Robust Authenticated-Encryption

AEZ and the Problem that it Solves

Viet Tung Hoang[1,2]   Ted Krovetz[3]   Phillip Rogaway[4]

[1] Dept. of Computer Science, University of Maryland, College Park, USA
[2] Dept. of Computer Science, Georgetown University, USA
[3] Dept. of Computer Science, California State University, Sacramento, USA
[4] Dept. of Computer Science, University of California, Davis, USA

March 31, 2017

Abstract. With a scheme for *robust* authenticated-encryption a user can select an arbitrary value $\lambda \geq 0$ and then encrypt a plaintext of any length into a ciphertext that's λ characters longer. The scheme must provide all the privacy and authenticity possible for the requested λ. We formalize and investigate this idea, and construct a well-optimized solution, AEZ, from the AES round function. Our scheme encrypts strings at almost the same rate as OCB-AES or CTR-AES (on Haswell, AEZ has a peak speed of about 0.7 cpb). To accomplish this we employ an approach we call *prove-then-prune*: prove security and then instantiate with a *scaled-down* primitive (e.g., reducing rounds for blockcipher calls).

Keywords: AEZ, authenticated encryption, CAESAR competition, misuse resistance, modes of operation, nonce reuse, provable security, prove-then-prune, robust AE.

Table of Contents

1   Introduction

We expose the low cost and high benefit of building authenticated-encryption (AE) schemes that achieve the unprecedentedly strong goal we call *robust* AE (henceforth RAE). We explain why RAE is desirable, define its syntax and security, and explore its guarantees. Then we construct an RAE scheme, AEZ, from AES4 and AES10 (four- and ten-round AES). AEZ's efficiency—nearly that of AES-based OCB [32] or CTR mode—flies in the face of a community's collective work [4, 11–13, 22–25, 35, 38–40, 52–54, 60] in which wide-block enciphering schemes—a special case of RAE—were always far more expensive than conventional blockciphers. Achieving this efficiency has entailed using a design paradigm, the *prove-then-prune* approach, with implications beyond AE.

CIPHERTEXT EXPANSION. One can motivate RAE from a syntactic point of view. Recall that in a nonce-based AE scheme, a plaintext M is mapped to a ciphertext $C = \mathcal{E}_K^{N,A}(M)$ under the control of a key K, nonce N, and associated data (AD) A. Typically the *ciphertext expansion* (or *stretch*) $\lambda = |C| - |M|$ is a constant or user-selectable parameter. For conventional AE, the stretch mustn't be too small, as customary definitions would break: a trivial adversary can get large advantage. This is because AE definitions "give up" when the first forgery occurs. The issue isn't *only* definitional: no prior AE scheme provides a desirable security guarantee when the ciphertext expansion is small.

Still, we know that meaningful security is possible even for zero-stretch: a strong pseudorandom permutation buys significant security, even from an AE point of view [5]. What is more, it would seem to be *useful* to allow small stretch, as, for example, short tags can save significant energy in resource-constrained environments (as discussed, e.g., by Struik [58]).

RAE takes a liberal approach towards ciphertext expansion, accommodating whatever stretch a user requests. This leads to schemes that deliver more than conventional AE even when the stretch is not small. Indeed we could have motivated RAE without considering small-λ, describing a desire to achieve nonce-reuse misuse-resistance [51], to automatically exploit novelty or redundancy in plaintexts [5], or to accommodate the release of unverified plaintexts [1, 21]. But our ideas are most easily understood by asking what it means, and what it takes, to do well for any stretch.

DEFINING RAE. So consider an AE scheme that expands a plaintext $M \in \{0,1\}^*$ by a user-selectable number of bits[5] $\tau \geq 0$. We ask: what's the best privacy and authenticity guarantee possible for some arbitrary, specified τ? Robust AE formalizes an answer.

Recall the definition of a *pseudorandom-injection* (PRI) [51]: for each nonce N and associated data A, for a fixed $\tau \geq 0$, the scheme's encryption algorithm should resemble a uniformly chosen injective function $\pi_{N,A,\tau}$ from binary strings to τ-bit longer ones. Decryption of an invalid ciphertext (one lacking a preimage under π) should return an indication of invalidity.

PRIs were introduced as an alternative characterization of nonce-reuse misuse-resistant AE (henceforth MRAE). But PRIs only approximate MRAE schemes with large stretch. We recast the PRI notion as prescriptive: the user selects $\tau \geq 0$ and then the scheme must look like a PRI for the chosen value. This is our basic definition for RAE.

RAE can be thought of as a bridge connecting blockciphers and AE. When $\tau = 0$ an RAE scheme is a kind of blockcipher—a tweakable blockcipher (TBC) [34] that operates on messages and tweaks of arbitrary length and is secure as strong pseudorandom permutation (PRP). The nonce and AD

---

[5] We'll later permit arbitrary alphabets. To avoid confusion, we use λ to measure ciphertext expansion in characters (bits, bytes, etc.) and τ to measure it in bits.

comprise the tweak. When $\tau \gtrsim 128$ an RAE scheme amounts to an MRAE scheme. An RAE scheme encompasses both objects, and everything in between.

In defining RAE we are actually a bit more generous than what was sketched above, allowing an RAE's decryption algorithm to return information about an invalid ciphertext beyond a single-valued indication of invalidity. The information just needs to be harmless. To formalize this the reference experiment uses a simulator S to provide responses to invalid decryption queries. It must do this without benefit of the family of random injections.

ENCIPHERING-BASED AE. We can achieve RAE with *enciphering-based AE*. The idea, rooted in folklore, was formalized by Bellare and Rogaway [5] and, in a different form, by Shrimpton and Terashima [56]. In its modern incarnation, enciphering-based AE works like this:

> Take the message you want to encrypt, augment it with τ bits of redundancy, and then encipher the resulting string by applying an arbitrary-input-length tweakable blockcipher. Tweak this using the nonce, AD, and an encoding of τ. On decryption, check for the presence of the anticipated redundancy and reject a ciphertext if it is not there.

We will prove that this method achieves RAE. In fact, we'll prove that this is so even if the decryption algorithm releases candidate plaintexts with incorrect redundancy.

AEZ. We construct a highly optimized RAE scheme, AEZ. We use the same name to refer to the arbitrary-input-length tweakable blockcipher from which it's built.[6] With the increasing ubiquity of hardware AES support, we choose to base AEZ on the AES round function.

How AEZ works depends on the length of the input; see Fig. 1. To encipher a plaintext of fewer than 32 bytes we use AEZ-tiny, a balanced-Feistel scheme with a round function based on AES4, a four-round version of AES. The construction builds on FFX [6, 17]. The more interesting case, AEZ-core, is used to encipher strings of 32 bytes or more. It builds on EME [22, 24] and OTR [36]. Look ahead to the top-left panel of Fig. 7. There are two enciphering layers, with consecutive pairs of blocks processed together using a two-round Feistel network. The round function for this is again based on AES4. The mask injected as the middle layer is determined, for each pair of consecutive blocks, using another AES4 call.

PERFORMANCE. AEZ-core is remarkably fast; as the description above implies, we need about five AES4 calls to encipher each consecutive pair of blocks, so ten AES rounds per block. Thus our performance approaches that of CTR-AES. An implementation of AEZ on Haswell using AES-NI has a peak speed of 0.72 cpb—about the same as OCB [32]. Look ahead to Fig. 8. Additionally, invalid strings can be rejected, and AD processed, in about 0.4 AES-equivalents per block, or 0.29 cpb peak (again on Haswell). Only the forward direction of AES is used, saving chip area in hardware realizations. The context size, about 128 bytes, is small, and key setup, about 1.2 AES-equivalents for a 128-bit key, is fast.

For a two-pass mode achieving MRAE, the cluster of performance characteristics described is unexpected. Part of the explanation as to how this is possible lies in the use of a design approach that benefits from both classical and provable-security design. Let us explain.

PROVE-THEN-PRUNE DESIGN. We designed AEZ using an approach we call *prove-then-prune*. It works like this:

---

[6] Since an RAE scheme trivially determines an arbitrary-input-length tweakable blockcipher (set $\tau = 0$) it makes sense to use a single name for both objects.
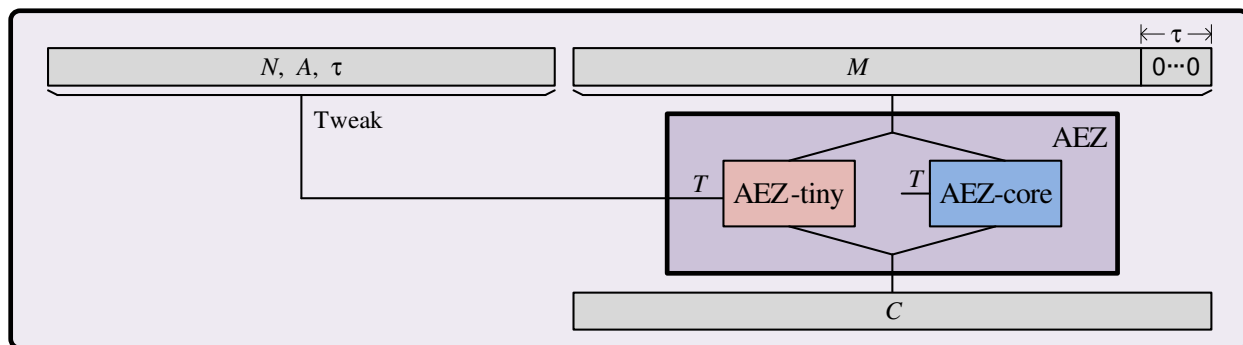
Fig. 1. High-level structure of AEZ. After appending to the message a block of τ zero bits we encipher it using a tweak T comprising the nonce N, associated data A, and stretch τ. Enciphering depends on the length of the plaintext: usually we use AEZ-core, but strings shorter than 32 bytes are enciphered by AEZ-tiny. Both depend on the underlying key K, which is not shown in the diagram above.

> To achieve some complex cryptographic goal, design a scheme in the provable-security tradition, choosing an underlying primitive and demonstrably achieving the goal when it's instantiated by an object achieving some standard assumption. Then, to improve speed, selectively instantiate some of the applications of the primitive using a *scaled-down* (e.g., reduced-round) construction. Use heuristic or cryptanalytic reasons to support the expectation that, despite scaling down, the scheme remains secure.

Specifically, AEZ is designed in terms of a tweakable blockcipher (TBC). If this TBC had been instantiated in the "usual" way, say using AES and the XE construction [34, 49], we would have a provably-sound design on message space $\{0,1\}^{\geq 128}$. The cost would be about 2.5 times the cost of AES. But to speed things up, we instantiate most TBC calls with an AES4-based construction. Heuristics reasons to suggest that security nonetheless remains. Our design was specifically chosen so as to make a scaled-down instantiation plausible.

The thesis underlying prove-then-prune approach is that it can be instrumental for devising highly efficient schemes for complex aims. We believe that *if* the instantiation is done judiciously, then the scaled-down scheme retains *some* assurance benefit. Still, it is important to emphasize the limitations of prove-then-prune. Naming an approach is not license to abuse it. The method is dangerous in the same sort of way that designing a confusion/diffusion primitive is: one has no guarantees for the object that will actually be used. Additionally, the set of people with provable-security competence is nearly disjoint from those with cryptanalytic competence. The authors think it essential that cryptanalysts study AEZ. This is all the more true because pruning was aggressive.

In some way, prove-then-prune is implicit in prior work: schemes like ALRED [15] typify a trend in which reduced-round AES is used in contexts where full AES would demonstrably do the job.

RAE BENEFITS. What do we hope to gain with RAE? Our definition and scheme are meant to achieve all of the following: (1) If (M, A) tuples are known *a priori* not to repeat, no nonce is needed to ensure semantic security. (2) If there's redundancy in plaintexts whose presence is verified on decryption, this augments authenticity. (3) Any authenticator-length can be selected, achieving best-possible authenticity for this amount of stretch. (4) Because of the last two properties, one can minimize length-expansion in many bandwidth-constrained applications. (5) If what's supposed to be a nonce should accidentally get repeated, the privacy loss is limited to revealing repetitions in (N, A, M) tuples, while authenticity is not damaged at all. (6) If a decrypting party leaks some

or all of a putative plaintext that was supposed to be squelched because of an authenticity-check failure, this won't compromise privacy or authenticity.

The authors believe that the properties enumerated would sometimes be worth a considerable computational price. Yet the overhead we pay is low: AEZ is almost as fast as OCB.

DISCUSSION. AEZ's name is meant to simultaneously suggest AE, AES, and EZ (easy), the last in the sense of ease of correct use. But the simplicity is for the user; we would not claim that the AEZ algorithm is simple.

Since McOE and COPA [2, 20], some recent AE schemes have been advertised as nonce-reuse misuse-resistant despite being online.[7] But online schemes are never misuse-resistant in the sense originally defined [51].[8] They never support automatic exploitation of novelty or verified redundancy [5] and are always vulnerable to a simple message-recovery attack [47]. We disagree with the presumption that two-pass AE schemes are routinely problematic; in fact, our work suggests that, on capable platforms, there isn't even a performance penalty. Finally, short messages routinely dominate networking applications, and we know of no application setting where it's important to limit latency to just a few bytes, the implicit expectation for proposed online schemes.

This paper upends some well-entrenched assumptions. Before, AE-quality was always measured with respect to an aspirational goal; now we're suggesting to employ an achievable one. Before, substantial ciphertext expansion was seen as necessary for any good AE; now we're allowing an arbitrary, user-supplied input. Before, AE schemes and blockciphers were considered fundamentally different species of primitives; now we're saying that, once the definitions are strengthened, they're pretty much the same thing. Before, one could either give a provable-security design or one that follows a more heuristic tradition; now we're doing the one and yet still finding need for the other.

AEZ is one of 57 CAESAR submissions [7]. It's distinguished by being the notionally strongest submission. We expect it to help clarify the potential cost and benefit of two-pass AE.

2   Prior AE Definitions

Fix an alphabet Σ. Typically Σ is $\{0, 1\}$ or $\{0, 1\}^8$, but other values, like $\Sigma = \{0, 1, \ldots, 9\}$, are fine. For $x \in \Sigma^*$ let $|x|$ denote its length. We write ε for the empty string and $x \twoheadleftarrow X$ for uniformly sampling from a distribution X. If X is a finite set, it has the uniform distribution.

SYNTAX. We formalize a nonce-based AE scheme as a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key space $\mathcal{K}$ is a set of strings with an associated distribution. The encryption algorithm $\mathcal{E}$ is deterministic and maps a four-tuple $(K, N, A, M) \in (\Sigma^*)^4$ to a value $C = \mathcal{E}_K^{N,A}(M)$ that is either a string in Σ^* or the distinguished symbol $\perp$. Later we will allow AD to be a vector of strings, $A \in (\Sigma^*)^*$. The distinction is insignificant insofar as we can always encode a vector of strings as a string. We require the existence of sets $\mathcal{N}$, $\mathcal{A}$ and $\mathcal{M}$ (the nonce space, AD space, and message space) such that $\mathcal{E}_K^{N,A}(M) \neq \perp$ iff $(K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$. The decryption algorithm $\mathcal{D}$ is deterministic and takes a four-tuple (K, N, A, C) to a value $\mathcal{D}_K^{N,A}(C) \in \Sigma^* \cup \{\perp\}$. The length of a string-valued $C = \mathcal{E}_K^{N,A}(M)$ is not allowed to depend on anything beyond $|N|$, $|A|$ and $|M|$. In fact, usually $\lambda = |C| - |M|$ is a constant, in which case we call the scheme λ-*expanding* and refer to λ as the

---

[7] By *online* we mean that the encryption algorithm can be realized in $O(1)$ memory and a single pass over M.
[8] If the first bit of ciphertext doesn't depend on the last bit of plaintext an adversary easily wins the MRAE game.

initialize $\quad$ $\boxed{\textbf{Ideal}_\Pi}$	initialize $\quad$ $\boxed{\textbf{Real}_\Pi}$	initialize $\quad$ $\boxed{\textbf{PRI}_\Pi}$
$K \twoheadleftarrow \mathcal{K}$	$K \twoheadleftarrow \mathcal{K}$	**for** $(N,A) \in \mathcal{N} \times \mathcal{A}$ **do** $\pi_{N,A} \twoheadleftarrow \mathrm{Inj}(\lambda)$
oracle Enc(N,A,M)	**oracle** Enc(N,A,M)	**oracle** Enc(N,A,M)
$C' \leftarrow \mathcal{E}_K(N,A,M)$	**return** $\mathcal{E}_K(N,A,M)$	**return** $\pi_{N,A}(M)$
$C \twoheadleftarrow \Sigma^{\lvert C' \rvert}$		
return C	**oracle** Dec(N,A,C)	**oracle** Dec(N,A,C)
	return $\mathcal{D}_K(N,A,C)$	**if** $\exists M \in \mathcal{M}$ s.t. $\pi_{N,A}(M) = C$ **then return** M
oracle Dec(N,A,C)		**return** $\perp$
return $\perp$		

initialize $\quad$ $\boxed{\textbf{REAL}_\Pi}$	initialize $\quad$ $\boxed{\textbf{RAE}_\Pi \text{ and } \textbf{RAE}_{\Pi,S}}$
$K \twoheadleftarrow \mathcal{K}$	**for** $(N,A,\lambda) \in \Sigma^* \times \Sigma^* \times \mathbb{N}$ **do** $\pi_{N,A,\lambda} \twoheadleftarrow \mathrm{Inj}(\lambda)$
	$\theta \leftarrow \varepsilon$
oracle Enc(N,A,λ,M)	
return $\mathcal{E}_K(N,A,\lambda,M)$	**oracle** Enc(N,A,λ,M)
	return $\pi_{N,A,\lambda}(M)$
oracle Dec(N,A,λ,C)	
return $\mathcal{D}_K(N,A,\lambda,C)$	**oracle** Dec(N,A,λ,C)
	if $\exists M \in \mathcal{M}$ s.t. $\pi_{N,A,\lambda}(M) = C$ **then return** M
	$M \leftarrow \perp$ $\qquad\qquad\qquad\qquad$ $\longleftarrow$ for $\textbf{RAE}_\Pi$
	$(M,\theta) \leftarrow S(N,A,\lambda,C,\theta)$ $\qquad$ $\longleftarrow$ for $\textbf{RAE}_{\Pi,S}$
	return M

Fig. 2. Games for defining security. The top three games are the usual ones for defining the AE and MRAE notions. The bottom two games are used to define RAE.

ciphertext expansion or *stretch*. We require that if $C = \mathcal{E}_K^{N,A}(M)$ is a string then $\mathcal{D}_K^{N,A}(C) = M$. Algorithm $\mathcal{D}$ *rejects* ciphertext C if $\mathcal{D}_K^{N,A}(C) = \perp$ and *accepts* it otherwise.

AE AND MRAE SECURITY. Both conventional-AE and MRAE security can be defined using a compact, all-in-one formulation [51]. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an AE-scheme. Consider an adversary $\mathcal{A}$ with access to an encryption oracle Enc and a decryption oracle Dec. We define the MRAE security of $\mathcal{A}$ as $\mathbf{Adv}_\Pi^{\mathrm{mrae}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{Real}_\Pi} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{Ideal}_\Pi} \Rightarrow 1]$, the difference in the probability that $\mathcal{A}$ outputs 1 when run in the **Real** and **Ideal** games of Fig. 2. Both begin by selecting $K \twoheadleftarrow \mathcal{K}$. Game **Real** answers encryption queries (N,A,M) with $\mathcal{E}_K^{N,A}(M)$ and decryption queries (N,A,C) with $\mathcal{D}_K^{N,A}(C)$. Game **Ideal** answers Dec(N,A,C) queries with $\perp$ and Enc(N,A,M) queries with $|C|$ uniformly chosen characters, where $C \leftarrow \mathcal{E}_K^{N,A}(M)$. For games **Real** and **Ideal**, adversaries may not repeat an Enc or Dec query, ask an Enc query $(N,A,M) \notin \mathcal{N} \times \mathcal{A} \times \mathcal{M}$, ask a Dec query $(N,A,C) \notin \mathcal{N} \times \mathcal{A} \times \Sigma^*$, or ask a Dec query (N,A,C) after an Enc query of (N,A,M) returned C.

The above definition captures MRAE security because repeated nonces were allowed and were properly serviced. For the conventional AE notion, $\mathbf{Adv}_\Pi^{\mathrm{ae}}(\mathcal{A})$, modify **Real** and **Ideal** by having an Enc(N,A,M) query following an earlier Enc(N,A',M') query return $\perp$. This has the same effect as prohibiting repeated N-values to the Enc oracle.

PRI SECURITY. We define security in the sense of a *pseudorandom-injection* (PRI) [51]. Fix a λ-expanding AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$; for now, λ is a constant associated to a (well-behaved) AE scheme. Let $\mathbf{Adv}_\Pi^{\mathrm{pri}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{Real}_\Pi} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{PRI}_\Pi} \Rightarrow 1]$ with the oracles again defined in Fig. 2. There $\mathrm{Inj}(\lambda)$ denotes the set of all one-to-one functions from Σ^* to Σ^* that increase the length of their inputs by λ characters. The same query restrictions apply as before.

Besides defining PRI security, Rogaway and Shrimpton showed that, for large ciphertext expansion λ, the notion essentially coincides with MRAE security [51]. Below we clarify the role of the ciphertext expansion by giving a sharper extended version of their result. To state our bound, define the *misuse count* as follows. Initially, set $r = 0$. Then, for each encryption query $\mathsf{Enc}(N, A, M)$, if there was a prior query (N, A, M') such that $|M'| = |M|$, increment r by 1. The final value of r is the misuse count. Below we show that good PRI security implies good MRAE security as long as q is small compared to $|\Sigma|^\lambda$ and r is small compared to $|\Sigma|^{(\lambda+m_{\min})/2}$ (with all variables defined below). The proof is in Appendix B.1.

