unable to come up with a new query Y for which $\mathcal{D}_K^H(Y) \neq \perp$.

It is possible to disentangle the privacy and authenticity notions in the DAE definition, defining separate notions for deterministic privacy and deterministic authenticity. We do this in Appendix B, and explain why asking for both of these conditions is equivalent to DAE. While the traditional approach for defining AE has been to split the goal into two separate properties, the unified definition seems to us nicer and more succinct.

We point out that the DAE notion does not formalize the idea that the party that produces a valid ciphertext (a value that decrypts to something other than $\perp$) necessarily *knows* the underlying key K. One could formalize this, but it would not coincide with DAE. Sometimes the key-wrap goal has been described in these terms. We suspect that when security-designers speak of having to know the key in order to produce a valid ciphertext what they typically mean is not a proof of knowledge, but just the inability for a party to produce a valid ciphertext in the absence of the key. It is the latter notion that is well captured by our DAE definition.

4 Building a DAE Scheme: The SIV Construction

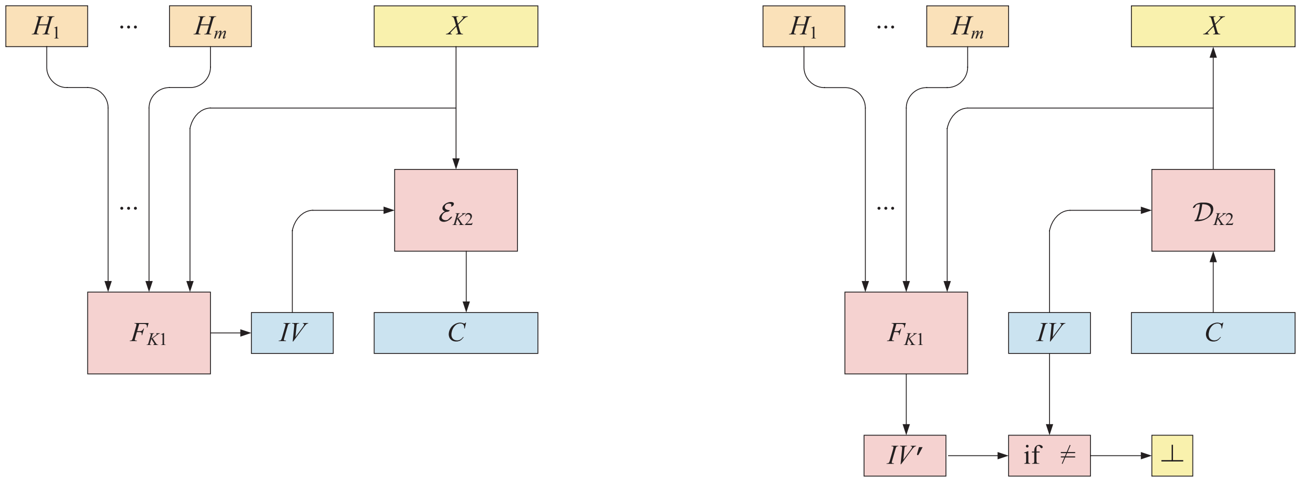
CONVENTIONAL IV-BASED ENCRYPTION SCHEMES. Encryption modes like CBC and CTR are what we call *conventional* IV-based encryption schemes. Such a scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is syntactically similar to a DAE but in this context the header space $\mathcal{H}$ is a set of strings and is renamed the *IV space*, $\mathcal{IV}$. We expect only privacy in a conventional IV-based encryption scheme, and demand a random IV. This makes the security notion rather weak, but sufficient for our purposes. The following definition captures the desired notion.

Fix a conventional IV-based encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with IV-space $\mathcal{IV} = \{0,1\}^n$. For simplicity, assume Π is length-preserving. Let $\mathcal{E}^\$$ be the probabilistic algorithm defined from $\mathcal{E}$ that, on input $K \in \mathcal{K}$ and $M \in \{0,1\}^*$, chooses an $IV \xleftarrow{\$} \{0,1\}^n$, computes $C \leftarrow \mathcal{E}_K^{IV}(M)$ and returns $IV \parallel C$. Then we define the advantage of adversary A in violating the privacy of Π by

$$\mathbf{Adv}_{\Pi}^{\mathrm{priv\$}}(A) \quad = \quad \Pr\left[K \xleftarrow{\$} \mathcal{K}:\ A^{\mathcal{E}_K^\$(\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$(\cdot)} \Rightarrow 1\right]$$

where the $\$(\cdot)$ oracle, on input M, returns a random string of length $n + |M|$. We assume that the adversary never asks a query M outside of the message space $\mathcal{X}$ of Π.

ARBITRARY-INPUT PSEUDORANDOM FUNCTIONS. Fix nonempty sets $\mathcal{K}$ and $\mathcal{X}$, the first being finite or otherwise endowed with a distribution and the second being finite or countably infinite. A *pseudorandom function* (PRF) is a map $F\colon\ \mathcal{K} \times \mathcal{X} \to \{0,1\}^n$ for some $n \geq 1$. We write $F_K(X)$ for $F(K, X)$. Let $\mathrm{Func}(\mathcal{X}, \mathcal{Y})$ be the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$ and let $\mathrm{Func}(\mathcal{X}, n) = \mathrm{Func}(\mathcal{X}, \{0,1\}^n)$. Regarding a function as the key, we can consider $\mathrm{Func}(\mathcal{X}, n)$ to be a PRF; to each $X \in \mathcal{X}$ associate a random string in $\{0,1\}^n$. Let A be an



```
**Algorithm** $\widetilde{\mathcal{E}}_{K1,K2}(H, X)$
$IV \leftarrow F_{K1}(H, X)$
$C \leftarrow \mathcal{E}^{IV}_{K2}(X)$
**return** $Y \leftarrow IV \parallel C$
```

```
**Algorithm** $\widetilde{\mathcal{D}}_{K1,K2}(H, Y)$
**if** $|Y| < n$ **then return** $\perp$
$IV \leftarrow Y[1\,..\,n],\quad C \leftarrow Y[n+1\,..\,|Y|]$
$X \leftarrow \mathcal{D}^{IV}_{K2}(C)$
$IV' \leftarrow F_{K1}(H, X)$
**if** $IV = IV'$ **then return** $X$ **else return** $\perp$
```

Figure 3: The SIV construction. The left side illustrates and defines encryption, the right side, decryption. The header is $H = (H_1, \ldots, H_m)$, the plaintext is X, the key is $(K1, K2)$, and the ciphertext is $Y = IV \parallel C$. Function $F\colon \mathcal{K}_1 \times \{0,1\}^{**} \rightarrow \{0,1\}^n$ is a PRF and $(\mathcal{K}_2, \mathcal{E}, \mathcal{D})$ is an IV-based encryption scheme, such as CTR mode.

adversary. The advantage of A in violating the pseudorandomness of F is

$$\mathbf{Adv}^{\mathrm{prf}}_F(A) = \Pr\left[K \leftarrow \mathcal{K}:\ A^{F_K(\cdot)} \Rightarrow 1\right] - \Pr\left[\rho \xleftarrow{\$} \mathrm{Func}(\mathcal{X}, n):\ A^{\rho(\cdot)} \Rightarrow 1\right] .$$

It is tacitly assumed that the adversary has a mechanism of naming points in $\mathcal{X}$ by strings; if $\mathcal{X} \subseteq \{0,1\}^*$ then a string names itself, but if $\mathcal{X}$ is not a set of strings then points of $\mathcal{X}$ are encoded as strings in some natural way. Our definition of PRFs is unusual for allowing the input X to be arbitrary (possibly not a string).

THE SIV CONSTRUCTION. Let $F\colon \mathcal{K}_1 \times \{0,1\}^{**} \rightarrow \{0,1\}^n$ be a PRF. Let $\Pi = (\mathcal{K}_2, \mathcal{E}, \mathcal{D})$ be a conventional IV-based encryption scheme with IV-length n and message space $\mathcal{X}$. We write $F_K(H, M)$ instead of $F_K((H, M))$. We construct from (F, Π) a DAE $\widetilde{\Pi} = \mathrm{SIV}[F, \Pi] = (\widetilde{\mathcal{K}}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{D}})$ with header space $\{0,1\}^{**}$ and message space $\mathcal{X}$ where $\widetilde{\mathcal{K}} = \mathcal{K}_1 \times \mathcal{K}_2$ and the encryption and decryption algorithms are as illustrated and defined in Figure 3. Recall that $Y[n+1..|Y|] = \varepsilon$ if $|Y| < n$.

We will now show that if F is PRF-secure and Π is IND\$-secure then $\widetilde{\Pi} = \mathrm{SIV}[F, \Pi]$ is DAE-secure. The intuition behind the proof is this. If any bit of the header H or plaintext X is new then the string IV will look like a random string and so $IV \parallel C$ will be difficult to distinguish from random bits. On decryption, the adversary must create a new (H, Y) where $Y = IV \parallel C$. Let's imagine giving the adversary the corresponding plaintext X for free. Now (H, X) is new because (H, X) determines (H, Y) and the adversary is not allowed to decipher values that it trivially knows the decipherment of. But if (H, X) is new then IV' is adversarially unpredictable and so its chance of being equal to IV is only about 2^{-n}.

In the following result we write $\mathrm{Time}_\Pi(\mu)$, where $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is an IV-based encryption scheme and $\mu > 0$ is an integer, for the sum of the worst-case times: to select $K \xleftarrow{\$} \mathcal{K}$, to compute $\mathcal{E}^{IV}_K$ on inputs of total length μ, and to compute $\mathcal{D}^{IV}_K$ on inputs of total length μ. Here, by convention, "time" means actual running time plus program size, all relative to some fixed RAM model of computation.

Theorem 2 Let $F\colon \mathcal{K}_1 \times \{0,1\}^{**} \to \{0,1\}^n$ be a PRF and let $\Pi = (\mathcal{K}_2, \mathcal{E}, \mathcal{D})$ be a conventional IV-based encryption scheme with message space $\mathcal{X}$ and IV-length n. Let $\widetilde{\Pi} = \mathrm{SIV}[F, \Pi]$. Let A be an adversary (for attacking $\widetilde{\Pi}$) that runs in time t and asks q queries, these of total length μ. Then there exists adversaries B and D such that

$$\mathbf{Adv}_{\Pi}^{\mathrm{priv\$}}(B) + \mathbf{Adv}_{F}^{\mathrm{prf}}(D) \quad \geq \quad \mathbf{Adv}_{\widetilde{\Pi}}^{\mathrm{dae}}(A) - q/2^n \ .$$

What is more, B and D run in time at most $t' = t + \mathrm{Time}_{\Pi}(\mu) + c\mu$ for some absolute constant c and ask at most q queries, these of total length μ. ▌

Proof: The proof proceeds in two stages. First we consider the DAE scheme $G = \mathrm{SIV}[\mathrm{Func}(\{0,1\}^{**}, n), \Pi]$ (replacing the function F_{K1} with a random function $\rho \in \mathrm{Func}(\{0,1\}^{**}, n)$). Then we extend this to account for the insecurity of the PRF F.

Denote the forward and reverse algorithms associated to G as $G_{\rho,K2}$ and $G^{-1}_{\rho,K2}$, with $(\rho, K2)$ being the key. Let $\delta = \mathbf{Adv}_{G}^{\mathrm{dae}}(A)$ and $q = q_{\mathrm{L}} + q_{\mathrm{R}}$ and $\mu = \mu_{\mathrm{L}} + \mu_{\mathrm{R}}$ where q_{L} and q_{R} are the number of left and right oracle queries, these totaling μ_{L} and μ_{R} bits, respectively. With the obvious simplifications in notation we have

$$\begin{aligned}
\delta &= \Pr\left[A^{G_{\rho,K2}(\cdot,\cdot),\, G^{-1}_{\rho,K2}(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right] \\
&= \left(\Pr\left[A^{G_{\rho,K2}(\cdot,\cdot),\, G^{-1}_{\rho,K2}(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{G_{\rho,K2}(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right]\right) \\
&\quad + \left(\Pr\left[A^{G_{\rho,K2}(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right]\right) \ = \ p_1 + p_2
\end{aligned}$$

where p_1 and p_2 represent the corresponding parenthesized expressions; it remains to bound these quantities. For p_2 we construct from A an adversary B^g for attacking the priv\$-security of Π. Let B run A. When A asks its left-oracle a query (H, X), let B ask $g(M)$ and return the result to A. When A asks a right-oracle query have B return $\perp$. When A halts with output bit b, let B output b. Notice that if $g = \mathcal{E}^{\$}_{K}$ then B properly simulates $G_{\rho,K2}(\cdot,\cdot), \perp(\cdot,\cdot)$ oracles for A (here we need the assumption that A never repeats a query). Similarly, if $g = \$$ then B simulates $\$(\cdot,\cdot), \perp(\cdot,\cdot)$ oracles for A. Hence $p_2 \leq \mathbf{Adv}_{\Pi}^{\mathrm{priv\$}}(B)$.

To bound p_1 consider giving the key $K2$ to the adversary and then asking it to carry out its distinguishing task. As this can only make the task easier we may assume

$$\begin{aligned}
p_1 &= \Pr\left[A^{G_{\rho,K2}(\cdot,\cdot),\, G^{-1}_{\rho,K2}(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{G_{\rho,K2}(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right] \\
&\leq \Pr\left[A(K2)^{G_{\rho,K2}(\cdot,\cdot),\, G^{-1}_{\rho,K2}(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A(K2)^{G_{\rho,K2}(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right] \ .
\end{aligned}$$

We can assume without loss of generality that A halts and outputs 1 as soon as a right-oracle query returns something other than $\perp$. Under this assumption, encryption queries are useless for distinguishing between these two oracle pairs, as prior to the right oracle returning $M \neq \perp$ both pairs behave as $G_{\rho,K2}(\cdot,\cdot), \perp(\cdot,\cdot)$. Hence p_1 is bounded by the probability that A asks a right-oracle query (H, Y) such that $G^{-1}_{\rho,K2}(H, Y) \neq \perp$. Examining the algorithm for $G^{-1}_{\rho,K2}$ we see that this occurs only when $\rho(H, X) = IV$, where $X = \mathcal{D}^{IV}_{K2}(C)$ (with Y having been parsed into IV and C). Since the adversary is given the key $K2$, it can compute $\mathcal{D}^{IV}_{K2}(C)$ for any strings IV, C of its choosing. In particular, when it asks a right-oracle query (H, Y) it knows what is the input to the random function ρ and what is the target output IV. But under our assumption that A never queries its right oracle (H, Y) when some left-oracle query (H, X) returned Y, either the input (H, X) is new, or the target IV is new. Thus, the probability that $\rho(H, X) = IV$ is at most $1/2^n$ for each right-oracle query, and we conclude that $p_1 \leq q_{\mathrm{R}}/2^n$. Since $q_{\mathrm{R}} \leq q$ we have $\delta \leq \mathbf{Adv}_{\Pi}^{\mathrm{priv\$}}(B) + q/2^n$.

For the second part of the proof note that

$$\mathbf{Adv}^{\mathrm{dae}}_{\widetilde{\Pi}}(A) \;=\; \delta + \Pr\left[A^{\widetilde{\mathcal{E}}_{K1,K2}(\cdot,\cdot),\widetilde{\mathcal{D}}_{K1,K2}(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{G_{\rho,K2}(\cdot,\cdot),G^{-1}_{\rho,K2}} \Rightarrow 1\right]$$

where $\widetilde{\Pi} = (\mathcal{K}1 \times \mathcal{K}2, \widetilde{\mathcal{E}}, \widetilde{\mathcal{D}})$ and we have suppressed the random selections $K1 \overset{\$}{\leftarrow} \mathcal{K}_1$ and $K2 \overset{\$}{\leftarrow} \mathcal{K}_2$. Let D^g be an adversary for attacking F as a PRF, and let it operate as follows. Adversary D picks $K2 \overset{\$}{\leftarrow} \mathcal{K}_2$ and runs A. When A asks a left oracle query (H,X), B answers by setting $IV \leftarrow g(H,X)$, computing $C \leftarrow \mathcal{E}^{IV}_{K2}(X)$ and returning to A the string $IV \,||\, C$. On a right oracle query (H,Y), adversary D parses $IV = Y[1..n]$, $C = Y[n+1..|Y|]$, computes $X \leftarrow \mathcal{D}^{IV}_{K2}(C)$ and tests if $IV = g((H,X))$, returning X to A if so and $\perp$ otherwise. When A halts with output bit b, let D output b. Clearly D correctly simulates $\widetilde{\mathcal{E}}_{K1,K2}(\cdot,\cdot), \widetilde{\mathcal{D}}_{K1,K2}(\cdot,\cdot)$ when its oracle $g = F_{K1}$ for some random key $K1$, and $G_{K1,K2}(\cdot,\cdot), G^{-1}_{K1,K2}(\cdot,\cdot)$ if instead $g = \rho$ for a random $\rho \in \mathrm{Func}(\mathcal{M}, n)$. So, $\mathbf{Adv}^{\mathrm{dae}}_{\widetilde{\mathcal{E}}}(A) \le \delta + \mathbf{Adv}^{\mathrm{prf}}_{F}(D)$ and rearranging gives the result. ▮

5 Enriching a PRF to take Vectors of Strings as Input: The S2V Construction

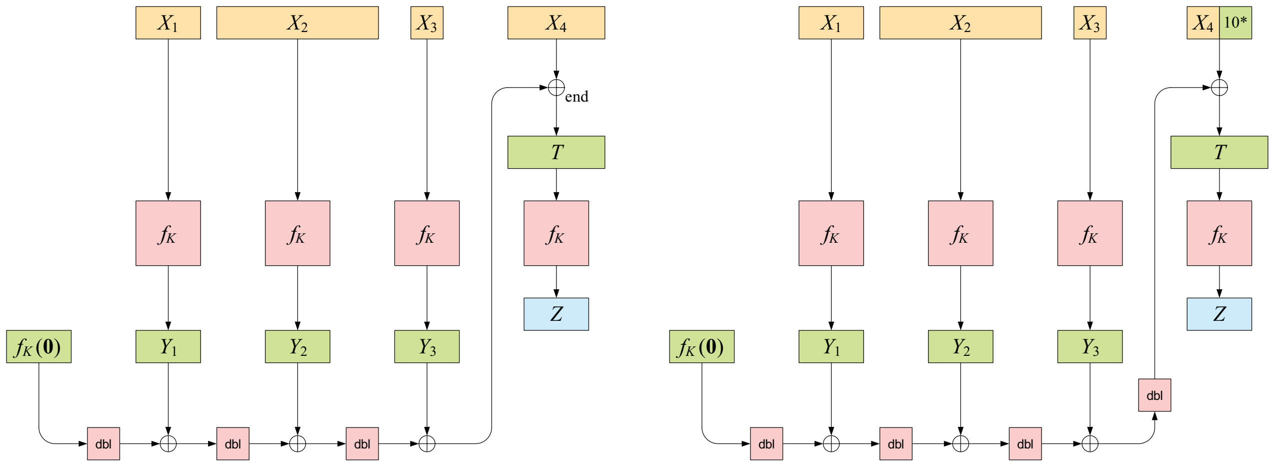
THE GOAL. Traditionally, a pseudorandom function (PRF) takes a single string as input: under the control of a key K, a PRF f maps a string $X \in \{0,1\}^*$ into a string $f_K(X)$. But SIV uses a non-traditional PRF—a function F that, under the control of a key K, maps a vector of strings $X = (X_1, \ldots, X_m) \in \{0,1\}^{**}$ into a string $F_K(X)$. Let us call a PRF that takes a string as input an sPRF (string-input PRF) and a PRF that takes a vector of strings as input a vPRF (vector-input PRF). This section is about efficient ways to turn an sPRF f into a vPRF f^*.

