Repeated Masks, Predictable Splices: Breaking AESpolyW and Its AEAD Applications

Shuping Mao[1], Peng Wang[2,*], Jiadong Han[2], Tingting Guo[3], Yan jia[4,5], Ying Chen[1]

1. Beijing Electronic Science & Technology Institute, Beijing, China
2. School of Cryptology, University of Chinese Academy of Sciences, Beijing, China
3. Ningbo Key Laboratory of Information Technology Application Innovation and Security, School of Cyber Science and Engineering, Ningbo University of Technology, Ningbo, China
4. State Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
5. School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

* Corresponding author. p-wang@ucas.ac.cn

Abstract

AESpolyW, presented at the 2026 IEEE Symposium on Security and Privacy, is a wide-block encryption mode that preserves the length of each message. Its design lets operations of the Advanced Encryption Standard (AES) overlap with hash computations based on polynomial evaluation. Our attacks splice ciphertexts by copying blocks from one ciphertext to the same positions in another. On five-block messages, two encryption queries and one decryption query at a fixed public parameter, called a tweak, distinguish AESpolyW from a random permutation at that tweak. The two probabilities of reporting AESpolyW differ by more than $1-2^{-3n}$, where n is the block size in bits. When AESpolyW decrypts the splice, an internal block-cipher inversion takes an output from an earlier encryption. The published proof treats such reuse as unlikely. The splice also works across distinct tweaks.

We also attack three AESpolyW-based schemes for authenticated encryption with associated data (AEAD). All three use encode-then-encipher, adding a zero block before encryption and checking it after decryption. The zero block goes at the end, after the first block, or at the beginning. The tweak $T = A\|\nu$ concatenates one block of associated data A, which is authenticated without encryption, and one block of nonce ν, a public value intended for one-time use per key. Two encryptions at distinct nonces suffice to forge a new ciphertext that passes verification with probability one. In the experiment for indistinguishability under chosen-ciphertext attack (IND-CCA), we use one auxiliary encryption at a nonce distinct from the challenge nonce and at most one decryption. This identifies which of two chosen plaintexts was encrypted in the challenge with probability one.

Keywords. AESpolyW, Splicing attacks, Encode-then-encipher

1 Introduction

Wide-block encryption (WBE) extends a fixed-block-length primitive to a length-preserving permutation on long messages. A public parameter T, called a tweak, selects a permutation for each supported message length. The strong tweakable pseudorandom permutation (STPRP) goal is for each pair of forward and inverse operations to behave as if it came from an independent random permutation. A tweakable WBE mode thus gives a family of permutations indexed by tweak and message length. In storage encryption, this lets a mode encipher an entire sector or data unit as one wide block.

Block-cipher-based WBE designs include CMC [HR03], EME [HR04], EME* [Hal04], XCB [MF07, BA25, WMX+25], HCTR [WFW05], TET [Hal07], FMix [BN15], and HCTR2 [CHB21]. They differ in parallelism, efficiency, implementation complexity, and security guarantees. The U.S. National Institute of Standards and Technology (NIST) sets out requirements for cryptographic accordions in IR 8552. An accordion is a block-cipher-based mode that provides STPRP security for variable-length inputs [CDD+25]. More recent wide-block proposals include ddd-AES [DMMT25], HCTR+ [DDG+25], and DbHCTR [Lee24].

Authenticated encryption (AE) protects the confidentiality and integrity of a plaintext. Authenticated encryption with associated data (AEAD) also authenticates associated data (AD), which remain unencrypted. Encode-then-encipher (EtE) [BR00] adds redundancy to a plaintext, encrypts the result with a wide-block mode, and checks the redundancy during decryption. Hoang et al. show that EtE with a wide-block mode that provides STPRP security can achieve robust authenticated encryption (RAE), a security notion that covers decryption that releases the invalid encoding after a failed redundancy check [HKR15].

EtE is one approach to deriving AEAD in NIST's ongoing work on cryptographic accordions. Figure 2 of NIST IR 8552 illustrates this approach [CDD+25, Section 4.1 and Figure 2]. The construction appends zero bits to the plaintext before enciphering and encodes the nonce and associated data in the accordion tweak. A nonce is a public value intended for one-time use per key. Reusing it with the same associated data repeats the tweak. The report discusses nonce misuse resistance, which limits the loss of security when a nonce repeats, and security under release of unverified plaintext (RUP), which covers decryption that releases data before verification [CDD+25, Sections 4.1 and 4.4.4].

AESpolyW [HNSS26], presented at the 2026 IEEE Symposium on Security and Privacy, is a wide-block mode that lets instructions for the Advanced Encryption Standard (AES) and polynomial hashing run concurrently. Polynomial hashing evaluates a polynomial whose coefficients are derived from data blocks. Figure 1 shows where AESpolyW uses each operation. With 128-bit AES keys and 64 KB messages, the authors measured fewer cycles per byte for AESpolyW than for HCTR2 on 11 of the 12 tested processors and than for EME on 9 of the 12.

AESpolyW splits the plaintext into a center block M_0, a left part M_l, and a right part M_r. Like HCTR [WFW05], the left branch combines polynomial hashing with counter-mode encryption. Counter-mode encryption masks the data with values generated from block counters. Like EME [HR04], the right branch has two layers that process blocks in parallel. Between them, AESpolyW combines each first-layer output block with the same value N using bitwise exclusive OR (XOR). We call N the center mask because the center of the construction computes it.

To splice two ciphertexts, we use one as the base ciphertext and copy selected blocks from the other into the same positions. The base tweak is the tweak that produced the base ciphertext. Each right-side layer XORs a block with a position mask, an n-bit value determined by the key and block index. Matching plaintext blocks at the selected positions therefore give matching first-layer outputs across tweaks. When decryption inverts the second right-side layer, each copied block introduces the same difference between the two center masks. The right input and output checksums are XOR sums taken before and after the center-mask mixing step. With an even number of copied blocks, the differences cancel in the right output checksum. Because the center and left ciphertext remain unchanged, decryption under the base tweak recovers the center mask of the base ciphertext. Removing that mask leaves differences that cancel in the right input checksum as well. Decryption therefore preserves the unspliced plaintext blocks. In an EtE instance, those blocks can include the zero redundancy and a block that distinguishes the two challenge plaintexts.