Theorem 1. $|\mathbf{Adv}_\Pi^{\mathrm{pri}}(\mathcal{A}) - \mathbf{Adv}_\Pi^{\mathrm{mrae}}(\mathcal{A})| \leq 2q/|\Sigma|^\lambda + (r^2 + r)/|\Sigma|^{\lambda+m_{\min}+1}$ for any λ-expanding AE scheme Π and adversary $\mathcal{A}$, where r is the misuse count of $\mathcal{A}$'s queries, q is the number of queries it asks, and $m_{\min}$ is the length of the shortest string in the message space.

In short, the PRI definition captures best-*possible* security of a λ-expanding AE scheme, while the MRAE formulation captures an unreachable ideal. The gap between the realizable and the ideal is formalized by Theorem 1. It is small if the ciphertext expansion is large, and it is large if the ciphertext expansion is small. The latter is so because any *actual* encryption algorithm must map distinct (N, A, M) and (N, A, M') to distinct ciphertexts, whence real encryption *can't* return uniformly random characters. Similarly, for any infinite message space, *some* unqueried ciphertexts *must* be valid, whence a decryption oracle that *always* returns an indication of invalidity is hoping for too much. Building on the PRI notion, we will now look towards an even more precise way to capture best-possible AE security.

3   RAE Security

SYNTAX. The principle difference between a PRI and an RAE scheme is that, for the latter, the ciphertext expansion λ is no longer a property of a scheme: it's an arbitrary *input* from the user. All values $\lambda \in \mathbb{N}$ should be allowed.[9] Corresponding to this change, we'll write $\mathcal{E}_K^{N,A,\lambda}(M)$ and $\mathcal{D}_K^{N,A,\lambda}(C)$. The difference may look small, but its consequences are not.

Fix an alphabet Σ. Our formal definition again has an RAE scheme being a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, but with the signature of $\mathcal{E}$ and $\mathcal{D}$ updated. The encryption algorithm $\mathcal{E}$ is deterministic and maps a five-tuple $(K, N, A, \lambda, M) \in (\Sigma^*)^3 \times \mathbb{N} \times \Sigma^*$ to a string $C = \mathcal{E}_K^{N,A,\lambda}(M)$ of length $|M| + \lambda$. For maximal utility when realized, we are not permitting a return value of $\bot$: an RAE scheme must be able to encrypt any M using any N, A, and λ. The decryption algorithm $\mathcal{D}$ is deterministic and takes a five-tuple (K, N, A, λ, C) to a value $\mathcal{D}_K^{N,A,\lambda}(C) \in \Sigma^* \cup \{\bot\}$. We require that $\mathcal{D}_K^{N,A,\lambda}(\mathcal{E}_K^{N,A,\lambda}(M)) = M$ for all K, N, A, λ, M. If there's no M such that $C = \mathcal{E}_K^{N,A,\lambda}(M)$ then $\mathcal{D}_K^{N,A,\lambda}(C) = \bot$. Later in this section we will relax this requirement as a way to model the possibility of decryption algorithms that reveal information beyond an indication of invalidity.

RAE SECURITY. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an RAE scheme over alphabet Σ. Its security is defined using the games $\mathbf{REAL}_\Pi$ and $\mathbf{RAE}_\Pi$ at the bottom of Fig. 2. (For the moment, ignore $\mathbf{RAE}_{\Pi,S}$.) The adversary $\mathcal{A}$ has two oracles, an encryption oracle Enc and a decryption oracle Dec. For game $\mathbf{REAL}$, these are realized by the actual encryption and decryption algorithms, which now take in the argument λ. For game $\mathbf{RAE}_\Pi$ we behave according to the family of random injections

---

[9] It might be OK to set some reasonable upperbound $\lambda \leq \lambda_{\max}$, but there shouldn't be a nonzero lowerbound.

$\pi_{N,A,\lambda}$ chosen at the beginning of the game, responding to each encryption query (N, A, λ, M) with $C = \pi_{N,A,\lambda}(M)$ and responding to each decryption query (N, A, λ, C) with $\pi_{N,A,\lambda}^{-1}(C)$, if that inverse exists, and $\perp$ if it does not. We let $\mathbf{Adv}_{\Pi}^{\mathrm{rae}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{REAL}_\Pi} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{RAE}_\Pi} \Rightarrow 1]$. There are no restrictions on the kinds of queries the adversary may make.

To gain some appreciation for the RAE definition, consider an adversary that asks to encrypt a message M using a single byte of stretch. Such a scheme would not be considered secure in the MRAE setting, as forging with probability $1/256$ is easy. But under the RAE viewpoint, that isn't a defect per se, as the user who requests one-byte expansion would *expect* $1/256$ of all ciphertexts to have some preimage. If a user should try to decrypt such a ciphertext C using the same K, N, A but $\lambda = 0$, a plaintext *will* emerge, never an indication of invalidity, but that plaintext should be unrelated to the originally encrypted one.

DECRYPTION-CALL LEAKAGE. An AE scheme will fail to approximate the $\mathbf{RAE}_\Pi$ abstraction if its decryption algorithm, when presented an invalid ciphertext, routinely returns anything beyond an indication of invalidity. We now explain how to relax this expectation so that it's OK to return additional material as long as it is known to be useless.

We said earlier that, for an RAE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and any N, A, λ, C, if there's no M such that $C = \mathcal{E}_K^{N,A,\lambda}(M)$ then we expect $\mathcal{D}_K^{N,A,\lambda}(C)$ to return $\perp$. Let us relax this requirement so that $\mathcal{D}_K^{N,A,\lambda}(C)$ may instead return a string, as long as its length is not $|C| - \lambda$. Any such string is trivially recognized as invalid, so, in effect, we are having $\mathcal{D}$ return both $\perp$ and an arbitrary piece of side information Y. We are not suggesting that the "real" decryption algorithm should return anything other than $\perp$ when presented an invalid ciphertext; instead, we are effectively overloading $\mathcal{D}$ by folding into it a "leakage function" that captures which a decryption algorithm's realization may leak about a presented ciphertext.

Using this generalized syntax, we define a game $\mathbf{RAE}_{\Pi,S}$ parameterized by a probabilistic algorithm S, the *simulator*. Again see Fig. 2. Simulator S is called upon to produce imitation ciphertexts when there's no preimage under $\pi_{N,A,\lambda}$. To accomplish this task S is provided nothing beyond the current oracle query and any saved state θ it wants to maintain. An RAE scheme is judged secure if there's a simulator S—preferably an efficient one—such that $(\mathcal{E}, \mathcal{D})$ is indistinguishable from the pair of oracles defined in $\mathbf{RAE}_{\Pi,S}$. We refine the RAE advantage by asserting that $\mathbf{Adv}_{\Pi,S}^{\mathrm{rae}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{REAL}_\Pi} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{RAE}_{\Pi,S}} \Rightarrow 1]$. The "basic" RAE definition corresponds to the case where simulator S ignores its input and returns $(\perp, \varepsilon)$.

The RAE definition effectively captures that, while it may be "nice" for decryption to reveal nothing but $\perp$ on presentation of an invalid ciphertext, there are plenty of other things we could return without damaging privacy or authenticity. In fact, it really doesn't matter *what* is returned just so long as it's recognizably invalid and doesn't depend on the key.

ILLUSTRATION. Fig. 3 illustrates two possibilities for how an RAE scheme might encrypt 2-bit strings with 2-bit ciphertext expansion ($\lambda = 2$). The key K, nonce N, and AD A are all fixed. For encryption, the four possible plaintexts are bijectively paired with four of the 16 possible ciphertexts. For decryption we show two possibilities. On the left is a *conventional* decryption algorithm: the 12 ciphertexts without a preimage decrypt to an indication of invalidity. One expects the simulator to always return $(\perp, \varepsilon)$. On the right is a *sloppy* decryption algorithm. The 12 ciphertexts with no preimage decrypt to 12 distinct strings, all recognizably invalid, all of the form $abcd \in \{0,1\}^4$ with $cd \neq 00$. Here the simulator S might sample without replacement from the named set of size 12.
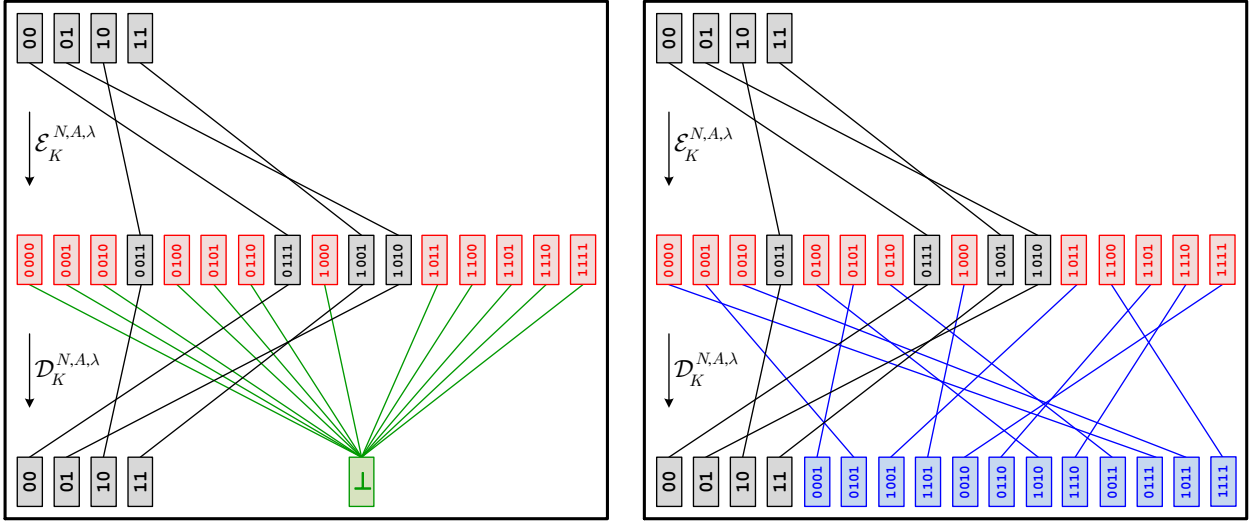
Fig. 3. Illustrating RAE. Two ways an RAE scheme might encrypt and decrypt a 2-bit string with 2-bit stretch.

DISCUSSION. The reader may have noticed that there is no distinction in the RAE security definition between the nonce N and associated data (AD) A. For this reason, either could be dropped—say the nonce—leaving us a signature $\mathcal{E}_K^{A,\lambda}(M)$ and $\mathcal{D}_K^{A,\lambda}(C)$. There's an especially good argument for doing this when the AD A is vector-valued: the user is already free to use one of its components as a nonce. Still, for greater uniformity in treatment across AE notions, and to encourage users to provide a nonce, we have retained both N and A.

We gave our definition of RAE into two stages only for pedagogical purposes: this paper offers only one definition for RAE. The simulator S may be trivial or not; that is the only distinction.

Andreeva *et. al* [1] recently provided several security definitions also meant to capture the requirement that a decryption algorithm releases only harmless information when presented an invalid ciphertext and a repeated nonce. Our own work is radically different from theirs insofar as we provide a single definition, RAE, that rolls into it this, among many, considerations.

4   Verified Redundancy Enhances Authenticity

If a plaintext contains redundancy, one naively expects that verifying its presence upon decryption should enhance the authenticity guarantee provided. For the case of enciphering-based encryption, which provides no authenticity guarantee on its own, this has been formally supported [5, 51]. But even in this case the existing results are with respect to conventional notions of AE, and such notions are too blunt to capture what one expects from verified redundancy. This is because the notions "give up" as soon as a single ciphertext forgery is made.

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be RAE scheme and let $v\colon \Sigma^* \to \{0,1\}$ be a function for indicating the "valid" strings: it determines $\mathcal{M}_v \subseteq \Sigma^*$ by $\mathcal{M}_v = \{M \in \Sigma^* : v(M) = 1\}$. Let $\Pi_v = (\mathcal{K}, \mathcal{E}, \tilde{\mathcal{D}})$ be the AE scheme built from Π that declares messages invalid if v says so: $\tilde{\mathcal{D}}_K^{N,A,\lambda}(C) = M$ if $|M| = |C| - \lambda$ and $v(M) = 1$, or if $|M| \neq |C| - \lambda$, where $M = \mathcal{D}_K^{N,A,\lambda}(C)$, while $\tilde{\mathcal{D}}_K^{N,A,\lambda}(C) = \mathbf{0} \,\|\, M$ otherwise, with $\mathbf{0}$ a canonical point in Σ. Let $d_v = \max_{\ell \in \mathbb{N}} \left\{ (|\mathcal{M}_v \cap \Sigma^\ell|)/|\Sigma|^\ell \right\}$ be the *density* of $\mathcal{M}_v$.

Suppose, for example, that $\Sigma = \{0,1\}$ and $d_v = 1/256$: there's a byte worth of redundancy in the message space. We'd like to be able to make statements about the authenticity of Π_v such as: the chance that an adversary can forge 10 successive, distinct ciphertexts is negligibly more than 2^{-80}. Conventional AE definitions don't let one say such a thing; they stop at the bound $q/|\Sigma|^\lambda$ where q is the number of queries and λ is the ciphertext expansion (assumed here to be a constant). One would like to obtain a much sharper bound via d_v and λ—in our example, the forgery probability should be about about $q(d_v/|\Sigma|^\lambda)^{10}$. This way, even if, say, $\lambda = 0$ and $d_v = 1/2$, we are *still* able to make strong statements about the security of Π_v. Intuitively, for an RAE scheme Π, the scheme Π_v should have about $(\lambda_{\min} + \log(1/d_v))\log(|\Sigma|)$ bits of authenticity, where $\lambda_{\min}$ is the minimum ciphertext expansion of any query—even after multiple successful forgeries and even in the presence of decryption leakage, future forgeries still remain just as hard.

To capture the intuition above, in Theorem 2 we show that Π_v itself is RAE-secure. The proof is in Appendix B.2. Consequently, in game **RAE**, for any query (N, A, λ, C) with $|C| = \ell + \lambda$ to Dec, the chance that this query is a successful forgery is about $|\mathcal{M}_v \cap \Sigma^\ell|/|\Sigma|^{\ell+\lambda} \leq d_v/|\Sigma|^\lambda$, despite any decryption leakage and past successful forgeries.

Theorem 2. Let Π and Π_v be defined as above. There is an explicitly given reduction $\mathcal{R}$ with the following property. For any simulator S and any adversary $\mathcal{A}$, there is a simulator S' such that the adversary $\mathcal{B} = \mathcal{R}(\mathcal{A})$ satisfies $\mathbf{Adv}^{\mathrm{rae}}_{\Pi,S}(\mathcal{B}) = \mathbf{Adv}^{\mathrm{rae}}_{\Pi_v,S'}(\mathcal{A})$. Adversary $\mathcal{B}$ makes the same queries as $\mathcal{A}$ and has essentially the same running time.

Note that for good RAE security, we want the simulator S to be efficient. This is important for privacy, but when the concern is authenticity, it's less of an issue: a computationally-unbounded simulator may give the adversary some information that it can't compute itself, but as long as the adversary can't forge, whatever the adversary learns from the simulator is irrelevant for authenticity. Still, in the proof of Theorem 2, for each query (N, A, λ, C), the simulator S' either runs S or samples from $\Sigma^\ell \cap \mathcal{M}_v$, where $\ell = |C| - \lambda$. For functions v that arise from real-world usage, sampling from $\Sigma^\ell \cap \mathcal{M}_v$ is likely to be simple and efficient, whence S' will be about as fast as S itself.

5   Robust AE from an Arbitrary-Input-Length TBC

We now show how to make an AE scheme that achieves RAE security. We begin with some basic definitions. Let $\mathcal{M} \subseteq \Sigma^*$ and $\mathcal{T}$ be sets. A *blockcipher* $\widetilde{\mathbb{E}} \colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ is a mapping such that $\widetilde{\mathbb{E}}^T_K(\cdot) = \widetilde{\mathbb{E}}(K, T, \cdot)$ is a length-preserving permutation on $\mathcal{M}$ for any K, T. Thus $|\widetilde{\mathbb{E}}^T_K(X)| = |X|$ and there's a unique $\widetilde{\mathbb{D}} \colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^* \to \mathcal{M} \cup \{\bot\}$ such that $\widetilde{\mathbb{E}}^T_K(M) = C$ implies $\widetilde{\mathbb{D}}^T_K(C) = M$ and $\widetilde{\mathbb{D}}^T_K(C) = \bot$ when there's no M such that $\widetilde{\mathbb{E}}^T_K(M) = C$. We call $\mathcal{T}$ the *tweak space* of $\widetilde{\mathbb{E}}$. When $|\mathcal{T}| = 1$ we make the tweak implicit, writing $\mathbb{E} \colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$, now with inverse $\mathbb{D}$. We define $\mathrm{Perm}(\mathcal{M})$ as the set of all length-preserving permutations on $\mathcal{M}$, and $\mathrm{Perm}(\mathcal{T}, \mathcal{M})$ the set of all mappings $\widetilde{\pi} \colon \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\widetilde{\pi}(T, \cdot) \in \mathrm{Perm}(\mathcal{M})$ for all $T \in \mathcal{T}$. We usually use *encipher* instead of *encrypt* when speaking of applying a blockcipher, and similarly for *decipher* and *decrypt*.

An *arbitrary-input-length* blockcipher is a blockcipher with message space $\mathcal{M} = \Sigma^*$. To be maximally useful, we will want a rich tweak space as well. These are versatile objects. A bit less general, a *wide-block* blockcipher has message space $\Sigma^{\geq n}$ for some fixed n. Again one prefers a rich tweak space. A *conventional* blockcipher has message space $\{0,1\}^n$ for some fixed n.

The strong, tweakable, PRP advantage of an adversary A attacking a blockcipher $\widetilde{\mathbb{E}}$ is defined as $\mathbf{Adv}^{\pm \widetilde{\mathrm{prp}}}_{\widetilde{\mathbb{E}}}(\mathcal{A}) = \Pr[K \twoheadleftarrow \mathcal{K} : \mathcal{A}^{\widetilde{\mathbb{E}}_K(\cdot,\cdot), \widetilde{\mathbb{D}}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\widetilde{\pi} \twoheadleftarrow \mathrm{Perm}(\mathcal{T}, \mathcal{M}) : \mathcal{A}^{\widetilde{\pi}(\cdot,\cdot), \widetilde{\pi}^{-1}(\cdot,\cdot)} \Rightarrow 1]$. We'll

write $\mathbf{Adv}_{\mathbb{E}}^{\pm\mathrm{prp}}(\mathcal{A}) = \Pr[K \leftarrow \mathcal{K}: \mathcal{A}^{\mathbb{E}_K(\cdot),\mathbb{D}_K(\cdot)} \Rightarrow 1] - \Pr[\pi \leftarrow \mathrm{Perm}(\mathcal{M}): \mathcal{A}^{\pi(\cdot),\pi^{-1}(\cdot)} \Rightarrow 1]$ if there's no tweak. If we prohibit the adversary $\mathcal{A}$ from querying the second oracle we drop the word "strong" and write $\mathbf{Adv}_{\mathbb{E}}^{\overline{\mathrm{prp}}}(\mathcal{A})$ and $\mathbf{Adv}_{\mathbb{E}}^{\mathrm{prp}}(\mathcal{A})$ respectively.

ENCODE-THEN-ENCIPHER. Fix Σ. Let $\widetilde{\mathbb{E}}: \mathcal{K} \times \mathcal{T} \times \Sigma^* \to \Sigma^*$ be an arbitrary-input-length tweakable blockcipher with tweak space $\mathcal{T} = \Sigma^* \times \Sigma^* \times \mathbb{N}$. Let $\widetilde{\mathbb{D}}$ be its inverse. Let Encode: $\Sigma^* \times \mathbb{N} \to \Sigma^*$ be an injective function satisfying $|\mathrm{Encode}(M,\lambda)| = |M| + \lambda$. We write the second argument to Encode as a subscript, $\mathrm{Encode}_\lambda(M) \in \Sigma^{|M|+\lambda}$. An example encoding function is $\mathrm{Encode}_\lambda(M) = M \parallel \mathbf{0}^\lambda$.

For any encoding function Encode there's a corresponding Decode: $\Sigma^* \times \mathbb{N} \to \Sigma^* \cup \{\bot\}$ such that $\mathrm{Decode}_\lambda(X) = M$ if there's an M satisfying $\mathrm{Encode}_\lambda(M) = X$, while $\mathrm{Decode}_\lambda(X) = \bot$ if there's no such M. We expect Encode and Decode to be trivially computable, as in the example.

From $\widetilde{\mathbb{E}}: \mathcal{K} \times \mathcal{T} \times \Sigma^* \to \Sigma^*$ and Encode we define the encode-then-encipher construction as the RAE scheme $\mathrm{EtE}[\mathrm{Encode}, \widetilde{\mathbb{E}}] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{E}_K^{N,A,\lambda}(M) = \widetilde{\mathbb{E}}_K^{(N,A,\lambda)}(\mathrm{Encode}_\lambda(M))$,
- $\mathcal{D}_K^{N,A,\lambda}(C) = M$ if $\widetilde{\mathbb{D}}_K^{(N,A,\lambda)}(C) = X$ and M satisfies $\mathrm{Encode}_\lambda(M) = X$,
- $\mathcal{D}_K^{N,A,\lambda}(C) = X$ if $\widetilde{\mathbb{D}}_K^{(N,A,\lambda)}(C) = X$ and no M satisfies $\mathrm{Encode}_\lambda(M) = X$.

We stress that decryption does not simply return $\bot$ when called on an invalid (N, A, λ, C), as is conventionally done; instead, we define decryption to "leak" the entire improperly encoded string X. Nonetheless, Theorem 3 shows that $\mathrm{EtE}[\mathrm{Encode}, \widetilde{\mathbb{E}}]$ is RAE-secure when $\widetilde{\mathbb{E}}$ is secure as a strong, tweakable PRP. Its proof appears in Appendix B.3.