At first glance it might seem like there'd be little to say about sPRF-to-vPRF conversion: there's an obvious approach for solving the problem, and it's obviously correct. Namely, encode any vector of strings $X = (X_1, \ldots, X_m)$ into a single string $\langle X \rangle$ and apply the sPRF to that, $f^*_K(X) = f_K(\langle X \rangle)$. By *encode* we mean any reversible, easily-computed map of a vector of strings into a single one, say

$$\langle X_1, \ldots, X_m \rangle \;=\; X_1 \;\|\; N_1 \;\|\; \cdots \;\|\; X_m \;\|\; N_m$$

where $N_i = |X_i|_{64}$ is the length of X_i encoded into 64 bits (assume that $|X_i| < 2^{64}$ for all i). The problem with making a vPRF in such a way is a diminution of efficiency. First, computing $f^*_K(X)$ may take longer than the total time to compute $f_K(X_i)$ for each component X_i since we have added $64m$ bits for length annotation. (As an example, if f = CMAC-AES then we are doubling the time to MAC $X = (X_1)$ where X_1 is 16-bytes. CMAC was designed to avoid any unnecessary blockcipher calls and it seems a shame to squander this effort with sloppy sPRF-to-vPRF conversion.) Second, even if some components of X stay fixed (say X_2 is constant), we must still re-process the entire encoded string each time we compute f^*_K at a new value. Third, the mechanism is not parallelizable; one cannot process X_i until one is done processing X_{i-1}. Fourth, the assumption that $|X_i| < 2^{64}$, while reasonable in practice, is artificial and potentially wasteful, yet use of a stingier encoding will lead to greater complexity. Finally, the given encoding disrupts word alignment: if, for example, the first argument is one byte and all subsequent arguments are multiples of eight bytes, an implementation will now be dealing with non-word-aligned data. Fixing this problem by a smarter encoding will lead to increased complexity. We aim to do sPRF-to-vPRF conversion in a way that fixes the problems above.

NOTATION. Fix a value $n \ge 2$. Let $\mathbf{0} = 0^n$ and $\mathbf{1} = 0^{n-1}1$ and $\mathbf{2} = 0^{n-2}10$. These are regarded as points in finite field $\mathbb{F}_{2^n}$ represented using a primitive polynomial in the customary way. For $S \in \{0,1\}^n$ let $\mathbf{2}S$ mean the n-bit string representing the product of $\mathbf{2}$ and S. This can be computed with a left shift of S followed by a conditional xor. By $\mathbf{2}^i S$ we mean to do this multiplication by $\mathbf{2}$ a total of i times. By $N \oplus_{\mathrm{end}} X$ ("xor-into-the-end") we mean to xor the n-bit string N into the end of the string X, which will have at least n bits;



```
Algorithm $f^*_K(X_1, \ldots, X_m)$                The S2V Construction, $f^* = \mathrm{S2V}[f]$
10  if $m = 0$ then return $f_K(\mathbf{1})$
11  $S \leftarrow f_K(\mathbf{0})$
12  for $i \leftarrow 1$ to $m-1$ do $S \leftarrow \mathbf{2}S \oplus f_K(X_i)$
13  if $|X_m| \geq n$ then $T \leftarrow S \oplus_{\text{end}} X_m$ else $T \leftarrow \mathbf{2}S \oplus X_m 10^*$
14  return $Z \leftarrow f_K(T)$
```

Figure 4: The S2V construction makes a PRF $f^*\colon \mathcal{K} \times \{0,1\}^{**} \to \{0,1\}^n$ from a PRF $f\colon \mathcal{K} \times \{0,1\}^* \to \{0,1\}^n$. **Bottom:** Definition of S2V. Strings $X_1, \ldots, X_m \in \{0,1\}^*$ and $m \geq 0$ are arbitrary. **Top:** Illustration of it, computing $Z = f^*_K(X_1, X_2, X_3, X_4)$. The left side shows the case for $|X_4| \geq n$, the right side for $|X_4| < n$

$N \oplus_{\text{end}} X = (0^{x-n} N) \oplus X$ where $x = |X|$. By $X10^*$ we mean $X10^i$ where $i \geq 0$ is the least number such that $|X| + 1 + i$ is divisible by n.

THE S2V CONSTRUCTION. Let $f\colon \mathcal{K} \times \{0,1\}^* \to \{0,1\}^n$ be an sPRF. We construct from it the vPRF $f^* = \mathrm{S2V}[f]$ where $f^*\colon \mathcal{K} \times \{0,1\}^{**} \to \{0,1\}^n$ is specified and illustrated in Figure 4. The special treatment of the last component of input, X_m, is to handle the case where $|X_m| < n$. The construction has the desired efficiency characteristics. The time to compute $f^*_K(X)$ is essentially the sum of the times to compute $f_K(X_i)$ on each component; in particular, when $f = \mathrm{CMAC}$, say, the number of blockcipher calls to compute $f^*_K(X)$ is the sum of the number of blockcipher calls to compute each $f_K(X_i)$. Also, one can preprocess invariant components so that the time to compute $f^*_K(X)$ will not significantly depend on them. The computation of f^* is on-line (assuming that f itself is on-line); in particular, the component lengths need not be known in advance. Word alignment is not disrupted. And the scheme is parallelizable: different arguments can be acted on simultaneously, so f^* will be parallelizable if f is.

In a related effort we have proven the following result. The complexity-theoretic analog of Theorem 3 follows in the usual way. We only prove security when queries are restricted to vectors with $n-1$ or fewer components. In practice $n \geq 64$ well exceeds the number of components in a vector of associated data, making the restriction irrelevant. The proof appears in Appendix E.

Theorem 3 Let $f = \mathrm{Func}(\{0,1\}^*, n)$ and $f^* = \mathrm{S2V}[f]$. Let A be an adversary that asks at most $q \geq 3$ vector-valued queries having p components in all, and each vector having fewer than n components. Then $\mathbf{Adv}^{\mathrm{prf}}_{f^*}(A) \leq pq/2^n$. ▮

The complexity-theoretic statement for the security of f^* follows from the information-theoretic statement in the standard way, so we omit a proof of the following:

Corollary 4 Let $f\colon \mathcal{K} \times \{0,1\}^* \to \{0,1\}^n$ be a PRF and let $f^* = \mathrm{S2V}[f]$. Let A be an adversary that runs in time t and asks $q \geq 3$ vector-valued queries, the q queries having a total of p components and μ bits and each vector having fewer than n components. Then there exists an adversary B where

$$\mathbf{Adv}_f^{\mathrm{prf}}(B) \geq \mathbf{Adv}_{f^*}^{\mathrm{prf}}(A) - pq/2^n$$

and B asks q queries having a total of p components and μ bits and B runs in time $t + c(\mu + p + q)$ for some absolute constant c. ▌

PRACTICAL USES OF S2V. In the next section we will use the S2V construction for sPRF-to-vPRF conversion to make a DAE scheme. But we point out that real-world security protocols already employ, implicitly, PRFs that operate on vectors of strings. They usually do this in a complex and inefficient manner. A good illustration is the TLS protocol; they define a PRF that operates on 2-vectors, the PRF defined in a complex and feedback-dependent way from HMAC-MD5 and HMAC-SHA1. Then, wanting to apply the PRF to vectors with more than two components, they concatenate logically-separate strings to form the second component. Similarly, IEEE 802.11r does key derivation by applying a PRF to input that includes long constants (host and user names) that remain fixed across many derivations. We suggest that when a security protocol wants to apply a PRF to what is logically a vector of strings the protocol should realize this with just such an abstraction. Concatenation should be avoided in achieving that abstraction (because it is, in general, both inefficient and wrong). The vPRF primitive should be realized in the protocol as a higher-level abstraction made from an sPRF.

6 The SIV Mode of Operation

SIV MODE. For $n \geq 64$, fix an n-bit blockcipher E. Let $\Pi = \mathrm{CTR}$ be counter mode [27] over E. For concreteness and implementation convenience, let the increment function used for CTR mode be the addition of one modulo 2^n. Before any increments, again for implementation convenience (see below), let us zero-out the leftmost bit in each of the last two 32-bit words of the counter. (The loss of two bits of a random IV has inconsequential impact on the priv\$-security of CTR mode.) Let $F = \mathrm{CMAC}^* = \mathrm{S2V}[\mathrm{CMAC}]$ be the result of applying the S2V construction to CMAC [28], with an underlying blockcipher of E. (Recall that CMAC [28] is a NIST-recommended CBC MAC variant with message space $\{0,1\}^*$.) Consider the scheme $\mathrm{SIV}[F, \Pi]$. By combining Theorems 2 and 3 and known results about CMAC and CTR mode [3, 18], the suggested mechanism is a provably secure DAE scheme assuming E is a secure PRP. The proven security falls off, as usual, in $\sigma^2/2^n$ where σ is the total number of blocks asked about. We overload the name SIV and call the mode of operation just described SIV mode. See Figure 5 for the specification. The addition operation shown there has the natural interpretation, modulo 2^n, while $B \,\&\, C$ denotes the bitwise-and of equal-length strings B and C. The only thing left unspecified in the definition of SIV mode is the underlying blockcipher E, which would typically be AES.

COMMENTS. Comparing SIV-AES and the X9.102 scheme AESKW, say, we note that, with SIV-AES, (1) the message space and header space are now $\{0,1\}^*$ instead of unusual sets; (2) message expansion is now independent of header length and message length; (3) the number of blockcipher calls is reduced by a factor of at least six; (4) vector-valued headers can now be handled, and the contribution of any component can be pre-processed if it is to be held fixed; (5) one now has a provable-security guarantee, falling off in $\sigma^2/2^n$, where σ is the total number of message blocks acted on. On the other hand, there is an effective attack on SIV if one can ask this many message blocks, while we do not know if this is true for AESKW.

In the instantiation of SIV we could have used, in place of CMAC, the composition of a universal hash function that gives n-bit outputs with an n-bit blockcipher. This demonstrates that the DAE goal can be achieved

```
**Algorithm** $\mathcal{E}_{K1\,K2}^{H_1,\ldots,H_t}(X)$
$IV \leftarrow \mathrm{CMAC}_{K1}^{*}(H_1,\ldots,H_t,X)$
$C \leftarrow \mathrm{CTR}_{K2}(IV,\ X)$
**return** $Y \leftarrow IV \parallel C$

**Algorithm** $\mathcal{D}_{K1\,K2}^{H_1,\ldots,H_t}(Y)$
**if** $|Y| < n$ **then return** $\perp$
$IV \leftarrow Y[1\,..\,n],\ C \leftarrow [n+1\,..\,|Y|]$
$X \leftarrow \mathrm{CTR}_{K2}(IV,\ C)$
$IV' \leftarrow \mathrm{CMAC}_{K1}^{*}(H_1,\ldots,H_t,X)$
**if** $IV = IV'$ **then return** $X$ **else return** $\perp$
```

```
**Algorithm** $\mathrm{CMAC}_K^*(X_1,\ldots,X_m)$
$S \leftarrow \mathrm{CMAC}_K(\mathbf{0})$          // *precompute*
**for** $i \leftarrow 1$ **to** $m-1$ **do** $S \leftarrow \mathbf{2}S \oplus \mathrm{CMAC}_K(X_i)$
**if** $|X_m| \geq n$
        **then return** $\mathrm{CMAC}_K(S \oplus_{\text{end}} X_m)$
        **else return** $\mathrm{CMAC}_K(\mathbf{2}S \oplus X_m 10^*)$

**Algorithm** $\mathrm{CTR}_K(IV, X)$
$Ctr \leftarrow IV \ \&\ 1^{n-64}\ 01^{31}\ 01^{31}$
$Pad \leftarrow E_K(Ctr) \parallel E_K(Ctr{+}1) \parallel E_K(Ctr{+}2) \parallel \cdots$
**return** $C \leftarrow X \oplus Pad\,[1..|X|]$
```

Figure 5: The SIV mode of operation. The mechanism is the generic SIV scheme instantiated using CMAC* and CTR modes, each of these based on a blockcipher E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$.

by a single "cryptographic" pass over the plaintext, plus a universal-hash-function computation over the header and plaintext. Similarly, a parallelizable MAC like PMAC [10] could have been used in place of CMAC, illustrating that DAE can be achieved by a parallelizable scheme. And if all messages to be encrypted were of one length, that length being a positive multiple of the blocksize, then the raw CBC MAC could have replaced CMAC.

Our earlier descriptions of SIV mode used a different incrementing function within CTR mode, multiplying by two in the finite field with 2^n points. We made this choice for reasons of economy of techniques: doubling in the finite field was already used within S2V, as well as in CMAC. But, in software, especially when coding in a high-level programming language, finite-field doubling is a little bit expensive to be doing with every n-bit word of plaintext. So we have switched to modulo 2^n increment, but where one first clears the most significant bit in each of the last two 32-bit words of the counter. This zeroing-out ensures that if $|M| \leq n2^{31}$ bits (ie, 32 GBytes for $n = 128$) there can be no carry-out of the last 32-bit word, making an increment of Ctr (modulo 2^n) equivalent to incrementing just its last 32-bit word (modulo 2^{32}). Similarly, if $|M| \leq n2^{63}$ bits (as it invariably will be), an increment of Ctr (modulo 2^n) is equivalent to incrementing its final 64 bits (modulo 2^{64}). Of course from a provable-security point of view, all of these details are irrelevant, since all reasonable instantiations of CTR mode will achieve essentially the same priv\$-security.

7 Misuse-Resistant AE

This section gives an application of DAEs motivated not by the key-wrap problem but by the goal of constructing symmetric encryption schemes that are resistant to misuse. We are specifically concerned with IV-misuse, meaning that the IV is used in a way other than the way mandated by the scheme; for example, using a counter when the scheme requires a random value, or repeating an IV when the scheme requires it to be a nonce. Experience has shown that IVs are frequently mishandled. An encryption scheme robust against misuse should at least be an AE scheme (as programmers, protocol designers, and even books often assume that encryption provides for authenticity) and so we will treat IV-misuse within the context of authenticated encryption and not privacy-only encryption. The notion is applicable to the latter context, too.

Designing an IV-based AE scheme that is secure when its IV is an arbitrary nonce—not just when it is a random value—is a first move in the direction of making schemes robust against IV-misuse. The current section takes this a step further; we aim for an AE scheme in which if the IV *is* a nonce then one achieves the usual notion for nonce-based AE; and if the IV *does* get repeated then authenticity remains and privacy is compromised only to the extent that some minimal amount of information may be revealed, the information being if this plaintext is equal to a prior one, and even that is revealed only if both the message and its header have been used with this particular IV. Our formalization will capture this intent.

REVISED SYNTAX FOR AN IV-BASED ENCRYPTION SCHEME. Let us update the syntax of a conventional IV-based encryption scheme to accommodate an associated header. In this case an IV-based encryption scheme is a tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where everything is as before except that the encryption algorithm and decryption algorithm take an extra argument: now they are deterministic algorithms that map $\mathcal{K} \times \{0,1\}^{**} \times \{0,1\}^* \times \{0,1\}^*$ to $\{0,1\}^* \cup \{\bot\}$. We write $\mathcal{E}_K(H, IV, X)$ or $\mathcal{E}_K^{H,IV}(X)$ in place of $\mathcal{E}(K, H, IV, X)$ and $\mathcal{D}_K(H, IV, C)$ or $\mathcal{D}_K^{H,IV}(Y)$ in place of $\mathcal{D}(K, H, IV, Y)$. There must be sets $\mathcal{H}$, $\mathcal{IV}$, and $\mathcal{X}$ such that $\mathcal{E}_K^{H,IV}(X) \in \{0,1\}^*$ iff $H \in \mathcal{H}$ and $IV \in \mathcal{IV}$ and $X \in \mathcal{X}$. We call $\mathcal{IV}$ the *IV space* of Π. We require that $\mathcal{D}_K^{H,IV}(Y) = X$ if $\mathcal{E}_K^{H,IV}(X) = Y$ and $\mathcal{D}_K^{H,IV}(Y) = \bot$ if there is no such X.

MISUSE-RESISTANT AE SECURITY. To measure the AE-security of an encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in the face of possible IV-reuse, imagine an adversary that may ask any sequence of encryption queries, even those that repeat IVs, and any sequence of decryption queries, which may likewise repeat IVs. We want the encryption oracle to return bits that look random except when this is impossible—on a repeated triple of (header, IV, message)—and the decryption oracle should return $\bot$ except when the triple is already known to have a valid decryption. For simplicity, assume as before that our IV-based encryption scheme is length-preserving.

Definition 5 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an IV-based encryption scheme that can handle an associated header and let A be an adversary. Then the **MRAE-advantage** of A in attacking Π is

$$\mathbf{Adv}_{\Pi}^{\mathrm{mrae}}(A) \;=\; \Pr\left[K \overset{\$}{\leftarrow} \mathcal{K} :\; A^{\mathcal{E}_K(\cdot,\cdot,\cdot),\, \mathcal{D}_K(\cdot,\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$(\cdot,\cdot,\cdot),\, \bot(\cdot,\cdot,\cdot)} \Rightarrow 1\right] .$$

The adversary may not repeat a left-query and may not ask a right-query (H, IV, Y) if some previous left-query (H, IV, X) returned Y. ▮

Of course the $\mathcal{E}_K$ oracle returns $\mathcal{E}_K(H, IV, X)$ on input (H, IV, X) and $\mathcal{D}_K$ returns $\mathcal{D}_K(H, IV, Y)$ on input (H, IV, Y). As before $\$(H, IV, X)$ returns a random string of length $n + |X|$ and $\bot(\cdot,\cdot,\cdot)$ always returns $\bot$.