We analyze fixed-tweak and cross-tweak splices on AESpolyW. The cross-tweak splice yields nonce-respecting attacks on AESpolyW-based EtE. In these attacks, all encryption queries use distinct nonces.

1. **Three-query breaks of AESpolyW.** We give two STPRP distinguishers, each using two

encryption queries and at most one decryption query. The center-difference attack varies only the center plaintext block under a fixed tweak and extends to any nonempty even set of right-side positions. The cross-tweak attack encrypts the same plaintext under two distinct tweaks. The distinguishing advantage is the probability of reporting AESpolyW with the real scheme minus that probability with random permutations. On five-block messages, both attacks achieve an advantage greater than $1 - 2^{-3n}$, where n is the block size.

2. **An explicit witness to the proof failure.** The splice produces a fresh ciphertext, meaning that no previous encryption at the base tweak and message length returned it. When AESpolyW decrypts that ciphertext, the block-cipher inversion at the center takes an output from an earlier encryption. This reuse occurs even when the proof experiment replaces the block cipher with a random permutation. It satisfies the condition for bad_3, the event that records reuse of an earlier block-cipher output during decryption. The forced reuse falls outside the encryption-side collision estimates that the proof applies to mixed encryption and decryption queries by symmetry.

3. **Nonce-respecting breaks of EtE.** We break ciphertext integrity and chosen-ciphertext confidentiality for AESpolyW-based EtE with appended, inserted, or prepended zero-block redundancy. When the associated data and nonce each occupy one block, two encryptions at distinct nonces and one verification yield a fresh, accepted ciphertext with probability one. In the experiment for indistinguishability under chosen-ciphertext attack (IND-CCA), we recover the challenge bit with probability one. We use one auxiliary encryption at a nonce distinct from the challenge nonce and at most one decryption. These attacks violate the standard security guarantees of nonce-based AEAD, even when the decoder releases plaintext only after verifying redundancy. The instances therefore fall short of the RAE objective.

Both splicing distinguishers and the three distinct-nonce EtE forgeries passed experimental verification (Appendix A).

Related work. Ciphertext splicing is an established technique in the cryptanalysis of wide-block encryption. Halevi and Rogaway give a two-encryption, one-decryption splicing attack on EME extended beyond its specified message lengths [HR04, Section 6.2 of the full version]. For an m-block message, EME multiplies a shared mask by $x, x^2, \ldots, x^{m-1}$ at positions $2, \ldots, m$, where x is the indeterminate in the field representation. These coefficients are linearly independent over the binary field within the prescribed $m \leq n$ range, where n is the block size in bits. At sufficiently long lengths, a subset whose coefficients XOR to zero enables splicing. Hoang et al. use the same cancellation principle against a simplified variant of AEZ-core, the wide-block component of AEZ, in which public coefficients multiply a common mask value in place of the original position-dependent function outputs [HKR15, Appendix D of the full version]. Copying selected ciphertext pairs preserves an internal checksum and allows the adversary to distinguish this variant.

In AESpolyW, all right-side coefficients are one, so two positions already cancel and the attack applies to five-block messages within the specified domain. Our analysis also supplies a concrete witness to the published proof's failure. The position masks stay fixed across tweaks, enabling the cross-tweak splices that we use to derive nonce-respecting forgery and IND-CCA attacks on AESpolyW-based EtE.

2 Preliminaries

2.1 Notation

A block cipher with n-bit blocks gives a permutation E_K under key K and an inverse E_K^{-1}. A tweakable wide-block mode with key κ gives a permutation for each public tweak T and supported input length. We write $\tilde{E}_\kappa(T, M)$ for its encryption of plaintext M, and $\tilde{E}_\kappa^{-1}(T, C)$ for its decryption of ciphertext C. A tweakable block cipher has a fixed block length.

For a bit string D, write $|D|$ for its length in bits, $|D|_n = \lceil |D|/n \rceil$ for its number of blocks, and ε for the empty string. The symbol $\|$ denotes concatenation, and $\oplus$ denotes bitwise XOR of strings of equal length. If D consists of full n-bit blocks $D_1, \ldots, D_{|D|_n}$, we use

$$\mathrm{XOR}_n(D) = \bigoplus_{i=1}^{|D|_n} D_i, \qquad \mathrm{Copy}_c(N) = \underbrace{N\| \cdots \|N}_{c \text{ copies}} \quad (c \geq 0,\ N \in \{0,1\}^n).$$

The strings 0^n and 1^n consist of n zeros and n ones, respectively. The XOR of zero blocks is 0^n, and $\mathrm{Copy}_0(N) = \varepsilon$. For integers $0 \leq i \leq j$, let $[i..j] = \{i, i+1, \ldots, j\}$. We write $[j] = [1..j]$ for $j \geq 1$ and $[0] = \emptyset$.

2.2 SPRP and STPRP Security

In a distinguishing experiment, the adversary submits encryption and decryption queries to an oracle, the interface that answers those queries. The adversary outputs 1 if it guesses that it is in the real experiment. For a strong pseudorandom permutation (SPRP), the real experiment uses a block cipher under a random key. The ideal experiment uses a uniformly random permutation π. Both experiments provide access to the inverse. The advantage of an adversary $\mathscr{A}$ is the difference between the real and ideal probabilities of outputting 1.

$$\mathrm{Adv}_E^{\mathrm{SPRP}}(\mathscr{A}) = \Pr\left[\mathscr{A}^{E_K, E_K^{-1}} = 1\right] - \Pr\left[\mathscr{A}^{\pi, \pi^{-1}} = 1\right].$$

For a strong tweakable pseudorandom permutation (STPRP), the real experiment uses a tweakable mode under a random key. The ideal experiment independently samples a uniformly random permutation $\pi_{T,\ell}$ on ℓ-bit strings for each tweak T and supported length ℓ. Write $\boldsymbol{\pi}$ for this family of permutations and $\boldsymbol{\pi}^{-1}$ for their inverses. The STPRP advantage is

$$\mathrm{Adv}_{\tilde{E}}^{\mathrm{STPRP}}(\mathscr{A}) = \Pr\left[\mathscr{A}^{\tilde{E}_\kappa, \tilde{E}_\kappa^{-1}} = 1\right] - \Pr\left[\mathscr{A}^{\boldsymbol{\pi}, \boldsymbol{\pi}^{-1}} = 1\right].$$

A ciphertext is fresh at a given tweak and length if no earlier encryption at that tweak and length returned it.