Theorem 3 (EtE is RAE-secure). Let Encode and $\widetilde{\mathbb{E}}: \mathcal{K} \times \mathcal{T} \times \Sigma^* \to \Sigma^*$ be defined as above. Then there's an explicitly given reduction $\mathcal{R}$ and an efficient simulator S with the following property. For any adversary $\mathcal{A}$, the adversary $\mathcal{B} = \mathcal{R}(\mathcal{A})$ satisfies $\mathbf{Adv}_{\mathrm{EtE}[\mathrm{Encode},\widetilde{\mathbb{E}}],S}^{\mathrm{rae}}(\mathcal{A}) \leq \mathbf{Adv}_{\widetilde{\mathbb{E}}}^{\pm\widetilde{\mathrm{prp}}}(\mathcal{B})$. It makes at most q queries whose total length is at most that of $\mathcal{A}$'s queries plus $q\lambda_{\max}$, where q is the number of $\mathcal{A}$'s queries and $\lambda_{\max}$ is the largest stretch among them. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$, plus the time associated to computations of Encode and Decode.

6   Wide-Block Enciphering: AEZ-core

Let $n \geq 1$ be an integer and let $\{0,1\}^{\geq 2n} = \{x \in \{0,1\}^*: |x| \geq 2n\}$. Define the *block length* of a string x as $\lceil |x|/n \rceil$. We show how to build a strong PRP on $\{0,1\}^{\geq 2n}$ from a TBC on $\{0,1\}^n$. We'll use about 2.5 TBC calls per n-bit block. Later we'll instantiate the TBC using mostly AES4, employing the prove-then-prune paradigm to selectively scale-down. This will reduce the amortized cost to about one AES call per block. Also see Appendix C for how to tweak a wide-block blockcipher.

We begin by recalling the definition of a pseudorandom function (PRF) $f: \mathcal{K} \times \mathcal{M} \to \{0,1\}^n$. For an adversary $\mathcal{A}$ attacking f, its PRF advantage is $\mathbf{Adv}_f^{\mathrm{prf}}(\mathcal{A}) = \Pr[K \leftarrow \mathcal{K}: \mathcal{A}^{f_K(\cdot)} \Rightarrow 1] - \Pr[\rho \leftarrow \mathrm{Func}(\mathcal{M}, n): \mathcal{A}^{\rho(\cdot)} \Rightarrow 1]$ where $\mathrm{Func}(\mathcal{M}, n)$ is the set of all functions from $\mathcal{M}$ to $\{0,1\}^n$.

AEZ-CORE. Let $\mathcal{T} = \{\mathsf{a}, \mathsf{u}, \mathsf{uu}, \mathsf{v}, \mathsf{vv}, \mathsf{x}, \mathsf{xx}, \mathsf{y}, \mathsf{yy}\} \cup (\{\mathsf{a}, \mathsf{aa}\} \times \mathbb{N})$ be the tweak space. Suppose we have a PRF $f: \mathcal{K} \times (\mathcal{T} \times \{0,1\}^n) \to \{0,1\}^n$. One can instantiate this with a TBC $\widetilde{E}$ on $\{0,1\}^n$ by setting $f_K(K, (T, X)) = \widetilde{E}_K^T(X)$. Consider the wide-block blockcipher AEZ-core$[f]$ defined and illustrated in Fig. 6. It loosely follows EME/EME2 [22, 24, 29], but avoids all doubling operations and only
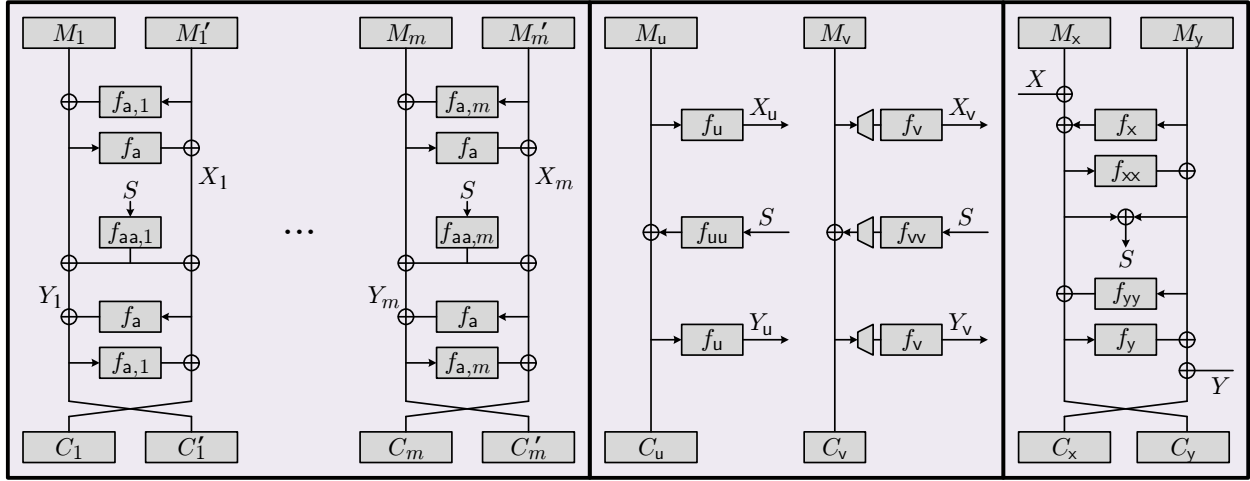
```
10    algorithm AEZ-core(K, M)                                                    // AEZ-core
11    M_1 M_1' ··· M_m M_m' M_uv M_x M_y ← M
12        where |M_1| = ··· = |M_m'| = |M_x| = |M_y| = n and |M_uv| < 2n
13    d ← |M_uv|;  if d < n then M_u ← M_uv; M_v ← ε
14    else M_u ← M_uv[1..n]; M_v ← M_uv[n + 1..|M_uv|] fi
15    for i ← 1 to m do W_i ← M_i ⊕ f_{a,i}(M_i');   X_i ← M_i' ⊕ f_a(W_i) od
16    if d = 0 then X ← X_1 ⊕ ··· ⊕ X_m ⊕ 0
17    else if d < n then X ← X_1 ⊕ ··· ⊕ X_m ⊕ f_u(M_u 10*)
18    else X ← X_1 ⊕ ··· ⊕ X_m ⊕ f_u(M_u) ⊕ f_v(M_v 10*) fi
19    S_x ← M_x ⊕ X ⊕ f_x(M_y);   S_y ← M_y ⊕ f_{xx}(S_x);   S ← S_x ⊕ S_y
20    for i ← 1 to m do
21        S' ← f_{aa,i}(S); Y_i ← W_i ⊕ S'; Z_i ← X_i ⊕ S'
22        C_i' ← Y_i ⊕ f_a(Z_i); C_i ← Z_i ⊕ f_{a,i}(C_i') od
23    if d = 0 then C_u ← C_v ← ε;  Y ← Y_1 ⊕ ··· ⊕ Y_m ⊕ 0
24    else if d < n then C_u ← M_u ⊕ f_{uu}(S); C_v ← ε; Y ← Y_1 ⊕ ··· ⊕ Y_m ⊕ f_u(C_u 10*)
25    else C_u ← M_u ⊕ f_u(S);   C_v ← M_v ⊕ f_{vv}(S)
26        Y ← Y_1 ⊕ ··· ⊕ Y_m ⊕ f_u(C_u) ⊕ f_v(C_v 10*) fi
27    C_y ← S_x ⊕ f_{yy}(S_y);   C_x ← S_y ⊕ Y ⊕ f_y(C_y)
28    return C_1 C_1' ··· C_m C_m' C_u C_v C_x C_y
```

Fig. 4. The AEZ-core[f] construction. The method builds a strong-PRP on $\{0,1\}^{\geq 2n}$ from an n-bit-output PRF f that operates on its subscript and argument. Its key K is implicit. The PRF can be realized by a TBC.

uses the forward direction of the underlying TBC. AEZ-core[f] operates on $\mathcal{M} = \{0,1\}^{\geq 2n}$ and itself takes in no tweak. Theorem 4 shows that it's a strong PRP. The proof is in Appendix B.4.

Theorem 4. Let $n \geq 1$ be an integer and let $\mathcal{T}$ and f be as above. There's an explicitly given reduction $\mathcal{R}$ with the following property. For any adversary $\mathcal{A}$, adversary $\mathcal{B} = \mathcal{R}(\mathcal{A})$ satisfies $\mathbf{Adv}^{\pm\mathrm{prp}}_{\mathrm{AEZ\text{-}core}[f]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prf}}_f(\mathcal{B}) + 2\sigma^2/2^n$ where σ is the total block length of $\mathcal{A}$'s queries. Adversary $\mathcal{B}$ uses the same running time as $\mathcal{A}$, and makes at most 2.5σ queries.

DISCUSSION.   AEZ-core and its inverse are almost the same: the only change needed is to take the rightmost column of tweaks in reverse order. Given that one must have *some* asymmetry in

an RAE scheme—an involution is certainly RAE-insecure—this is about as symmetric a design as one could hope for. A high degree of symmetry can help maximize efficiency of both hardware and software. Symmetry is the reason for the wire-crossing just before each $C_i \, C'_i$.

Among the efficiency characteristics of AEZ-core is that one can selectively decrypt a chosen block about 2.5 times more quickly than decrypting everything. When AEZ-core is turned into an RAE scheme by the EtE construction, this observation is put to good use in achieving fast rejection of ciphertexts whose final 0^τ bits is plaintext is not correct. That it is undamaging to release this timing information is guaranteed by results already show—in particular, that it is ok to release the *entire* speculative plaintext.

AEZ-core confines "specialized" processing to the final 2–4 blocks. This helps with efficiency and simplicity compared to having specialized processing at the beginning or at the beginning and end. In particular, the 0^τ authenticator used to make an RAE scheme will be put at the *end* of the message (adding a variable number of zero-bits at beginning could destroy word alignment) and, as long as $\tau \leq 2n$, it will be found in the final two blocks.

Numerous alternatives to AEZ-core were considered before arriving at our design. Correct alternatives we know are slower or more complex, while most simplifications are wrong. For example, consider trying to cheapen the design by using $c_i \cdot f_{\mathsf{aa},1}(S)$ instead of $f_{\mathsf{aa},i}(S)$ where each c_i is a public constant and the product is in $\mathrm{GF}(2^n)$. This fails for *any* choice of c_i. See Appendix D.

One variant of AEZ-core that *does* work is to eliminate the "left-hand" xor coming out of $f_{\mathsf{aa},i}$. (One then has to define X_i as the output of f_{a} instead of that output xor'ed with M'_1, and change Y_i similarly.) We have kept this xor because it's needed for symmetry.

7   Definition of AEZ

So far we have described two key elements of AEZ: the EtE construction and the AEZ-core$[f]$ wide-block blockcipher. Now we give AEZ's complete description. First a bit of notation.

NOTATION.   The bit length of a string X is written $|X|$. For the bitwise xor of unequal-length strings, drop the necessary number of rightmost bits from the longer ($10 \oplus 0100 = 11$). For X a string, let $X0^* = X0^p$ with p the smallest number such that 128 divides $|X| + p$. By $\mathcal{X}^*$ we denote the set of all strings over the alphabet $\mathcal{X}$, including ε. By $(\mathcal{X}^*)^*$ we denote the set of all vectors over $\mathcal{X}^*$, including the empty vector.

If $|X| = n$ and $1 \leq i \leq j \leq n$ then $X(i)$ is the ith bit of X (indexing from the left starting at 1), $\mathrm{msb}(X) = X(1)$, and $X(i..j) = X(i) \cdots X(j)$. Let $[n]_t$ be the t-bit string representing $n \bmod 2^t$ and let $[n]$ be shorthand for $[n]_8$; for example $[0]^{16} = ([0]_8)^{16} = 0^{128}$ and $[1]^{16} = (00000001)^{16}$. A block is 128 bits. Let $\mathbf{0} = 0^{128}$. If $X = a_1 \cdots a_{128}$ is a block ($a_i \in \{0, 1\}$) then we define $X \lll 1 = a_2 \cdots a_{128}\, 0$. For $n \in \mathbb{N}$ and $X \in \{0, 1\}^{128}$ define $n \cdot X$ by asserting that $0 \cdot X = \mathbf{0}$ and $1 \cdot X = X$ and $2 \cdot X = (X \lll 1) \oplus [135 \cdot \mathrm{msb}(X)]_{128}$ and $2n \cdot X = 2 \cdot (n \cdot X)$ and $(2n + 1) \cdot X = (2n \cdot X) \oplus X$.

For $K, X \in \{0, 1\}^{128}$ we write $\mathtt{aesenc}(X, K)$ for a single round of AES: $\mathtt{SubBytes}$, $\mathtt{ShiftRows}$, $\mathtt{MixColumns}$, then an $\mathtt{AddRoundKey}$ with K. For $\boldsymbol{K} = (K_0, K_1, K_2, K_3, K_4)$ a list of five blocks, let $\mathrm{AES4}_{\boldsymbol{K}}(X) = \mathrm{AES4}(\boldsymbol{K}, X)$ be $\mathtt{aesenc}(\mathtt{aesenc}(\mathtt{aesenc}(\mathtt{aesenc}(X \oplus K_0, K_1), K_2), K_3), K_4)$. For $\boldsymbol{K}$ a list of 11 blocks, $\boldsymbol{K} = (K_0, K_1, \ldots, K_{10})$, define $\mathrm{AES10}_{\boldsymbol{K}}(X) = \mathrm{AES10}(\boldsymbol{K}, X)$ like we defined AES4 but with ten rounds of $\mathtt{aesenc}$. We do not omit the final-round $\mathtt{MixColumns}$.

AEZ DEFINITION.   See Figs. 5 and 6 for the definition of AEZ, and Fig. 7 for an illustration. Most of it is self-explanatory. We briefly explain some of the algorithm's more unusual elements.

100	**algorithm** ENCRYPT(K, N, A, τ, M)	//AEZ authenticated encryption
101	$X \leftarrow M \parallel 0^\tau$; $(A_1, \ldots, A_m) \leftarrow A$	
102	$T \leftarrow ([\tau]_{128}, N, A_1, \ldots, A_m)$	
103	**if** $M = \varepsilon$ **then return** AEZ-prf(K, T, τ) **else return** Encipher(K, T, X)	

110	**algorithm** DECRYPT(K, N, A, τ, C)	//AEZ authenticated decryption
111	$(A_1, \ldots, A_m) \leftarrow A$; $T \leftarrow ([\tau]_{128}, N, A_1, \ldots, A_m)$	
112	**if** $|C| < \tau$ **then return** $\perp$	
113	**if** $|C| = \tau$ **then if** $C = $ AEZ-prf(K, T, τ) **then return** ε **else return** $\perp$ **fi fi**	
114	$X \leftarrow$ Decipher(K, T, C); $M \parallel Z \leftarrow X$ where $|Z| = \tau$	
115	**if** $(Z = 0^\tau)$ **then return** M **else return** $\perp$	

200	**algorithm** Encipher(K, T, X)	//AEZ enciphering
201	**if** $|X| < 256$ **then return** Encipher-AEZ-tiny(K, T, X)	
202	**if** $|X| \geq 256$ **then return** Encipher-AEZ-core(K, T, X)	

210	**algorithm** Encipher-AEZ-tiny(K, T, M)	//AEZ-tiny enciphering
211	$m \leftarrow |M|$; $n \leftarrow m/2$; $\Delta \leftarrow$ AEZ-hash(K, T)	
212	**if** $m = 8$ **then** $k \leftarrow 24$ **else if** $m = 16$ **then** $k \leftarrow 16$	
213	**else if** $m < 128$ **then** $k \leftarrow 10$ **else** $k \leftarrow 8$ **fi**	
214	$L \leftarrow M(1 \mathinner{..} n)$; $R \leftarrow M(n+1 \mathinner{..} m)$; **if** $m \geq 128$ **then** $j \leftarrow 6$ **else** $j \leftarrow 7$ **fi**	
215	**for** $i \leftarrow 0$ **to** $k-1$ **do**	
216	$R' \leftarrow L \oplus ((\mathrm{E}_K^{0,j}(\Delta \oplus R10^* \oplus [i]_{128}))(1 \mathinner{..} n))$; $L \leftarrow R$; $R \leftarrow R'$ **od**	
217	$C \leftarrow R \parallel L$; **if** $m < 128$ **then** $C \leftarrow C \oplus (\mathrm{E}_K^{0,3}(\Delta \oplus (C0^* \vee 10^*)) \wedge 10^*$ **fi**	
218	**return** C	

220	**algorithm** Encipher-AEZ-core(K, T, M)	//AEZ-core enciphering
221	$M_1 M_1' \cdots M_m M_m' \, M_{\mathsf{uv}} \, M_{\mathsf{x}} M_{\mathsf{y}} \leftarrow M$	
222	where $|M_1| = \cdots = |M_m'| = |M_{\mathsf{x}}| = |M_{\mathsf{y}}| = 128$ and $|M_{\mathsf{uv}}| < 256$	
223	$\Delta \leftarrow$ AEZ-hash(K, T); $d \leftarrow |M_{\mathsf{uv}}|$	
224	**if** $d \leq 127$ **then** $M_{\mathsf{u}} \leftarrow M_{\mathsf{uv}}$; $M_{\mathsf{v}} \leftarrow \varepsilon$	
225	**else** $M_{\mathsf{u}} \leftarrow M_{\mathsf{uv}}[1..128]$; $M_{\mathsf{v}} \leftarrow M_{\mathsf{uv}}[129..|M_{\mathsf{uv}}|]$ **fi**	
226	**for** $i \leftarrow 1$ **to** m **do** $W_i \leftarrow M_i \oplus \mathrm{E}_K^{1,i}(M_i')$; $X_i \leftarrow M_i' \oplus \mathrm{E}_K^{0,0}(W_i)$ **od**	
227	**if** $d = 0$ **then** $X \leftarrow X_1 \oplus \cdots \oplus X_m \oplus \mathbf{0}$	
228	**else if** $d \leq 127$ **then** $X \leftarrow X_1 \oplus \cdots \oplus X_m \oplus \mathrm{E}_K^{0,4}(M_{\mathsf{u}}10^*)$	
229	**else** $X \leftarrow X_1 \oplus \cdots \oplus X_m \oplus \mathrm{E}_K^{0,4}(M_{\mathsf{u}}) \oplus \mathrm{E}_K^{0,5}(M_{\mathsf{v}}10^*)$ **fi**	
230	$S_{\mathsf{x}} \leftarrow M_{\mathsf{x}} \oplus \Delta \oplus X \oplus \mathrm{E}_K^{0,1}(M_{\mathsf{y}})$; $S_{\mathsf{y}} \leftarrow M_{\mathsf{y}} \oplus \mathrm{E}_K^{-1,1}(S_{\mathsf{x}})$; $S \leftarrow S_{\mathsf{x}} \oplus S_{\mathsf{y}}$	
231	**for** $i \leftarrow 1$ **to** m **do**	
232	$S' \leftarrow \mathrm{E}_K^{2,i}(S)$; $Y_i \leftarrow W_i \oplus S'$; $Z_i \leftarrow X_i \oplus S'$	
233	$C_i' \leftarrow Y_i \oplus \mathrm{E}_K^{0,0}(Z_i)$; $C_i \leftarrow Z_i \oplus \mathrm{E}_K^{1,i}(C_i')$ **od**	
234	**if** $d = 0$ **then** $C_{\mathsf{u}} \leftarrow C_{\mathsf{v}} \leftarrow \varepsilon$; $Y \leftarrow Y_1 \oplus \cdots \oplus Y_m \oplus \mathbf{0}$	
235	**else if** $d \leq 127$ **then**	
236	$C_{\mathsf{u}} \leftarrow M_{\mathsf{u}} \oplus \mathrm{E}_K^{-1,4}(S)$; $C_{\mathsf{v}} \leftarrow \varepsilon$; $Y \leftarrow Y_1 \oplus \cdots \oplus Y_m \oplus \mathrm{E}_K^{0,4}(C_{\mathsf{u}}10^*)$	
237	**else** $C_{\mathsf{u}} \leftarrow M_{\mathsf{u}} \oplus \mathrm{E}_K^{-1,4}(S)$; $C_{\mathsf{v}} \leftarrow M_{\mathsf{v}} \oplus \mathrm{E}_K^{-1,5}(S)$	
238	$Y \leftarrow Y_1 \oplus \cdots \oplus Y_m \oplus \mathrm{E}_K^{0,4}(C_{\mathsf{u}}) \oplus \mathrm{E}_K^{0,5}(C_{\mathsf{v}}10^*)$ **fi**	
239	$C_{\mathsf{y}} \leftarrow S_{\mathsf{x}} \oplus \mathrm{E}_K^{-1,2}(S_{\mathsf{y}})$; $C_{\mathsf{x}} \leftarrow S_{\mathsf{y}} \oplus \Delta \oplus Y \oplus \mathrm{E}_K^{0,2}(C_{\mathsf{y}})$	
240	**return** $C_1 C_1' \cdots C_m C_m' \, C_{\mathsf{u}} C_{\mathsf{v}} \, C_{\mathsf{x}} C_{\mathsf{y}}$	

Fig. 5. Main routines of AEZ. The tweakable blockcipher E, the hash AEZ-hash, and the PRF AEZ-prf are defined in Fig. 6. The ciphertext expansion is $\tau = 8\lambda$ bits.