The MRAE-notion of security trivially implies nonce-based AE-scheme security: the latter is the special case where the adversary is not allowed to repeat an IV to any left query. Note that all proposed AE schemes to date [19, 21, 26, 29, 33] do fail should an IV get repeated: existing AE schemes are not MRAE-secure.

BUILDING A MISUSE-RESISTANT AE SCHEME. We can turn a DAE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with header space $\{0,1\}^{**}$ and message space $\mathcal{X}$ into a misuse-resistant AE scheme $\widetilde{\Pi} = (\mathcal{K}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{D}})$ by regarding the IV as one of the components, say the last component, of the header. In particular, SIV mode can be regarded as an MRAE scheme by asserting that one of the header components, say the last one specified, is an IV.

CORRECTNESS. Correctness of the MRAE scheme described above is nearly immediate. Given an adversary A for breaking the misuse-resistant AE scheme (it distinguishes $\mathcal{E}_K(\cdot,\cdot,\cdot),\ \mathcal{D}_K(\cdot,\cdot,\cdot)$ from $\$(\cdot,\cdot,\cdot),\ \bot(\cdot,\cdot,\cdot)$) we get a comparably good adversary B for breaking the DAE, distinguishing $\mathcal{E}_K(\cdot,\cdot),\ \mathcal{D}_K(\cdot,\cdot)$ from $\$(\cdot,\cdot),\ \bot(\cdot,\cdot)$: adversary B runs A and maps left queries (H, IV, X) to queries $(\langle H, IV\rangle, X)$, and maps right queries (H, IV, Y) to queries $(\langle H, IV\rangle, Y)$. The syntax and DAE-security notion for a PRI have been designed to "match up" so that there is nothing to do.

COMMENTS. Since all we have done in the construction is to hijack a component of the header as an IV, it seems as though nothing has actually been done. Yet the MRAE goal is conceptually different from the DAE goal, the former employing an IV and gaining for this a stronger notion of security. The header and the IV are conceptually different, the one being user-supplied data that the user wants authenticated, the other being a mechanism-supplied value needed to obtain a strong notion of security.

In retrospect, it is easy to construct an MRAE scheme by a sequence of simple steps. One can achieve this goal in a trivial way from a DAE scheme that takes a vector-valued header. Such a DAE scheme is easily built from a vector-input PRF and an IND$-secure conventional encryption scheme. At least if one is unconcerned with optimizing efficiency, a vector-input PRF is easily made from a string-input PRF. String-input PRFs and IND$-secure conventional encryption schemes can be built from blockciphers by well-known means. So each step along our path is easy or well-known. Still, the direct construction of an MRAE or DAE scheme from a blockcipher is not a simple matter, as evidenced by the long history of buggy or baroque AE schemes Perhaps simple is how things seem *after* finding the right abstraction boundaries.

8 The PRI Characterization of DAE Security

A secure pseudorandom injection (PRI) resembles a random injective function with the desired amount of length-expansion. We allow a chosen-ciphertext attack in our definition (that is, we focus on a "strong" PRI, analogous to a strong PRP [24]), giving the adversary both the forward and backward direction of the function. We allow the PRI to be tweakable [23], so that the scheme can be used to authenticate an associated header. We allow the domain to be fairly arbitrary—in particular, we consider message spaces that contain strings of various lengths.

Formally, let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a DAE with header space $\mathcal{H}$ and message space $\mathcal{X}$. Imagine an adversary A given access to two oracles—one for $\mathcal{E}$ and one for $\mathcal{D}$. We want to say that this pair looks just like a random injection f and its inverse f^{-1}, the random injection f having the same signature as $\mathcal{E}$. For e: $\mathcal{H} \times \mathcal{X} \to \mathbb{N}$ let $\mathsf{Inj}_e^{\mathcal{H}}(\mathcal{X}, \mathcal{Y})$ be the set of all injective functions f from $\mathcal{H} \times \mathcal{X}$ to $\mathcal{Y}$ such that $|f(H, X)| = |X| + e(H, X)$.

Definition 6 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a DAE with header space $\mathcal{H}$, message space $\mathcal{X}$, and expansion e. The **PRI-advantage** of adversary A in breaking Π is

$$\mathbf{Adv}_{\Pi}^{\text{pri}}(A) \;=\; \Pr\left[K \xleftarrow{\$} \mathcal{K}:\; A^{\mathcal{E}_K(\cdot,\cdot),\, \mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[f \xleftarrow{\$} \mathsf{Inj}_e^{\mathcal{H}}(\mathcal{X}, \mathcal{Y}):\; A^{f(\cdot,\cdot),\, f^{-1}(\cdot,\cdot)} \Rightarrow 1\right]. \qquad \blacksquare$$

The f^{-1} oracle above, on input (H, Y) returns the point X such that $f(H, X) = Y$; if there is no such point then it returns the distinguished value $\perp$. Recall that oracle queries outside the domain of the oracle return $\perp$. As before, we may assume without loss of generality that the adversary does not repeat a query, that it does not ask (H, Y) of its right oracle if some previous left oracle query (H, X) returned Y, that it does not ask (H, X) of its left oracle if some previous right-oracle query (H, Y) returned X, and that it does not ask any query (H, X) outside of $\mathcal{H} \times \mathcal{X}$. When a PRI is length-preserving we call it an *enciphering scheme* and use the notation $\mathbf{Adv}_{\Pi}^{\pm\text{prp}}(A)$ or $\mathbf{Adv}_{\Pi}^{\pm\widetilde{\text{prp}}}(A)$ according to whether or not it accommodates a nontrivial tweak space.

Assuming a reasonable amount of stretch, the PRI and DAE notions of security are very close, as the following theorem shows.

Theorem 7 Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a DAE with header space $\mathcal{H}$, message space $\mathcal{X}$, and stretch s, and let $\tau = \min_{X \in \mathcal{X}}\{|X|\}$ be the length of a shortest plaintext. Let A be an adversary that asks at most q_{L} left-oracle queries, q_{R} right-oracle queries, for a total of $q = q_{\text{L}} + q_{\text{R}}$ queries. Then

$$\left|\mathbf{Adv}_{\Pi}^{\text{pri}}(A) - \mathbf{Adv}_{\Pi}^{\text{dae}}(A)\right| \;\le\; q^2/2^{s+\tau+1} + 4q_{\text{R}}/2^s.$$

In other words, as the stretch s grows, the DAE and PRI notions converge. The quantitative difference between the measures is small if the stretch is, say, $s = 128$ bits. Among other reasons, it is to achieve this equivalence with PRIs that our definition for them used indistinguishability from random bits rather than, say, indistinguishability from the encryption of random bits.

Proof: Let A be an adversary that has access to two oracles. Let it ask q_{L} queries of its left oracle and q_{R} queries of its right oracle, and let $q = q_{\text{L}} + q_{\text{R}}$. With the obvious notational simplifications we have

$$\begin{aligned} \left|\mathbf{Adv}_{\Pi}^{\text{pri}}(A) - \mathbf{Adv}_{\Pi}^{\text{dae}}(A)\right| &= \left|\Pr\left[A^{f(\cdot,\cdot),\, f^{-1}(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right]\right| \\ &= \left|\Pr\left[A^{\text{G1}} \Rightarrow 1\right] - \Pr[A^{\text{G0}} \Rightarrow 1]\right| \end{aligned}$$

```
On query left$(H,X)$:
10  $c \leftarrow |X| + e(H,X)$
11  $Y \xleftarrow{\$} \{0,1\}^c$
12  if $Y \in \mathrm{Image}(f(H,\cdot)) \cup \mathit{Invalid}^H$ then
13      $\mathit{bad} \leftarrow$ true , $Y \xleftarrow{\$} \{0,1\}^c - \mathrm{Image}(f(H,\cdot)) - \mathit{Invalid}^H$
14  return $f(H,X) \leftarrow Y$

On query right$(H,Y)$:
20  $c \leftarrow |Y|$
21  $\mathit{EligibleX} \leftarrow \{X \in \{0,1\}^{\le c}\colon\ |X| + e(H,X) = |Y| \text{ and } f(H,X) = \texttt{undef}\}$
22  $\mathit{EligibleY} \leftarrow \{0,1\}^c - \mathrm{Image}(f(H,\cdot)) - \mathit{Invalid}^H$
23  $x \xleftarrow{\$} [1\,..\,|\mathit{EligibleY}|]$
24  if $x \in [1..|\mathit{EligibleX}|]$ then
25      $\mathit{bad} \leftarrow$ true , $X \leftarrow$ the $x^{\text{th}}$ string of $\mathit{EligibleX}$, $f(H,X) \leftarrow Y$, return $X$
26  $\mathit{Invalid}^H \leftarrow \mathit{Invalid}^H \cup \{Y\}$
27  return $\perp$
```

Figure 6: Games used in the proof of Theorem 7. Game G1 is the complete code; game G0 omits the shaded statements.

for the games G0 and G1 defined in Figure 6. Recall that booleans are initialized to `false`, sets are initialized to empty, and partial functions are initialized to everywhere undefined with the symbol `undef`. The set $\mathrm{Image}(f(H,\cdot))$ contains all points $Y \neq \texttt{undef}$ such that $f(H,X) = Y$ for some $X \in \mathcal{X}$. Set difference is indicated with a minus sign. Look first at game G0. Much of the code (lines 12–13 and 20–26) is irrelevant to what the adversary sees. Each query $\mathrm{left}(H,X)$ returns a random string of $|X| + e(H,X)$ bits and each query $\mathrm{right}(H,Y)$ returns $\perp$. Thus game G0's $(\mathrm{left}, \mathrm{right})$ oracles faithfully simulate a pair of oracles $(\$, \perp)$ and we have that $\Pr[A^{\mathrm{G0}} \Rightarrow 1] = \Pr[A^{\$,\perp} \Rightarrow 1]$.

Game G1 is more subtle. We claim that its $(\mathrm{left}, \mathrm{right})$ oracles are simply a lazy evaluation of a pair of oracles (f, f^{-1}) with the desired domain and range. To see this, understand first that the partial function $f(H,\cdot)$ maintains the correspondence $X \mapsto f(H,X)$ for those domain points that we have already assigned values to, while the set $\mathit{Invalid}^H$ maintains the set of points Y that have become ineligible to be $f(H,X)$ values, for any X, by virtue of having been asked $\mathrm{right}(H,Y)$ and having returned $\perp$, effectively asserting that $f^{-1}(H,Y) = \perp$ and so Y is outside the image of $f(H,\cdot)$. Now, starting at $\mathrm{left}(H,X)$ queries, we begin at line 10 by calculating the length c of the ciphertext that we must return. The code at lines 11–14 returns a random string Y of length c subject to the constraint that Y is outside of the image of $f(H,\cdot)$ and not ineligible to be an $f(H,X)$ value by virtue of having asserted that there is no preimage for Y with tweak H. Looking next at $\mathrm{right}(H,Y)$ queries, we calculate at line 21 the set $\mathit{EligibleX}$ of values X that could possibly map to Y using tweak H, and we calculate at line 22 the set of strings Y that could, at this moment be paired with strings in $\mathit{EligibleX}$. By our conventions on the adversary making no "pointless" queries, the string Y will necessarily be among the strings in $\mathit{EligibleY}$. Since we aim to randomly and injectively pair points in $\mathit{EligibleX}$ with points in $\mathit{EligibleY}$, the chance that a given point Y in $\mathit{EligibleY}$ has a preimage in $\mathit{EligibleX}$ is just $|\mathit{EligibleX}|/|\mathit{EligibleY}|$. Lines 23 and 24 effectively flip a coin with this bias, deciding if the string $Y \in \mathit{EligibleY}$ should or should not be given a (random) preimage in $\mathit{EligibleX}$. If it is not given a preimage, we record this decision by augmenting $\mathit{Invalid}^H$ at line 26. If it is given a preimage, it is given a random one by lines 23–25, the choice is recorded, and the random preimage is returned. We have thus provided a perfect simulation of an (f, f^{-1}) oracle, and so $\Pr[A^{\mathrm{G1}} \Rightarrow 1] = \Pr[A^{f,f^{-1}} \Rightarrow 1]$.

To bound $|\Pr[A^{\mathrm{G1}} \Rightarrow 1] - \Pr[A^{\mathrm{G0}} \Rightarrow 1]|$ we can now invoke the fundamental lemma of game-playing [7], since games G1 and G0 have been defined to be identical apart from the sequel of statements $\mathit{bad} \leftarrow$ `true`. The lemma assures us that $|\Pr[A^{\mathrm{G1}} \Rightarrow 1] - \Pr[A^{\mathrm{G0}} \Rightarrow 1]| \le \Pr[A^{\mathrm{G0}} \text{ sets } \mathit{bad}]$.

Let BAD be the event that A^{G0} causes *bad* to get set to `true`. We must bound the probability of BAD. Remember that the shaded statements have been expunged from the game. Prior to BAD occurring, each left-query adds a single point to a set $\mathrm{Image}(f(H,\cdot))$ but has no impact on any set $\mathit{Invalid}^H$, while each right-query adds a single point to a set $\mathit{Invalid}^H$ but has no impact on any set $\mathrm{Image}(f(H,\cdot))$. If the i^{th} query is left-query then the set $\mathrm{Image}(f(H,\cdot)) \cup \mathit{Invalid}^H$ will have at most $i-1$ points and the chance that *bad* will get set at line 13 will be at most $(i-1)/2^{s+\tau}$ and so, overall, the probability that *bad* gets set at line 13 is at most $\sum_{i=1}^{q}(i-1)/2^{s+\tau} \leq q^2/2^{s+\tau+1}$. If the i^{th} query is a right-query then *bad* will be set with probability $|\mathit{EligibleX}|/|\mathit{EligibleY}|$ for the current sets $\mathit{EligibleX}$ and $\mathit{EligibleY}$. How big can $|\mathit{EligibleX}|$ be? Asked a query Y of length c, even if *every* string of length at most $c-s$ (the maximal possible length) is in $\mathit{EligibleX}$, still we will have that $|\mathit{EligibleX}| < 2^{c+1-s}$. Conversely, how small can $|\mathit{EligibleY}|$ be? On the i^{th} query we know that $|\mathit{EligibleY}| > 2^c - i$. So on the i^{th} query we have that $|\mathit{EligibleX}|/|\mathit{EligibleY}| < 2^{c+1-s}/(2^c-i) \leq 2^{2-s}$ assuming $i \leq 2^{c-1}$ or, more strongly, assuming $q \leq 2^{s+\tau-1}$. Summing over all q_{R} right-queries we have that the probability that *bad* gets set at line 25 is at most $4q_{\mathrm{R}}/2^s$. Since the result becomes vacuous when $q > 2^{s+\tau-1}$, we may now drop that technical condition and conclude the theorem. ▌

Acknowledgments

Many thanks to the X9F1 working group, whose draft standard motivated this paper, and Morris Dworkin, who made this work known to us [13]. Thanks to Jesse Walker for an enormous number of valuable comments; to Susan Langford for noticing a significant error in an earlier draft; to Steve Bellovin for voicing his concerns about IV-misuse at a meeting back in 2000 (his comments ultimately motivated Section 7); to Mihir Bellare for his typically perceptive reading; to the Eurocrypt 2006 PC for their comments; and to Dan Harkins for comments, test vectors, and encouraging the choice of a more efficient incrementing function in SIV mode. Phil Rogaway was supported by NSF 0208842 and a gift from Intel Corp. Much of this paper was written while Rogaway was a visitor to the School of Information Technology at Mae Fah Luang University, Thailand. Many thanks to MFLU and, in particular, to Dr. Thongchai Yooyativong and Dr. Tatsanee Mallanoo, for their generous hospitality.

References

[1] J. An and M. Bellare. Does encryption with redundancy provide authenticity? *Advances in Cryptology – Eurocrypt '01*, LNCS vol. 2045, Springer, pp. 512–528, 2001.

[2] M. Bellare, A. Boldyreva, L. Knudsen, and C. Namprempre. On-Line ciphers and the Hash-CBC constructions. *Advances in Cryptology – Crypto '01*, LNCS vol. 2139, Springer, pp. 292–309, 2001.

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: analysis of the DES modes of operation. *Proc. of the 38th Symposium on Foundations of Computer Science*, IEEE Press, pp. 394–403, 1997.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. *Advances in Cryptology – Crypto'98*, LNCS vol.1462, Springer, pp. 26–45, 1998.

[5] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *J. of Computer and System Science (JCSS)*, vol. 61, no. 3, pp. 362–399, Dec 2000.

[6] M. Bellare and C. Namprempre. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology – Asiacrypt '00*, LNCS vol. 1976, Springer, pp. 531–545, 2000.

[7] M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint report 2004/331, 2004.

[8] 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*, LNCS vol. 1976, Springer, pp. 317–330, 2000.

[9] M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation. *Fast Software Encryption (FSE 2004)*, LNCS vol. 3017, Springer, pp. 389–407, 2004.

[10] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. *Advances in Cryptology – Eurocrypt ’02*, LNCS vol. 2332, Springer, pp. 384-397, 2001.

[11] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: the three-key constructions. *Advances in Cryptology – Crypto ’00*, LNCS vol. 1880, Springer, pp. 197–215, 2000.

[12] Y. Dodis and A. Smith. Entropic security and the encryption of high entropy messages. *Theory of Cryptography (TCC 2005)*, LNCS vol. 3378, Springer, pp. 556-577, 2005.

[13] M. Dworkin. Request for review of key wrap algorithms. Cryptology ePrint report 2004/340, 2004. Contents are excerpts from a draft standard of the Accredited Standards Committee, X9, entitled *ANS X9.102 — Wrapping of Keys and Associated Data.*

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

[15] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.

[16] S. Halevi and P. Rogaway. A tweakable enciphering mode. *Advances in Cryptology – Crypto ’03*, LNCS vol. 2727, Springer, pp. 482–499, 2003.

[17] R. Housley. Triple-DES and RC2 key wrapping. IETF RFC 3217, Dec. 2001. Earlier version in RFC 2630, June 1999.

[18] T. Iwata and K. Kurosawa. OMAC: One-key CBC MAC. *Fast Software Encryption (FSE 2003)*, LNCS vol. 2887, Springer, pp. 129–153, 2003.

[19] C. Jutla. Encryption modes with almost free message integrity. *Advances in Cryptology – Eurocrypt ’01,* LNCS vol. 2045, Springer, pp. 529–544, 2001.

[20] J. Katz and M. Yung. Unforgeable encryption and adaptively secure modes of operation. *Fast Software Encryption (FSE 2000)*, LNCS vol. 1978, Springer, pp. 284–299, 2000.