2.3 The AESpolyW Mode

For a message consisting of full blocks, AESpolyW [HNSS26] partitions $M = M_0 \| M_l \| M_r$ and $C = C_0 \| C_l \| C_r$ into a center block and two branches. The right branch contains $m_r = \lfloor (|M|_n - 1)/2 \rfloor$ blocks, and the left branch contains the rest. AESpolyW splits the tweak as $T = T_l \| T_r$, with the last $\lfloor |T|_n/2 \rfloor$ full blocks forming the right tweak T_r. We write $M_{r,j}$ and $C_{r,j}$ for block j of M_r and C_r.

The block cipher uses key K, and the left branch uses a tweakable hash H_H with key H. The hash takes T_l and a data string as inputs and returns an n-bit value. AESpolyW derives two values $\Delta_0 = E_K(0^n)$ and $\Delta_1 = E_K(1^n)$ to generate position masks. The function $h_\Delta(j)$ gives the position mask for block index j from the starting value Δ, using the mask family specified by

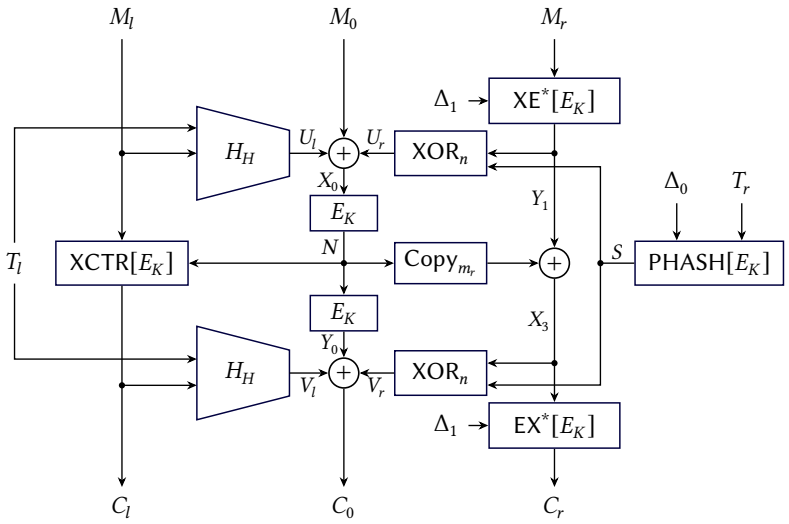

Figure 1: Encryption of AESpolyW.

AESpolyW [HNSS26, Sec. 4.3]. Let $T_{r,k}$ be block k of T_r. The right-tweak hash PHASH applies the block cipher to each masked block and XORs the outputs.

$$S = \mathsf{PHASH}[E_K](\Delta_0, T_r) = \bigoplus_{k=1}^{|T_r|_n} E_K(T_{r,k} \oplus h_{\Delta_0}(k)), \tag{1}$$

where the empty sum is 0^n. For a one-block right tweak, $S = E_K(T_r \oplus h_{\Delta_0}(1))$ is injective in T_r for a fixed key.

For $j \in [m_r]$, the two right-side layers use the same position mask $h_j = h_{\Delta_1}(j)$ for every tweak under a given key. The first layer XE^* XORs the mask before applying the block cipher. The second layer EX^* applies the block cipher before XORing the mask. Between the two layers, AESpolyW XORs the center mask N, defined below, into each right-side block. The blockwise relations are

$$Y_{1,j} = E_K(M_{r,j} \oplus h_j), \quad X_{3,j} = Y_{1,j} \oplus N, \quad C_{r,j} = E_K(X_{3,j}) \oplus h_j. \tag{2}$$

The strings Y_1 and X_3 are the concatenations of the blocks $Y_{1,j}$ and $X_{3,j}$, respectively, in index order. The left and right input checksums U_l, U_r enter the computation of the center mask. The left and right output checksums V_l, V_r enter the computation of the center ciphertext block.

$$U_l = H_H(T_l, M_l), \quad U_r = \mathsf{XOR}_n(Y_1 \| S),$$
$$V_l = H_H(T_l, C_l), \quad V_r = \mathsf{XOR}_n(X_3 \| S).$$

AESpolyW computes $X_0 = M_0 \oplus U_l \oplus U_r$ and $N = E_K(X_0)$. To compute the left ciphertext $C_l = \mathsf{XCTR}[E_K](N, M_l)$, XOR counter mode (XCTR) generates a keystream, a sequence of masking blocks. Block j of this sequence is $E_K(N \oplus \mathrm{str}_n(j))$, where $\mathrm{str}_n(j)$ encodes j in n bits. The mode XORs the data with the required number of keystream bits. AESpolyW computes $Y_0 = E_K(N)$ and sets the center ciphertext block to $C_0 = Y_0 \oplus V_l \oplus V_r$. Figure 1 shows the full encryption, and Figure 2 shows the right branch.

Decryption uses the supplied tweak to compute S and V_l. It recovers $X_{3,j} = E_K^{-1}(C_{r,j} \oplus h_j)$ at each right-side position and uses these blocks to compute V_r. The center mask is then $N = E_K^{-1}(C_0 \oplus V_l \oplus V_r)$. Applying XCTR to the left ciphertext gives $M_l = \mathsf{XCTR}[E_K](N, C_l)$. For a fixed N and length, the mode uses the same keystream in both directions. On the right, decryption computes $Y_{1,j} = X_{3,j} \oplus N$ and $M_{r,j} = E_K^{-1}(Y_{1,j}) \oplus h_j$. It recomputes U_l, U_r from M_l, Y_1 and the supplied tweak to obtain $M_0 = E_K^{-1}(N) \oplus U_l \oplus U_r$. Algorithm 1 in Appendix B gives the full pseudocode.