300	**algorithm** AEZ-hash(K,T)	//AXU hash. T is a vector of strings
301	$(T_1, \ldots, T_t) \leftarrow T$	
302	**for** $i \leftarrow 1$ **to** t **do**	
303	$\quad m \leftarrow \max(1, \lceil |T_i|/128 \rceil); \; X_1 \cdots X_m \leftarrow T_i \;\; /\!/|X_1| = \cdots = |X_{m-1}| = 128$	
304	$\quad$ **if** $|X_m| = 128$ **then** $\Delta_i \leftarrow \mathrm{E}_K^{2+i,1}(X_1) \oplus \cdots \oplus \mathrm{E}_K^{2+i,m}(X_m)$	
305	$\quad$ **if** $|X_m| < 128$ **then**	
306	$\qquad \Delta_i \leftarrow \mathrm{E}_K^{2+i,1}(X_1) \oplus \cdots \oplus \mathrm{E}_K^{2+i,m-1}(X_{m-1}) \oplus \mathrm{E}_K^{2+i,0}(X_m 10^*)$	
307	**return** $\Delta_1 \oplus \cdots \oplus \Delta_t \oplus \mathbf{0}$	

310	**algorithm** AEZ-prf(K,T,τ)	//PRF used when $M = \varepsilon$
311	$\Delta \leftarrow$ AEZ-hash(K,T)	
312	**return** $(\mathrm{E}_K^{-1,3}(\Delta) \,\|\, \mathrm{E}_K^{-1,3}(\Delta \oplus [1]_{128}) \,\|\, \mathrm{E}_K^{-1,3}(\Delta \oplus [2]_{128}) \,\|\, \cdots)[1..\tau]$	

|		
400	**algorithm** $\mathrm{E}_K^{i,j}(X)$	//Scaled-down TBC
401	$I \,\|\, J \,\|\, L \leftarrow \mathrm{Extract}(K)$ where $|I| = |J| = |L| = 128$	
402	$\boldsymbol{k}_0 \leftarrow (\mathbf{0}, I, J, L, \mathbf{0}); \;\; \boldsymbol{k}_1 \leftarrow (\mathbf{0}, J, L, I, \mathbf{0}); \;\; \boldsymbol{k}_2 \leftarrow (\mathbf{0}, L, I, J, I)$	
403	$\boldsymbol{K} \leftarrow (\mathbf{0}, I, L, J, I, L, J, I, L, J, I)$	
404	**if** $i = -1$ and $0 \le j \le 7$ **then return** $\mathrm{AES10}_{\boldsymbol{K}}(X \oplus jJ)$	
405	**if** $i = 0$ and $0 \le j \le 7$ **then return** $\mathrm{AES4}_{\boldsymbol{k}_0}(X \oplus jJ)$	
406	**if** $1 \le i \le 2$ and $j \ge 1$ **then return** $\mathrm{AES4}_{\boldsymbol{k}_i}(X \oplus (j \bmod 8)J \oplus 2^{\lfloor (j-1)/8 \rfloor}L)$	
407	**if** $i \ge 3$ and $j \ge 1$ **then**	
408	$\quad$ **return** $\mathrm{AES4}_{\boldsymbol{k}_0}(X \oplus (j \bmod 8)J \oplus 2^{\lfloor (j-1)/8 \rfloor} \cdot L \oplus (i-2)8J)$	
409	**if** $i \ge 3$ and $j = 0$ **then return** $\mathrm{AES4}_{\boldsymbol{k}_0}(X \oplus (i-2)8J)$	

Fig. 6. AEZ's hash, PRF, and TBC. The last is the locus of prove-then-prune scaling-down. The key K is turned into 384 bits by a key-derivation function Extract.

AEZ operates on arbitrary byte strings. Not only is the plaintext $M \in \mathrm{BYTE}^*$ arbitrary, but so too the key $Key \in \mathrm{BYTE}^*$ and nonce $N \in \mathrm{BYTE}^*$. The AD is even more general: an arbitrary-length vector of byte strings, $A \in (\mathrm{BYTE}^*)^*$. The requested ciphertext expansion of $\lambda \in \mathbb{N}$ bytes is measured in $\tau = 8\lambda$ bits.

At line 217, Encipher-AEZ-tiny may xor a bit into the ciphertext just before the algorithm's conclusion. This is done to avoid a simple random-permutation distinguishing attacks, for very short strings, based on the fact that Feistel networks only generate *even* permutations [30]. A similar trick, conditionally swapping two fixed points, has been used before [45]. Our approach has the benefit that the natural implementation is constant-time.

We define Decipher(K, T, Y) as the unique X such that Encipher$(K, T, X) = Y$. Logically, this is all we need say for the specification to be well-defined. Still, the additional pseudocode is easy to describe. AEZ-tiny deciphering is identical to AEZ-tiny enciphering except we must count backwards instead of forwards, and must do the even-cycles correction (line 217) at the beginning instead of the end. Specifically, Decipher-AEZ-tiny(K, T, M) is identical to Encipher-AEZ-tiny(K, T, M) except that line 215 is changed to count from $k-1$ down to 0, while line 217 has each C replaced by M before moving the line up to just after line 213. And AEZ-core deciphering is identical to AEZ-core enciphering except that we must take the xy-tweaks in reverse order. Specifically, Decipher-AEZ-core(K, T, M) is identical to Encipher-AEZ-core(K, T, M) except we swap tweaks $(0, 1)$ and $(0, 2)$, and we swap tweaks $(-1, 1)$ and $(-1, 2)$. These appear at lines 230 and 239.

The TBC $\mathrm{E}_K^{i,j}(X)$ takes a tweak $(i, j) \in \{-1, 0\} \times [0..7] \cup \{1, 2, 3\} \times \mathbb{N}$. The first component selects between AES10 (when $i = -1$) and AES4 (when $i \ge 0$). Either way, the construction is based on XE [34, 49]. We emphasize that E is *not* secure as a tweakable-PRP, since AES4 itself is completely
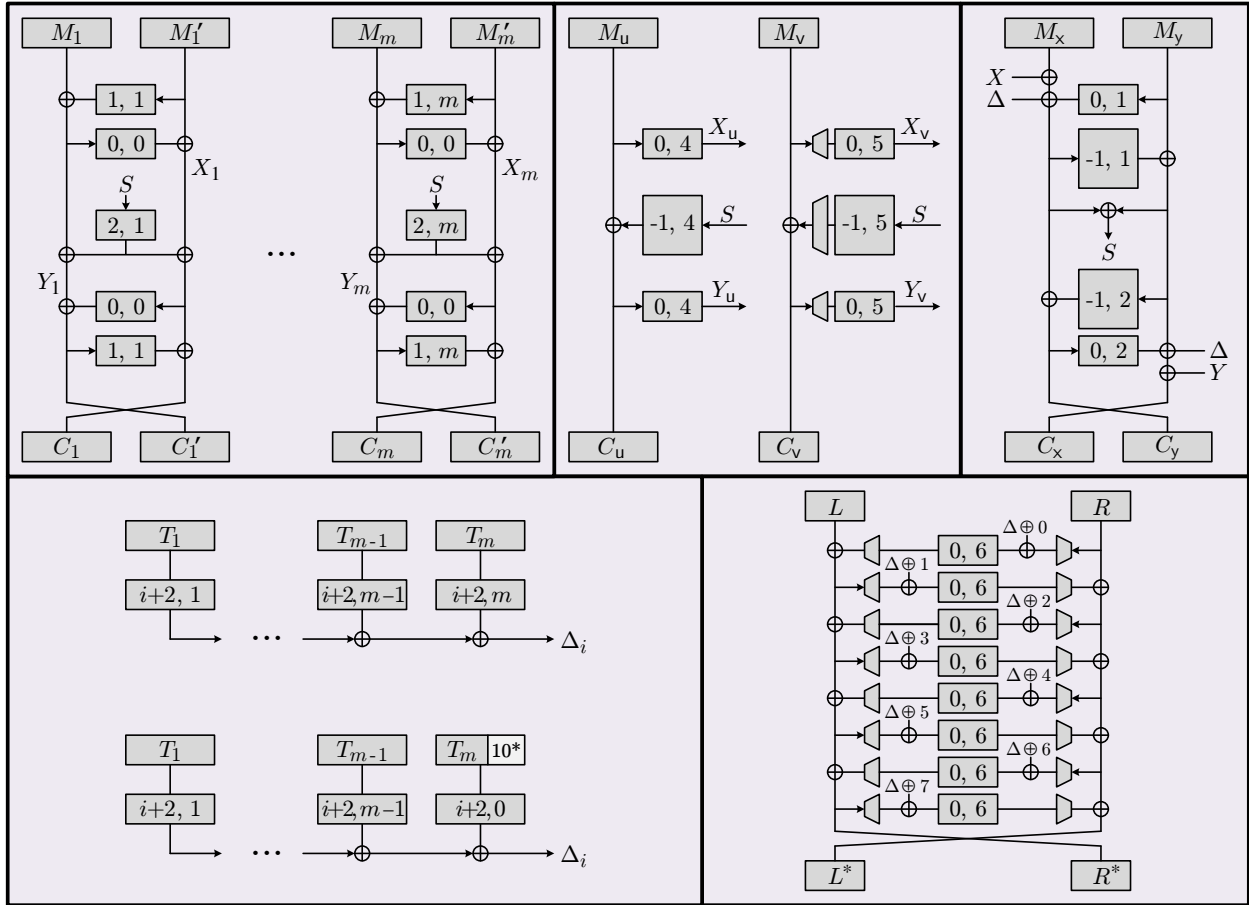
Fig. 7. Illustrating AEZ enciphering. Rectangles with pairs of numbers are TBCs, the pair being the tweak (the key, always K, is not shown). **Top row:** enciphering a message M of (32 or more bytes) with AEZ-core. The diagram shows processing a string that is (exclude the middle panel) or isn't (include the middle panel) a multiple of 16 bytes. **Bottom left:** AEZ-hash is an xor-universal hash built from AES4. It computes $\Delta = \bigoplus \Delta_i$ from a vector-valued tweak T comprising A, N, and τ. Its i-th component $T_1 \cdots T_m$ is hashed as shown. **Bottom right:** AEZ-tiny, when operating on a string $M = L \parallel R$ of 16–31 bytes. More rounds are used if M has 1–15 bytes.

insecure as a PRP: it is easily broken by the "Square" attack [14]. Use of an AES4-based TBC *despite* this fact is where the scaling-down has been done in AEZ.

The key $K \in \text{BYTE}^*$ is mapped to three 16-byte subkeys (I, J, L) using the key-derivation function (KDF) named Extract that is called at line 401. The definition of Extract is omitted from the figures and regarded as orthogonal to the rest of AEZ. See the AEZ spec [26] for the current Extract : $\text{BYTE}^* \to \text{BYTE}^{48}$. In our view, it is an unresolved matter what the security properties (and even what signature) of a good KDF should be. Work has gone off in very different directions [33, 46, 61], and the area is currently the subject of a Password Hashing Competition (PHC) running concurrently with CAESAR.

Note the mod 8's at lines 406 and 408. Unlike the offset sequence used for OCB [32], we limit ourselves to eight successive J values; after that, we add in the next power-of-two times L. This allows a small table of $2^j \cdot J$ values to be precomputed and used regardless of the length of

the message. In this way we limit the frequency of doublings yet avoid number-of-trailing-zeros calculations.

We impose a limit that AEZ be used for at most 2^{48} bytes of data (about 280 TB); by that time, the user should rekey. This usage limit stems from the existence of birthday attacks on AEZ, as well as the use of AES4 to create a universal hash function.

COST ACCOUNTING.   Let us summarize the computational cost of AEZ in "AES-equivalents," where 1 AES-equivalent is 10 AES rounds. Assume a message of m blocks, the last of which may be fragmentary. To encipher or decipher $m \geq 2$ blocks takes at most $m + 2.4$ AES-equivalents (latency 3.6). This assumes K, N, τ, and A have already been processed. To encrypt or decrypt $m \geq 2$ blocks: at most $m + 3.8$ AES-equivalents (latency 3.6). This assumes that K, A, and τ have already been processed and that $|N| \leq 128$ and $\tau = 128$. To reject an invalid ciphertext of $m \geq 2$ blocks: at most $0.4m + 2.4$ AES-equivalents (latency 2.8). Same assumptions. To setup an m block key: $1.2m$ AES-equivalents (latency 0.4). This assumes that needed constants have been precomputed. To setup a string-values AD: $0.4m$ (latency 0.4). To encipher or decipher messages of 1–15 bytes is somewhat slower: 10, 6.8, and 4.4 AES-equivalents for 1, 2, and 3 bytes.

PARAMETERIZED COUNTERPARTS.   For a TBC-parameterized generalization of AEZ, let $\mathrm{AEZ}[\widetilde{E}]$ be identical to AEZ except for using the TBC $\widetilde{E} \colon \mathcal{K} \times \mathcal{T}_{\mathrm{aez}} \times \{0,1\}^{128} \to \{0,1\}^{128}$ in place of E (assume the correct tweak-space $\mathcal{T}_{\mathrm{aez}}$). The key space of $\widetilde{E}$ is then taken as the key space for the constructed RAE scheme. Note that $\mathrm{AEZ} = \mathrm{AEZ}[\mathrm{E}]$ with E the algorithm defined by lines 400–409.

Taking the above a step further, given a conventional blockcipher $E \colon \mathcal{K} \times \{0,1\}^{128} \to \{0,1\}^{128}$ we can define $\mathrm{AEZ}[E]$ as $\mathrm{AEZ}[\widetilde{E}]$ where $\widetilde{E}_K^{i,j}(X) = E_K(X \oplus (i+1)I \oplus jJ)$ for $I = E_K(\mathbf{0})$ and $J = E_K(\mathbf{1})$. The scheme $\mathrm{AEZ}[\mathrm{AES}]$ can be regarded as a natural "scaled up" version of AEZ. We emphasize that AEZ is *not* $\mathrm{AEZ}[\mathrm{AES}]$, which is about 2.5 times as slow.

Schemes $\mathrm{AEZ}[\widetilde{E}]$ and $\mathrm{AEZ}[E]$ are close to AEZ, but enjoy conventional provable-security guarantees, as we now describe.

8   Security of $\mathbf{AEZ}[\widetilde{E}]$ and $\mathbf{AEZ}[E]$

We show that if $\widetilde{E}$ is secure as a tweakable PRP then $\mathrm{AEZ}[\widetilde{E}]$ is RAE-secure. In fact, the statement holds even if the decryption algorithm is modified so as to leak the entire improperly encoded string obtained by deciphering an invalid ciphertext. So, for the remainder of this section, assume the modification of AEZ in which the **else** clause of line 115 returns the deciphered message X rather than $\perp$. This change only makes our results stronger, explicitly modeling the *possibility* of a decryption implementation leaking some or all of X. The *actual* decryption algorithm returns $\perp$.

Our provable-security results for AEZ need to assume that the adversary avoids enciphering or deciphering extremely short strings—at least those under 16 bytes, say, for which AEZ-tiny, a Feistel-based construction, will not enjoy a desirable bound. While provably-secure options are now available for enciphering very short strings, they still do not have competitive efficiency.

As the alphabet for AEZ is $\varSigma = \mathrm{BYTE}$, in this this section we write $|x|$ for the byte length of x. For an encryption query (N, A, λ, M), define the number of blocks processed as $\lceil |N|/16 \rceil + \sum_i \lceil |A_i|/16 \rceil + \lceil (|M| + \lambda)/16 \rceil$. This query is *small* if $M \neq \varepsilon$ and $16 \leq |M| + \lambda < 32$, and *tiny* if $M \neq \varepsilon$ and $|M| + \lambda < 16$. Likewise, for a decryption query (N, A, λ, C), the number of blocks processed is $\lceil |N|/16 \rceil + \sum_i \lceil |A_i| \rceil + \lceil (|C|)/16 \rceil$. The query is *small* if $16 \leq |C| < 32$ and $|C| \neq \lambda$, and *tiny* if $|C| \neq \lambda$ and $|C| < 16$. The proof for the following is in Appendix B.5.

Theorem 5. Let $\widetilde{E}\colon \mathcal{K} \times \mathcal{T}_{\mathrm{aez}} \times \{0,1\}^{128} \to \{0,1\}^{128}$ be a TBC and $\Pi = \mathrm{AEZ}[\widetilde{E}]$. Then there are efficient, explicitly given algorithms $\mathcal{R}$ and S with the following property. Let $\mathcal{A}$ be an adversary for attacking Π. Assume it never asks any small or tiny query. Then $\mathcal{B} = \mathcal{R}(\mathcal{A})$ satisfies $\mathbf{Adv}_{\Pi,S}^{\mathrm{rae}}(\mathcal{A}) \leq 3.5 s^2/2^{128} + \mathbf{Adv}_{\widetilde{E}}^{\widetilde{\mathrm{prp}}}(\mathcal{B})$, where s is the total number of processed blocks, plus 2 blocks per message. Adversary $\mathcal{B}$ makes at most $2.5 s$ queries and has about the same running time as $\mathcal{A}$.

An alternative approach to justifying the security of AEZ is to speak of the security of AEZ[E], the cousin of AEZ defined from a conventional blockcipher E using the XE construction to make the needed TBC. Its security can be captured by the following result. The proof is in Appendix B.6.

Theorem 6. Let $E\colon \mathcal{K} \times \{0,1\}^{128} \to \{0,1\}^{128}$ be a blockcipher and $\Pi = \mathrm{AEZ}[E]$. Then there are efficient, explicitly given algorithms $\mathcal{R}$ and S with the following property. Let $\mathcal{A}$ be an adversary for attacking Π. Assume it never asks a small or tiny query. Then $\mathcal{B} = \mathcal{R}(\mathcal{A})$ satisfies $\mathbf{Adv}_{\Pi,S}^{\mathrm{rae}}(\mathcal{A}) \leq 13 s^2/2^{128} + \mathbf{Adv}_{E}^{\mathrm{prp}}(\mathcal{B})$, where s is the total number of processed blocks, plus 2 blocks per message. Adversary $\mathcal{B}$ makes at most $2.5 s$ queries and has about the same running time as $\mathcal{A}$.

If one wants to accommodate small queries then we still have a provable, albeit much inferior result. Let Feistel$[r, n]$ denote an ideal r-round Feistel network on $\{0,1\}^{2n}$. The best known provable bound for Feistel networks [43, Theorem 7] states that if an adversary makes $q \leq \frac{2^n}{128n}$ queries then $\mathbf{Adv}_{\mathrm{Feistel}[6,n]}^{\pm\mathrm{prp}}(\mathcal{A}) \leq \frac{8q}{2^n} + \frac{q^2}{2^{2n+1}}$. Translating this to our setting, one is bound to make at most $q \leq \frac{2^{64}}{128 \cdot 64} = 2^{51}$ small queries, and the security advantage is $q/2^{61} + 4s^2/2^{128}$. These restrictions seem to be more of the artifacts of the analysis in [43, Theorem 7] than reflecting the actual security of Feistel networks: assuming that the round functions of Feistel$[6, n]$ are instantiated from full AES, the fastest known attack, for $n \geq 64$, is still the exhaustive key search on AES.

9   Estimated Security of AEZ Itself

Consider enciphering a message M, $|M| \geq 256$, by AEZ[AES] (which, recall, is not AEZ, but a scaled-up version using an AES-based TBC). The design would seem excessive: each block M_i would be subjected to 30 rounds of AES (ten shared with a neighboring block), not counting the additional AES rounds to produce the highly unpredictable, M-dependent value S, a value derived from which gets injected into the process while 20 rounds yet remain. It is in light of such apparent overkill that AEZ selectively prunes some of the AES calls that AEZ[AES] would perform. In particular, we prune invocations where we aim to achieve computational xor-universal hashing. We leave enough AES rounds so that each block M_i is effectively processed with 12 AES rounds, eight of these subsequent to injection of the highly-unpredictable S and four of them shared with a neighboring block. The key steps in calculating S are not pruned, nor are the TBCs used to mask u- and v-blocks.

To estimate the security of AEZ it seems appropriate to replace the $s^2/2^{128}$ term of Theorem 5 by $s^2/2^{113}$, resulting in the bound $4s^2/2^{113} + t/2^{128}$, because of the higher maximal expected differential probability of AES4 [31] compared to an ideal hash or cipher, where t is the time (including the description size) in which the adversary runs.

Moreover, we contend that the assumption that the adversary avoids asking tiny or small queries can be lifted. To justify this heuristically, consider a collection of independent, ideal, k-round Feistel networks on $\{0,1\}^{2n}$; the round functions are all uniformly random and independent. The best
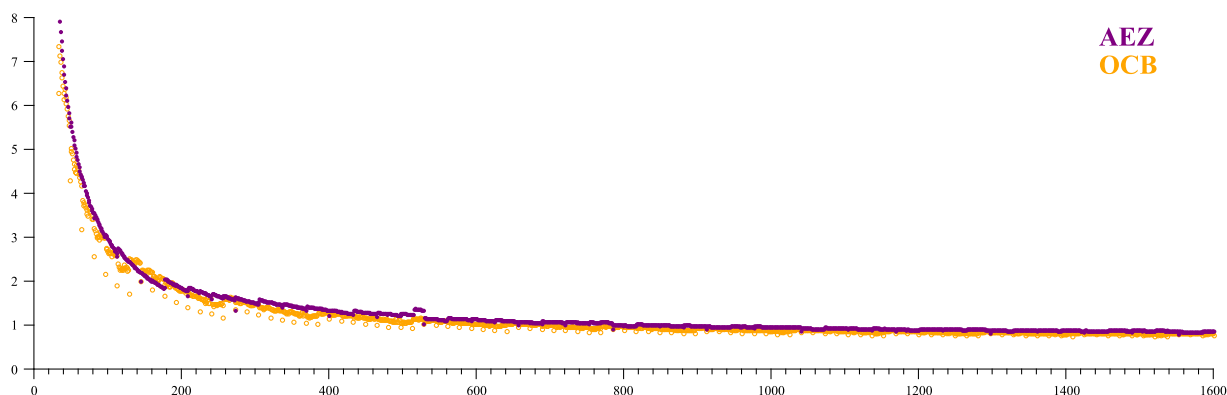
Fig. 8. AEZ vs. OCB performance. The x-axis is message length, in bytes, and the y-axis is cycles per byte (cpb). The graph is best viewed in color: solid purple circles are for AEZ; unfilled yellow circles are for OCB3 [32]. Performance of the two is close, both having peak speeds around 0.7 cpb and being similar on most shorter messages as well. The execution vehicle is an Intel Haswell processor using AES-NI.

attack known, due to Patarin [41], that distinguishes them from a family of independent, truly random even permutations requires at least $2^{(k-4)n}$ plaintext/ciphertext pairs. From our choice of the number of rounds, this attack needs at least 2^{72} plaintext/ciphertext pairs, and thus doesn't violate our up-to-the-birthday-bound security goal.