[21] T. Kohno, J. Viega, and D. Whiting. CWC: A high-performance conventional authenticated encryption mode. *Fast Software Encryption (FSE 2004)*, LNCS vol. 3017, Springer, pp. 427–445, 2004.

[22] H. Krawczyk. The order of encryption and authentication for protecting communications (or: how secure is SSL?) *Advances in Cryptology – Crypto ’01,* LNCS vol. 2139, Springer, pp. 310–331, 2001.

[23] M. Liskov, R. Rivest, and D. Wagner. Tweakable block ciphers. *Advances in Cryptology – Crypto ’02,* LNCS vol. 2442, Springer, pp. 31–46, 2002.

[24] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, vol. 17, no. 2, pp. 373–386, 1988.

[25] S. Matyas. Key handling with control vectors. *IBM Systems Journal*, vol. 30, no. 2, pp. 151–174, 1991.

[26] D. McGrew and J. Viega. The Galois/Counter mode of operation (GCM). Manuscript, May 2005. Available from the NIST website.

[27] National Institute of Standards and Technology, M. Dworkin, author. Recommendation for block cipher modes of operation, methods and techniques. NIST Special Publication 800-38A, 2001.

[28] National Institute of Standards and Technology, M. Dworkin, author. Recommendation for block cipher modes of operation: the CMAC mode for authentication. NIST Special Publication 800-38B, May 2005.

[29] National Institute of Standards and Technology, M. Dworkin, author. Recommendation for block cipher modes of operation: the CCM mode for authentication and confidentiality. NIST Special Publication 800-38C, May 2004.

[30] D. Phan and D. Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). *Selected Areas in Cryptography (SAC 2004)*, LNCS vol 3357, Springer, pp. 182-197, 2004.

[31] P. Rogaway. Authenticated-encryption with associated-data. *Proceedings of the 9th Annual Conference on Computer and Communications Security (CCS-9)*, ACM, pp. 98–107, 2002.

[32] P. Rogaway. Nonce-based symmetric encryption. *Fast Software Encryption (FSE 2004)*, LNCS vol. 3017, Springer, pp. 348–359, 2004.

[33] P. Rogaway, M. Bellare, and J. Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security* (TISSEC), vol. 6, no. 3, pp. 365–403, Aug. 2003.

[34] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. *Advances in Cryptology – Eurocrypt '06*, LNCS vol. 4004, Springer, pp. 373–390, 2006. Proceedings version of this paper.

[35] A. Russell and H. Wong. How to fool an unbounded adversary with a short key. *Advances in Cryptology – Eurocrypt '02*, LNCS vol. 2332, Springer, pp. 133–148, 2002.

[36] R. Schroeppel. The hasty pudding cipher. AES candidate submitted to NIST, 1998.

[37] S/MIME Working Group, IETF. Mailing list archives, 1997. http://www.imc.org/ietf-smime/index.html

A Critique of the Draft X9.102 Standard

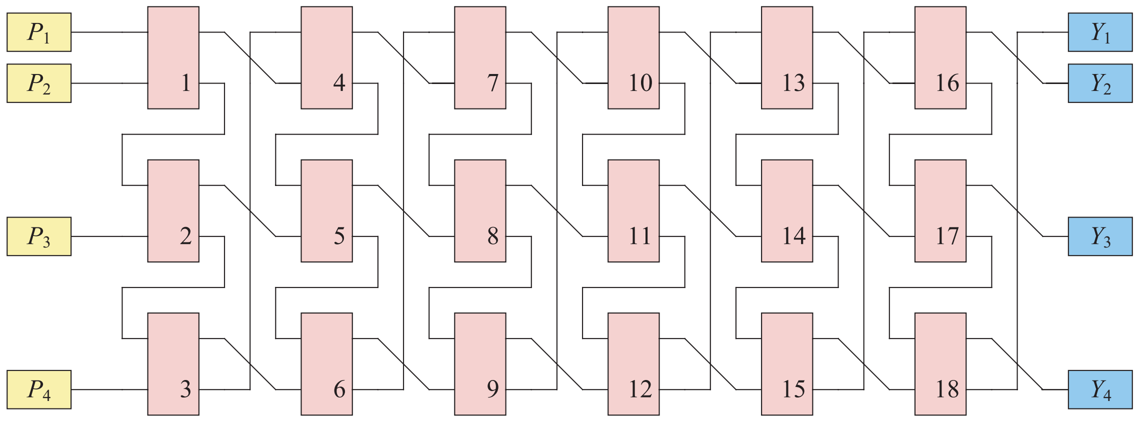
Review of the X9.102 proposal [13] motivated the current paper. The following summary comments on that proposal address the models/definitions it describes, and then each of its schemes.

MODELS AND DEFINITIONS. The specification document outlines an attack model and goal [13, Section 2] for the four key-wrap schemes. This is more then specs usually do, but the description is not very precise.

▷ The stated security goals for AESKW, TDKW, and AKW1 are indistinguishability of ciphertexts [3, 15] under an adaptive chosen-ciphertext attack (IND-CCA2), and unforgeability of ciphertexts [8, 20] under an adaptive chosen-ciphertext attack. In effect, the goal would then be to provide authenticated-encryption [6, 8, 20]. But indistinguishability—at least as it is traditionally defined and understood—cannot be achieved by schemes AESKW and TDKW because they are deterministic and stateless; the usual formulation of indistinguishability demands that one conceal in a sequence of ciphertexts whether or not a given plaintext was encrypted twice. While deterministic encryption schemes have been considered in the literature, usually going under the name of a blockcipher or an enciphering scheme, security is typically understood to be in the sense of a pseudorandom permutation (PRP) [5, 24] and the scheme must therefore be length-preserving.

▷ It is unnecessary to ask for indistinguishability and unforgeability under a chosen-*ciphertext* attack. Unforgeability under a chosen-plaintext attack implies unforgeability under a chosen-ciphertext attack, since the decryption oracle will only return a valid plaintext if it is asked a valid ciphertext, which is then a forgery. Moreover, a scheme that provides indistinguishability and unforgeability under a chosen-plaintext will automatically provide indistinguishability under a chosen-ciphertext attack [6, 20].

▷ The model section limits the number of key-wrapping oracle queries to 2^{48} for AESKW and 2^{32} for the other schemes. Where do these numbers come from? No limit is placed on the total lengths of all queries (beyond that which can be inferred by using maximal-length messages), but one expects that a security proof, if it existed, would show a dependency on that.



```
Algorithm AESKW.Encrypt$_K^{ICV,\,H}(X)$
10  if $X \notin \{0,1\}^{\le 2^{38}-64}$ or $H \notin \text{BYTE}^{\le 255}$ or $H = X = \varepsilon$ or $|ICV| \ne 48$ then return $\perp$
11  $s \leftarrow 64 - (|H| + |X|) \bmod 64$
12  $P_1 \cdots P_n \leftarrow ICV \parallel [s]_8 \parallel [|H|/8]_8 \parallel H \parallel X$        // each $P_j$ of length 64. Necessarily $n \ge 2$
13  for $t \leftarrow 0$ to 5 do
14      for $i \leftarrow 1$ to $n-1$ do
15          $P_i \parallel P_{i+1} \leftarrow \text{AES}(P_{i+1} \parallel P_i) \oplus [6t+i]_{128}$        // each $P_j$ of length 64
16          $(P_1, P_2, P_3, \cdots, P_n) \leftarrow (P_n, P_1, P_2, \cdots, P_{n-1})$
17  return $P_1 \cdots P_n$
```

Figure 7: **Top:** Illustration of AESKW encryption, one of the four key-wrap algorithms in the X9.102 draft standard. Wires carry 64 bits and each block represents an AES call keyed with the underlying encryption key. On the left side of each block the input block's most significant 64 bits are on top, while on the right side of each block, the output block's most significant 64 bits are on bottom. The odd convention makes for fewer wire crossings. **Bottom:** Definition of the AESKW encryption algorithm. Decryption works in the natural way, verifying ICV, H, and the validity of the encoding.

▷ The definitional suggestions for AKW2 in [13, Section 2.4] are weak in focusing on random plaintexts. It would seem that a definition only needs to make the first block of the message be random, and even this block does not need to be hidden from the adversary. to this scheme.

In summary, a more precise definition would be desirable. For AESKW, TDKW, and a deterministic version of AKW1, we advocate a PRI as the desired notion. For AKW2, a specialized notion of security is required. We sketch one following our subsequent comments on AKW2.

THE AESKW AND TDKW SCHEMES. Encryption under AESKW is a deterministic function that maps a key K, a bit string X, an octet string H, and a six-byte integrity check vector ICV into a ciphertext a little longer than the sum of the lengths of $|X|$ and $|H|$. The mechanism uses an apparently new six-round Feistel-network variant; see Figure 7. We comment:

▷ The description of AESKW/TDKW in [13] is very awkward. The specification does not indicate the encryption and decryption signature; the length restriction on input X is not clearly stated (restrictions are stated on derived strings); plaintext formatting is viewed as a separate mechanism from encryption rather than a part of it; integrity-checking is viewed as a separate mechanism from decryption rather than a part of it; integrity checking is described before the encryption method is described; the algorithm specification repeatedly renames variables; and the provided picture does little to illustrate the algorithm's actual structure. Extracting the definition and drawing of Figure 7 took much work.

▷ The message space is unnatural; one can encrypt bit strings up to $2^{38} - 64$ bits? Algorithms should be designed to work on "natural" message spaces. Similarly, the restriction that either X is nonempty or H is nonempty is unnatural; what's allowed for header should be independent of what's allowed for a message.

▷ The length of the ciphertext increases with the length of the header. This goes against the notion of a header, which should be authenticated but *not* encrypted. It would be preferable if the ciphertext length were independent of the header length. In addition, the length of the ciphertext increases by at least eight bytes and at most 16 bytes minus one bit. It would be preferable if the length increased by exactly eight bytes regardless of the length of the message.

▷ It is not clear what is the intended semantics of "prerequisites"—is the ICV under the adversary's control or not? We interpret that the ICV should be treated definitionally just like the header.

▷ There is a mixture of bit-orientation and byte-orientation in the spec. It seems preferable to make everything bit strings or make everything byte strings.

▷ The xoring of the counter $6t+i$ into the blockcipher output is not explained; why is this done?

▷ The mechanism would be more natural if it reversed the most-significant 64-bits and the least-significant 64-bits in each AES call; that is, replace $\mathrm{AES}(P_{i+1} \parallel P_i)$ at line 15 of Figure 7 by $\mathrm{AES}(P_i \parallel P_{i+1})$ or, alternatively, replace $P_i \parallel P_{i+1}$ by $P_{i+1} \parallel P_i$. The current convention looks odd in the pseudocode and seems to make it impossible to draw a picture of the mechanism with a small number of wire crossings without establishing peculiar conventions.

▷ The string X will be "misaligned" (not fall on a word boundary) if H is of non-word length. This would cause unnecessary inefficiency in typical implementations. It seems preferable if mechanisms don't disrupt the alignment of H and X in doing internal work.

▷ The number of blockcipher calls seems large: roughly 12 per block of data (the same price paid for X or H). This is six times more than that used for AKW1.

▷ Even if the header H is held fixed, one must spend time (as well as bits) to re-authenticate it with each message that is encrypted or decrypted.

▷ There is no proof of security, and the mechanism is so complex that providing one would be difficult.

The above criticism notwithstanding, we find it likely that the mechanism is correct. Namely, the modified Feistel network illustrated in Figure 7 is, we conjecture, a secure enciphering scheme (in the sense of a strong, variable-input-length PRP). Scheme AESKW is then seen as an instance of the PTE paradigm, except that the header is folded into the plaintext instead of used to tweak the enciphering scheme.

Our comments on AESKW apply equally to TDKW. But for TDKW we appreciate the use of a multiround Feistel network more, for it is more important to go beyond mechanisms that have security degrading in $\sigma^2/2^\eta$, where η is the blockcipher blocksize and σ is the total number of blocks acted on. The modified Feistel network used here probably does have security (as a strong PRP) better than $\sigma^2/2^\eta$, but it would be hard to prove.

THE AKW1 SCHEME. We recreate an illustration of the AKW1 scheme in Figure 8. High-level comments about the mechanism are as follows.

▷ Whereas AESKW and TDKW are deterministic and stateless, and therefore have no chance to achieve semantic security [3, 15], algorithm AKW1 is probabilistic and can achieve that goal; it would seem to be a probabilistic AE scheme [6, 8, 20].

▷ But as a probabilistic AE scheme, AKW1 is highly atypical. It does not employ generic composition, nor is it obtained by optimizing a generic-composition scheme, nor does it employ techniques associated to one-pass AE. Furthermore, it is straightforward to achieve AE using two blockcipher calls per block, but AKW1 uses, beyond that, an application of SHA1. Why did the designers choose such an odd and comparatively expensive design? Perhaps the scheme wasn't actually meant to be "just" an AE scheme; maybe it should work even if the random-number generator used to make the IV fails (cf. [13, page 3, item 2]). But if one regards the IV as part of the header and looks to see if the resulting algorithm is a secure DAE scheme, the answer is *no*; for an attack,

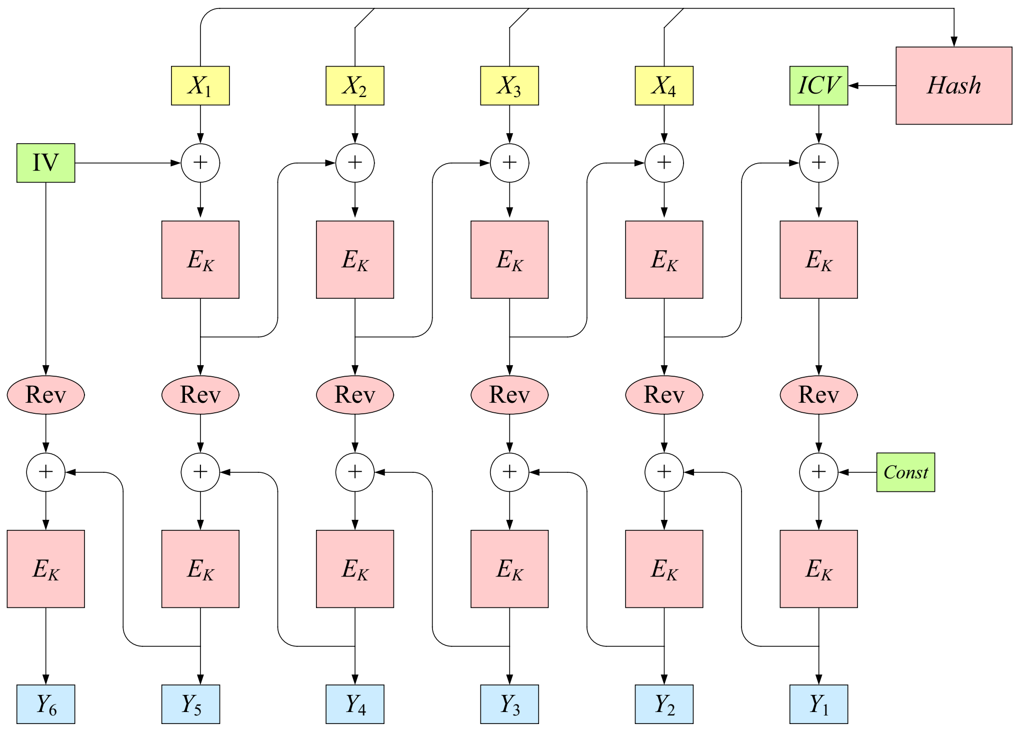


Figure 8: Encryption under AKW1. The boxes are TDEA, all keyed by the underlying encryption key, and the ovals reverse the bytes of their eight-byte inputs. The value $Const = \mathrm{0x4adda22c79e82105}$ is a fixed eight-byte string. The string IV is a random eight-byte string. The value $ICV = Hash(X_1X_2X_3X_4)$ is the first 64 bits of $\mathrm{SHA1}(X_1X_2X_3X_4)$.

find 64-bit strings A and B such that $Hash(A) = Hash(B)$ (this takes about 2^{32} time) and then notice that the encryption of (A, A) and (B, B) will have the same first block Y_1, which violates the goal of a DAE scheme. Perhaps the scheme is intended to function as a deterministic AE scheme when $IV = 0^n$, say. Probably the best explanation for the odd structure of AKW1 is that there is no explanation, according to a participant, and as revealed by the S/MIME working group's mail log, the scheme grew by accretion, with different people having their own goals and ideas, with no underlying design rationale.

▷ The constraint that the input to the algorithm must be a positive multiple of 64 bits seems an unfortunate limitation for a general-purpose algorithm.

▷ The lack of a header/associated-data is a significant limitation for a general-purpose PRI or AE scheme.

▷ The byte-reversal operations seem gratuitous; what is their purpose?

▷ There is no provable security result associated to AKW1.

▷ The AKW1 mechanism resembles CMC [16], which is a tweakable, proven-secure, wide-blocksize blockcipher. The paradigm of adding redundancy (even 0-bits) to a wide-blocksize blockcipher and then enciphering is the PTE construction of this paper. This suggests one way to eliminate the hash function and obtain a provable-secure construction at the same time.

THE AKW2 SCHEME. We recreate an illustration of the AKW2 scheme in Figure 9.

▷ The AKW2 mechanism is deterministic, but the goal cannot be that of a secure PRI, or even deterministic indistinguishability $\mathrm{detPriv}$, since encryption of plaintext block i does not impact any prior ciphertext block. As a consequence, ciphertexts leak equality of prefixes: the encryption of (H, P) and (H', P') reveals the length of the longest block-aligned prefix of P and P', assuming the first blocks of H and H' agree.