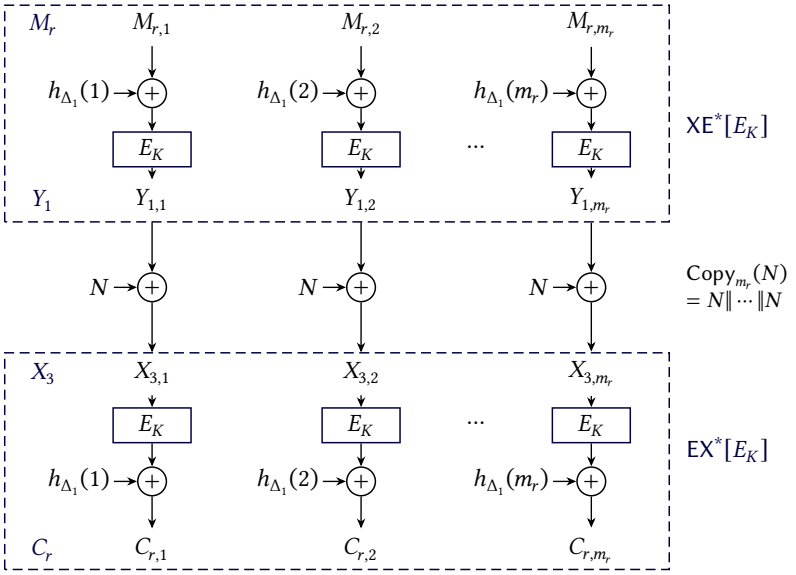

Figure 2: Blockwise processing in the right branch of AESpolyW.

2.4 Encode-then-Encipher and Authenticated Encryption

Encode-then-encipher (EtE) maps a plaintext P to an encoded plaintext $M = \mathsf{Encode}(P)$ and then encrypts M with a tweakable wide-block mode. The injective encoding assigns distinct encodings to distinct plaintexts and adds τ bits of redundancy. For $\tau = n$, we use the three zero-block placements studied for EtE based on HCTR2 [CCH+26]. Let P_0 be the first block of P and P_{tail} the rest. The appended, inserted, and prepended variants are

$$\begin{aligned}
\mathsf{EtE}_A &: \quad \mathsf{Encode}_A(P) = P\|0^n, \\
\mathsf{EtE}_S &: \quad \mathsf{Encode}_S(P_0\|P_{\text{tail}}) = P_0\|0^n\|P_{\text{tail}}, \\
\mathsf{EtE}_P &: \quad \mathsf{Encode}_P(P) = 0^n\|P.
\end{aligned}$$

For a wide-block mode W_κ and $\mathsf{X} \in \{A, S, P\}$, encryption is $\mathsf{Enc}^{\mathsf{X}}_\kappa(T, P) = \mathsf{W}_\kappa(T, \mathsf{Encode}_{\mathsf{X}}(P))$. Decryption applies $\mathsf{W}^{-1}_\kappa(T, C)$ and passes the result to the decoder. The decoder checks the encoding and removes the redundancy to recover P if the encoding is valid. Otherwise it returns the rejection symbol $\perp$. Our instances use AESpolyW as W_κ, with key $\kappa = (K, H)$.

Hoang et al. show that EtE achieves robust authenticated encryption (RAE) when the underlying variable-length mode provides STPRP security [HKR15, Theorem 3 of the full version].

Tweak formation and splitting. For nonce-based EtE, the tweak T encodes the associated data A and nonce ν. In the AESpolyW-based instances studied here, $A, \nu \in \{0,1\}^n$, and both encryption and decryption use $T = A\|\nu$ as the AESpolyW tweak. This tweak has two blocks, so Algorithm 1 takes the last $n\lfloor |T|_n/2 \rfloor = n$ bits as the right tweak. Thus

$$T = A\|\nu, \qquad T_l = A, \qquad T_r = \nu. \tag{3}$$

The left keyed hash takes A as its tweak, and PHASH processes ν on the right.

In the following nonce-respecting experiments, all encryption queries use distinct nonces.

Ciphertext integrity (INT-CTXT). The adversary makes encryption queries with distinct nonces and outputs a tuple (A, ν, C). The tuple is a successful ciphertext forgery if C is fresh at $T = A\|\nu$ and the decoder accepts it. The final tuple may use a previously queried nonce.

Indistinguishability under chosen-ciphertext attack (IND-CCA). The adversary chooses two equal-length messages, along with the nonce and associated data for the challenge. The challenger encrypts the message selected by a uniform bit b. Ordinary encryption queries and the challenge must all use distinct nonces. Decryption queries may use the challenge nonce, excluding the exact challenge tuple (A, ν, C). If b' is the adversary's guess, its normalized advantage is $2\Pr[b' = b] - 1$.

3 Splicing Attacks on AESpolyW

3.1 Center-Difference Splicing Attack

In the five-block attack, we encrypt two plaintexts under the same tweak. They differ only in the center block. The splice combines the first three blocks of the first ciphertext with the last two of the second (Figure 3). The first response is the base ciphertext. Decryption under the base tweak preserves the first three blocks of its plaintext.

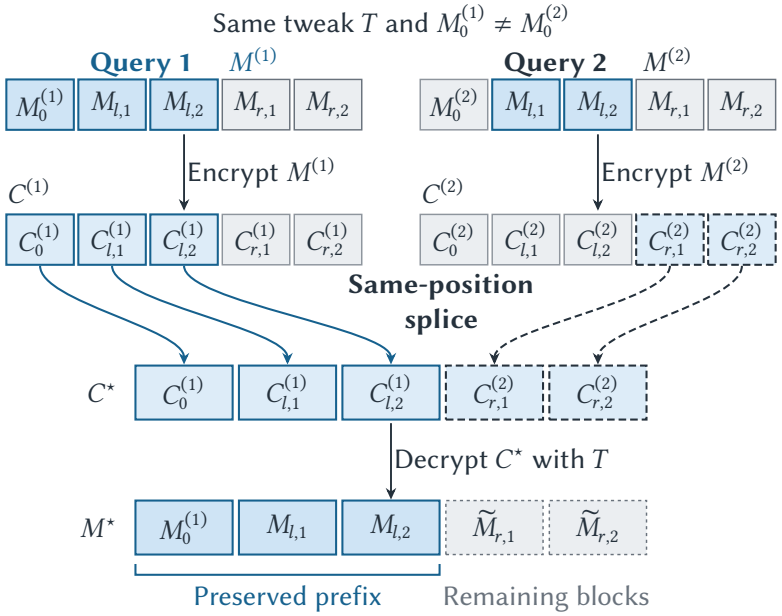

Figure 3: Five-block center-difference splicing attack on AESpolyW.