AEZ was specifically designed so that scaling-down most of its AES calls would seem safe. This is design-specific; one cannot indiscriminately scale a scheme's primitives. A previous design, where AEZ-core followed the NR approach [39, 40], could not be as effectively scaled-down.

10   Software Performance

The development of AEZ has generally presumed an instruction set architecture (ISA) with round-level support for AES, such as Intel's AES-NI or ARM's version 8 ISA. On these systems the AES unit can be kept busy processing several AES4 computations in parallel while idle processing units handle load, store, and xor overhead. On Intel's Haswell architecture, for example, unrelated AES rounds can issue every cycle and take seven cycles to retire, so seven parallel AES4 calculations can complete in 34 CPU cycles, while idle superscalar processing units can handle other computations. This observation has led us to design AEZ to conveniently process eight blocks at a time.

AEZ overhead beyond AES rounds has been minimized. As an example of this, our AES4 key schedule omits the final round key, allowing `aesenc`'s included xor operation to be used for other purposes. Such optimizations lead to AEZ peak speeds, on Haswell, of around 0.72 cpb—not far from the theoretical maximum for the number of rounds executed of 0.63 cpb.

Fig. 8 compares the performance of AEZ and OCB on messages of all byte lengths up to 1600 bytes. The two are not only similar for long messages but for short strings too. Only when messages are shorter than 16 bytes, where AEZ-tiny increases the number of AES4 calls used, does OCB become significantly faster.

The performance of AEZ is on par with OCB even on processors that are not superscalar or do not support AES rounds at the assembly level. On a Marvell 88F6283 embedded CPU—a single-issue, 32-bit, ARM version 5 ISA—we see an experimental version of AEZ peaking at 86 cpb while

OCB's optimized reference code runs at 84 cpb. For comparison, GCM, CCM and CTR run at 124, 134 and 67 cpb, respectively. The figures use the OpenSSL libraries.

One might expect the two-pass nature of AEZ to be a performance burden because data must be dragged into cache from memory twice. We have found that modern processors, like Intel's Haswell, have such efficient hardware prefetching that bringing data into cache twice, in a sequential streaming fashion, is not expensive at all. It requires no explicit prefetching. Encrypting 1MB on Haswell is as efficient as encrypting 32KB despite 1MB exceeding the 256KB level-2 cache. Two passes may have a more significant cost on systems with poor prefetching facilities, although this might be mitigated by software prefetching.

Another benefit of AEZ's two passes is that the second pass is not needed to discover that a ciphertext is inauthentic, leading to message rejection costing as little as 0.28 cpb on Haswell. On long messages, approximately 2/5 of AES4 calls are performed during the first pass, which aligns perfectly with the peak times we've observed for encryption and fast-rejection.

All Haswell timings reported in this paper were gathered on a 2.9 GHz Intel Core i5-4570S CPU using its time-stamp counter to gather elapsed CPU cycles over encryption calls. Our implementation is written in C using "intrinsic" functions to access CPU-specific functionality. It was compiled using GCC 4.9 with options `-march=native -O3`. Our optimized implementation will be made publicly available and freely licensed.

Acknowledgments

References

1. E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha, and K. Yasuda. How to securely release unverified plaintext in authenticated encryption. Cryptology ePrint report 2014/144. Feb 25, 2014.
2. E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, and K. Yasuda. Parallelizable and authenticated online ciphers. *ASIACRYPT 2013*. LNCS 8269, Springer, pp. 424–443, 2013.
3. M. Bellare, A. Boldyreva, L. Knudsen, and C. Namprempre. Online ciphers and the Hash-CBC construction. *CRYPTO 2001*. LNCS 2139, pp. 292–309, 2001. Also Cryptology ePrint Archive 2007/197.
4. M. Bellare and P. Rogaway. On the construction of variable-input-length ciphers. *FSE 1999*. LNCS 1636, Springer, pp. 321-344, 1999.

5. M. Bellare and P. Rogaway. Encode-then-encipher encryption: how to exploit nonces or redundancy in plaintexts for efficient cryptography. *ASIACRYPT 2000*. LNCS 1976, Springer, pp. 317–330, 2000.

6. M. Bellare, P. Rogaway, and T. Spies. The FFX mode of operation for format-preserving encryption. Draft 1.1. Submission to NIST. Feb 20, 2010.

7. D. Bernstein. Cryptographic competitions: CAESAR call for submissions, final (2014.01.27). Available at http://competitions.cr.yp.to/caesar-call.html.

8. J. Black and M. Cochran. MAC reforgeability. LNCS 5665, Springer, FSE 2009, pp. 345–362, 2009.

9. J. Black and P. Rogaway. Ciphers with arbitrary finite domains. *CT-RSA 2002*. LNCS 2271, Springer, pp. 114–130, 2002.

10. A. Boldyreva, J. Degabriele, K. Paterson, and M. Stam. On symmetric encryption with distinguishable decryption failures. Cryptology ePrint Report 2013/433, 2013.

11. D. Chakraborty and M. Nandi. An improved security bound for HCTR. *FSE 2008*. LNCS 5086, Springer, pp. 289–302, 2008.

12. D. Chakraborty and P. Sarkar. HCH: A new tweakable enciphering scheme using the hash-encrypt-hash approach. *IEEE Transactions on Information Theory*, 54(4), pp. 1683–1699, 2008.

13. D. Chakraborty and P. Sarkar. A new mode of encryption providing a tweakable strong pseudorandom permutation. *FSE 2006*. LNCS 4047, Springer, pp. 293–309, 2006.

14. J. Daemen and V. Rijmen. *The Design of Rijndael: AES — The Advanced Encryption Standard*. Springer-Verlag, 2002.

15. J. Daemen and V. Rijmen. A new MAC construction ALRED and a specific instance ALPHA-MAC. *Fast Software Encryption*. LNCS 3557, Springer, pp. 1–17, 2005.

16. J. Daemen and V. Rijmen. The Pelican MAC function. Cryptology ePrint report 2005/088. 2005.

17. M. Dworkin. Recommendation for block cipher modes of operation: methods for format-preserving encryption. NIST Special Publication 800-38G: Draft. July 2013.

18. N. Ferguson. Authentication weaknesses in GCM. Manuscript. May 20, 2005.

19. R. Fisher and F. Yates. *Statistical tables for biological, agricultural and medical research*. London: Oliver & Boyd, 1938.

20. E. Fleischmann, C. Forler, S. Lucks, and J. Wenzel. McOE: A family of almost foolproof on-line authenticated encryption schemes. *FSE 2012*. LNCS 7549, Springer, pp. 196–215, 2011. Also Cryptology ePrint Report 211/644 (29 Nov 2011; revised 5 Dec 2013).

21. P. Fouque, A. Joux, G. Martinet, and F. Valette. Authenticated on-line encryption. *SAC 2003*. LNCS 3006, pp. 145–159, 2004.

22. S. Halevi. EME*: extending EME to handle arbitrary-length messages with associated data. *INDOCRYPT 2004*. LNCS 3347, Springer, pp. 315–327, 2004.

23. S. Halevi. Invertible universal hashing and the TET encryption mode. Cryptology ePrint report 2007/014.

24. S. Halevi and P. Rogaway. A parallelizable enciphering mode. *CT-RSA 2004*. LNCS 2964, Springer, pp. 292–304, 2004.

25. S. Halevi and P. Rogaway. A tweakable enciphering mode. *CRYPTO 2003*. LNCS 2729, Springer, pp. 482–499, 2003.

26. V. T. Hoang, T. Krovetz, and P. Rogaway. AEZ v3: authenticated encryption by enciphering. CAESAR submission. 2014.

27. J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. *FSE 2000*, LNCS 1978, Springer, pp. 284–299, 2000.

28. V. T. Hoang, T. Krovetz, and P. Rogaway. Robust authenticated-encryption: AEZ and the problem that it solves. Cryptology ePrint report 2014/793. Jan 2015. Full version of this paper.

29. IEEE. 1619.2-2010 – IEEE standard for wide-block encryption for shared storage media. IEEE press, 2010.

30. B. Kaliski, R. Rivest, and A. Sherman. Is DES a pure cipher? (Results of more cycling experiments on DES). *CRYPTO 85*. LNCS 218, Springer, pp. 212–226, 1986.

31. L. Keliher and J. Sui. Exact maximum expected differential and linear probability for two-round Advanced Encryption Standard. *IET Information Security*, 1(2), pp. 53–57, 2007.

32. T. Krovetz and P. Rogaway. The software performance of authenticated-encryption modes. *FSE 2011*. LNCS 6733, Springer, pp. 306–327, 2011.

33. H. Krawczyk. Cryptographic extraction and key derivation: the HKDF scheme. *CRYPTO 2010*. LNCS 6223, Springer, pp. 631–648, 2010.

34. M. Liskov, R. Rivest, and D. Wagner. Tweakable block ciphers. *CRYPTO 2002*. LNCS 2442, Springer, pp. 31–46, 2002.

35. D. McGrew and S. Fluhrer. The security of the extended codebook (XCB) mode of operation. *SAC 2007*. LNCS 4876, Springer, pp. 311–327, 2007.

36. K. Minematsu. Parallelizable rate-1 authenticated encryption from pseudorandom functions. *EURO-CRYPT 2014*. LNCS 8441, Springer, pp. 275–292, 2014.

37. K. Minematsu and Y. Tsunoo. Provably secure MACs from differentially-uniform permutations and AES-based implementations. *FSE 2006*. LNCS 4047, Springer, pp. 226–241, 2006.

38. M. Nandi. Improving upon HCTR and matching attacks for Hash-Counter-Hash approach. Cryptology ePrint report 2008/090. Feb 28, 2008.

39. M. Naor and O. Reingold. On the construction of pseudo-random permutations: Luby-Rackoff revisited. *Journal of Cryptology*, 12(1), pp. 29-66, 1999.

40. M. Naor and O. Reingold. The NR mode of operation. Undated manuscript realizing the mechanism of [39].

41. J. Patarin. Generic attacks on Feistel schemes. *ASIACRYPT 2001*. LNCS 2248, Springer, pp. 222–238, 2001. Also see Cryptology ePrint report 2008/036.

42. S. Patel, Z. Ramzan, and G. Sundaram. Efficient constructions of variable-input-length block ciphers. *SAC 2004*. LNCS 3357, Springer, 2004.

43. J. Patarin. Security of balanced and unbalanced Feistel schemes with linear non equalities. Cryptology ePrint report 2010/293. May 2010.

44. J. Patarin. Security of random Feistel schemes with 5 or more rounds. *CRYPTO 2004*. LNCS 3152, Springer, pp. 106–122, 2004.

45. J. Patarin, B. Gittins, and J. Treger. Increasing block sizes using Feistel networks: the example of the AES. *Cryptography and Security: From Theory to Applications*. LNCS 6805, Springer, pp. 67–82, 2012.

46. C. Percival. Stronger key derivation via sequential memory-hard functions. *The BSD Conference* (BSDCan), May 2009.

47. R. Reyhanitabar and D. Vizár. Careful with misuse resistance of online AEAD. Unpublished manuscript distributed on the `crypto-competitions` mailing list. August 24, 2014.

48. P. Rogaway. Authenticated-encryption with associated-data. *ACM CCS 2002*. ACM Press, pp. 98–107, 2002.

49. P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. *ASIACRYPT 2004*. LNCS 3329, Springer, pp. 16–31, 2004.

50. P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM CCS 2001, pp. 196–205.

51. P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. *EUROCRYPT 2006*. LNCS 4004, Springer, pp. 373–390, 2006. Also Cryptology ePrint Report 2006/221, retitled Deterministic authenticated-encryption: a provable-security treatment of the key-wrap problem.

52. P. Sarkar. Efficient tweakable enciphering schemes from (block-wise) universal hash functions. Cryptology ePrint report 2008/004.

53. P. Sarkar. Improving upon the TET mode of operation. *Information Security and Cryptology* (ICISC 2007). LNCS 4817, Springer, pp. 180–192, 2007.

54. P. Sarkar. Tweakable enciphering schemes using only the encryption function of a block cipher. Cryptology ePrint report 2009/216.

55. R. Schroeppel. Hasty Pudding Cipher Specification. AES candidate submitted to NIST. June 1998 (revised May 1999). http://richard.schroeppel.name/hpc/hpc-spec

56. T. Shrimpton and S. Terashima. A modular framework for building variable-input-length tweakable ciphers. *ASIACRYPT (1) 2013*. LNCS 8269, Springer, pp. 405–423, 2013.

57. M. Simplício, P. Barbuda, P. Barreto, T. Carvalho, and C. Margi. The MARVIN message authentication code and the LETTERSOUP authenticated encryption scheme. *Security and Communications Networks*, 2(2), pp. 165–180, 2009.

58. R. Struik. AEAD ciphers for highly constrained networks. DIAC 2013 presentation. Aug 13, 2013.

59. P. Wang, D. Feng, C. Lin, and W. Wu. Security of truncated MACs. *Inscrypt 2008*. LNCS 5487, Springer, pp. 96–114, 2009.

60. P. Wang, D. Feng, and W. Wu. HCTR: a variable-input-length enciphering mode. *Information Security and Cryptology* (ICISC 2005). LNCS 3822, Springer, pp. 175–188, 2005.

61. F. Yao and Y. L. Yin. Design and analysis of password-based key derivation functions. *IEEE Trans. on Information Theory*, 51(9), pp. 3292–3297, 2005.

A   More on Related Work

RAE and AEZ build on a large body of related work. While we have summarized much of this throughout this paper, here we give some additional context and high points.

Blockciphers accommodating truly arbitrary inputs were first realized by Schroeppel's Hasty Pudding Cipher (HPC) [55]. Ahead of its time, the work not only built a blockcipher on all of $\{0,1\}^*$, but also provided it a tweak. If one were to first overcome the problem that HPC's tweak is limited in length, it could be used with the EtE construction to make an RAE scheme.

The problem of constructing from conventional blockciphers those with arbitrary or near-arbitrary domains was first identified Bellare and Rogaway [4], who wanted to construct these objects with a conventional-looking mode. But the mechanism they suggested was somewhat slow, was limited to a domain of $(\{0,1\}^n)^+$, and only achieves conventional (not strong) PRP security.

In a follow-up paper [5] the same authors evidenced the utility of arbitrary-input-length blockciphers by explaining how semantic security could be achieved by enciphering messages with novelty, and they showed how authenticity could be achieved by enciphering messages with redundancy (this time using a *strong* PRP). These observations formed the basis for our work.

Around the same time as the last two work, Naor and Reingold (NR) constructed a blockcipher on $(\{0,1\}^n)^+$ by sandwiching a layer of ECB between layers of a "blockwise-universal" hashing [39, 40]. The approach came to be used in many proposals, including XCB [35], which was standardized in the IEEE [29].

The other method inspiring further wide-block blockciphers was EME [24], which involves two layers of blockcipher-based enciphering and a light layer of mixing in between. A follow-on design, EME2 [22], become the other wide-block blockcipher of IEEE 1619.2 [29], Both it and XCB are tweakable and operate on a message space of $\{0,1\}^{\geq n}$. EME/EME2 provides the starting point for AEZ-core.

As for extending blockciphers to short blocks, a different line of work was begun [9]. *Format-preserving encryption* aimed to deal not only with small domains but also those defined as arbitrary finite sets, sets of numbers $[0..N-1]$, or strings over arbitrary alphabets. Adapting Feistel designs to arbitrary alphabets, realizations of FFX [6], now a draft NIST standard [17], would form the basis of AEZ-tiny.

Meanwhile, notions of AE were appearing. Probabilistic versions came first [5, 27], then a nonce-based version [50], then AD finally appeared [49]. Next the MRAE goal—RAE's closest definition counterpart—was defined [51]. The main motivation for that work was to minimize the damage that could be done by nonce-reuse.

Other authors had the same concern but weren't willing to use two-pass schemes. Fleischmann *et. al* [20] built on Bellare *et. al* [3] to define a security notion for online-AE intended to confer some lower level of nonce-reuse misuse-resistance. The approach has gained popularity—many CAESAR submissions follow it, especially after COPA [2] made clear that one could achieve this weakened flavor of nonce-reuse misuse-resistance with a parallelizable scheme. The RAE definition goes a different direction, strengthening instead of weakening the original MRAE definition.

Following up on directions from prior work [10, 20, 21], AE security in the face of decryption-algorithm leakage was studied by Andreeva *et. al* [1] in work concurrent with our own. A principle motivation for those authors has been to express when it is OK for an online decryption algorithm to be incrementally releasing unverified plaintext. For us, this is a direction not taken, for such

leakage can never be generically harmless [47]. In effect, leaking equality of message prefixes is leaking an enormous amount of information.

Ferguson made clear early on that AE algorithms could fail badly when tags are too short [18]. Still, no definitions for AE security were ever offered appropriate to the short-tag setting. But the general concern for making short MACs work well goes back to Black and Cochran [8] and Wang *et. al* [59].

Some examples of using AES4 where AES itself would do include ALRED, LETTERSOUP, MARVIN, and Pelican [15, 16, 57]. These inspired our predilection to cut certain AES rounds even when provable security couldn't promise this was fine. The approach should not be confused with that of Minematsu and Tsunoo [37], where AES4 provably *does* suffice for the protocol devised [37]. The approach leverages the low MEDP for AES4, a line of work culminating in the bound of Keliher and Sui [31].

Many authors have proposed ideas to eliminate use of the inverse-direction of a blockcipher in modes that previously needed this. The method we us in AEZ is inspired by Minematsu's OTR [36].

The CAESAR competition [7], organized by Dan Bernstein, was the proximal motivation to define RAE and to try to develop a nice scheme for achieving it.

B Deferred Proofs

B.1 Proof of Theorem 1

It suffices to show that

$$\left|\Pr[\mathcal{A}^{\mathbf{Ideal}_\Pi} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{PRI}_\Pi} \Rightarrow 1]\right| \le (r^2 + r)/|\Sigma|^{\lambda + m_{\min} + 1} + 2q/|\Sigma|^\lambda .$$

Without loss of generality, assume that $q \le |\Sigma|^{\lambda - 1}$; otherwise the claim is trivial. Consider games G_1–G_4 in Fig. 9. Game G_1 corresponds to game $\mathbf{Ideal}_\Pi$ and game G_4 corresponds to game $\mathbf{PRI}_\Pi$. We explain the game chain up to the terminal one. Game G_2 is identical to game G_1, except that in procedure Enc, it ensures that ciphertexts C are distinct. Partition the encryption queries based on the nonce, the associated data, and the size of the message. Suppose that in game G_1 we have p partitions of size $s_1, \ldots, s_p \ge 1$. Games G_1 and G_2 are identical-until-bad, and thus

$$\left|\Pr[\mathcal{A}^{G_1} \Rightarrow 1] - \Pr[\mathcal{A}^{G_2} \Rightarrow 1]\right| \le \Pr[\mathcal{A}^{G_1} \text{ sets bad}]$$
$$\le \sum_{i=1}^{p} \frac{s_i(s_i - 1)}{|\Sigma|^{m_{\min} + \lambda + 1}}$$
$$= \sum_{i=1}^{p} \frac{(s_i - 1)^2 + (s_i - 1)}{|\Sigma|^{m_{\min} + \lambda + 1}} \le \frac{r^2 + r}{|\Sigma|^{m_{\min} + \lambda + 1}};$$

the last inequality is due to the fact that $(s_1 - 1) + \cdots + (s_p - 1) = r$. Game G_3 is a simplified version of game G_2; the change is conservative. Game G_4 is identical to game G_3, except that in procedure Dec, it samples a λ-character string v and returns a non-$\bot$ answer if $v = \mathbf{0}^\lambda$, where $\mathbf{0}$ is a canonical point in Σ. Let L' be the multiset of $|C|$ in $\mathcal{A}$'s decryption queries in game G_4, and

proc $\mathsf{Enc}(N, A, M)$ Games G_1 / $\boxed{G_2}$
$\ell \leftarrow |M|; \; C \twoheadleftarrow \Sigma^{\ell+\lambda}$
if $C \in \mathit{Ran}_{N,A,\ell}$ **then**
   $\mathsf{bad} \leftarrow \mathsf{true}; \; \boxed{C \twoheadleftarrow \Sigma^{\ell+\lambda} \backslash \mathit{Ran}_{N,A,\ell}}$
$\mathit{Ran}_{N,A,\ell} \leftarrow \mathit{Ran}_{N,A,\ell} \cup \{C\}$
$\mathit{Dom}_{N,A} \leftarrow \mathit{Dom}_{N,A} \cup \{(M, \mathbf{0}^\lambda)\}$
return C

proc $\mathsf{Dec}(N, A, \lambda, C)$
if $|C| < \lambda$ **then return** $\bot$
$\ell \leftarrow |C| - \lambda$
$(M, v) \twoheadleftarrow (\Sigma^\ell \times \Sigma^\lambda) \backslash \mathit{Dom}_{N,A}$
$\mathit{Dom}_{N,A} \leftarrow \mathit{Dom}_{N,A} \cup \{(M, v)\}$
return $\bot$

proc $\mathsf{Enc}(N, A, M)$ Games G_3 / $\boxed{G_4}$
$\ell \leftarrow |M|; \; C \twoheadleftarrow \Sigma^{\ell+\lambda}$
$\mathit{Dom}_{N,A} \leftarrow \mathit{Dom}_{N,A} \cup \{(M, \mathbf{0}^\lambda)\}$
return C

proc $\mathsf{Dec}(N, A, C)$
if $|C| < \lambda$ **then return** $\bot$
$\ell \leftarrow |C| - \lambda$
$(M, v) \twoheadleftarrow (\Sigma^\ell \times \Sigma^\lambda) \backslash \mathit{Dom}_{N,A}$
$\mathit{Dom}_{N,A} \leftarrow \mathit{Dom}_{N,A} \cup \{(M, v)\}$
if $v = \mathbf{0}^\lambda$ **then**
   $\mathsf{bad} \leftarrow \mathsf{true}; \; \boxed{\textbf{return } M}$
return $\bot$

Fig. 9. Games used to prove Theorem 1. Here $\mathbf{0}$ is a canonical element of Σ. Games G_2 and G_4 contain the boxed statements, but games G_1 and G_3 do not.

let L be the multiset $\{\ell \mid \ell \geq 0 \text{ and } \ell + \lambda \in L'\}$. Then

$$\left|\Pr[\mathcal{A}^{G_3} \Rightarrow 1] - \Pr[\mathcal{A}^{G_4} \Rightarrow 1]\right| \leq \Pr[\mathcal{A}^{G_3} \text{ sets } \mathsf{bad}]$$

$$\leq \sum_{\ell \in L} \frac{|\Sigma|^\ell}{|\Sigma|^{\ell+\lambda} - q}$$

$$= \sum_{\ell \in L} \frac{1}{|\Sigma|^\lambda - (q/|\Sigma|^\ell)}$$

$$\leq \sum_{\ell \in L} \frac{1}{|\Sigma|^\lambda - q} \leq \frac{q}{|\Sigma|^\lambda - q} \leq \frac{2q}{|\Sigma|^\lambda};$$

the last inequality is due to the assumption that $q \leq |\Sigma|^{\lambda-1}$. Summing up,

$$\left|\Pr[\mathcal{A}^{\mathbf{Ideal}_\Pi} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{PRI}_\Pi} \Rightarrow 1]\right| \leq \sum_{i=1}^{3} \left|\Pr[\mathcal{A}^{G_i} \Rightarrow 1] - \Pr[\mathcal{A}^{G_{i+1}} \Rightarrow 1]\right|$$

$$\leq \frac{r^2 + r}{|\Sigma|^{\lambda+m_{\min}+1}} + \frac{2q}{|\Sigma|^\lambda}$$

as claimed.