▷ There is no provable-security claim associated to the mode, and the key-separation method used in AKW2 precludes the possibility of proving security relative to a standard assumption. Provable security could be

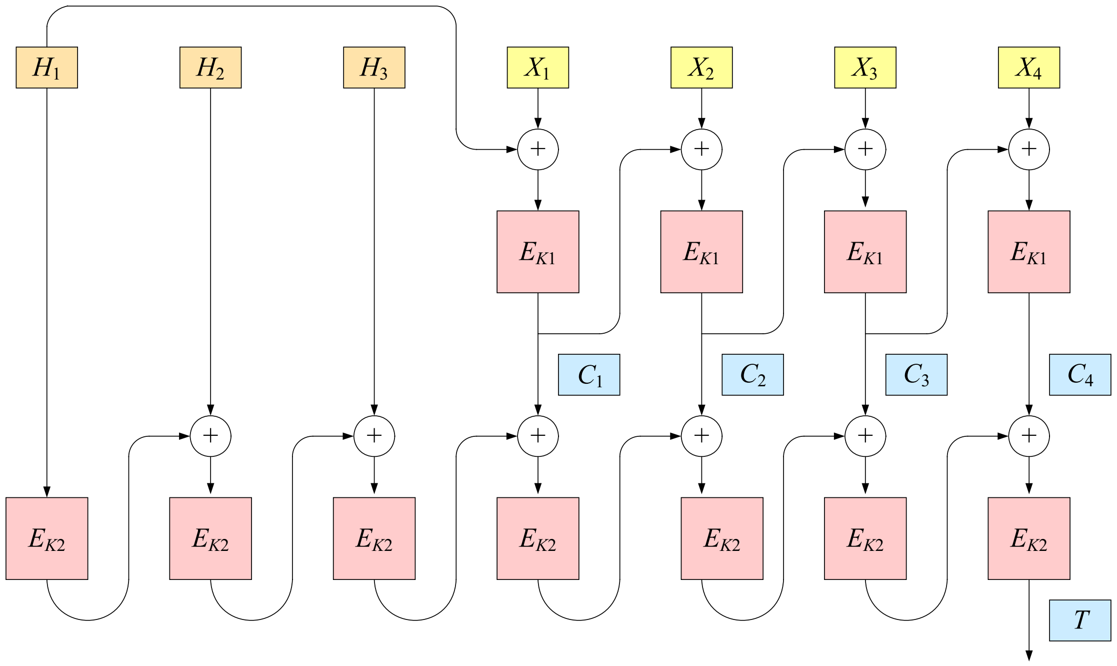


Figure 9: Encryption under AKW2. The header is $H_1 H_2 H_3$ and the message is $P_1 P_2 P_3 P_4$. The boxes are TDEA, where the first row is keyed by the confidentiality subkey $K' = K \oplus \mathrm{0x4545454545454545}$ and the second row is keyed by authentication subkey $K'' = K \oplus \mathrm{0x4d4d4d4d4d4d4d4d}$, where K is the underlying key. The ciphertext is $C_1 C_2 C_3 C_4 T$.

pursued by regarding the key as $K' \parallel K''$, or it is easy to describe a (nonstandard) assumption under which the style of key separation used by the mode works.

▷ The SIV construction of this paper is an alternative design approach having similar efficiency characteristics and that does achieve PRI-security.

Let us consider how to define security for this mode. Begin with privacy. For $b \in \{0,1\}$ the adversary is given an oracle $\mathrm{Enc1}_b$ that behaves as follows: on receipt of (H, M_0, M_1), where $|M_0| = |M_1|$, the oracle chooses a random $R \xleftarrow{\$} \{0,1\}^\eta$, where $\eta = 64$, and returns $(R,\ \mathcal{E}_K^H(R \parallel M_b))$. The adversary's goal for violating privacy is to ascertain if it has a "left oracle" ($b = 0$) or a "right oracle" ($b = 1$). One can measure the adversary's effectiveness by $\mathbf{Adv}_\Pi^{\mathrm{detPriv1}}(A) = \Pr[A^{\mathrm{Enc1}_1} \Rightarrow 1] - \Pr[A^{\mathrm{Enc1}_0} \Rightarrow 1]$. This is a weakening of the $\mathrm{detPriv}$-notion that is defined in Appendix B. The adversary's goal for violating authenticity would be the $\mathrm{detAuth}$-notion of authenticity from Appendix B: given a pair of oracles F_K, F_K^{-1} the adversary aims to make a right-oracle call of (H, C) where C was not the return value to a prior left-query (H, M) and where $F_K^{-1}(H, C) \neq \perp$. The goal for AKW2 would then be $\mathrm{detPriv1} + \mathrm{detAuth}$.

Let us assume that AKW2 actually achieves security in the $\mathrm{detPriv1}$+auth sense. Then one can distill out a simple and concrete usage restriction that could be stated in the mechanism's documentation: *the first block of plaintext P_1 should be random.* This may be a reasonable restriction for a key-wrapping mechanism.

B All-in-One vs. Two-Requirement Notions for AE

An alternative approach for defining DAE-security is to specify a notion for deterministic privacy, $\mathrm{detPriv}$, a notion for deterministic authenticity, $\mathrm{detAuth}$, and demand both. This "two-requirement" approach is the one that has been taken in all prior work on AE. In this section we specify the two-requirement definition for DAE and show where it leads: to a notion equivalent to our "all-in-one" definition. We go on to recall prior variants for AE security and explain that, in each case, the two-requirement definition is equivalent to the all-in-one definition.

DETERMINISTIC PRIVACY. We adapt the indistinguishability-from-random-bits notion of privacy [32] to the setting where the encryption scheme takes an header. Fix a DAE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with header space $\mathcal{H}$ and message space $\mathcal{X}$. Then, for A an adversary, define its **detPriv**-advantage in attacking Π as

$$\mathbf{Adv}_{\Pi}^{\mathrm{detPriv}}(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K}:\ A^{\mathcal{E}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$(\cdot,\cdot)} \Rightarrow 1\right]$$

where we assume that A does not repeat a query. Informally, adversary A is trying to determine if its oracle is enciphering its queries or returning random bits, and the trivial way to make that determination is barred.

DETERMINISTIC AUTHENTICITY. The usual notion of integrity of ciphertexts [6, 8, 20] must be adapted to the deterministic setting (the difference is just a matter of syntax). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a DAE with header space $\mathcal{H}$ and message space $\mathcal{X}$, and consider an adversary A with access to oracles for $\mathcal{E}_K$ and $\mathcal{D}_K$. We define A's **detAuth-advantage** in attacking Π as

$$\mathbf{Adv}_{\Pi}^{\mathrm{detAuth}}(A) \quad = \quad \Pr\left[K \xleftarrow{\$} \mathcal{K}:\ A^{\mathcal{E}_K(\cdot,\cdot),\, \mathcal{D}_K(\cdot,\cdot)} \text{ forges}\right] .$$

Above, when we say that A *forges* it means that it asks a right-query (H, Y) and gets a response other than $\perp$, and A did not earlier ask a left-query (H, X) that returned Y. We assume without loss of generality that A never asks a right-query (H, Y) having already asked a left-query (H, X) that returned Y.

EQUIVALENCE OF DETPRIV+DETAUTH-SECURITY AND DAE-SECURITY. Here we show that our all-in-one notion of DAE security is equivalent to the two-part notion that requires $\mathrm{detPriv}$ and $\mathrm{detAuth}$.

Proposition 8 [detPriv+detAuth implies DAE] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a DAE with header space $\mathcal{H}$ and message space $\mathcal{X}$. Let A be an adversary with access to two oracles. Suppose A runs in time t and asks q_L queries to its left oracle, these totaling μ_L bits, and asks q_R queries to its right oracle, these totaling μ_R bits. Then there exist adversaries D and F such that

$$\mathbf{Adv}_{\Pi}^{\mathrm{dae}}(A) \le \mathbf{Adv}_{\Pi}^{\mathrm{detPriv}}(D) + q_R\, \mathbf{Adv}_{\Pi}^{\mathrm{detAuth}}(F)$$

where D runs in time $t + O(\mu_L + \mu_R)$ and asks q_{L} queries totaling μ_L bits, and F runs in time $t + O(\mu_L + \mu_R)$, asking at most q_{L} left-queries and one right-query, these totaling at most $\mu_L + \mu_R$ bits. ▮

Proof: Let D^g operate by running A, answering left oracle queries (H, X) with $g(H, X)$, and responding to all right oracle queries with $\perp$. When A halts with output bit b, let D return b. Then

$$\begin{aligned}
\mathbf{Adv}_{\Pi}^{\mathrm{dae}}(A) &= \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] \\
&= \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] \\
&\qquad\qquad + \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] \\
&= \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] \\
&\qquad\qquad + \Pr[D^{\mathcal{E}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[D^{\$(\cdot,\cdot)} \Rightarrow 1] \\
&= \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] + \mathbf{Adv}_{\Pi}^{\mathrm{detPriv}}(D)
\end{aligned}$$

where $K \xleftarrow{\$} \mathcal{K}$ throughout. Let $\delta = \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1]$; it remains to bound this quantity. Let E be the event that A asks at least one valid right-oracle query (H, Y) (ie, $D_K(H, Y) \neq \perp$). We can write

$$\begin{aligned}
\delta &= \Big(\Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1 \wedge \mathsf{E}] + \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1 \wedge \overline{\mathsf{E}}]\Big) \\
&\quad - \Big(\Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1 \wedge \mathsf{E}] + \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1 \wedge \overline{\mathsf{E}}]\Big) \\
&= \Big(\Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1 \wedge \mathsf{E}] - \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1 \wedge \mathsf{E}]\Big) \\
&\quad + \Big(\Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1 \wedge \overline{\mathsf{E}}] - \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1 \wedge \overline{\mathsf{E}}]\Big) \\
&= \Big(\Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1 \wedge \mathsf{E}] - \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1 \wedge \mathsf{E}]\Big)
\end{aligned}$$

where the last equality holds since, if E does not occur, all right-oracle queries are answered by $\perp$ whether A had been provided with a $\mathcal{D}_K(\cdot,\cdot)$ oracle or a $\perp(\cdot,\cdot)$ oracle. Conditioning on event E we obtain

$$\begin{aligned}
\delta &= \Big(\Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1 \,|\, \mathsf{E}] - \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1 \,|\, \mathsf{E}]\Big) \Pr[\mathsf{E}] \\
&\leq \Pr[\mathsf{E}]
\end{aligned}$$

For $j \in [1..q_R]$ let E_j be the event that E occurs on the j^{th} right-oracle query. Then $\delta \leq \Pr[\mathsf{E}] \leq \sum_{i=1}^{q_R} \Pr[\mathsf{E}_j]$. It must be the case that $\Pr[\mathsf{E}_j] \geq \delta/q_R$ for some j. Fix this value of j and let F be the following forging adversary. Adversary F runs A, answering all of A's left-oracle queries with its own $\mathcal{E}_K$ oracle, and answering the first $j-1$ of A's right-oracle queries with $\perp$. When A asks its j^{th} right oracle query (H,Y), adversary F asks $\mathcal{D}_K(H,Y)$. Then $\mathbf{Adv}_{\Pi}^{\text{detAuth}}(F) \geq \Pr[\mathsf{E}_j] \geq \delta/q_R =$ so $\delta \leq q_R\, \mathbf{Adv}_{\Pi}^{\text{detAuth}}(A)$ and we are done. ▌

Proposition 9 [DAE implies detPriv+detAuth] Let $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$ be a DAE with header space $\mathcal{H}$ and message space $\mathcal{X}$. Let D be a detPriv-adversary that runs in time t and asks q queries to its oracle, these totaling μ bits. Let F be a detAuth-adversary that runs in time t' and asks q' queries totaling μ' bits. Then there exists adversaries A and A' such that

$$\begin{aligned}
\mathbf{Adv}_{\Pi}^{\text{dae}}(A) &\geq \mathbf{Adv}_{\Pi}^{\text{detPriv}}(D) \\
\mathbf{Adv}_{\Pi}^{\text{dae}}(A') &\geq \mathbf{Adv}_{\Pi}^{\text{detAuth}}(F)
\end{aligned}$$

where A runs in time t and asks at most q queries totaling μ bits, and where A' runs in time t' and asks at most q' queries totaling μ' bits. ▌

Proof: The first result is trivial, so we do not bother with it. The second is also simple. Let A run F, answering left-oracle queries with its left oracle (either $\mathcal{E}_K(\cdot,\cdot)$ or $\$(\cdot,\cdot)$) and right-oracle queries with its right oracle (either $\mathcal{D}_K(\cdot,\cdot)$ or $\perp(\cdot,\cdot)$). If any right oracle query returns a value other than $\perp$ then let A output 1; otherwise, it outputs 0. Notice that $\Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1] = \Pr[F^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \text{ forges }]$, and that $\Pr[A^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] = 0$, since in the latter case the right oracle always returns $\perp$. ▌

ALL-IN-ONE AND TWO-REQUIREMENT NOTIONS FOR AE ARE INVARIABLY EQUIVALENT. There are now several variants of AE: the encryption scheme may be probabilistic, nonce-based, deterministic, or misuse-resistant; the privacy requirement can be indistinguishability from random bits or conventional indistinguishability; and message headers may be present or absent, strings or vectors. For any of these variants one can give a two-requirement definition or an all-in-one definition. In all cases the results come out as above, showing that the all-in-one definition and the two-requirement definition are equivalent.

As a first example, the indistinguishability-from-random-bits notion of privacy we selected for detPriv and within DAE can be relaxed to conventional indistinguishability, formalized, say, by indistinguishability from the encryption of random bits. Each oracle $\$(\cdot,\cdot)$ gets changed to a $\mathcal{E}_K(\cdot,\$^{|\cdot|})$ oracle that encrypts as many random bits as the message-portion of its query is long. The all-in-one and two-requirements definitions will again be equivalent, with a proof just as before.

As a second example, consider probabilistic AE, no headers, privacy in the sense of conventional indistinguishability. The usual two-requirement definition [6, 8, 20] would specify

$$\begin{aligned}\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) &= \Pr[K \xleftarrow{\$} \mathcal{K}:\ A^{\mathcal{E}_K(\cdot)} \Rightarrow 1] - \Pr[K \xleftarrow{\$} \mathcal{K}:\ A^{\mathcal{E}(\$^{|\cdot|})} \Rightarrow 1]\\ \mathbf{Adv}^{\mathrm{auth}}_{\Pi}(A) &= \Pr[K \xleftarrow{\$} \mathcal{K}:\ A^{\mathcal{E}_K(\cdot)} \text{ forges}]\end{aligned}$$

and a good AE scheme would have to be secure in both of these senses. Then all-in-one definition would define

$$\mathbf{Adv}^{\mathrm{ae}}_{\Pi}(A) = \Pr[K \xleftarrow{\$} \mathcal{K}:\ A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)} \Rightarrow 1] - \Pr[K \xleftarrow{\$} \mathcal{K}:\ A^{\mathcal{E}_K(\$^{|\cdot|}),\perp(\cdot)} \Rightarrow 1]$$

where the adversary may not ask a right-query of C after this is returned by a left-query. It is again simple to show that the all-in-one definition and the two-requirement one are equivalent (where, as before, the all-in-one notion will have quantitatively tighter authenticity).

In general, we prefer the all-in-one definitions for authenticated-encryption, finding them more aesthetic and concise.

AE AS A FORM OF CHOSEN-CIPHERTEXT SECURITY. All-in-one definitions for AE resemble the definition for chosen-ciphertext-attack (CCA2) security [3, 4]: in the definition just given, say, change the $\perp(\cdot)$ oracle to a $\mathcal{D}_K(\cdot)$ oracle to recover the CCA2 notion for the same setting. The definition of AE thus strengthens CCA2 security in a simple and natural way. Perhaps it is only "historical accident" that our community came to think of AE as privacy+authenticity and not as "CCA3 security."

C DAEs Achieve Semantic Security when Plaintexts Carry a Key

A folklore justification for using a key-wrap scheme instead of a probabilistic, semantically secure encryption scheme is that, in the key-wrap setting, one expects the plaintext to carry a random cryptographic key, and so a probabilistic encryption scheme ought not be needed. In this section we provide a result that validates this intuition. We show that encoding a random key into the plaintext (the key may be dropped into the message in any fashion) and then applying a DAE will achieve what amounts to probabilistic AE—in particular, it achieves what amounts to semantic security. We begin with some definitions.

KEY INSERTION. A *key-insertion scheme* is a pair of algorithms $\Phi = (\mathrm{InsertKey},\ \mathrm{ExtractKey})$. The first algorithm is used to insert a key into a plaintext and the second algorithm is used to extract it. For the remainder, fix a constant κ, the length of the key to be inserted. Algorithm $\mathrm{InsertKey}$, on input of $X \in \{0,1\}^*$, chooses a random $R \xleftarrow{\$} \{0,1\}^\kappa$ and, depending on $|X|$, returns either $M \xleftarrow{\$} \mathrm{InsertKey}(X) \in \{0,1\}^*$ or the distinguished value $\perp$. An equivalent viewpoint is that $\mathrm{InsertKey}$ is a deterministic function that takes as input a string $X \in \{0,1\}^*$ and a random string $R \in \{0,1\}^\kappa$; then we write $M \leftarrow \mathrm{InsertKey}(X, R)$. The set of all strings X such that $M \xleftarrow{\$} \mathrm{InsertKey}(X)$ is a string is called the *message space* of Φ. We insist that if $M = \mathrm{InsertKey}(X, R)$ is a string then $|M| = |X| + e(|X|)$ for some fixed *expansion function* e. (Recall that we have fixed the key length κ and so, implicitly, the expansion depends on $\kappa = |R|$ as well as on $|X|$.) Algorithm $\mathrm{ExtractKey}$ takes a string $M \in \{0,1\}^*$ and, depending on $|M|$, returns either $\perp$ or the encoding of a pair of strings $\langle X, R\rangle$ with $|R| = \kappa$. The set of strings M such that $M = \mathrm{InsertKey}(X, R)$ for some R is called the *image*, $\mathcal{M}$, of Φ. We insist that if $M = \mathrm{InsertKey}(X, R) \neq \perp$ then $\mathrm{ExtractKey}(M) = \langle X, R\rangle$, and $\mathrm{ExtractKey}(M) = \perp$ for all $M \notin \mathcal{M}$. To simplify the subsequent theorem statement and capture the intent that $\mathrm{InsertKey}$ and $\mathrm{ExtractKey}$ are simple mappings, we require that they be computable in linear time.