Attack steps. Fix a supported tweak T. Encrypt

$$M^{(b)} = M_0^{(b)} \| M_l \| M_{r,1} \| M_{r,2}, \qquad b \in \{1, 2\},$$

with distinct center blocks $M_0^{(1)}, M_0^{(2)}$ and a common two-block left part M_l. Write each response as $C^{(b)} = C_0^{(b)} \| C_l^{(b)} \| C_{r,1}^{(b)} \| C_{r,2}^{(b)}$. Form

$$C^\star = C_0^{(1)} \| C_l^{(1)} \| C_{r,1}^{(2)} \| C_{r,2}^{(2)}. \tag{4}$$

If $C^\star$ equals either encryption response, output 0. Otherwise, decrypt $C^\star$ under the base tweak T and output 1 exactly when the first three plaintext blocks equal $M_0^{(1)} \| M_l$.

Attack result. In AESpolyW, $C^\star$ is fresh and the test succeeds with probability one. The attack makes two encryption queries and at most one decryption query. Its STPRP advantage is at least

$$1 - \frac{2^{2n} - 1}{2^{5n} - 2} > 1 - 2^{-3n}.$$

Real experiment. The two encryptions share a tweak and the left and right plaintext parts, so S, Y_1, U_l, U_r agree. Since $X_0^{(b)} = M_0^{(b)} \oplus U_l \oplus U_r$, the distinct center blocks give distinct values of $X_0^{(b)}$. The masks $N^{(b)} = E_K(X_0^{(b)})$ therefore differ. Write $\delta = N^{(1)} \oplus N^{(2)} \neq 0^n$. At each of the two right-side positions, $X_{3,j}^{(b)} = Y_{1,j} \oplus N^{(b)}$, so the two values differ by δ. The second right-side layer $C_{r,j}^{(b)} = E_K(X_{3,j}^{(b)}) \oplus h_j$ is injective at each position, so both copied blocks differ from the corresponding blocks of $C^{(1)}$.

Decryption applies $E_K^{-1}(C_{r,j}^{\star} \oplus h_j)$ at each right-side position and recovers $X_{3,j}^{(2)}$ at both. The right output checksum therefore satisfies

$$V_r^{\star} \oplus V_r^{(1)} = \bigoplus_{j=1}^{2} (X_{3,j}^{(2)} \oplus X_{3,j}^{(1)}) = \delta \oplus \delta = 0^n.$$

The retained left part gives $V_l^{\star} = V_l^{(1)}$, and the retained center gives $C_0^{\star} = C_0^{(1)}$. Thus

$$Y_0^{\star} = C_0^{(1)} \oplus V_l^{(1)} \oplus V_r^{(1)} = E_K(N^{(1)}), \qquad N^{\star} = N^{(1)}.$$

With $N^{(1)}$, XCTR uses the same keystream and recovers M_l from $C_l^{(1)}$. Decryption then recomputes U_l. On the right, removing $N^{(1)}$ gives $Y_{1,j}^{\star} = Y_{1,j} \oplus \delta$ for $j = 1, 2$. The two differences cancel in the right input checksum, so $U_r^{\star} = U_r$. Finally,

$$M_0^{\star} = E_K^{-1}(N^{(1)}) \oplus U_l \oplus U_r = M_0^{(1)}.$$

The copied blocks give $C^{\star} \neq C^{(1)}$. Decryption correctness gives $C^{\star} \neq C^{(2)}$ because the recovered center block differs from $M_0^{(2)}$. The splice is fresh and passes the test.

Ideal experiment. Condition on a pair of encryption responses for which the splice is fresh. The ideal inverse at tweak T maps $C^{\star}$ uniformly to one of the $2^{5n} - 2$ unqueried inputs. Of the 2^{2n} inputs with the tested prefix, only $M^{(1)}$ was queried, since $M^{(2)}$ has a different center. The conditional probability of passing is $(2^{2n}-1)/(2^{5n}-2)$. The adversary outputs 0 if the splice equals either encryption response, so the same fraction bounds the overall ideal acceptance probability.

Generalization to an even number of positions. At a supported length ℓ that is a multiple of n, make two encryption queries at the same tweak with plaintexts that differ only in the center block. Choose a set $J \subseteq [m_r]$ with a nonzero even number of positions and replace those right-side blocks of $C^{(1)}$ with the corresponding blocks of $C^{(2)}$. Apply the same freshness test and check that decryption preserves the center, the left part, and the right-side blocks outside J. In both right checksum computations, the difference is now $\bigoplus_{j \in J} \delta = 0^n$. The center mask, the left plaintext, and the center plaintext block follow as above. For $j \notin J$, the same $X_{3,j}$ and center mask recover the corresponding plaintext block from the first query. A changed block in J distinguishes the splice from $C^{(1)}$, and the preserved center distinguishes its decryption from $M^{(2)}$, so the splice also differs from $C^{(2)}$. In the ideal experiment, the test leaves $n|J|$ bits free and excludes one qualifying input from the two previous queries. Thus the STPRP advantage is at least

$$1 - \frac{2^{n|J|} - 1}{2^{\ell} - 2}. \tag{5}$$

3.2 Where the AESpolyW Security Proof Fails

In the proof of Theorem 1 in [HNSS26], Game 2 uses a random permutation in place of the block cipher. Game 3 replaces that permutation with the forward and inverse lookup tables in

Algorithm 3 of the original paper. When a lookup finds no entry, the game samples a uniform n-bit value and records the pair in both tables. Game 4 uses the ideal tweakable permutations of the STPRP experiment.

Appendix B.3.2 of the original proof defines bad_3 to detect reuse of internal block-cipher values across construction queries during decryption. In particular, the event occurs if an inverse call in the center computation takes as input a block-cipher output from an earlier construction query under the same user key. For encryption, Appendix B.3.3 applies the almost-XOR-universal (AXU) property to collisions between forward inputs $X_{0,1} = M_0 \oplus U_l \oplus U_r$. The AXU property bounds the probability, over the choice of hash key, that the hash values of two distinct inputs have a prescribed XOR difference. Appendix B.3.5 invokes symmetry to use the encryption-side estimates to bound the probability that bad_3 occurs first [HNSS26].