B.2   Proof of Theorem 2

The reduction $\mathcal{R}$ creates from $\mathcal{A}$ the adversary $\mathcal{B}$ as follows. It runs $\mathcal{A}$. When the latter makes an encryption query (N, A, λ, M), if $v(M) = 1$ then the former sends the same query to its encryption oracle and returns the answer to $\mathcal{A}$; otherwise it returns $\bot$. When $\mathcal{A}$ makes a query (N, A, λ, C), adversary $\mathcal{B}$ sends the same query to its decryption oracle to get M. If $|M| = |C| - \lambda$ and $v(M) \neq 1$ then it returns $\mathbf{0} \parallel M$ to $\mathcal{A}$, where $\mathbf{0}$ is a canonical point in Σ. Otherwise, it returns M. Finally, it outputs the same guess as $\mathcal{A}$.

For any query (N, A, λ, C) that it receives, S' stores $(N, A, \lambda, |C|)$ in a set L_λ. It also maintains, for each (N, A, λ, ℓ) in L_λ, a set $B_{N,A,\lambda,\ell}$. Initially, $B_{N,A,\lambda,\ell} = \Sigma^{\ell-\lambda} \backslash \mathcal{M}_v$. The simulator S' works by

running the simulator S. For each query (N, A, λ, C), the simulator S' tosses a biased coin, heads landing land with probability $|B_{N,A,\lambda,\ell}|/(|B_{N,A,\lambda,\ell}| + |\Sigma|^\ell - |\Sigma|^{\ell-\lambda})$, where $\ell = |C|$. If heads shows up, simulator S' will sample $M \twoheadleftarrow B_{N,A,\lambda,\ell}$, remove M from $B_{N,A,\lambda,\ell}$, and return M. Otherwise, it runs S on query (N, A, λ, C) and output whatever S returns. Then

$$\Pr[\mathcal{A}^{\mathbf{REAL}_{\Pi_v,S'}} \Rightarrow 1] = \Pr[\mathcal{B}^{\mathbf{REAL}_{\Pi,S}} \Rightarrow 1], \text{ and}$$
$$\Pr[\mathcal{A}^{\mathbf{RAE}_{\Pi_v,S'}} \Rightarrow 1] = \Pr[\mathcal{B}^{\mathbf{RAE}_{\Pi,S}} \Rightarrow 1] \ .$$

Subtracting, we get $\mathbf{Adv}^{\mathrm{rae}}_{\Pi,S}(\mathcal{B}) = \mathbf{Adv}^{\mathrm{rae}}_{\Pi_v,S'}(\mathcal{A})$.

B.3   Proof of Theorem 3

The reduction $\mathcal{R}$ creates from $\mathcal{A}$ the adversary $\mathcal{B}$ as follows. It runs $\mathcal{A}$. Whenever $\mathcal{A}$ makes an encryption query (N, A, λ, M), adversary $\mathcal{B}$ sends $(T, \mathrm{Encode}_\lambda(M))$ to its left oracle, with $T = (N, A, \lambda)$, and returns the answer to $\mathcal{A}$. When $\mathcal{A}$ makes a decryption query (N, A, λ, C), adversary sends (T, C), with $T = (N, A, \lambda)$ to its right oracle to get answer X, and then returns $\mathrm{Decode}_\lambda(X)$ to $\mathcal{A}$. Finally, it outputs the same guess as $\mathcal{A}$.

Let $\Pi = \mathrm{EtE}[\mathrm{Encode}, \mathrm{Perm}(\mathcal{T}, \Sigma^*)]$ be the variant of $\mathrm{EtE}[\mathrm{Encode}, \widetilde{\mathbb{E}}]$ in which $\widetilde{\mathbb{E}}_K$ is replaced by an ideal tweakable, length-preserving random permutation $\widetilde{\pi} : \mathcal{T} \times \Sigma^* \to \Sigma^*$. It suffices to build a simulator S such that $\mathbf{Adv}^{\mathrm{rae}}_{\Pi,S}(\mathcal{A}) = 0$ for any adversary $\mathcal{A}$. Wlog, assume that the simulator is given only queries (N, A, λ, C) of $\lambda > 0$, and assume that the adversary never repeats its queries. For each $\lambda \in \mathbb{Z}^+$ and each $\ell \geq \lambda$, let $V_{\ell,\lambda} = \Sigma^\ell \backslash R_\lambda$, where R_λ is the range of $\mathrm{Encode}_\lambda(\cdot)$. Then $|V_{\ell,\lambda}| = |\Sigma|^\ell - |\Sigma|^{\ell-\lambda} \geq \frac{1}{2}|\Sigma|^\ell$. For each query (N, A, λ, C) it receives, the simulator S stores $(N, A, \lambda, |C|)$ in a set L_λ. It also maintains, for each $(N, A, \ell, \lambda) \in L_\lambda$, an (initially empty) set $B_{N,A,\ell,\lambda}$. On input (N, A, λ, C), if $|B_{N,A,\ell,\lambda}| < |V_{\ell,\lambda}|$ then S picks $M \twoheadleftarrow V_{\ell,\lambda} \backslash B_{N,A,\ell,\lambda}$, adds M to $B_{N,A,\ell,\lambda}$, and returns M, where $\ell = |C|$. Otherwise, the simulator returns $\bot$. Then $\mathbf{Adv}^{\mathrm{rae}}_{\Pi,S}(\mathcal{A}) = 0$.

What's left is to show how to implement the sampling $M \twoheadleftarrow V_{\ell,\lambda} \backslash B_{N,A,\ell,\lambda}$ efficiently, for $\ell > 5$. If $|V_{\ell,\lambda} \backslash B_{N,A,\ell,\lambda}| > \frac{1}{4}|\Sigma|^\ell$ then S keeps sampling in Σ^ℓ until it hits a point $M \in V_\ell \backslash B_{N,A,\ell}$. The expected number of sampling operations is $O(1)$, and each sampling runs in $O(\ell + \lambda)$ time. The set $B_{N,A,\ell,\lambda}$ is implemented via a hash, and thus the amortized cost of checking if $M \in V_{\ell,\lambda} \backslash B_{N,A,\ell,\lambda}$ is $O(\ell+\lambda)$. At the moment $|V_{\ell,\lambda} \backslash B_{N,A,\ell,\lambda}| = \frac{1}{4}|\Sigma|^\ell$, the simulator stores the elements of $V_{\ell,\lambda} \backslash B_{N,A,\ell,\lambda}$ in an array $H_{N,A,\ell,\lambda}$, and then performs a Fisher–Yates shuffle [19] (commonly known as Knuth shuffle) over $H_{N,A,\ell,\lambda}$. The running time of this shuffling is $O(|\Sigma|^\ell \cdot (\ell+\lambda))$, but it's done after $|V_{\ell,\lambda}| - \frac{1}{4}|\Sigma|^\ell \geq \frac{1}{4}|\Sigma|^\ell$ invocations of S, and thus the amortized cost is only $O(\ell+\lambda)$. If $|V_{\ell,\lambda} \backslash B_{N,A,\ell,\lambda}| \leq \frac{1}{4}|\Sigma|^\ell$, since the array $H_{N,A,\ell}$ is already built, S picks $M \leftarrow H_{N,A,\ell,\lambda}[m]$, where $m = \lfloor \frac{1}{4}|\Sigma|^\ell \rfloor + 1 - |V_\ell \backslash B_{N,A,\ell}|$.

B.4   Proof of Theorem 4

The reduction $\mathcal{R}$ creates from adversary $\mathcal{A}$ the adversary $\mathcal{B}$ as follows. For each encryption query M (respectively, decryption query C) of $\mathcal{A}$, adversary $\mathcal{B}$ runs the encryption of AEZ-core on M (respectively, decryption of AEZ-core on C), but each call to $f_K(T, x)$ is replaced by querying (T, x) to the oracle. It then outputs the same guess as $\mathcal{A}$.

Let $\mathrm{Func}(\mathcal{T}, n)$ be the set of all functions $\rho : \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$. Let AEZ-core[$\mathrm{Func}(\mathcal{T}, n)$] denote the ideal variant of AEZ-core[$\mathbb{E}$], where f_K is replaced by $\rho \twoheadleftarrow \mathrm{Func}(\mathcal{T}, n)$. It suffices to show that

$$\mathbf{Adv}^{\pm\mathrm{prp}}_{\mathrm{AEZ\text{-}core}[\mathrm{Func}(\mathcal{T},n)]}(\mathcal{A}) \leq \frac{2\sigma^2}{2^n} \tag{1}$$

We now prove claim (1) for an even computationally unbounded adversary $\mathcal{A}$. Without loss of generality, assume that $\mathcal{A}$ is deterministic. Assume that $\sigma \leq 2^n$; otherwise the theorem is trivial. Assume further that $\mathcal{A}$ doesn't make redundant queries: it never repeats a prior query, once it makes an encryption query (T, M) to get C then it won't query (T, C) to the decryption oracle, and likewise, once it makes a decryption query (T, C) to get M then it won't query (T, M) to the encryption oracle. Let $q \leq 2\sigma$ be the number of queries of $\mathcal{A}$ and let q_i be the number of queries that have at least $2i + 2$ full blocks. We claim that $2\sum_{i \in \mathbb{N}} q_i \leq \sigma$. Let $\mathcal{P}$ be a $q \times \sigma$ matrix. For each $j \leq q$, let ℓ_j be the length of the ith query and color the first $2 \cdot \lfloor \ell_j/2n \rfloor$ entries of the jth column of $\mathcal{P}$. Then the total number of colored entries in $\mathcal{P}$ is

$$\sum_{j=1}^{q} 2 \cdot \left\lfloor \frac{\ell_j}{2n} \right\rfloor \leq \sum_{j=1}^{q} \left\lceil \frac{\ell_j}{n} \right\rceil = \sigma \ .$$

On the other hand, each q_i is the number of colored entries in the $(2i + 1)$th row of $\mathcal{P}$, and also the number of colored entries in the $(2i + 2)$th row. Hence $2\sum_{i \in \mathbb{N}} q_i$ is the number of colored entries of $\mathcal{P}$, justifying the claim above.

Consider games G_1 and G_2 in Fig. 10. Adversary can call $\text{ENC}(\cdot)$ for encryption queries, and $\text{DEC}(\cdot)$ for decryption queries. Game G_1 is the random game: ENC and DEC implements π and π^{-1}, where π is an ideal random length-preserving permutation on $\{0, 1\}^{\geq 2n}$. In game G_2, for each $\text{ENC}(M)$ query, we output a ciphertext $C = (C_1, C_1', \ldots, C_m, C_m', C_\mathsf{u}, C_\mathsf{v}, C_\mathsf{x}, C_\mathsf{y})$ uniformly from $\{0, 1\}^{|M|} \backslash Ran$, where Ran is the set of all prior ciphertexts, that is subject to the following conditions:

- $C_i' \neq M_i'$ and C_i' is also different from $\tilde{M}_i'$ and $\tilde{C}_i'$ produced by any prior $(\tilde{M}, \tilde{C})$, for every $i \leq m$.
- $C_\mathsf{y} \neq M_\mathsf{y}$ and C_y is also different from $\tilde{M}_\mathsf{y}$ and $\tilde{C}_\mathsf{y}$ produced by any prior $(\tilde{M}, \tilde{C})$.

Likewise, for each $\text{DEC}(C)$ query, we output a message $M = (M_1, M_1', \ldots, M_m, M_m', M_\mathsf{u}, M_\mathsf{v}, M_\mathsf{x}, M_\mathsf{y})$ uniformly from $\{0, 1\}^{|C|} \backslash Dom$, where Dom is the set of all prior messages, that is subject to the conditions that:

- $M_i' \neq C_i'$ and M_i' is also different from $\tilde{M}_i'$ and $\tilde{C}_i'$ produced by any prior $(\tilde{M}, \tilde{C})$, for every $i \leq m$.
- $M_\mathsf{y} \neq C_\mathsf{y}$ and M_y is also different from $\tilde{M}_\mathsf{y}$ and $\tilde{C}_\mathsf{y}$ produced by any prior $(\tilde{M}, \tilde{C})$.

Since the two games G_1 and G_2 are identical-until-bad, and thus

$$\Pr[\mathcal{A}^{G_1} \Rightarrow 0] - \Pr[\mathcal{A}^{G_2} \Rightarrow 0] \leq \Pr[\mathcal{A}^{G_1} \text{ sets bad}] \ .$$

We now bound the chance that game G_1 sets bad. Consider a query of the adversary. Due to symmetry, without loss of generality, suppose that this is an encryption query $\text{ENC}(M)$. Recall that this query triggers bad to true only if when we sample $C \leftarrow \{0, 1\}^{|M|} \backslash Ran$, the ciphertext C violates the conditions described above. Had we instead sampled $C \leftarrow \{0, 1\}^{|M|}$, the chance that this query sets bad to true would have been higher, because for any $C' \in Ran$, setting $C \leftarrow C'$ will certainly set bad to true. Thus in bounding $\Pr[\mathcal{A}^{G_1} \text{ sets bad}]$, we can pretend that ciphertexts C in encryption queries are sampled uniformly from $\{0, 1\}^{|M|}$, and likewise, in decryption queries, messages M are sampled uniformly from $\{0, 1\}^{|C|}$. Hence

$$\Pr[\mathcal{A}^{G_1} \text{ sets bad}] \leq \sum_{i \in \mathbb{N}} \frac{2q_i(2q_i - 1)}{2^{n+1}} \leq \frac{2q}{2^n} \sum_{i \in \mathbb{N}} q_i \leq \frac{\sigma q}{2^n} \ .$$

$$
\begin{array}{ll}
\textbf{proc } \text{ENC}(M) & \textbf{proc } \text{DEC}(C) \qquad\qquad \text{Games } G_1 \ / \ \boxed{G_2} \\
\end{array}
$$

proc $\text{ENC}(M)$

$(M_1, M_1', \ldots, M_m, M_m', M_u, M_v, M_x, M_y) \leftarrow M$
$Dom_y \leftarrow Dom_y \cup \{M_y\}; \ C \twoheadleftarrow \{0,1\}^{|M|} \backslash Ran$
$Dom \leftarrow Dom \cup \{M\}; \ Ran \leftarrow Ran \cup \{C\}$
$(C_1, C_1', \ldots, C_m, C_m', C_u, C_v, C_x, C_y) \leftarrow C$
for $i = 1$ **to** m **do**
$\quad Dom_i \leftarrow Dom_i \cup \{M_i'\}$
$\quad$ **if** $C_i' \in Dom_i$ **then**
$\qquad$ bad $\leftarrow$ true; $\boxed{C_i' \twoheadleftarrow \{0,1\}^n \backslash Dom_i}$
$\quad Dom_i \leftarrow Dom_i \cup \{C_i'\}$
if $C_y \in Dom_y$ **then**
$\quad$ bad $\leftarrow$ true; $\boxed{C_y \twoheadleftarrow \{0,1\}^n \backslash Dom_y}$
$C \leftarrow (C_1, C_1', \ldots, C_m, C_m', C_u, C_v, C_x, C_y)$
$Dom_y \leftarrow Dom_y \cup \{C_y\}; \ $ **return** C

proc $\text{DEC}(C)$

$(C_1, C_1', \ldots, C_m, C_m', C_u, C_v, C_x, C_y) \leftarrow C$
$Dom_y \leftarrow Dom_y \cup \{C_y\}; \ M \twoheadleftarrow \{0,1\}^{|C|} \backslash Dom$
$Dom \leftarrow Dom \cup \{M\}; \ Ran \leftarrow Ran \cup \{C\}$
$(M_1, M_1', \ldots, M_m, M_m', M_u, M_v, M_x, M_y) \leftarrow M$
for $i = 1$ **to** m **do**
$\quad Dom_i \leftarrow Dom_i \cup \{C_i'\}$
$\quad$ **if** $M_i' \in Dom_i$ **then**
$\qquad$ bad $\leftarrow$ true; $\boxed{M_i' \twoheadleftarrow \{0,1\}^n \backslash Dom_i}$
$\quad Dom_i \leftarrow Dom_i \cup \{M_i'\}$
if $M_y \in Dom_y$ **then**
$\quad$ bad $\leftarrow$ true; $\boxed{M_y \twoheadleftarrow \{0,1\}^n \backslash Dom_y}$
$M \leftarrow (M_1, M_1', \ldots, M_m, M_m', M_u, M_v, M_x, M_y)$
$Dom_y \leftarrow Dom_y \cup \{M_y\}; \ $ **return** M

Fig. 10. Games G_1 and G_2 in the proof of Theorem 4. Game G_2 includes the corresponding boxed statement, but game G_1 does not.

Let H_2 be the game in which $\mathcal{A}$ is given oracle access to the encryption and decryption of AEZ-core[Func$(\mathcal{T}, n)$]. Because

$$
\begin{aligned}
\mathbf{Adv}^{\pm\text{prp}}_{\text{AEZ-core[Func}(\mathcal{T},n)]}(\mathcal{A}) &= \Pr[\mathcal{A}^{G_1} \Rightarrow 0] - \Pr[\mathcal{A}^{H_2} \Rightarrow 0] \\
&\leq \Pr[\mathcal{A}^{G_1} \Rightarrow 0] - \Pr[\mathcal{A}^{G_2} \Rightarrow 0] + \Pr[\mathcal{A}^{G_2} \Rightarrow 0] - \Pr[\mathcal{A}^{H_2} \Rightarrow 0],
\end{aligned}
$$

it suffices to prove that

$$
\Pr[\mathcal{A}^{G_2} \Rightarrow 0] - \Pr[\mathcal{A}^{H_2} \Rightarrow 0] \leq \frac{2\sigma^2 - \sigma q}{2^n} \ . \tag{2}
$$