INSERTKEY-THEN-DAE ENCRYPTION. Let $\Phi = (\text{InsertKey}, \text{ExtractKey})$ be a key-insertion scheme with message space $\mathcal{X}$, image $\mathcal{M}$, and key length κ. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a DAE with header space $\mathcal{H}$ and message space $\mathcal{M}$. We define from Φ and Π the probabilistic encryption scheme $\widetilde{\Pi} = (\mathcal{K}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{D}}) = \text{InsKey}[\Phi, \Pi]$ by

```
Algorithm $\widetilde{\mathcal{E}}_K(H, X)$
$R \xleftarrow{\$} \{0,1\}^\kappa$
$M \leftarrow \text{InsertKey}(X, R)$
if $M = \perp$ return $\perp$
return $\mathcal{E}_K(H, M)$
```

```
Algorithm $\widetilde{\mathcal{D}}_K(H, Y)$
$M \leftarrow \mathcal{D}_K(H, Y)$
if $M = \perp$ then return $\perp$
return $\text{ExtractKey}(M)$
```

The encryption scheme $\widetilde{\Pi}$ is nonstandard insofar as decryption of a ciphertext Y returns not only the underlying plaintext X but also the random bits R that were inserted (algorithm ExtractKey returns such a pair). The formalization should not be interpreted as meaning that the encrypting party that does not "know" R—indeed if it follows the algorithm above then it chooses R and therefore knows it. The return value from encrypt does not include R because the ciphertext that is to be sent to the receiver already incorporates it. On the other hand, the decryption algorithm does return R, as this value is conceptually a part of the plaintext. We must correspondingly strengthen the notion of security, providing the random bits R to the attacker. To do this, we must adapt the definition of AE. Consider an encryption oracle $\mathbb{E}_K(\cdot,\cdot)$ that behaves exactly as the encryption algorithm in $\widetilde{\Pi}$, above, but returns the random string R as part of the ciphertext. Specifically, on input (H, X), where $H \in \mathcal{H}$ and $X \in \mathcal{X}$, it computes: $R \xleftarrow{\$} \{0,1\}^\kappa$, $M \leftarrow \text{InsertKey}(X, R)$, $Y \leftarrow \mathcal{E}_K(H, M)$, and then returns an encoded string $\langle R, Y\rangle$. Let oracle $\$(\cdot,\cdot)$, on input (H, X), where $H \in \mathcal{H}$ and $X \in \mathcal{X}$, operate identically to $\mathbb{E}_K(\cdot,\cdot)$ but return $|\langle R, Y\rangle|$ random bits. Finally, consider a decryption oracle $\mathbb{D}_K(\cdot,\cdot)$ that, on input (H, Y) where $H \in \mathcal{H}$, computes $M \leftarrow \mathcal{D}_K(H, Y)$; then if $M \neq \perp$ then it computes $(R, X) \leftarrow \text{ExtractKey}(M)$ and returns (R, X), while otherwise it returns $\perp$. Define

$$\mathbf{Adv}^{\text{kiae}}_{\widetilde{\Pi}}(A) \;=\; \Pr[K \xleftarrow{\$} \mathcal{K}:\; A^{\mathbb{E}_K(\cdot,\cdot),\,\mathbb{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[K \xleftarrow{\$} \mathcal{K}:\; A^{\$(\cdot,\cdot),\,\perp(\cdot,\cdot)} \Rightarrow 1]\,.$$

Basically, when the adversary asks for the encryption of X we embellish the string to (X, R) for a random key R, inform the adversary of the random key that was inserted, and give the adversary the resulting ciphertext. We are saying that this looks like random bits, even in the presence of a decryption oracle. As usual, the adversary may not ask a right-query (H, C) following a left-query (H, M) that returned C.

We emphasize that the KIAE-notion is in effect the usual notion for probabilistic AE as it must be interpreted for a key-insertion scheme; some change is essential because, if nothing else, the syntax of a scheme has changed. But we have given the adversary all the abilities it would normally have in the probabilistic AE setting, and have taken away nothing. The adversary cannot specify the inserted key—that it does not control—but it learns the inserted key and it is otherwise in full control of the plaintexts.

INSERTKEY-THEN-DAE ACHIEVES KIAE. We now show that as long as the inserted key is "long enough" the InsertKey-then-DAE scheme achieves the version of probabilistic authenticated-encryption we have defined.

Theorem 10 Let $\Phi = (\text{InsertKey}, \text{ExtractKey})$ be a key-insertion scheme with message space $\mathcal{M}$ and image $\mathcal{X}$, and let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a DAE scheme with message space $\mathcal{X}$. Define $\widetilde{\Pi} = (\mathcal{K}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{D}}) = \text{InsKey}[\Phi, \Pi]$ and let B be an adversary (for attacking $\widetilde{\Pi}$). Suppose that B runs in time t and asks q queries totaling μ bits. Then there exists an adversary A (for attacking Π) where

$$\mathbf{Adv}^{\text{kiae}}_{\widetilde{\Pi}}(B) \le \mathbf{Adv}^{\text{dae}}_{\Pi}(A) + q^2/2^{\kappa-1}$$

where A runs in time at most $t' = t + O(\mu)$, and asks at most $q' = q$ queries of total length at most $\mu' = \mu + O(q)$. ▮

Proof: Let $\delta = \mathbf{Adv}^{\rm kiae}_{\widetilde{\Pi}}(B)$ and let $\mathbb{E}^*_K(\cdot,\cdot)$ be an oracle that behaves exactly as $\mathbb{E}$, except that it never uses the same random string R twice. Now, suppressing obvious notation, we have

$$\begin{aligned}
\delta &= \Pr[B^{\mathbb{E}_K(\cdot,\cdot),\mathbb{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[B^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] \\
&= \Pr[B^{\mathbb{E}_K(\cdot,\cdot),\mathbb{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[B^{\mathbb{E}^*_K(\cdot,\cdot),\mathbb{D}_K(\cdot,\cdot)} \Rightarrow 1] \\
&\quad + \Pr[B^{\mathbb{E}^*_K(\cdot,\cdot),\mathbb{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[B^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] \\
&\leq q^2/2^\kappa + \Pr[B^{\mathbb{E}^*_K(\cdot,\cdot),\mathbb{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[B^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] \\
&= q^2/2^\kappa + p
\end{aligned}$$

where the inequality holds because the observable behavior of $\mathbb{E}$ and $\mathbb{E}^*$ differs only when the former uses twice some randomly chosen string R, and this happens with probability at most $q^2/2^\kappa$ (by the sum bound).

To bound the probability p, we construct a DAE adversary $A^{g,h}$ that will run B and faithfully simulate either the pair of oracles $\mathbb{E}^*_K, \mathbb{D}_K$ (if $g = \mathcal{E}_K$ and $h = \mathcal{D}_K$), or the pair $\$, \perp$ (if $g = \$$ and $h = \perp$). Specifically, let $\mathcal{R}$ be initialized to the empty set and let A run B. When B asks a left-oracle query (H, X), adversary A chooses $R \xleftarrow{\$} \{0,1\}^\kappa$. If $R \in \mathcal{R}$, A outputs 0 and halts. Otherwise, it computes $M \leftarrow \mathrm{InsertKey}(X, R)$, $Y \leftarrow g(H, M)$, sets $\mathcal{R} \leftarrow \mathcal{R} \cup \{R\}$, and returns $\langle R, Y\rangle$ to B. When B asks a right-oracle query (H, Y), let A compute $M \leftarrow h(H, Y)$. If $M = \perp$ then A returns $\perp$ to B; otherwise A computes $\langle R, X\rangle \leftarrow \mathrm{ExtractKey}(M)$ and returns $\langle R, X\rangle$ to B. When B halts with bit b, let A output b.

$$\begin{aligned}
\mathbf{Adv}^{\rm dae}_{\Pi}(A) &= \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1] \\
&= \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1|\mathsf{BAD}]\Pr[\mathsf{BAD}] + \Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1|\overline{\mathsf{BAD}}]\Pr[\overline{\mathsf{BAD}}] \\
&\quad - \Pr[A^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1|\mathsf{BAD}]\Pr[\mathsf{BAD}] - \Pr[A^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1|\overline{\mathsf{BAD}}]\Pr[\overline{\mathsf{BAD}}] \\
&= \Big(\Pr[A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1|\overline{\mathsf{BAD}}] - \Pr[A^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1|\overline{\mathsf{BAD}}]\Big)\Pr[\overline{\mathsf{BAD}}] \\
&= \Big(\Pr[B^{\mathbb{E}^*_K(\cdot,\cdot),\mathbb{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[B^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1]\Big)\Pr[\overline{\mathsf{BAD}}] \\
&= p\,\Pr[\overline{\mathsf{BAD}}] \\
&= p - p\,\Pr[\mathsf{BAD}] \\
&\geq p - \Pr[\mathsf{BAD}] \\
&\geq p - q^2/2^\kappa
\end{aligned}$$

where, as before, we have bounded the probability of an R-repeat (ie,BAD) by $q^2/2^\kappa$. Rearranging to $p \leq \mathbf{Adv}^{\rm dae}_{\Pi}(A) + q^2/2^\kappa$, and putting it all together, we have $\mathbf{Adv}^{\rm kiae}_{\widetilde{\Pi}}(B) \leq \mathbf{Adv}^{\rm dae}_{\Pi}(A) + q^2/2^{\kappa-1}$.

Noting that the $\mathrm{InsertKey}$ is computed in linear time (necessary for the time bound t', the theorem follows.

▌

D Building a DAE Scheme: The PTE Constructions

A folklore approach for achieving authenticity is to add redundancy and then encrypt, an approach investigated in works like [1, 8]. One pads the plaintext (for example, by appending a particular number of zero-bits) and then applies a length-preserving enciphering scheme (that is, a wide-blocksize blockcipher, like CMC [16]). We call this the *pad-then-encipher* (PTE) approach.

To accommodate an associated header under this paradigm either (a) use it as a tweak for the enciphering scheme, or (b) incorporate it into the plaintext before enciphering. The former will be more efficient in terms of the length of the resulting ciphertext, but it requires the underlying enciphering scheme to be tweakable.

Three of the four X9.102 key-wrap schemes (AESKW, TDKW, and AKW1) can be seen as instances of pad-then-encipher (although they use enciphering schemes for which there has been offered no proof of security). In this section we formalize and prove security for pad-then-encipher, for both options (a) and (b).

PADDING SCHEMES. A *padding scheme* is a pair of deterministic algorithms $\Phi = (\mathrm{Pad}, \mathrm{Unpad})$ where $\mathrm{Pad}, \mathrm{Unpad}\colon \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$. The set $\mathcal{X} = \{X \in \{0,1\}^*\colon \mathrm{Pad}(X) \in \{0,1\}^*\}$ is the *domain* of Φ and $\mathcal{M} = \{\mathrm{Pad}(X)\colon X \in \mathcal{X}\}$ is the *range* of Φ. Our convention is that $\mathrm{Pad}(\perp) = \mathrm{Unpad}(\perp) = \perp$. We insist that $\mathrm{Unpad}(\mathrm{Pad}(X)) = X$ for all $X \in \mathcal{X}$, that $\mathrm{Unpad}(M) = \perp$ for all $M \notin \mathcal{M}$, that $|\mathrm{Pad}(X)| = |X| + e(|X|)$ for some *expansion function* e, that Pad and Unpad are linear-time computable, and that $X \in \mathcal{X} \Rightarrow \{0,1\}^{|X|} \subseteq \mathcal{X}$. Call $s = \min_{X\in\mathcal{X}}\{e(|X|)\}$ the *stretch* of Φ. We emphasize that unpadding not only extracts the padding but, equally important, it returns $\perp$ if the presented point is not a properly padded domain point. If a padding function has stretch $s \geq 1$ then a fraction at most 2^{-s} of the points in $\mathcal{M}$ will unpad to give strings, while the remainder will unpad to give $\perp$.

ENCODING SCHEMES. We have already spoken of encoding schemes as reversible and easily computable mappings from tuples of vectors to strings. Here we will be more formal. An *encoding scheme* is a pair of deterministic algorithms $\Lambda = (\mathrm{Encode}, \mathrm{Decode})$ where $\mathrm{Encode}\colon \{0,1\}^{**} \to \{0,1\}^* \cup \{\perp\}$ and $\mathrm{Decode}\colon \{0,1\}^* \to \{0,1\}^{**} \cup \{\perp\}$. The *domain* of Λ is the set of tuples $\mathcal{Y} = \{Y \in \{0,1\}^{**}\colon \mathrm{Encode}(Y) \in \{0,1\}^*\}$.

We assume this to be a cross-product of sets of strings, and insist that if $Y \in \mathcal{Y}$ then $Y' \in \mathcal{Y}$ when $|Y'| = |Y|$ and the corresponding components of Y and Y' have equal lengths. The *range* of Λ is the set of strings $\mathcal{M} = \{\mathrm{Encode}(Y)\colon Y \in \mathcal{Y}\}$. We insist that $\mathrm{Decode}(\mathrm{Encode}(Y)) = Y$ for all $Y \in \mathcal{Y}$, that $\mathrm{Decode}(M) = \perp$ if $M \notin \mathcal{M}$, and that $\mathrm{Encode}(\perp) = \mathrm{Decode}(\perp) = \perp$. We assume that Encode and Decode are linear-time computable.

THE PTE1 CONSTRUCTION. The PTE1 construction builds a DAE out of an enciphering scheme by padding the input message prior to enciphering, and by using the header directly as the header (or tweak) of the underlying enciphering scheme. (We recall that an enciphering scheme is a length-preserving DAE.) Fix a padding scheme $\Phi = (\mathrm{Pad}, \mathrm{Unpad})$ with domain $\mathcal{X}$ and range $\mathcal{M}$, and an enciphering scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with header space $\mathcal{H}$ and message space $\mathcal{M}$. Then we define the DAE scheme $\widetilde{\Pi} = (\mathcal{K}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{D}})$ with header space $\mathcal{H}$ and message space $\mathcal{X}$, written $\widetilde{\Pi} = \mathrm{PTE1}[\Phi, \Pi]$, as

```
**Algorithm** $\widetilde{\mathcal{E}}_K(H, X)$
$M \leftarrow \mathrm{Pad}(X)$
**return** $\mathcal{E}_K(H, M)$

**Algorithm** $\widetilde{\mathcal{D}}_K(H, Y)$
$M \leftarrow \mathcal{D}_K(H, Y)$
**return** $X \leftarrow \mathrm{Unpad}(M)$
```

Theorem 11 tells us that if Π is a an enciphering scheme that is secure in the PRI sense, then $\widetilde{\Pi}$ is a secure length-increasing DAE. In the proof, we will make use of the notation $\mathrm{Perm}^{\mathcal{H}}(\mathcal{M})$ to mean the space of all maps $\pi\colon \mathcal{H} \times \mathcal{M} \to \mathcal{M}$ such that $|\pi(H, M)| = |M|$ and $\pi(H, \cdot)$ is a permutation; ie., the space of all length-preserving injections from $\mathcal{H} \times \mathcal{M}$ to $\mathcal{M}$. Also, recall our notational convention is to use $\mathbf{Adv}_{\Pi}^{\pm\widetilde{\mathrm{prp}}}$ in place of $\mathbf{Adv}_{\Pi}^{\mathrm{pri}}$ when Π is an enciphering scheme with a nontrivial header (tweak) space.

Theorem 11 Let $\Phi = (\mathrm{Pad}, \mathrm{Unpad})$ be a padding scheme with domain $\mathcal{X}$, range $\mathcal{M}$, expansion function e, stretch s, and let $\tau = \min_{X\in\mathcal{X}}\{|X|\}$. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an enciphering scheme with header space $\mathcal{H}$ and message space $\mathcal{M}$. Let $\widetilde{\Pi} = (K, \widetilde{\mathcal{E}}, \widetilde{\mathcal{D}}) = \mathrm{PTE1}[\Phi, E]$. Let B be a DAE-adversary that runs in time t, asks q_{L} left-queries, these of total length μ_{L} bits, and asks q_{R} right-queries, these of total length μ_{R} bits. Let $q = q_{\mathrm{L}} + q_{\mathrm{R}}$ and $\mu = \mu_{\mathrm{L}} + \mu_{\mathrm{R}}$. Then there exists an adversary A such that

$$\mathbf{Adv}_{\Pi}^{\pm\widetilde{\mathrm{prp}}}(A) \quad \geq \quad \mathbf{Adv}_{\widetilde{\Pi}}^{\mathrm{dae}}(B) - (q^2/2^{s+\tau+1} + 4q_{\mathrm{R}}/2^s)$$

where A runs in time $t + O(\mu)$ and asks q queries of total length $\mu + O(q)$. ▮

Proof: We construct an adversary $A^{g,h}$ for attacking the PRI-security of Π as follows. Let A run B answering left-oracle queries (H, X) by computing $M \leftarrow \mathrm{Pad}(X)$ and returning $g(H, M)$ to B. To answer right-oracle queries (H, Y), adversary A asks $M \leftarrow h(H, Y)$ returns $\perp$ to B if $M = \perp$, and otherwise returns to B the result of $\mathrm{Unpad}(M)$. When B halts with output bit b, let A output b as well.

Recall that A's oracles are instantiated either as $g = \mathcal{E}_K, h = \mathcal{D}_K$ for a random $K \in \mathcal{K}$, or as $g = \pi, h = \pi^{-1}$ for a random element $\pi \in \mathrm{Perm}^{\mathcal{H}}(\mathcal{M})$. In the former case, A perfectly simulates for B a left oracle $\widetilde{\mathcal{E}}_K$ and right oracle $\widetilde{\mathcal{D}}_K$. In the latter case, A simulates for B a left oracle $\widetilde{\mathcal{E}}_\pi$ that computes $\widetilde{\mathcal{E}}$ but with the underlying enciphering algorithm $\mathcal{E}$ replaced by π, and a right oracle $\widetilde{\mathcal{D}}_\pi$ that computes $\widetilde{\mathcal{D}}$ but with the underlying deciphering algorithm $\mathcal{D}$ replaced by π^{-1}. Now,

$$\begin{aligned}
\mathbf{Adv}_{\Pi}^{\pm\widetilde{\mathrm{prp}}}(A) &= \Pr\left[A^{\mathcal{E}_K(\cdot,\cdot),\,\mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\pi(\cdot,\cdot),\,\pi^{-1}(\cdot,\cdot)} \Rightarrow 1\right] \\
&= \Pr\left[B^{\widetilde{\mathcal{E}}_K(\cdot,\cdot),\,\widetilde{\mathcal{D}}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[B^{\widetilde{\mathcal{E}}_\pi(\cdot,\cdot),\,\widetilde{\mathcal{D}}_\pi(\cdot,\cdot)} \Rightarrow 1\right] \\
&= \Pr\left[B^{\widetilde{\mathcal{E}}_K(\cdot,\cdot),\,\widetilde{\mathcal{D}}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[B^{\$(\cdot,\cdot),\,\perp(\cdot,\cdot)} \Rightarrow 1\right] \\
&\quad - \left(\Pr\left[B^{\widetilde{\mathcal{E}}_\pi(\cdot,\cdot),\,\widetilde{\mathcal{D}}_\pi(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[B^{\$(\cdot,\cdot),\,\perp(\cdot,\cdot)} \Rightarrow 1\right]\right) \\
&= \mathbf{Adv}_{\widetilde{\Pi}}^{\mathrm{dae}}(B) - \alpha
\end{aligned}$$

where $\alpha = \Pr\left[B^{\widetilde{\mathcal{E}}_\pi(\cdot,\cdot),\,\widetilde{\mathcal{D}}_\pi(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[B^{\$(\cdot,\cdot),\,\perp(\cdot,\cdot)} \Rightarrow 1\right]$ and, throughout, $K \xleftarrow{\$} \mathcal{K}$ and $\pi \leftarrow \mathrm{Perm}^{\mathcal{H}}(\mathcal{M})$ are understood. It remains to bound the information-theoretic quantity α.