The five-block splice of Section 3.1 forces this reuse whenever the lookup tables agree on the recorded pairs. To compute the right output checksum, decryption of $C^\star$ inverts the second right-side layer at the two copied positions. These lookups read the entries used in the second encryption. The two differences cancel, so $V_r^\star = V_r^{(1)}$. The splice also keeps $C_0^{(1)}$ and $C_l^{(1)}$, so $V_l^\star = V_l^{(1)}$. The center inverse call on line 6 of the original decryption algorithm therefore takes

$$Y_0^\star = C_0^\star \oplus V_l^\star \oplus V_r^\star = C_0^{(1)} \oplus V_l^{(1)} \oplus V_r^{(1)} = Y_0^{(1)}.$$

The original proof denotes Y_0, the output of the second center block-cipher call in encryption, by $Y_{0,2}$. Thus $Y_{0,2}^{(3)} = Y_{0,2}^{(1)}$ meets the condition for bad_3. The third construction query is a decryption, and its inverse lookup takes the output of the first construction query's center computation. The inverse table returns the recorded $N^{(1)}$ even though $C^\star$ is fresh at the base tweak.

The argument holds in Game 2 and carries over to Game 3 under the coupling in Appendix B.2 of [HNSS26]. The coupling samples the two games jointly. Their answers agree until a newly sampled table value collides with a previously assigned value.

With one user and the three queries of our attack, the probability that any of $\mathrm{bad}_1, \ldots, \mathrm{bad}_5$ occurs in Game 3 is at least $1 - \varepsilon_{\mathrm{sw}}$, where $\varepsilon_{\mathrm{sw}}$ is the collision probability bound in the coupling of Appendix B.2. To bound this union, the original proof sums $\Pr[\mathrm{bad}_i^*]$ for $i = 1, \ldots, 6$, where bad_i^* is the event that bad_i occurs first. If any of the six events stops the proof's experiment earlier, that event also contributes to the sum. The estimate in Appendix B.3.5 misses the forced reuse. The claimed bounds in Appendix B.3.8 therefore cannot justify the transition from Game 3 to Game 4.

Quantitative Game 3–4 gap. Run the five-block distinguisher at a fixed two-block tweak and let p_i be its acceptance probability in Game i. The attack in Section 3.1 works for every underlying block-cipher permutation, so $p_2 = 1$. Each construction query uses at most eleven block-cipher calls in either direction. This includes two for the initial masks, one for the right-tweak hash, two for the center, two for the left-branch counter mode, and four for the two right-side layers. The three queries thus use at most 33 block-cipher calls.

Under the Game 2–3 coupling, a fresh table sample at the j-th primitive call can collide with at most $j - 1$ previously assigned values on the sampled side. Repeated lookups agree while the tables remain consistent. Therefore

$$\varepsilon_{\mathrm{sw}} \le \sum_{j=1}^{33} \frac{j-1}{2^n} = 528 \cdot 2^{-n}, \qquad p_3 \ge 1 - 528 \cdot 2^{-n}.$$

Game 4 is the ideal experiment of Section 3.1, so $p_4 \le (2^{2n} - 1)/(2^{5n} - 2) < 2^{-3n}$. Consequently,

$$p_3 - p_4 > 1 - 528 \cdot 2^{-n} - 2^{-3n}. \tag{6}$$

At $n = 128$, the right-hand side is $1 - 528 \cdot 2^{-128} - 2^{-384}$. Appendix B.3.8 of [HNSS26] claims an upper bound of $2.5\omega\sigma_u\sigma/2^n$ for the same difference. With one user, both block-cipher call budgets

in Appendix B can be set to 33. For the concrete parameters with $\omega = 1$ given in Section 4.4 of [HNSS26], the claimed upper bound is then $2.5 \cdot 33^2 \cdot 2^{-128}$, contradicting Equation (6).

3.3 Cross-Tweak Two-Block Splicing Attack

The center-difference attack also gives a same-tweak forgery for a seven-block EtE encoding with appended zero redundancy. The plaintexts differ only in the center block. Splicing the first two of the three right-side blocks preserves the final zero block. The cross-tweak distinguisher uses distinct tweaks for its encryption queries.

Attack steps.

1. Choose two distinct supported tweaks $T^{(1)}, T^{(2)}$ and a message M with $m \geq 5$ full n-bit blocks at a supported length $\ell = mn$. Encrypt M once under each tweak to obtain $C^{(1)}$ and $C^{(2)}$. Use $C^{(1)}$ as the base ciphertext and fix two right-side positions $J = \{i, j\}$.

2. Compare corresponding ciphertext blocks at each of the $m - 1$ positions other than the center. If all pairs are equal, output 1. If some pairs are equal and others differ, output 0.

3. If all pairs differ, form $C^\star$ by replacing the two right-side blocks of the base ciphertext at J with the corresponding blocks of $C^{(2)}$. Decrypt $C^\star$ under the base tweak $T^{(1)}$. Output 1 exactly when the result agrees with M outside J and differs from M at both positions in J.

Attack result. The distinguisher uses two encryption queries at distinct tweaks and at most one decryption query. It accepts AESpolyW with probability one. Its STPRP advantage is at least

$$1 - 2^{-(\ell - 2n)}. \tag{7}$$

In particular, at five blocks the advantage exceeds $1 - 2^{-3n}$.

Block preservation in AESpolyW. The splice works for any two equal-length messages $M^{(1)}, M^{(2)}$ consisting of full n-bit blocks that agree at the selected right-side positions J. Let $C^{(1)}, C^{(2)}$ be their encryptions at $T^{(1)}, T^{(2)}$, respectively. In Equation (2), the position mask h_k depends only on the key and k. Thus $Y^{(1)}_{1,k} = Y^{(2)}_{1,k}$ for each $k \in J$. Writing $\delta = N^{(1)} \oplus N^{(2)}$, we have $X^{(2)}_{3,k} = X^{(1)}_{3,k} \oplus \delta$ at both selected positions. The second right-side layer is injective at each position. Thus the corresponding ciphertext blocks differ at both selected positions exactly when $\delta \neq 0^n$. The splice $C^\star$ is fresh at the base tweak $T^{(1)}$ exactly when $\delta \neq 0^n$, since the only encryption response at that tweak is $C^{(1)}$.