Since $\mathcal{A}$ is deterministic, for any $i \geq 0$, the $(i+1)$th query of $\mathcal{A}$ is completely determined from the transcript $(M^1, C^1, cmd_1), \ldots, (M^i, C^i, cmd_i)$ of the interaction between $\mathcal{A}$ and the oracles up to the ith queries, and $\mathcal{A}$'s guess is uniquely determined from the full transcript $(M^1, C^1, cmd_1), \ldots, (M^q, C^q, cmd_q)$, with each $cmd_j \in \{\text{'enc'}, \text{'dec'}\}$. Let $\mathcal{S}$ be the set of all possible transcripts in game G_2 that makes $\mathcal{A}$ output 0. For each $s \in \mathcal{S}$, let $\ell(s)$ be the total length of $\mathcal{A}$'s queries, according to s. In game G_2 the chance that s is produced is at least $2^{-\ell(s)}$, and thus

$$
\sum_{s \in \mathcal{S}} 2^{-\ell(s)} \leq 1
$$

because the left-hand side doesn't exceed the probability that $\mathcal{A}$ outputs 0 in game G_2. On the other hand, the chance that $s = (M^1, C^1, cmd_1), \ldots, (M^q, C^q, cmd_q)$ is produced in game G_2 is at most

$$
\prod_{i=0}^{q-1} \frac{1}{2^{|M^i|} - \sigma/2} \leq 2^{-\ell(s)} \prod_{i=0}^{q-1} \frac{1}{1 - \sigma/2^{2n+1}} \leq \frac{2^{-\ell(s)}}{1 - \sigma q/2^{2n+1}} \leq 2^{-\ell(s)} \cdot \left(1 + \frac{\sigma q}{2^{2n}}\right) \leq 2^{-\ell(s)} \cdot \left(1 + \frac{\sigma q}{2^n}\right),
$$

where the second inequality is due to the fact that $(1 - x_1) \cdots (1 - x_q) \geq 1 - (x_1 + \cdots + x_q)$ for any $0 \leq x_1, \ldots, x_q \leq 1$. We claim that for any $s \in \mathcal{S}$, the chance that s is produced in game H_2 is

proc VALIDATE(M, C, cmd)
$(M_1, M_1', \ldots, M_m, M_m', M_\mathsf{u}, M_\mathsf{v}, M_\mathsf{x}, M_\mathsf{y}) \leftarrow M; \ (C_1, C_1', \ldots, C_m, C_m', C_\mathsf{u}, C_\mathsf{v}, C_\mathsf{x}, C_\mathsf{y}) \leftarrow C; \ X, Y \leftarrow 0^n$
for $i = 1$ **to** m **do**
  if $cmd = $ 'enc' **then**
    $V_i \leftarrow \rho_{\mathsf{a},i}(M_i') \oplus M_i$
    if $(H[i, M_i M_i'] = \bot) \wedge (V_i \in Dom_\mathsf{a})$ **then** bad $\leftarrow$ true **else** $Dom_\mathsf{a} \leftarrow Dom_\mathsf{a} \cup \{V_i\}$
    $H[i, M_i M_i'] \leftarrow X_i \leftarrow \rho_\mathsf{a}(V_i) \oplus M_i'; \ H[i, C_i C_i'] \leftarrow Y_i \leftarrow X_i \oplus V_i \oplus \rho_{\mathsf{a},i}(C_i') \oplus C_i; \ U_i \leftarrow \rho_{\mathsf{a},i}(C_i') \oplus C_i$
    if $U_i \in Dom_\mathsf{a}$ **then** bad $\leftarrow$ true **else** $Dom_\mathsf{a} \leftarrow Dom_\mathsf{a} \cup \{U_i\}$
  else
    $U_i \leftarrow \rho_{\mathsf{a},i}(C_i') \oplus C_i$
    if $(H[i, C_i C_i'] = \bot) \wedge (U_i \in Dom_\mathsf{a})$ **then** bad $\leftarrow$ true **else** $Dom_\mathsf{a} \leftarrow Dom_\mathsf{a} \cup \{U_i\}$
    $H[i, C_i C_i'] \leftarrow Y_i \leftarrow \rho_\mathsf{a}(U_i) \oplus C_i'; \ H[i, M_i M_i'] \leftarrow X_i \leftarrow Y_i \oplus U_i \oplus \rho_{\mathsf{a},i}(M_i') \oplus M_i; \ V_i \leftarrow \rho_{\mathsf{a},i}(M_i') \oplus M_i$
    if $V_i \in Dom_\mathsf{a}$ **then** bad $\leftarrow$ true **else** $Dom_\mathsf{a} \leftarrow Dom_\mathsf{a} \cup \{V_i\}$
  $X \leftarrow X \oplus X_i; \ Y \leftarrow Y \oplus Y_i$
if $M_\mathsf{u} \neq \varepsilon$ **then** $X \leftarrow X \oplus \rho_\mathsf{u}(\mathrm{pad}(M_\mathsf{u})); \ Y \leftarrow Y \oplus \rho_\mathsf{u}(\mathrm{pad}(C_\mathsf{u}))$
if $|M_\mathsf{u}| = n$ **then** $X \leftarrow X \oplus \rho_\mathsf{u}(\mathrm{pad}(M_\mathsf{v})); \ Y \leftarrow Y \oplus \rho_\mathsf{v}(\mathrm{pad}(C_\mathsf{v}))$
$A \leftarrow \rho_\mathsf{x}(M_\mathsf{y}) \oplus X \oplus M_\mathsf{x}; \ B \leftarrow \rho_\mathsf{y}(C_\mathsf{y}) \oplus Y \oplus C_\mathsf{x}; \ S \leftarrow A \oplus B$
if $(A \in Dom_\mathsf{xx}) \vee (B \in Dom_\mathsf{yy})$ **then** bad $\leftarrow$ true
$Dom_\mathsf{xx} \leftarrow Dom_\mathsf{xx} \cup \{A\}; \ Dom_\mathsf{yy} \leftarrow Dom_\mathsf{yy} \cup \{B\}$
if $S \in Set$ **then** bad $\leftarrow$ true **else** $Set \leftarrow Set \cup \{S\}$
//Constraints for ρ if bad and coll are not set
// $\rho_\mathsf{xx}(A) = M_\mathsf{y} \oplus B; \ \rho_\mathsf{yy}(B) = C_\mathsf{y} \oplus A$
// $\rho_{\mathsf{aa},i}(S) = V_i \oplus Y_i$ for every $i = 1, \ldots, m$
// $\rho_\mathsf{vv}(S)[1..|M_\mathsf{v}|] = C_\mathsf{v} \oplus M_\mathsf{v}$ if $M_\mathsf{v} \neq \varepsilon$
// $\rho_\mathsf{uu}(S)[1..|M_\mathsf{u}|] = C_\mathsf{u} \oplus M_\mathsf{u}$ if $M_\mathsf{u} \neq \varepsilon$
// $\rho_\mathsf{a}(U_i) = Y_i \oplus C_i'$ if $cmd = $ 'enc', and $\rho_\mathsf{a}(V_i) = X_i \oplus M_i'$ otherwise, for every $i = 1, \ldots, m$

Fig. 11. Procedure VALIDATE **to determine if ρ is valid.** If ρ is indeed valid, the commented code dictates the constraints on ρ so that the real game results in the given transcript $(M^1, C^1, cmd_i), \ldots, (M^q, C^q, cmd_q)$. Sets are initialized to $\emptyset$, and the flag bad is initialized to false. Function pad takes as input a string x with length $|x| \leq n$, and then returns x if $|x| = n$, and returns $x10^{n-|x|}$ otherwise.

at least $2^{-\ell(s)}(1 - (2\sigma^2 - 2\sigma q)/2^n)$, and thus

$$\Pr[\mathcal{A}^{G_2} \Rightarrow 0] - \Pr[\mathcal{A}^{H_2} \Rightarrow 0] \leq \sum_{s \in \mathcal{S}} 2^{-\ell(s)} \cdot \left(1 + \frac{\sigma q}{2^n}\right) - 2^{-\ell(s)}\left(1 - \frac{2\sigma^2 - 2\sigma q}{2^n}\right)$$
$$\leq \sum_{s \in \mathcal{S}} 2^{-\ell(s)} \cdot \frac{2\sigma^2 - \sigma q}{2^n}$$

establishing (2). To justify the claim above, consider an arbitrary transcript $s \in \mathcal{S}$. Let $s = (M^1, C^1, cmd_1), \ldots, (M^q, C^q, cmd_q)$. Recall that in game H_2, we use $\rho \twoheadleftarrow \mathrm{Func}(\mathcal{T}, n)$ instead of f_K. We say that the function ρ is *valid* if it doesn't trigger the flag bad to true when we run VALIDATE$(M^1, C^1, cmd_1), \ldots,$ VALIDATE(M^q, C^q, cmd_q), where the code of VALIDATE is specified in Fig. 11. If bad is not set then the commented code in VALIDATE specifies the constraints for ρ so that game H_2 results in the given transcript $(M^1, C^1, cmd_i), \ldots, (M^q, C^q, cmd_q)$. If ρ is valid then the constraints never give conflicting assignments, and thus the conditional probability that ρ satisfies the prescribed constraints, given that ρ is valid, is at least $2^{-\ell(s)}$. Hence the chance that H_2 results in the transcript s is at least $2^{-\ell(s)}$ times the probability that $\rho \twoheadleftarrow \mathrm{Func}(\mathcal{T}, n)$ is valid. In other words, it suffices to prove that $\rho \twoheadleftarrow \mathrm{Func}(\mathcal{T}, n)$ sets bad to true with probability at most $(2\sigma^2 - 2\sigma q)/2^n$.

CODE INTERPRETATION. In the code, we maintain a map $H : \mathbb{N} \times \{0,1\}^n \to \{0,1\}^n$ such that $H[i, M_i M_i'] \leftarrow X_i$ and $H[i, C_i C_i'] \leftarrow Y_i$. Wlog, consider the case $cmd = $ 'enc'. First, the assignment $H[i, M_i M_i'] \leftarrow X_i$ doesn't overwrite an existing point in H, because if there is a prior query $(\tilde{M}, \tilde{C}, \widetilde{cmd})$ such that $\tilde{M}_i \tilde{M}_i' = M_i M_i'$ then $X_i = \tilde{X}_i$, and if there is a prior query $(\tilde{M}, \tilde{C}, \widetilde{cmd})$ such that $\tilde{C}_i \tilde{C}_i' = M_i M_i'$ then $X_i = \tilde{Y}_i$. When we run the assignment $H[i, M_i M_i'] \leftarrow X_i \leftarrow \rho_{\mathsf{a}}(V_i) \oplus M_i'$, if bad is not set and $H[i, M_i M_i'] = \bot$ then ρ_{a} has never been evaluated on V_i, and thus X_i is a fresh random string. Next, the assignment $H[i, C_i C_i'] \leftarrow Y_i$ doesn't overwrite an existing point in H, as $C_i' \neq M_i'$ and $C_i' \neq \tilde{C}_i'$ for any prior query $(\tilde{M}, \tilde{C}, \widetilde{cmd})$. Moreover, when we run the assignment $H[i, C_i C_i'] \leftarrow Y_i \leftarrow X_i \oplus V_i + \rho_{\mathsf{a},i}(C_i') \oplus C_i$, as $\rho_{\mathsf{a},i}$ has never been evaluated on C_i', the string Y_i is a fresh random string. In other words, H is a random oracle, and $X_i \leftarrow H[i, M_i M_i']$ and $Y_i \leftarrow H[i, C_i C_i']$ for any query (M, C) and any $i \leq \lfloor |M|/2n \rfloor - 2$.

A TECHNICAL LEMMA . We show that for any two queries (M, C, cmd) and $(\tilde{M}, \tilde{C}, \widetilde{cmd})$, it's unlikely that $X \oplus M_{\mathsf{x}} = \tilde{X} \oplus \tilde{M}_{\mathsf{x}}$, unless $M[1..|M| - n] = \tilde{M}[1..|\tilde{M}| - n]$. By symmetry, it's unlikely that $Y \oplus C_{\mathsf{x}} = \tilde{Y} \oplus \tilde{C}_{\mathsf{x}}$, unless $C[1..|C| - n] = \tilde{C}[1..|\tilde{C}| - n]$.

Lemma 1. For any two queries (M, C, cmd) and $(\tilde{M}, \tilde{C}, \widetilde{cmd})$, (a) if $M[1..|M| - n] \neq \tilde{M}[1..|\tilde{M}| - n]$ then $\Pr[X \oplus M_{\mathsf{x}} = \tilde{X} \oplus \tilde{M}_{\mathsf{x}}] \leq 2^{-n}$, and (b) if $C[1..|C| - n] \neq \tilde{C}[1..|\tilde{C}| - n]$ then $\Pr[Y \oplus C_{\mathsf{x}} = \tilde{Y} \oplus \tilde{C}_{\mathsf{x}}] \leq 2^{-n}$.

Proof. We'll give a proof for part (a); part (b) follows due to symmetry. If $M[1..|M| - 2n] = \tilde{M}[1..|\tilde{M}| - 2n]$ then $M_{\mathsf{x}} \neq \tilde{M}_{\mathsf{x}}$ and $X = \tilde{X}$. Hence $X \oplus M_{\mathsf{x}} \neq \tilde{X} \oplus \tilde{M}_{\mathsf{x}}$. Consider the case that $M[1..|M| - 2n] \neq \tilde{M}[1..|\tilde{M}| - 2n]$. Let $m = \lfloor |M|/2n \rfloor - 2$ and $m' = \lfloor |\tilde{M}|/2n \rfloor - 2$. Wlog, assume that $m \geq m'$. We consider the following cases.

CASE 1: $m > m'$. Then X_m is independent of $\tilde{X}$, and thus the chance that $X \oplus M_{\mathsf{x}} = \tilde{X} \oplus \tilde{M}_{\mathsf{x}}$ is at most 2^{-n}.

CASE 2: $m = m'$ and there is an index $i \leq m$ such that $M_i M_i' \neq \tilde{M}_i \tilde{M}_i'$, and thus X_i and $\tilde{X}_i$ are independent. Then $X \oplus M_0' = \tilde{X} \oplus \tilde{M}_0'$ with probability at most 2^{-n}.

CASE 3: $m = m'$ and $M_i M_i' = \tilde{M}_i \tilde{M}_i'$ for every $i \leq m$. If $M_{\mathsf{u}} \neq \tilde{M}_{\mathsf{u}}$ then one of them must be nonempty; wlog, suppose that $M_{\mathsf{u}} \neq \varepsilon$. Then $X_{\mathsf{u}} = \rho_{\mathsf{u}}(M_{\mathsf{u}})$ is independent of $\tilde{X}$, and $X \oplus M_{\mathsf{x}} = \tilde{X} \oplus \tilde{M}_{\mathsf{x}}$ with probability at most 2^{-n}. So suppose that $M_{\mathsf{u}} = \tilde{M}_{\mathsf{u}}$. Then $M_{\mathsf{v}} \neq \tilde{M}_{\mathsf{v}}$, and thus one of them must be nonempty; wlog assume that $M_{\mathsf{v}} \neq \varepsilon$. Then $X_{\mathsf{v}} = \rho_{\mathsf{v}}(M_{\mathsf{v}} 10^*)$ is independent of $\tilde{X}$, and $X \oplus M_{\mathsf{x}} = \tilde{X} \oplus \tilde{M}_{\mathsf{x}}$ with conditional probability at most 2^{-n}. $\square$

ACCOUNTING FOR BAD EVENTS. The flag bad is set only if one of the following happens:

- There are some (M, C, cmd) and $(\tilde{M}, \tilde{C}, \widetilde{cmd})$ such that (1) $M[1..|M| - n] \neq \tilde{M}[1..|\tilde{M}| - n]$, but (2) $X \oplus M_{\mathsf{x}} = \tilde{X} \oplus \tilde{M}_{\mathsf{x}}$. From Lemma 1, this happens with conditional probability at most $q^2/2^{n+1}$.

- There are some (possibly the same) queries (M, C, cmd) and $(\tilde{M}, \tilde{C}, \widetilde{cmd})$ and $i, j \geq 1$ and such that (1) either $i \neq j$ or $M_i M_i' \neq \tilde{M}_j \tilde{M}_j'$ but (2) $\rho_{\mathsf{a},i}(M_i') \oplus M_i = \rho_{\mathsf{a},j}(\tilde{M}_j') \oplus \tilde{M}_j$. Summing over for at most $\sigma - 2q$ pairs $M_i M_i'$ and $\tilde{M}_j \tilde{M}_j'$, this event happens with probability at most $(\sigma - 2q)^2/2^{n+1}$.

- There are some (M, C, cmd) and $(\tilde{M}, \tilde{C}, \widetilde{cmd})$ such that (1) $X \oplus M_{\mathsf{x}} \neq \tilde{X} \oplus \tilde{M}_{\mathsf{x}}$, but (2) $\rho_{\mathsf{x}}(M_{\mathsf{y}}) \oplus X \oplus M_{\mathsf{x}} = \rho_{\mathsf{x}}(\tilde{M}_{\mathsf{y}}) \oplus \tilde{X} \oplus \tilde{M}_{\mathsf{x}}$. This event happens with probability at most $q^2/2^{n+1}$.

- The dual of the bad events above, where M_i becomes C_i, X becomes Y, and so on. This happens with probability at most $(2q^2 + (\sigma - q)^2)/2^{n+1}$.

- There are some (possibly the same) queries (M, C, cmd) and $(\tilde{M}, \tilde{C}, \widetilde{cmd})$ and $i, j \geq 1$ and such that (1) either $i \neq j$ or $M_i M_i' \neq \tilde{C}_j \tilde{C}_j'$ but (2) $\rho_{\mathsf{a},i}(M_i') \oplus M_i = \rho_{\mathsf{a},j}(\tilde{C}_j') \oplus \tilde{C}_j$. Summing over for at most $\sigma - 2q$ strings $M_i M_i'$ at most $\sigma - 2q$) strings $\tilde{C}_j \tilde{C}_j'$, this event happens with probability at most $(\sigma - 2q)^2/2^n$.

- There is a query (M, C, cmd) and a prior query $(\tilde{M}, \tilde{C}, \widetilde{cmd})$ such that $S = \tilde{S}$. As $S = \rho_{\mathsf{x}}(M_{\mathsf{y}}) \oplus M_{\mathsf{x}} \oplus X \oplus \rho_{\mathsf{y}}(C_{\mathsf{y}}) \oplus C_{\mathsf{x}} \oplus Y$, $\tilde{S} = \rho_{\mathsf{x}}(\tilde{M}_{\mathsf{y}}) \oplus \tilde{M}_{\mathsf{x}} \oplus \tilde{X} \oplus \rho_{\mathsf{y}}(\tilde{C}_{\mathsf{y}}) \oplus \tilde{C}_{\mathsf{x}} \oplus \tilde{Y}$, and either $C_{\mathsf{y}} \notin \{M_{\mathsf{y}}, \tilde{M}_{\mathsf{y}}, \tilde{C}_{\mathsf{y}}\}$ or $M_{\mathsf{y}} \notin \{C_{\mathsf{y}}, \tilde{M}_{\mathsf{y}}, \tilde{C}_{\mathsf{y}}\}$, this happens with probability at most $q^2/2^{n+1}$.

Summing up, the chance that bad is set is at most

$$\frac{2.5q^2 + 2(\sigma - 2q)^2}{2^n} = \frac{2\sigma^2 - 8\sigma q + 10.5q^2}{2^n} \leq \frac{2\sigma^2 - 2\sigma q}{2^n};$$

the last inequality is due to the fact that $\sigma \geq 2q$.

B.5   Proof of Theorem 5

The reduction $\mathcal{R}$ creates from $\mathcal{A}$ the adversary $\mathcal{B}$ as follows. It runs $\mathcal{A}$. Each time the latter makes an Enc query, the former runs the encryption procedure of $\mathrm{AEZ}[\widetilde{E}]$, but each $\widetilde{E}_K^T(X)$ is replaced by querying (T, X) to the oracle of $\mathcal{B}$. Likewise, for each Dec query of $\mathcal{A}$, adversary $\mathcal{B}$ runs the decryption procedure of $\mathrm{AEZ}[\widetilde{E}]$, but each $\widetilde{E}_K^T(X)$ is replaced by querying (T, X) to the oracle of $\mathcal{B}$. Finally, $\mathcal{B}$ outputs the same guess as $\mathcal{A}$.

Let $\mathrm{AEZ}[\mathrm{Perm}(\mathcal{T}_{\mathrm{aez}}, 128)]$ be the ideal variant of $\mathrm{AEZ}[\widetilde{E}]$ where $\widetilde{E}$ is replaced by an ideal TBC $\widetilde{\pi} : \mathcal{T}_{\mathrm{aez}} \times \{0,1\}^{128} \to \{0,1\}^{128}$. Let S be the simulator in Theorem 3. It suffices to show that

$$\mathbf{Adv}^{\mathrm{rae}}_{\mathrm{AEZ}[\mathrm{Perm}(\mathcal{T}_{\mathrm{aez}}, 128)], S}(\mathcal{A}) \leq \frac{3s^2}{2^{128}} \ . \tag{3}$$

Let H be the ideal variant of AEZ-hash in which E is replaced by $\widetilde{\pi}$. Note that s is the sum of the number of processed blocks from two sources: AEZ-core, and AEZ-prf; let these numbers be σ and t respectively. Equation (3) is obtained via the following results:

- Theorem 3 in Section 5 to justify the RAE security from the EtE construction,
- Theorem 4 in Section 6 to justify that AEZ-core is a secure length-preserving strong tweakable-PRP on $\mathrm{BYTE}^{\geq 32}$, contributing $2\sigma^2/2^{128}$ to the total advantage,
- Theorem 7 in Appendix C for tweaking AEZ-core. This part contributes at most $\frac{(\sigma/4)^2}{2^{128}} + \mathbf{Adv}^{\mathrm{axu}}_H(\sigma) \leq \frac{0.1\sigma^2}{2^{128}} + \mathbf{Adv}^{\mathrm{axu}}_H(\sigma)$ to the total advantage.

What remains is to bound (i) the AXU advantage of H, (ii) the PRF-advantage of the ideal variant of AEZ-prf and (iii) the advantage in distinguishing AEZ-core on $\widetilde{\pi}$ and AEZ-core on an ideal random function $f : \mathcal{T}_{\mathrm{aez}} \times \{0,1\}^{128} \to \{0,1\}^{128}$. For (i), note that $\mathrm{Coll}_H(m, m') \leq 1/(2^{128} - 1)$ for every $m, m' \in \mathbb{N}$, and thus

$$\mathbf{Adv}^{\mathrm{axu}}_H(z) \leq \frac{0.5z(z-1)}{2^{128} - 1} \leq \frac{0.5z^2}{2^{128}}$$

for any $z \in \{0, \ldots, 2^{128}\}$. For (ii), the PRF advantage of the ideal variant of AEZ-prf is at most $t^2/2^{128}$. For (iii), note that $\mathcal{A}$'s queries results in calling $\widetilde{\pi}^{(0,0)}$ at most σ times. Suppose that $\mathcal{A}$'s queries result in calling $\widetilde{\pi}$ on tweaks $T_1, \ldots, T_m \neq (0,0)$ with $q_1, \ldots, q_m$ queries respectively. Note that $q_1, \ldots, q_m \leq \sigma/2$ and $q_1 + \cdots + q_m \leq 1.5\sigma$. By the PRP/PRF switching lemma, the advantage in distinguishing $\widetilde{\pi}$ and f on tweaks $T_1, \ldots, T_m$ with $q_1, \ldots, q_m$ queries respectively, and on tweak $(0,0)$ with σ queries, is at most

$$\frac{\sigma^2}{2 \cdot 2^{128}} + \sum_{i=1}^{m} \frac{q_i^2}{2 \cdot 2^{128}} \leq \frac{\sigma^2}{2 \cdot 2^{128}} + \sum_{i=1}^{m} \frac{q_i \sigma}{4 \cdot 2^{128}} \leq \frac{0.9\sigma^2}{2^{128}} \ .$$

Summing up,

$$\mathbf{Adv}^{\mathrm{rae}}_{\mathrm{AEZ}[\mathrm{Perm}(\mathcal{T}_{\mathrm{aez}}, 128)], S}(\mathcal{A}) \leq \frac{3.5\sigma^2 + t^2}{2^{128}} \leq \frac{3.5s^2}{2^{128}}$$

as claimed.