It is easy to see that, for a random $\pi \in \mathrm{Perm}^{\mathcal{H}}(\mathcal{M})$, $\widetilde{\mathcal{E}}_\pi$ is a random element from $\mathsf{Inj}_e^{\mathcal{H}}(\mathcal{X}, \mathcal{M})$. To see this, notice that $\widetilde{\mathcal{E}}_\pi(H, X) = \pi(H, \mathrm{Pad}(X))$ where $\mathrm{Pad}(X)$ deterministically and reversible maps $X \in \mathcal{X}$ into a string in $M \in \mathcal{M}$. Thus we can appeal directly to Theorem 7, and conclude that $\alpha \le q^2/2^{s+\tau+1} + 4q_{\mathrm{R}}/2^s$. ▮

THE PTE2 CONSTRUCTION. The PTE2 construction builds a DAE from an enciphering scheme by encoding the header and the plaintext into a string, padding that string, and then enciphering the result. Fix an encoding scheme $\Lambda = (\mathrm{Encode}, \mathrm{Decode})$ with domain $\mathcal{H} \times \mathcal{X}$ and range $\mathcal{M}$. Fix a padding scheme $\Phi = (\mathrm{Pad}, \mathrm{Unpad})$ with domain $\mathcal{M}$ and range $\mathcal{M}^*$. Fix an enciphering scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with message space $\mathcal{M}^*$ and no tweak space (ie, a singleton set, which is ignored). Then we define the DAE scheme $\widetilde{\Pi} = (\mathcal{K}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{D}})$ with header space $\mathcal{H}$ and message space $\mathcal{X}$, written $\widetilde{\Pi} = \mathrm{PTE2}[\Lambda, \Phi, \Pi]$, as

```
Algorithm $\widetilde{\mathcal{E}}_K(H, X)$
$M \leftarrow \mathrm{Encode}(H, X)$
$M^* \leftarrow \mathrm{Pad}(M)$
return $\mathcal{E}_K(M^*)$
```

```
Algorithm $\widetilde{\mathcal{D}}_K(H, Y)$
$M^* \leftarrow \mathcal{D}_K(Y)$
$M \leftarrow \mathrm{Unpad}(M^*)$
if $\mathrm{Decode}(M) = \perp$ then return $\perp$
$(H', X) \leftarrow \mathrm{Decode}(M)$
if $H' = H$ then return $X$
return $\perp$
```

We point out that the header and message spaces of $\widetilde{\Pi}$ are determined by the domain of the encoding scheme. The following theorem tells us that if Π is an enciphering scheme secure in the PRI sense, then $\widetilde{\Pi}$ is a secure length-increasing DAE. Since we consider enciphering schemes with no tweak space, we use $\mathrm{Perm}(\mathcal{M})$ instead of $\mathrm{Perm}^{\{\varepsilon\}}(\mathcal{M})$, and use $\mathbf{Adv}_{\Pi}^{\pm\mathrm{prp}}$ in place of $\mathbf{Adv}_{\Pi}^{\mathrm{pri}}$.

Theorem 12 Let $\Lambda = (\mathrm{Encode}, \mathrm{Decode})$ be an encoding scheme with domain $\mathcal{H} \times \mathcal{X}$, range $\mathcal{M}$, and let $\tau = \min_{X \in \mathcal{X}} |X|$. Let $\Phi = (\mathrm{Pad}, \mathrm{Unpad})$ be a padding scheme with domain $\mathcal{M}$, range $\mathcal{M}^*$, and stretch s. Let

$\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an enciphering scheme with header space $\mathcal{H}$ and message space $\mathcal{M}^*$. Let $\widetilde{\Pi} = (K, \widetilde{\mathcal{E}}, \widetilde{\mathcal{D}}) = \mathrm{PTE2}[\Lambda, \Phi, \Pi]$. Let B be a DAE-adversary that runs in time t, asks $q_{\mathrm{L}} \leq 2^{s-1}$ left queries of total length μ_{L} bits, and q_{R} right queries of total length μ_R bits. Let $q = q_{\mathrm{L}} + q_{\mathrm{R}}$ and $\mu = \mu_{\mathrm{L}} + \mu_{\mathrm{R}}$. Then there exists an adversary A such that $\mathbf{Adv}_{\Pi}^{\pm\mathrm{prp}}(A) \geq \mathbf{Adv}_{\widetilde{\Pi}}^{\mathrm{dae}}(B) - (2q_{\mathrm{R}}/2^s + q_{\mathrm{L}}^2/2^{s+\tau+1})$ where A runs in time $t + O(\mu)$ and asks q queries of total length $\mu + O(q)$. ▌

Proof: We construct an adversary $A^{g,h}$ for attacking the PRI-security of Π as follows. Let A run B answering left-oracle queries (H, X) by computing $M \leftarrow \mathrm{Encode}(H, X)$, $M^* \leftarrow \mathrm{Pad}(X)$ and returning $g(M^*)$ to B. To answer right-oracle queries (H, Y), adversary A: asks $M \leftarrow h(Y)$; computes $M \leftarrow \mathrm{Unpad}(M^*)$; computes $\mathrm{Decode}(M)$, returning $\perp$ to B if this results in $\perp$, and otherwise assigning $(H', X) \leftarrow \mathrm{Decode}(M)$; returns X to B if $H = H$, and $\perp$ if not. When B halts with output bit b, let A output b as well.

Recall that A's oracles are instantiated either as $g = \mathcal{E}_K, h = \mathcal{D}_K$ for a random $K \in \mathcal{K}$, or as $g = \pi, h = \pi^{-1}$ for a random element $\pi \in \mathrm{Perm}(\mathcal{M})$. In the former case, A perfectly simulates for B a left oracle $\widetilde{\mathcal{E}}_K$ and right oracle $\widetilde{\mathcal{D}}_K$. In the latter case, A simulates for B a left oracle $\widetilde{\mathcal{E}}_\pi$ that computes $\widetilde{\mathcal{E}}$ but with the underlying enciphering algorithm replaced by π, and a right oracle $\widetilde{\mathcal{D}}_\pi$ that computes $\widetilde{\mathcal{D}}$ but with the underlying deciphering algorithm replaced by π^{-1}. Now,

$$\begin{aligned}
\mathbf{Adv}_{\Pi}^{\pm\mathrm{prp}}(A) &= \Pr\left[A^{\mathcal{E}_K(\cdot,\cdot),\, \mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\pi(\cdot,\cdot),\, \pi^{-1}(\cdot,\cdot)} \Rightarrow 1\right] \\
&= \Pr\left[B^{\widetilde{\mathcal{E}}_K(\cdot,\cdot),\, \widetilde{\mathcal{D}}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[B^{\widetilde{\mathcal{E}}_\pi(\cdot,\cdot),\, \widetilde{\mathcal{D}}_\pi(\cdot,\cdot)} \Rightarrow 1\right] \\
&= \Pr\left[B^{\widetilde{\mathcal{E}}_K(\cdot,\cdot),\, \widetilde{\mathcal{D}}_K(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[B^{\$(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right] \\
&\quad - \left(\Pr\left[B^{\widetilde{\mathcal{E}}_\pi(\cdot,\cdot),\, \widetilde{\mathcal{D}}_\pi(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[B^{\$(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right]\right) \\
&= \mathbf{Adv}_{\widetilde{\Pi}}^{\mathrm{dae}}(B) - \alpha
\end{aligned}$$

where $\alpha = \Pr\left[B^{\widetilde{\mathcal{E}}_\pi(\cdot,\cdot),\, \widetilde{\mathcal{D}}_\pi(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[B^{\$(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right]$ and, throughout, $K \xleftarrow{\$} \mathcal{K}$ and $\pi \leftarrow \mathrm{Perm}(\mathcal{M})$ are understood. It remains to bound the information-theoretic quantity α.

Claim: $\alpha \leq 2q_{\mathrm{R}}/2^s + q_{\mathrm{L}}^2/2^{s+\tau+1}$

Proof: Write $\alpha = \alpha_1 + \alpha_2$ where

$$\begin{aligned}
\alpha_1 &= \Pr\left[B^{\widetilde{\mathcal{E}}_\pi(\cdot,\cdot),\, \widetilde{\mathcal{D}}_\pi(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[B^{\widetilde{\mathcal{E}}_\pi(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right] \\
\alpha_2 &= \Pr\left[B^{\widetilde{\mathcal{E}}_\pi(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[B^{\$(\cdot,\cdot),\, \perp(\cdot,\cdot)} \Rightarrow 1\right]
\end{aligned}$$

We will show that $\alpha_1 \leq 2q_{\mathrm{R}}/2^s$ and $\alpha_2 \leq q_{\mathrm{L}}^2/2^{s+\tau+1}$, establishing the claim.

For bounding α_1 we can assume that adversary B halts and outputs 1 as soon as some right-oracle query returns a valid string. Let BAD be the event B asks a right-oracle query that returns a string X. Conditioning probabilities on BAD leads to $\alpha_1 \leq \Pr[\mathsf{BAD}]$. Notice that $\widetilde{\mathcal{D}}_\pi(H, Y)$ returns a string if and only if $\mathrm{Unpad}(M^*) = M \neq \perp$ and $\mathrm{Decode}(M) = (H', X) \neq \perp$ and $H' = H$. Moreover, if $\mathrm{Unpad}(M^*) = \perp$, then by our conventions the other two conjuncts must be false. Letting U be the event that some right-query causes Unpad to return a string, we have $\Pr[\mathsf{BAD}] \leq \Pr[\mathsf{U}]$. Let U_i be the event that U occurs on the i^{th} query, $i \in [1..q_{\mathrm{R}}]$. We will now bound $\Pr[\mathsf{U}]$ by bounding $\Pr[\mathsf{U}_i]$.

Fix an $n \geq 0$ such that $\{0,1\}^n \subset \mathcal{M}^*$ and let

$$\mathcal{V}(n) = |\{M^* \in \{0,1\}^n : \mathrm{Unpad}(M) \neq \perp\}| \ .$$

```
On query left(H, X):
10  $M \leftarrow \text{Encode}(H,X)$;    $M^* \leftarrow \text{Pad}(M)$
11  if $M^* = \perp$ then return $\perp$
12  $c \leftarrow |M^*|$;    $Y \xleftarrow{\$} \{0,1\}^c$
13  if $Y \in \text{Image}(\pi)$ then
14      bad ← true  , $Y \xleftarrow{\$} \{0,1\}^c - \text{Image}(\pi)$
14  return $\pi(M^*) \leftarrow Y$
On query right(H, Y):
27  return $\perp$
```

Figure 10: Games used to bound α_2 in the proof of Theorem 12. Game G1 is the complete code; game G0 omits the shaded statement.

Since the padding scheme has stretch s, we know that $\mathcal{V}(n)/2^n \leq 2^{-s}$ for all n. At the time of the i^{th} right oracle query, adversary B can know at most q_{L} valid encipherings under $\widetilde{\mathcal{E}}_\pi$. Since B is forbidden to ask (H,Y) of its right oracle if some left-oracle query (H,X) returned Y, the probability that U_i occurs is at most $\mathcal{V}(n)/(2^n - q_{\text{L}})$. If $\mathcal{V}(n) = 0$ this probability is zero (which is certainly less than the claimed upperbound), so assume that $\mathcal{V}(n) \geq 1$. Then $\Pr[\mathsf{U}_i] \leq \mathcal{V}(n)/(2^n - q_{\text{L}}) \leq (2^n)(2^{-s})/(2^n - q_{\text{L}})$. Now, we have assumed that $q_{\text{L}} \leq 2^{s-1}$; by substitution for 2^{s-1} and since $\mathcal{V}(n) \geq 1$, we have $q_{\text{L}} \leq 2^n/2\mathcal{V}(n) \leq 2^{n-1}$. Hence $\Pr[\mathsf{U}_i] \leq (2^n)(2^{-s})/2^{n-1} = 2(2^{-s})$ and, finally, $\Pr[\mathsf{U}] \leq \sum_{i=1}^{q_{\text{R}}} \Pr[\mathsf{U}_i] \leq q_{\text{R}}/2^{s-1}$. Putting it all together yields the claimed bound $\alpha_1 \leq 2q_{\text{R}}/2^s$.

For α_2, a simple game playing argument shows that $\alpha_2 \leq q_{\text{L}}^2/2^{s+\tau+1}$. Consider the games G1 and G0 in Figure 10. Recall that booleans are initialized to `false`, sets are initialized to empty, and partial functions are initialized to everywhere undefined with the symbol `undef`. The set $\text{Image}(\pi)$ contains all points $Y \neq \texttt{undef}$ such that $\pi(M^*) = Y$ for some $M \in \mathcal{M}^*$. Set difference is indicated with a minus sign. Both games G1 and G0 simulate a $\perp$ oracle on the right. We claim that game G1 faithfully simulates an $\widetilde{\mathcal{E}}_\pi$ oracle on the left, while game G0 faithfully simulates an \$ left oracle. Let's examine what happens when a left query (H,X) is made. The result of $\text{Pad}(\text{Encode}(H,X))$ is assigned to M^*, and if $M^* = \perp$, then $\perp$ is returned. Since oracles are always defined to return $\perp$ when queried outside of their domains, this is consistent with both $\widetilde{\mathcal{E}}_\pi$ and \$. A random string of $c = |M^*|$ bits is then selected and assigned to Y. If Y is in $\text{Image}(\pi)$, then the flag *bad* is set to `true`; here is where the games begin to behave differently. In game G1, to observe the permutivity of π, a new point is selected from among the unused c-bit strings, and this is subsequently assigned to $\pi(M^*)$ and returned. Game G0, on the other hand, continues on with the uniform value Y, ultimately returning it.

Under our convention that adversaries not repeat queries, it is clear that $\alpha_2 = \Pr[B^{\text{G1}} \Rightarrow 1] - \Pr[B^{\text{G0}} \Rightarrow 1]$. Moreover, since these games are identical until *bad* is set, we can invoke the fundamental lemma of game-playing [7] and state that $\alpha_2 \leq \Pr[B^{\text{G0}}$ sets *bad*]. Now, prior to *bad* being set in game G0, each left query adds a single point to $\text{Image}(\pi)$. Accordingly, the probability that *bad* is set on the i^{th} query is at most $(i-1)/2^{s+\tau}$, so the probability that it is ever set is at most $q_{\text{L}}^2/2^{s+\tau+1}$ and we are done. ▮

E Proof of Security for S2V

Proof: Consider the game S0 defined in Figure 11. The game is a faithful simulation of $F = f^*$. The intuition underlying this formulation of F is as follows. We grow a random function ρ to compute $\rho(X_1), \ldots, \rho(X_{m-1})$ for each query $(X_1, \ldots, X_m)$, and we grow a separate random function ρ' for the final call $\rho(T)$. But whenever we need a value $\rho(I)$ we force it to take on the value $\rho'(I)$ if the latter has already been defined, and whenever we need a value $\rho'(I)$ we force it to take on the value $\rho(I)$ if the latter has already been defined, but if either

```
**Initialize**                 Game S0 (as written) and S1 (without the highlighted statements)
100  $\rho(\mathbf{0}) \xleftarrow{\$} \{0,1\}^n, \quad \rho(\mathbf{1}) \xleftarrow{\$} \{0,1\}^n$

**On query** $F(X_1, \ldots, X_m)$
100  **if** $m = 0$ **then return** $\rho(\mathbf{1})$
101  $S \leftarrow \rho(\mathbf{0})$
112  **for** $i \leftarrow 1$ **to** $m - 1$ **do**
113      **if** $X_i \in \text{Domain}(\rho')$ **then** *bad* $\leftarrow$ true, $\rho(X_i) \leftarrow \rho'(X_i)$
114      **if** $X_i \notin \text{Domain}(\rho)$ **then** $\rho(X_i) \xleftarrow{\$} \{0,1\}^n$
115      $S \leftarrow \mathbf{2}S \oplus \rho(X_i)$
116  **if** $|X_m| \geq n$ **then** $T \leftarrow S \oplus_{\text{end}} X_m$ **else** $T \leftarrow \mathbf{2}S \oplus X_m 10^*$
117  **if** $T \in \text{Domain}(\rho)$ **then** *bad* $\leftarrow$ true, **return** $\rho'(T) \leftarrow \rho(T)$
118  **if** $T \in \text{Domain}(\rho')$ **then** *bad* $\leftarrow$ true, **return** $\rho'(T)$
119  **return** $\rho'(T) \xleftarrow{\$} \{0,1\}^n$
```

```
200  $\rho(\mathbf{0}) \xleftarrow{\$} \{0,1\}^n, \quad \rho(\mathbf{1}) \leftarrow C$                Game S2
201  **for** $t \leftarrow 1$ **to** $q$ **do**
202      $S \leftarrow \rho(\mathbf{0})$
203      **for** $i \leftarrow 1$ **to** $m_t - 1$ **do**
204          **if** $X_i^t \in \text{Domain}(\rho')$ **then** *bad* $\leftarrow$ true
205          **if** $X_i^t \notin \text{Domain}(\rho)$ **then** $\rho(X_i^t) \xleftarrow{\$} \{0,1\}^n$
206          $S \leftarrow \mathbf{2}S \oplus \rho(X_i^t)$
207      **if** $|X_{m_t}^t| \geq n$ **then** $T^t \leftarrow S \oplus_{\text{end}} X_{m_t}^t$ **else** $T^t \leftarrow \mathbf{2}S \oplus X_{m_t}^t 10^*$
208      **if** $T^t \in \text{Domain}(\rho)$ **then** *bad* $\leftarrow$ true
209      **if** $T^t \in \text{Domain}(\rho')$ **then** *bad* $\leftarrow$ true
210      $\rho'(T^t) =$ arbitrary
```

```
300  **for** $I \in \{0,1\}^*$ **do** $\rho(I) \xleftarrow{\$} \{0,1\}^*$ **od**, $\rho(\mathbf{1}) \leftarrow C$                Game S3
301  **for** $t \leftarrow 1$ **to** $q$ **do**
302      $S \leftarrow \rho(\mathbf{0})$
303      **for** $i \leftarrow 1$ **to** $m_t - 1$ **do** $S \leftarrow \mathbf{2}S \oplus \rho(X_i^t)$
304      **if** $|X_{m_t}^t| \geq n$ **then** $T^t \leftarrow S \oplus_{\text{end}} X_{m_t}^t$ **else** $T^t \leftarrow \mathbf{2}S \oplus X_{m_t}^t 10^*$
305  **if** $[\exists\, r, s, t, i]\ [\ (T^t = X_i^r)$ **or** $(T^t = \mathbf{0})$ **or** $(T^t = \mathbf{1})$ **or** $(T^t = T^s)\ ]$ **then** *bad* $\leftarrow$ true
```

```
400  **for** $I \in \{0,1\}^*$ **do** $\rho(I) \xleftarrow{\$} \{0,1\}^*$ **od**, $\rho(\mathbf{1}) \leftarrow C$                Game S4
401  **for** $t \leftarrow 1$ **to** $q$ **do**
402      **if** $|X_{m_t}^t| \geq n$
403      **then** $T^t \leftarrow \left[\mathbf{2}^{m_t-1}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_t-2}\rho(X_1^t) \oplus \mathbf{2}^{m_t-3}\rho(X_2^t) \oplus \cdots \oplus \mathbf{2}\rho(X_{m_t-2}^t) \oplus \rho(X_{m_t-1}^t)\right] \oplus_{\text{end}} X_{m_t}^t$
404      **else** $T^t \leftarrow \left[\mathbf{2}^{m_t}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_t-1}\rho(X_1^t) \oplus \mathbf{2}^{m_t-2}\rho(X_2^t) \oplus \cdots \oplus \mathbf{2}^2\rho(X_{m_t-2}^t) \oplus \mathbf{2}\rho(X_{m_t-1}^t)\right] \oplus X_{m_t}^t 10^*$
405  **if** $[\exists\, r, s, t, i]\ [\ (T^t = X_i^r)$ **or** $(T^t = \mathbf{0})$ **or** $(T^t = \mathbf{1})$ **or** $(T^t = T^s)\ ]$ **then** *bad* $\leftarrow$ true
```

Figure 11: Games used in the proof of security for S2V.

happens we give up in the analysis by setting *bad*. Since ρ and ρ' are grown by adding uniform random values and kept in sync, their joint effect is the same as choosing a single random function ρ for both purposes. We have that $\Pr[f \xleftarrow{\$} \text{Func}(\{0,1\}^*, n) :\ A^{f^*(\cdot)} \Rightarrow 1] = \Pr[A^{\text{S0}} \Rightarrow 1]$.