To decrypt $C^\star$ at the base tweak $T^{(1)}$, AESpolyW inverts the second right-side layer with the same position masks and obtains

$$X^\star_{3,k} = \begin{cases} X^{(1)}_{3,k} \oplus \delta, & k \in J, \\ X^{(1)}_{3,k}, & k \notin J. \end{cases}$$

The right output checksum uses $S^{(1)}$, as in the base encryption. The two changes cancel, so

$$V^\star_r \oplus V^{(1)}_r = \bigoplus_{k \in J} (X^\star_{3,k} \oplus X^{(1)}_{3,k}) = \delta \oplus \delta = 0^n.$$

The retained $C^{(1)}_l$ and the base tweak give $V^\star_l = V^{(1)}_l$. Together with the retained $C^{(1)}_0$, these checksums recover the base center mask

$$N^\star = E_K^{-1}\big(C^{(1)}_0 \oplus V^\star_l \oplus V^\star_r\big) = N^{(1)}.$$

Decryption uses the base center mask and retained left ciphertext to recover $M_l^{(1)}$, so $U_l^\star = U_l^{(1)}$.

On the right, $Y_{1,k}^\star = X_{3,k}^\star \oplus N^{(1)}$, so $Y_{1,k}^\star \oplus Y_{1,k}^{(1)}$ is δ on J and zero elsewhere. Inverting the first right-side layer therefore recovers the base right-side plaintext at each unspliced position. The first layer is injective, so if $\delta \neq 0^n$, both recovered blocks in J differ from their base values. The right input checksum uses the same $S^{(1)}$. The two differences cancel again, giving

$$U_r^\star \oplus U_r^{(1)} = \bigoplus_{k \in J} (Y_{1,k}^\star \oplus Y_{1,k}^{(1)}) = 0^n.$$

Decryption therefore recovers the center block $E_K^{-1}(N^{(1)}) \oplus U_l^{(1)} \oplus U_r^{(1)} = M_0^{(1)}$. Thus the splice preserves every base plaintext block outside J.

Real experiment. For the distinguisher, the two encryptions use the same message. At every right-side position k, the first layer gives the same $Y_{1,k}$ in both encryptions, so the ciphertext blocks are equal exactly when $\delta = 0^n$. In the left branch, the k-th full block uses the keystream $E_K(N \oplus \mathrm{str}_n(k))$, as specified in Algorithm 1. Since the left plaintext is common and the keystream block is injective in N, corresponding left ciphertext blocks are also equal exactly when $\delta = 0^n$. If $\delta = 0^n$, all ciphertext blocks other than the center agree and the attack accepts in step 2. Otherwise they all differ, and the block-preservation argument shows that the fresh splice passes the plaintext test in step 3. The attack accepts with probability one for every key.

Ideal experiment. The two tweaks index independent random permutations, so $C^{(1)}, C^{(2)}$ are independent uniform ℓ-bit strings. For each of the $m-1$ positions other than the center, the corresponding ciphertext blocks agree with probability 2^{-n}, independently across positions. The all-equal branch accepts with probability $2^{-(\ell-n)}$, and the all-different branch occurs with probability $(1-2^{-n})^{m-1}$.

Fix any pair of encryption responses in the all-different branch. The resulting $C^\star$ differs from $C^{(1)}$. The response at $T^{(2)}$ imposes no constraint on the inverse permutation at $T^{(1)}$. Decryption at $T^{(1)}$ therefore returns a uniform sample from the $2^\ell - 1$ plaintexts other than M. The test fixes every block outside J and allows $2^n - 1$ choices at each selected position, giving exactly $(2^n-1)^2$ accepting plaintexts. Adding the probabilities of acceptance in the two disjoint branches gives the ideal acceptance probability. It is strictly less than $2^{-(\ell-2n)}$, yielding Equation (7), since $(2^n-1)^2/(2^\ell-1) < 2^{-(\ell-2n)}$ and $(1-2^{-n})^{m-1} \leq 1-2^{-n}$.

4 Nonce-Respecting Attacks on AESpolyW-Based EtE

The Cross-Tweak Two-Block Splicing Attack in Section 3.3 applies to all three encodings in Section 2.4. The associated data and nonce occupy one block each, as in Equation (3).

Distinct nonces give distinct center masks. Encrypt the same encoded plaintext at $T^{(b)} = A \| v^{(b)}$ for $b \in \{1,2\}$, with fixed A and distinct nonces. Both encryptions have the same M_0, U_l, Y_1. By the one-block case of Equation (1), $S^{(b)} = E_K(v^{(b)} \oplus h_{\Delta_0}(1))$, so $S^{(1)} \neq S^{(2)}$. It follows that $X_0^{(1)} \oplus X_0^{(2)} = S^{(1)} \oplus S^{(2)} \neq 0^n$. Injectivity of E_K then gives $N^{(1)} \neq N^{(2)}$.

Table 1 shows where each encoding places the blocks of the encoded plaintext M when there is one redundancy block $Z = 0^n$. In each row, U denotes the first block of the plaintext P, and the labels L_i denote blocks in the left part of M. The splice uses the two right-side positions that hold R_1 and R_2. Decryption under the base tweak preserves U and Z.

4.1 Ciphertext Forgery

Attack steps.

Table 1: Encoded layouts for the nonce-respecting EtE attacks.

Encoding	Encoded blocks	Encoded layout (M_0, M_l, M_r)
EtE_A	7	$(U, L_1 \| L_2 \| L_3, R_1 \| R_2 \| Z)$
EtE_S	5	$(U, Z \| L_2, R_1 \| R_2)$
EtE_P	5	$(Z, U \| L_2, R_1 \| R_2)$

1. Choose a plaintext P whose encoding has the layout in Table 1. Request encryption of P under $T_i = A \| \nu_i$ for $i \in \{1, 2\}$, with $\nu_1 \neq \nu_2$, obtaining $C^{(1)}, C^{(2)}$.

2. In $C^{(1)}$, replace right-side blocks $1, 2$ with the corresponding blocks of $C^{(2)}$. Submit the resulting ciphertext $C^\star$ for verification at the base tweak T_1.