B.6   Proof of Theorem 6

Let q be the number of calls to E. Then $q \leq 2.5s - 2$. Let $\pi : \{0,1\}^{128} \to \{0,1\}^{128}$ be an ideal random permutation, and let $\tilde{\pi}^{i,j}(X) = \pi(X \oplus (i+1)I \oplus jJ)$ where $I = \pi(\mathbf{0})$ and $J = \pi(\mathbf{1})$ for every $(i,j) \in \mathcal{T}_{\mathrm{aez}}$. From Theorem 5, what remains is to show that, $\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{\pi}}(\mathcal{D}) \leq 1.5(q+2)^2/2^{128}$ for any adversary $\mathcal{D}$ that makes at most q queries, which is implied by [49, Theorem 7]. (The theorem statement of [49, Theorem 7] states that $\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{\pi}}(\mathcal{D}) \leq 4.5q^2/2^{128}$, but in our setting, the proof of [49, Theorem 7] actually yields $\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{\pi}}(\mathcal{D}) \leq 0.5q^2/2^{128} + (q+2)^2/2^{128}$ and then upperbounds $q + 2 \leq 2q$.)

C   Tweaking a Wide-Block Blockcipher

In Section 5 we have shown that RAE can be achieved from the EtE construction using a an arbitrary-input-length tweakable blockcipher with tweak space $\mathcal{T} = \Sigma^* \times \Sigma^* \times \mathbb{N}$. In AEZ, we consider $\Sigma = \mathrm{BYTE}$ and then build an arbitrary-input-length tweakable blockcipher but without the tweaks, which is AEZ-core for messages at least 32 bytes, and AEZ-tiny for shorter messages. We now show how to tweak this arbitrary-input-length tweakable blockcipher. We'll use the XEX processing [34, 49] to tweak AEZ-core. Recall that AEZ-tiny is a Feistel network whose round functions are derived from a TBC $\mathrm{E} : \mathcal{T}' \times \{0,1\}^{128} \to \{0,1\}^{128}$. To tweak AEZ-tiny, we'll expand the tweak space of E to $\mathcal{T} \times \mathcal{T}'$ via the XE processing [34, 49]. (Since there's no query to E_K^{-1}, we don't need XEX processing.) In both cases, we'll hash the tweak $T \in \mathcal{T}$ via an almost-xor universal (AXU) hash. To save the context size, AEZ-tiny and AEZ-core will use the same key for the AXU hash. To justify this sharing, we'll extend the $\pm\widetilde{\mathrm{prp}}$ security as follows. The tweak space $\mathcal{T}$ is partitioned into two sets $\mathcal{T}^{\pm}$ and $\mathcal{T}^+$. The adversary can only make decryption queries for tweaks $T \in \mathcal{T}^{\pm}$. This definition, put forward by Krovetz and Rogaway [32], aims at minimizing the overhead of tweaking a blockcipher. Below, we'll extend the classic notion of AXU hash for a possibly infinite message space.

AXU HASH. Let $n \geq 1$ be an integer and let $H \colon \mathcal{K} \times \mathcal{U} \to \{0,1\}^n$. For our application, each $U \in \mathcal{U}$ will be a vector whose components are string vectors. We write $\|U\|$ to denote the length of the string encoding of U. For each $m, m' \in \mathbb{N}$, define $\mathrm{Coll}_H(m, m')$ as the maximum, taken over all $y \in \{0,1\}^n$ and distinct $U, U' \in \mathcal{U}$ such that $\lceil \|U\|/n \rceil = m$ and $\lceil \|U'\|/n \rceil = m'$, of $\Pr[K \twoheadleftarrow \mathcal{K} \colon H_K(U) \oplus H_K(U') = y]$. If there are no such U and U' then $\mathrm{Coll}_H(m, m') = 0$. For each $s \in \mathbb{N}$, let

$$\mathbf{Adv}_H^{\mathrm{axu}}(s) = \max_{\substack{q, m_1, \ldots, m_q \\ m_1 + \cdots + m_q = s}} \left\{ \sum_{1 \leq i < j \leq q} \mathrm{Coll}_H(m_i, m_j) \right\}$$

EXPANDING THE TWEAK SPACE. Let $n \geq 1$ be an integer and $\mathcal{M} \subseteq \{0,1\}^*$ be a set such that (i) $|x| \geq n$ for every $x \in \mathcal{M}$, and (ii) if $x \in \mathcal{M}$ then $\{0,1\}^{|x|} \subseteq \mathcal{M}$. Suppose that we have an AXU hash $H \colon \mathcal{K} \times \mathcal{U} \to \{0,1\}^n$ and a tweakable blockcipher $\widetilde{\mathbb{E}} \colon \mathcal{K}' \times \mathcal{V} \times \mathcal{M} \to \mathcal{M}$, where $\mathcal{V}$ is partitioned to $\mathcal{V}^\pm$ and $\mathcal{V}^+$. Consider the following way $\mathrm{XT}[H, \widetilde{\mathbb{E}}]$ to build a tweakable blockcipher $\widetilde{\mathrm{E}} \colon (\mathcal{K} \times \mathcal{K}') \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$, with $\mathcal{T} = \mathcal{U} \times \mathcal{V}$ and $\mathcal{T}^\pm = \mathcal{U} \times \mathcal{V}^\pm$ and $\mathcal{T}^+ = \mathcal{U} \times \mathcal{V}^+$. On input M and tweak $T = (U, V)$, let $S = 0^m \| H_K(U) \| 0^{|M|-m-n}$, where $m = \max\{0, |M| - 2n\}$. If $T \in \mathcal{T}^\pm$ then $\widetilde{\mathrm{E}}_{K,K'}^T(M) = S \oplus \widetilde{\mathbb{E}}_{K'}^V(M \oplus S)$, otherwise $\widetilde{\mathrm{E}}_{K,K'}^T(M) = \widetilde{\mathbb{E}}_{K'}^V(M \oplus S)$. The following says that $\widetilde{\mathrm{E}}$ achieves the extended $\pm\widetilde{\mathrm{prp}}$ security. This result is close to one of Liskov, Rivest, and Wagner [34].

We can justify the key sharing of AEZ-tiny and AEZ-core by applying Theorem 7 for $n = 128$, $\mathcal{M} = \mathrm{BYTE}^{16} \cup \mathrm{BYTE}^{\geq 32}$, $\mathcal{V}^+ = \{0\} \times \mathcal{N} \times \mathcal{A}$, $\mathcal{V}^\pm = \{(1,0)\}$, $\widetilde{\mathbb{E}}_K^{(1,0)}$ as AEZ-core, and $\widetilde{\mathbb{E}}_K^{(0,T)}$ as the TBC implementing the round functions of AEZ-tiny with tweak T.

Theorem 7. Let $\widetilde{\mathbb{E}} \colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ be a tweakable blockcipher and and $H \colon \mathcal{K}' \times \mathcal{T} \to \{0,1\}^n$ be an AXU hash. There is an explicitly given reduction $\mathcal{R}$ with the following property. For any adversary $\mathcal{A}$, adversaries $\mathcal{B} = \mathcal{R}(\mathcal{A})$ satisfies

$$\mathbf{Adv}_{\mathrm{XT}[H,\widetilde{\mathbb{E}}]}^{\pm\widetilde{\mathrm{prp}}}(\mathcal{A}) \leq q^2/2^n + \mathbf{Adv}_{\widetilde{\mathbb{E}}}^{\pm\widetilde{\mathrm{prp}}}(\mathcal{B}) + \mathbf{Adv}_H^{\mathrm{axu}}(s),$$

where s is the total block length of the string encodings of the tweaks in $\mathcal{A}$'s queries and q is the number of $\mathcal{A}$'s queries. Adversary $\mathcal{B}$ has about the same running time as $\mathcal{A}$ plus the time using H to process the tweaks in $\mathcal{A}$'s queries. It makes q queries as $\mathcal{A}$, and the total block length of its queries is the same as that of $\mathcal{A}$.

Proof. The reduction $\mathcal{R}$ creates from $\mathcal{A}$ the adversary $\mathcal{B}$ as follows. It samples $K_1 \twoheadleftarrow \mathcal{K}_1$ and runs $\mathcal{A}$. For each encryption query (T, M) of $\mathcal{A}$, with $T = (U, V)$, adversary $\mathcal{B}$ computes $S \leftarrow 0^m \| H_K(U) \| 0^{|M|-m-n}$, where $m = \max\{0, |M| - 2n\}$, and queries $(V, M \oplus S)$ to its left oracle to gets answer C. It returns $C \oplus S$ to $\mathcal{A}$ if $T \in \mathcal{T}^\pm$, and returns C to $\mathcal{A}$ otherwise. For each decryption query (T, C) of $\mathcal{A}$, with $T = (U, V)$, adversary $\mathcal{B}$ computes $S \leftarrow 0^m \| H_K(U) \| 0^{|C|-m-n}$, where $m = \max\{0, |C| - 2n\}$, and queries $(V, C \oplus S)$ to its right oracle to gets answer M, and returns $M \oplus S$ to $\mathcal{A}$. Finally, it outputs the same guess as $\mathcal{A}$.

Let $\mathrm{Perm}(\mathcal{V}, \mathcal{M})$ denote the set of all functions $f \colon \mathcal{V} \times \mathcal{M} \to \mathcal{M}$ such that each $f(V, \cdot)$ is a length-preserving permutation on $\mathcal{M}$, for every $V \in \mathcal{V}$. Let $\mathrm{XT}[H, \mathrm{Perm}(\mathcal{V}, \mathcal{M})]$ denote the variant of $\mathrm{XT}[H, \mathbb{E}]$ in which $\mathbb{E}_{K'}$ is replaced by an ideal $\widetilde{\pi} \twoheadleftarrow \mathrm{Perm}(\mathcal{V}, \mathcal{M})$. It suffices to show that

$$\mathbf{Adv}_{\mathrm{XT}[H,\mathrm{Perm}(\mathcal{V},\mathcal{M})]}^{\pm\widetilde{\mathrm{prp}}}(\mathcal{A}) \leq q^2/2^n + \mathbf{Adv}_H^{\mathrm{axu}}(s) \ .$$

Consider games G_1–G_6 in Fig. 12. In each game, adversary $\mathcal{A}$ interrogates oracle ENC for encryption queries, and oracle DEC for decryption queries. Without loss of generality, assume that the adversary

proc $\textsc{Enc}(T, M)$

$(U, V) \leftarrow T$; $m \leftarrow \max\{0, |M| - 2n\}$
$S \leftarrow 0^m \parallel H_K(U) \parallel 0^{|M|-m-n}$
$x \leftarrow M \oplus S$; $y \twoheadleftarrow \{0,1\}^{|M|}$
if $T \in \mathcal{T}^{\pm}$ **then** $Mask \leftarrow S$ **else** $Mask \leftarrow 0^{|M|}$
if $x \in Dom(V)$ **then**
    $y \leftarrow P[V, x]$; **return** $y \oplus Mask$
if $y \in Ran(V)$ **then**
    $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{y \twoheadleftarrow \{0,1\}^{|M|} \backslash Ran(V)}$
$P[V, x] \leftarrow y$; $Dom(V) \leftarrow Dom(V) \cup \{x\}$
$R[V, y] \leftarrow x$; $Ran(V) \leftarrow Ran(V) \cup \{y\}$
return $y \oplus Mask$

proc $\textsc{Dec}(T, C)$ Games $\boxed{G_1}$ / G_2

$(U, V) \leftarrow T$; $m \leftarrow \max\{0, |C| - 2n\}$
$S \leftarrow 0^m \parallel H_K(U) \parallel 0^{|C|-m-n}$
$y \leftarrow C \oplus S$; $x \twoheadleftarrow \{0,1\}^{|C|}$
if $y \in Ran(V)$ **then** $(x \leftarrow R[V, y]$; **return** $x \oplus S)$
if $x \in Dom(V)$ **then**
    $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{x \twoheadleftarrow \{0,1\}^{|C|} \backslash Dom(V)}$
$P[V, x] \leftarrow y$; $Dom(V) \leftarrow Dom(V) \cup \{x\}$
$R[V, y] \leftarrow x$; $Ran(V) \leftarrow Ran(V) \cup \{y\}$
$M \leftarrow x \oplus S$; **return** M

proc $\textsc{Enc}(T, M)$

$(U, V) \leftarrow T$; $m \leftarrow \max\{0, |M| - 2n\}$
$S \leftarrow 0^m \parallel H_K(U) \parallel 0^{|M|-m-n}$
$x \leftarrow M \oplus S$; $C \twoheadleftarrow \{0,1\}^{|M|}$
if $T \in \mathcal{T}^{\pm}$ **then** $Mask \leftarrow S$ **else** $Mask \leftarrow 0^{|M|}$
if $x \in Dom$ **then**
    $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{C \leftarrow P[V, x] \oplus Mask}$
$y \leftarrow C \oplus Mask$; $P[V, x] \leftarrow y$
$Dom(V) \leftarrow Dom(V) \cup \{x\}$
$R[V, y] \leftarrow x$; $Ran(V) \leftarrow Ran(V) \cup \{y\}$
return C

proc $\textsc{Dec}(T, C)$ Games $\boxed{G_3}$ / G_4

$(U, V) \leftarrow T$; $m \leftarrow \max\{0, |C| - 2n\}$
$S \leftarrow 0^m \parallel H_K(U) \parallel 0^{|C|-m-n}$
$y \leftarrow C \oplus S$; $M \twoheadleftarrow \{0,1\}^{|C|}$
if $y \in Ran(V)$ **then** $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{M \leftarrow R[V, y] \oplus S}$
$x \leftarrow M \oplus S$; $P[V, x] \leftarrow y$; $Dom(V) \leftarrow Dom(V) \cup \{x\}$
$R[V, y] \leftarrow x$; $Ran(V) \leftarrow Ran(V) \cup \{y\}$
return M

proc $\textsc{Enc}(T, M)$

$(U, V) \leftarrow T$; $C \twoheadleftarrow \{0,1\}^{|M|}$
if $C \in Ran(V)$ **then**
    $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{C \twoheadleftarrow \{0,1\}^{|M|} \backslash Ran(V)}$
$Dom(V) \leftarrow Dom(V) \cup \{M\}$
$Ran(V) \leftarrow Ran(V) \cup \{C\}$
return C

proc $\textsc{Dec}(T, C)$ Games G_5 / $\boxed{G_6}$

$(U, V) \leftarrow T$; $M \twoheadleftarrow \{0,1\}^{|C|}$
if $M \in Dom(V)$ **then**
    $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{M \twoheadleftarrow \{0,1\}^{|C|} \backslash Dom(V)}$
$Dom(V) \leftarrow Dom(V) \cup \{M\}$; $Ran(V) \leftarrow Ran(V) \cup \{C\}$
return M

Fig. 12. Games G_1–G_6 in the proof of Theorem 7. Games G_1, G_3, G_6 include the corresponding boxed statements, but games G_2, G_4, G_5 do not. In each game, there is an implicit procedure INITIALIZE that samples $K_1 \twoheadleftarrow \mathcal{K}_1$. Sets are initialized to $\emptyset$. Every entry of all arrays is initialized to $\perp$.

doesn't make redundant queries, that is, (i) it doesn't repeat prior queries for each oracle, (ii) once it queries (T, M) to ENC to get C, it won't query (T, C) to DEC, and (iii) once it queries (T, C) to DEC to get M, it won't query (T, M) to ENC. Game G_1 corresponds to the real setting and game G_6 to the random setting.

We explain the game chain to the terminal one. Let $\text{Func}(\mathcal{V}, \mathcal{M})$ be the set of functions $f : \mathcal{V} \times \mathcal{M} \to \mathcal{M}$ such that each $f(V, \cdot)$ is length-preserving, for every $V \in \mathcal{V}$. Game G_2 is identical to G_1, except that instead of using an ideal $\tilde{\pi} \twoheadleftarrow \text{Perm}(\mathcal{V}, \mathcal{M})$ we'll use an ideal $f \twoheadleftarrow \text{Func}(\mathcal{V}, \mathcal{M})$. Suppose that $\mathcal{A}$ uses ℓ tweaks $T_1, \ldots, T_\ell$, and makes q_i queries for tweak T_i. Games G_1 and G_2 are identical-until-bad, and thus

$$\Pr[\mathcal{A}^{G_1} \Rightarrow 1] - \Pr[\mathcal{A}^{G_2} \Rightarrow 1] \le \Pr[\mathcal{A}^{G_2} \text{ sets } \mathsf{bad}] \le \sum_{i=1}^{\ell} \frac{q_i^2}{2^{n+1}} \le \frac{q^2}{2^{n+1}} \ .$$

In game G_3, instead of sampling y at random and computing C as $y \oplus Mask$, we sample C at random at let $y \leftarrow C \oplus Mask$. The change is conservative. In game G_4, both ENC and DEC always

return a fresh random answer. Games G_3 and G_4 are identical-until-bad, and thus

$$\Pr[\mathcal{A}^{G_3} \Rightarrow 1] - \Pr[\mathcal{A}^{G_4} \Rightarrow 1] \le \Pr[\mathcal{A}^{G_4} \text{ sets } \mathsf{bad}] \ .$$

We now bound the chance that G_4 sets bad. Without loss of generality, we can assume that the adversary is non-adaptive, because the answers from the oracles are always independent, uniformly random strings. Game G_4 sets bad only if the adversary can find distinct (U, M) and (U', M') such that $|M| = |M'|$ and

$$(0^m \parallel H_K(U) \parallel 0^{|M|-n-m}) \oplus M = (0^m \parallel H_K(U') \parallel 0^{|M'|-m-n}) \oplus M', \tag{4}$$

where $m = \max\{0, |M| - 2n\}$. If $U = U'$ then $M \ne M'$ because $(U, M) \ne (U', M')$, and thus Equation (4) won't happen. If $U \ne U'$ then Equation (4) happens with probability at most $\mathrm{Coll}(\lceil \|U\|/n \rceil, \lceil \|U'\|/n \rceil)$. Hence $\Pr[\mathcal{A}^{G_4} \text{ sets } \mathsf{bad}] \le \mathbf{Adv}_H^{\mathrm{axu}}(s)$. Next, game G_5 is the simplified version of game G_4. Game G_6 is identical to G_5, except that we maintain the consistency among the queries. Then

$$\Pr[\mathcal{A}^{G_5} \Rightarrow 1] - \Pr[\mathcal{A}^{G_6} \Rightarrow 1] \le \Pr[\mathcal{A}^{G_5} \text{ sets } \mathsf{bad}] \le \sum_{i=1}^{\ell} \frac{q_i^2}{2^{n+1}} \le \frac{q^2}{2^{n+1}} \ .$$

Summing up, $\mathbf{Adv}_{\mathrm{XT}[H,\mathrm{Perm}(\mathcal{V},\mathcal{M})]}^{\pm\widetilde{\mathrm{prp}}}(\mathcal{A}) \le \Pr[\mathcal{A}^{G_1} \Rightarrow 1] - \Pr[\mathcal{A}^{G_6} \Rightarrow 1] \le q^2/2^n + \mathbf{Adv}_H^{\mathrm{axu}}(s). \qquad \square$

D   An Insecure Variant of AEZ-core

Numerous variants of AEZ-core were considered to arrive at AEZ-core. Most simplifications of the final version do not work. As an example, consider trying to cheapen the design by using $c_i \cdot f_{\mathsf{aa},1}(S)$ instead of $f_{\mathsf{aa},i}(S)$ to whiten the middle of each Feistel network, where each c_i is a public constant, and the dot is the multiplication in $\mathrm{GF}(2^n)$. For example, one might hope this works for $c_i = 1$ or $c_i = i$. But this modification is insecure for any choice of c_i values.

For each $L \subseteq \{1, \ldots, n+1\}$ let $\theta(L) = \oplus_{i \in L} c_i$. Let $D \ne \emptyset$ be a subset of $\{1, \ldots, n+1\}$ such that $\theta(D) = 0^n$. Such a set D must exists. Assume to the contrary that $\theta(L) \ne 0^n$ for all nonempty $L \subseteq \{1, \ldots, n+1\}$. Then for any distinct nonempty subsets $L, L' \subseteq \{1, \ldots, n+1\}$, we have $\theta(L) \ne \theta(L')$. This means that for $2^{n+1} - 1$ nonempty subsets $L \subseteq \{1, \ldots, n+1\}$ we have $2^{n+1} - 1 > 2^n$ corresponding distinct elements $\theta(L)$ of $\mathrm{GF}(2^n)$, which is a contradiction.

We now describe an attack to the modified AEZ-core. Our attack only uses strings of length $\ell = 2n(n+3)$. Let M and $\tilde{M}$ be arbitrary distinct ℓ-bit strings such that they agree everywhere except the last two blocks. Query M and $\tilde{M}$ to the first oracle to get answers C and $\tilde{C}$ respectively. In the real game, we'll have $X_i = \tilde{X}_i$ and $\tilde{Y}_i = Y_i \oplus (c_i \cdot (S \oplus \tilde{S}))$ for every $1 \le i \le n+2$. Next, let C^* be the "mixed" ciphertext such that, for every $1 \le i \le n+3$, the $(2i-1)$'th and $2i$'th blocks of C^* are the same as those of $\tilde{C}$ if $i \in D$, otherwise C^* would borrow the corresponding two blocks of C. Query C^* to the second oracle to get an answer M^*. Let $\overline{D} = \{1, \ldots, n+2\} \backslash D$. In the real game, the query C^* will generate $Y_i^* = \tilde{Y}_i$ for every $i \in D$, and $Y_i^* = Y_i$ for every $i \in \overline{D}$. Then

$$Y^* = \bigoplus_{i \in D} \tilde{Y}_i \oplus \bigoplus_{j \in \overline{D}} Y_j = Y \oplus \bigoplus_{i \in D} ((S \oplus \tilde{S}) \cdot c_i) = Y \ .$$

Consequently, $S^* = S$ and thus M^* and M agree at the $(2n+3)$th and $(2n+4)$th blocks. The latter event happens with probability at mos 2^{-n} in the random game. Hence this attack wins with advantage at least $1 - 2^{-n}$.