Game S1 is a faithful simulation of a random function from $\{0,1\}^{**}$ to $\{0,1\}^n$ (recall that the adversary may not repeat a query); $\Pr[R \xleftarrow{\$} \text{Func}(\{0,1\}^{**}, n) :\ A^{R(\cdot)} \Rightarrow 1] = \Pr[A^{\text{S1}} \Rightarrow 1]$. Furthermore, games S1 and S0 differ only by the sequels of statements that set the flag *bad*. So by the fundamental lemma of game-playing, the advantage we wish to bound is at most $\Pr[A^{\text{S1}}$ sets *bad* $]$.

To bound this assume for a moment that the adversary never asks the no-argument query ($m = 0$) and so all values returned to the adversary are returned at line 119. The n-bit strings, call them $Z_1, \ldots, Z_q$, that are returned to the adversary and placed into the range of ρ' have no impact on the running of the game—they are never even referred to—what matters is the domain of ρ', not its range. Thus one could let the adversary choose whatever return values it likes as $Z_1, \ldots, Z_q$ and it would not matter in the setting of *bad*. Now as for the no-argument query, we let the adversary choose the value $C = f_K(\mathbf{1})$ that is best for it, along with an optimal sequence of queries $X_1, \ldots, X_q$ (each query $X_t = (X_1^t, \ldots, X_{m_t}^t) \in \{0,1\}^{**}$) for the adversary to ask having total length of at most μ bits. Fixing all of these values, Game S1 has now been reduced to the non-interactive Game S2 that is specified in Figure 11.

Examining game S2, notice that points are added into the domain of ρ only at line 200, where ρ gets defined at **0** and **1**, and at line 205, when ρ gets defined at X_i^t. Points are added into the domain of ρ' only at line 210, when ρ' is defined at T^t. Thus *bad* gets set to `true` in exactly the following situations: at line 204, when an X_i^t value is equal to some T^s value for some $s < t$; at line 208, when a T^t value is equal to an X_i^r value for some $r \le t$ and $1 \le i < m_r$; at line 208, when a T^t value is equal to **0** or **1**; and at line 209, when a T^t value is equal to a T^s value for some $s < t$. Recalling that p is the assumed total number of vector components, we see that, all in all, there are a total of: $(p-q)q$ pairs (T^t, X_i^r) which, if equal, set *bad*; q pairs $(T^t, \mathbf{0})$ which, if equal, set *bad*; q pairs $(T^t, \mathbf{1})$ which, if equal, set *bad*; and $\binom{q}{2}$ pairs (T^t, T^s), for $s < t$, which, if equal, set *bad*. In a moment we will show that for each of these pairs the probability that the first and second components are equal is at most 2^{-n}. Given this, we can conclude that the probability that *bad* gets set is in game S2 at most $\left((p-q)q + 2q + \binom{q}{2}\right) \cdot 2^{-n}$. This value is at most pq for $q \ge 3$, the result we want.

Now, to show that each of the specified pairs collide with probability at most 2^{-n} we first rewrite game S2 as game S3, which makes its random choices up front and checks for domain collisions (ie., when *bad* begin set to `true` in game s2) at the end. Then, in game S4, we unroll the loop to more explicitly specify T^t. Now our job is to show that, for any valid r, s, t, i, each of the four equalities at line 405, namely, (Case 1) $T^t = X_i^r$ and (Case 2) $T^t = \mathbf{0}$ and (Case 3) $T^t = \mathbf{1}$ and (Case 4) $T^t = T^s$, the equality holds with probability at most 2^{-n}. The implicit quantification for a "valid" r, s, t, i at lines 305 and 405 is $r, s, t \in [1..q]$, $r \le t$, $s < t$, and $i \in [1..m_r - 1]$. Fix a valid r, s, t, i.

Case 1. First we show that $\Pr[T^t = X_i^r] \le 2^{-n}$. Keep clear that all of the X_i^r values are constants. There are two subcases to consider, depending on $|X_{m_t}^t|$.

Case 1.0. Assume first that $|X_{m_t}^t| \ge n$. In this case we are looking to bound

$$\Pr\left[\left(\mathbf{2}^{m_t-1}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_t-2}\rho(X_1^t) \oplus \mathbf{2}^{m_t-3}\rho(X_2^t) \oplus \cdots \oplus \mathbf{2}\rho(X_{m_t-2}^t) \oplus \rho(X_{m_t-1}^t)\right) \oplus_{\text{end}} X_{m_t}^t = X_i^r\right].$$

If $|X_i^r| \ne |X_{m_t}^t|$ then the above probability is zero and we are done. So assume $|X_i^r| = |X_{m_t}^t|$. If X_i^r and $X_{m_t}^t$ differ in some bit before their last n bits then the above probability is again zero (because of the behavior of $\oplus_{\text{end}}$) and we are done. Hence there is no loss of generality to assume that $|X_i^r| = |X_{m_t}^t| = n$; just strip away the leading, irrelevant prefix. The probability we wish to bound is then

$$\Pr\left[\mathbf{2}^{m_t-1}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_t-2}\rho(X_1^t) \oplus \mathbf{2}^{m_t-3}\rho(X_2^t) \oplus \cdots \oplus \mathbf{2}\rho(X_{m_t-2}^t) \oplus \rho(X_{m_t-1}^t) \oplus X_{m_t}^t = X_i^r\right].$$

In this formula, various pairs of the X_i^t values may coincide. Whenever this happens, combine the multipliers of the coinciding quantities by xoring them. This will never form the zero multiplier because each coefficient is of the form $\mathbf{2}^j$ where $j \in [0..n-1]$. Combine all terms where $X_i^t = \mathbf{1}$ and collect them into a single constant, together with the constant X_i^r, to make the single constant B. (Thus B contains all of the quantities in the formula that the adversary controlled.) Rename variables and coefficients to get an expression

$$\Pr[c_0\, \rho(\mathbf{0}) \oplus c_1\, \rho(C_1) \oplus \cdots \oplus c_u\, \rho(C_u) = B]$$

for some $u \ge 0$ and where $C_1, \ldots, C_u$ are distinct strings different from **1** and where each $c_i \ne 0$. As every $\rho(I)$ value is random with the exception of $\rho(\mathbf{1})$, the above probability is at most $\mathbf{2}^{-n}$.

Case 1.1. Assume instead that $|X^t_{m_t}| < n$. We are looking to bound

$$\Pr\left[\left(\mathbf{2}^{m_t}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_t-1}\rho(X^t_1) \oplus \mathbf{2}^{m_t-2}\rho(X^t_2) \oplus \cdots \oplus \mathbf{2}^2\rho(X^t_{m_t-2}) \oplus \mathbf{2}\rho(X^t_{m_t-1})\right) \oplus X^t_{m_t}10^* = X^r_i\right] .$$

Showing that this is at most 2^{-n} is done in a manner directly analogous to Case 1.0, so details are omitted.

Case 2 and *Case 3*. We immediately have that $\Pr[T^t = \mathbf{0}] \le 2^{-n}$ and $\Pr[T^t = \mathbf{1}] \le 2^{-n}$, since these are just specializations of Case 1.0 where $X^r_i = \mathbf{0}$ or $X^r_i = \mathbf{1}$.

Case 4, that $\Pr[T^t = T^s] \le 2^{-n}$. There are four subcases, depending on whether each of $X^t_{m_t}$ and $X^s_{m_s}$ are or are not longer than n bits.

Case 4.0. Suppose first that both $|X^t_{m_t}| \ge n$ and $|X^s_{m_s}| \ge n$. Then we must show that

$$\Pr\left[\left(\mathbf{2}^{m_t-1}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_t-2}\rho(X^t_1) \oplus \mathbf{2}^{m_t-3}\rho(X^t_2) \oplus \cdots \oplus \mathbf{2}\rho(X^t_{m_t-2}) \oplus \rho(X^t_{m_t-1}) \oplus_{\text{end}} X^t_{m_t}\right) = \\ \left(\mathbf{2}^{m_s-1}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_s-2}\rho(X^s_1) \oplus \mathbf{2}^{m_s-3}\rho(X^s_2) \oplus \cdots \oplus \mathbf{2}\rho(X^s_{m_s-2}) \oplus \rho(X^s_{m_s-1}) \oplus_{\text{end}} X^s_{m_s}\right)\right] \le 2^{-n} .$$

If $|X^t_{m_t}| \ne |X^s_{m_s}|$ then the above probability is zero and we are done. Likewise if $X^t_{m_t}$ and $X^s_{m_s}$ differ in bits before their final n bits. So we can assume that $|X^t_{m_t}| = |X^s_{m_s}| = n$ (again by stripping away the irrelevant prefix) and the xor-at-the-end becomes an ordinary xor. Letting $Z = X^t_{m_t} \oplus X^s_{m_s}$, we are thus aiming to bound the probability

$$\Pr\left[\mathbf{2}^{m_t-1}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_t-2}\rho(X^t_1) \oplus \mathbf{2}^{m_t-3}\rho(X^t_2) \oplus \cdots \oplus \mathbf{2}\rho(X_{m_t-2}) \oplus \rho(X^t_{m_t-1}) = \\ \mathbf{2}^{m_s-1}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_s-2}\rho(X^s_1) \oplus \mathbf{2}^{m_s-3}\rho(X^s_2) \oplus \cdots \oplus \mathbf{2}\rho(X_{m_s-2}) \oplus \rho(X^s_{m_s-1}) \oplus Z\right] .$$

Since the adversary may not repeat a query, $(X^t_1, \ldots, X^t_{m_t-2}, X^t_{m_{t-1}}) = (X^s_1, \ldots, X^s_{m_s-2}, X^s_{m_{s-1}})$ implies $Z \ne \mathbf{0}$ and so the probability above is zero in this case. Consequently, we may assume that $(X^t_1, \ldots, X^1_{m_t-2}) \ne (X^s_1, \ldots, X^s_{m_s-1})$. Collect all terms as before to get an expression

$$\Pr[c_0\, \rho(C_0) \oplus c_1\, \rho(C_1) \oplus \cdots \oplus c_u\, \rho(C_u) = B]$$

for some $u \ge 0$ and where $C_0, C_1, \ldots, C_u$ are distinct strings different from $\mathbf{1}$ (all $\rho(\mathbf{1})$ terms are included in B) and where each $c_i \ne 0$. The probability of this event is at most 2^{-n}.

Case 4.1. Suppose next that $|X^t_{m_t}| \ge n$ and $|X^s_{m_s}| < n$. Then we must show that

$$\Pr\left[\left(\mathbf{2}^{m_t-1}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_t-2}\rho(X^t_1) \oplus \mathbf{2}^{m_t-3}\rho(X^t_2) \oplus \cdots \oplus 2\rho(X^t_{m_t-2}) \oplus \rho(X^t_{m_t-1})\right) \oplus_{\text{end}} X^t_{m_t} = \\ \left(\mathbf{2}^{m_s}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_s-1}\rho(X^s_1) \oplus \mathbf{2}^{m_s-2}\rho(X^s_2) \oplus \cdots \oplus \mathbf{2}^2\rho(X^s_{m_s-2}) \oplus 2\rho(X^s_{m_s-1})\right) \oplus X^s_{m_s}10^*\right] \le 2^{-n} .$$

If $|X^t_{m_t}| \ne |X^s_{m_s}10^*| = n$ then the probability above is zero, so assume $|X^t_{m_t}| = n$ and the xor-at-then-end becomes an ordinary xor. We are now considering

$$\Pr\left[\left(\mathbf{2}^{m_t-1}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_t-2}\rho(X^t_1) \oplus \mathbf{2}^{m_t-3}\rho(X^t_2) \oplus \cdots \oplus 2\rho(X^t_{m_t-2}) \oplus \rho(X^t_{m_t-1})\right) \oplus X^t_{m_t} = \\ \left(\mathbf{2}^{m_s}\rho(\mathbf{0}) \oplus \mathbf{2}^{m_s-1}\rho(X^s_1) \oplus \mathbf{2}^{m_s-2}\rho(X^s_2) \oplus \cdots \oplus \mathbf{2}^2\rho(X^s_{m_s-2}) \oplus 2\rho(X^s_{m_s-1})\right) \oplus X^s_{m_s}10^*\right] \le 2^{-n} .$$

We must separately examine the subcases that $m_t \ne m_s + 1$ and $m_t = m_s + 1$. In the former subcase, we again gather together terms that coincide and write the probability as

$$\Pr[c_0\, \rho(\mathbf{0}) \oplus c_1\, \rho(C_1) \oplus \cdots \oplus c_u\, \rho(C_u) = B]$$

where $C_1, \ldots, C_u$ are distinct strings different from $\mathbf{1}$ (all $\rho(\mathbf{1})$ terms are included in B). We must argue that one of the $c_i \ne 0$; in particular, we will show that $c_0 \ne 0$. To see this, notice that the xor of coefficients that describes c_0 contains $\mathbf{2}^{m_s}$ and $\mathbf{2}^{m_t-1} \neq \mathbf{2}^{m_s}$, and that one of these is a term of greatest degree. The inequality holds because we have restricted both m_t and m_s to be less than n. Hence the string c_0 contains at least one nonzero bit, and it follows that

$$\Pr[c_0\, \rho(\mathbf{0}) = c_1\, \rho(C_1) \oplus \cdots \oplus c_u\, \rho(C_u) \oplus B] \leq 2^{-n} \; .$$

On the other hand, assume that $m_t = m_s + 1$. Then we have

$$\begin{aligned} \Pr \Big[\big(\mathbf{2}^{m_s} \rho(\mathbf{0}) \oplus \mathbf{2}^{m_s-1} \rho(X_1^t) \oplus \mathbf{2}^{m_2-2} \rho(X_2^t) \oplus \cdots \oplus 2\rho(X_{m_s-1}^t)\big) \oplus \mathbf{2}^0 \rho(X_{m_s}^t) \oplus X_{m_t}^t = \\ \big(\mathbf{2}^{m_s} \rho(\mathbf{0}) \oplus \mathbf{2}^{m_s-1} \rho(X_1^s) \oplus \mathbf{2}^{m_s-2} \rho(X_2^s) \oplus \cdots \oplus 2\rho(X_{m_s-1}^s)\big) \oplus X_{m_s}^s 10^* \Big]. \end{aligned}$$

Once more, gather together terms that coincide and write the probability as

$$\Pr[c_0\, \rho(C_0) \oplus c_1\, \rho(C_1) \oplus \cdots \oplus c_{u-1}\, \rho(C_{u-1}) \oplus c_u\, \rho(X_{m_s}^t) = B]$$

where $C_1, \ldots, C_{u-1}, X_{m_s}^t$ are distinct strings different from $\mathbf{1}$. In this case, notice that the xor of coefficients that describes c_u contains $\mathbf{2}^0$ (ie, $c_u\, \rho(X_{m_s}^t)$ contains an unshifted copy of $\rho(X_{m_2}^t)$. Since we have restricted both m_t and m_s to be less than n, there can be no wrap-around of the coefficients $\mathbf{2}^i$, and so the string c_u contains at least one nonzero bit. It follows that the probability in question is 2^{-n}.

Case 4.2. The case in which $|X_{m_s}^s| \geq n$ and $|X_{m_t}^t| < n$ is the same as Case 4.1 after a renaming of variables.
Case 4.3. The case in which $|X_{m_t}^t| < n$ and $|X_{m_s}^s| < n$ is just like Case 4.0 and is therefore omitted. This completes the proof. ▮

F Key Rap

Mihir Bellare has asked why *key wrap* needs that apparently superfluous *w*, inspiring this appendix.[1]

Yo! We'z gonna' take them keys
an' whatever you pleaze
We gonna' wrap 'em all up
looks like some ran'om gup
Make somethin' gnarly and funky
won't fool no half-wit junkie
So the game's like AE
but there's one major hitch
No coins can be pitched
there's no state to enrich
the IV's in a ditch
dead drunk on cheap wine

Now NIST and X9
and their friends at the fort
suggest that you stick it
in a six-layer torte
S/MIME has a scheme
there's even one more
So many ways
that it's hard to keep score
And maybe they work
and maybe they're fine
but I want some proofs
for spendin' my time

After wrappin' them keys
gonna' help out some losers
chronic IV abusers
don't read no directions
risk a deadly infection
If a rusty IV's drippin' into yo' veins
and ya never do manage
to get it exchanged
Then we got ya somethin'
and it comes at low cost
When you screw up again
not all 'ill be lost

[1] The contents of this appendix also appear in the *Journal of Craptology*, volume 3, November 2006.