Forgery success probability. The nonce choice gives $N^{(1)} \neq N^{(2)}$. By Section 3.3, the spliced ciphertext differs from $C^{(1)}$, the only ciphertext returned at T_1. Decrypting it preserves every block of the encoded plaintext outside the two selected positions, including Z. The decoder therefore accepts with probability one. The attack uses two encryption queries at distinct nonces and one final verification.

4.2 Challenge-Bit Recovery

Attack steps.

1. Choose two equal-length plaintexts $P^{(0)}, P^{(1)}$ for the same encoding in Table 1. Their encoded plaintexts differ only in U, which takes the distinct values U_0, U_1.

2. Make an auxiliary encryption query on $P^{(0)}$ at $T_{\text{aux}} = A \| \nu_{\text{aux}}$, obtaining C^{aux}. Submit $P^{(0)}, P^{(1)}$ as the challenge pair at $T_{\text{ch}} = A \| \nu_{\text{ch}}$ with $\nu_{\text{ch}} \neq \nu_{\text{aux}}$. Let C^{ch} be the encryption of the chosen $P^{(b)}$.

3. If $C^{\text{ch}}_{r,1} = C^{\text{aux}}_{r,1}$, output 1. Otherwise, replace right-side blocks $1, 2$ of C^{ch} with the corresponding blocks of C^{aux} and query decryption at T_{ch}. Read the first block of the returned plaintext and output the bit corresponding to U_0 or U_1.

Attack result. If $b = 0$, the auxiliary and challenge encryptions use the same encoded plaintext and associated data, with distinct nonces. The center masks differ, and the outputs of the first right-side layer agree at position 1, so the ciphertext blocks there differ. Equality in step 3 thus identifies $b = 1$.

When the compared blocks differ, the splice changes at least the first right-side block of the challenge ciphertext, so the decryption query at the challenge tweak is legal. For either challenge bit, the challenge and auxiliary encoded plaintexts agree at the two selected positions. The cross-tweak splice therefore preserves the redundancy Z and the challenge block U_b. The decoder accepts and returns a plaintext with first block U_b, which identifies b.

The adversary recovers the challenge bit with probability one and has normalized advantage one. It uses one auxiliary encryption and one challenge encryption at distinct nonces, and at most one decryption query. It makes no decryption query in the equality branch.

Redundancy spanning multiple full blocks. For $\tau = qn$ with $q \geq 1$, choose a supported encoded length with two right-side positions outside the q zero blocks. For IND-CCA, choose a plaintext block whose encoded position lies outside the splice and vary it between the two candidates. The two-block splice then preserves every zero block, and the same attacks apply.

5 Conclusion and Discussion

The five-block distinguisher breaks the STPRP security of AESpolyW. The construction uses the same center mask at every right-side position. Copying blocks at an even set of positions can therefore leave both right-side checksums unchanged. This cancellation forces the block-cipher inversion at the center to take a block-cipher output from an earlier encryption as its input. The published proof overlooks this case.

The key fixes the right-side position masks across tweaks. A two-block splice across distinct tweaks preserves the base plaintext at every other position if the plaintexts match at the selected positions. The splice changes the base ciphertext when the center masks differ. With one block each of associated data and nonce in $T = A\|\nu$, all three EtE encodings admit forgeries and IND-CCA challenge-bit recovery with probability one, even when the encryption queries use distinct nonces.

Position-dependent center masks. One possible repair uses a position-dependent multiple of the center mask at each right-side position, $X_{3,j} = Y_{1,j} \oplus \lambda_j N$. For up to n right-side blocks, the coefficients $1, x, \ldots, x^{n-1}$ in the polynomial representation of the field are linearly independent over $\mathbb{F}_2$. Every nonempty set of positions then has a nonzero coefficient sum. Any sequence of more than n coefficients in the field $\mathbb{F}_{2^n}$ of 2^n elements has a nonempty subset whose XOR is zero. This property accounts for the attack on overlong EME inputs [HR04].

For longer right branches, EME* divides the data into chunks of at most n blocks and uses a distinct mask for each chunk [Hal04]. IEEE Std 1619.2-2025 specifies the related EME2-AES mode [IEE25]. AESpolyW could derive a separate center mask for each right-side chunk and multiply it by independent coefficients within that chunk. Consider a splice within one chunk where the first-layer outputs match at the selected positions. If the center masks differ between the two queries, the splice produces a nonzero checksum difference. Generating these masks and processing the chunks adds work. A security proof for the modified construction would need to cover same-position splicing across chunks, mixed encryption and decryption queries, variable input lengths, and queries under multiple independently chosen keys.

Disclosure. We disclosed our initial AESpolyW findings to all four authors of the design in July 2026.

References

[BA25] Amit Singh Bhati and Elena Andreeva. Breaking the IEEE encryption standard XCB-AES in two queries. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025, Proceedings, Part V*, volume 16004 of *Lecture Notes in Computer Science*, pages 172–199. Springer, 2025. https://doi.org/10.1007/978-3-032-01901-1_6. 2

[BN15] Ritam Bhaumik and Mridul Nandi. An inverse-free single-keyed tweakable enciphering scheme. In *Advances in Cryptology - ASIACRYPT 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 159–180. Springer, 2015. https://doi.org/10.1007/978-3-662-48800-3_7. 2

[BR00] Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330. Springer, 2000. https://doi.org/10.1007/3-540-44448-3_24. 2

[CCH+26] Donghoon Chang, Yu Long Chen, Yukihito Hiraga, Kazuhiko Minematsu, Nicky Mouha, Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Key committing security of HCTR2, revisited. In *Advances in Cryptology – CRYPTO 2026, Proceedings, Part VI*, volume 16805 of *Lecture Notes in Computer Science*, pages 98–130. Springer, 2026. https://doi.org/10.1007/978-3-032-35412-9_4. 6

[CDD+25] Yu Long Chen, Michael Davidson, Morris Dworkin, John Kelsey, Yu Sasaki, Meltem Sönmez Turan, Donghoon Chang, Nicky Mouha, and Alyssa Thompson. Requirements for cryptographic accordions. NIST Interagency Report 8552, National Institute of Standards and Technology, Gaithersburg, MD, April 2025. https://doi.org/10.6028/NIST.IR.8552. 2

[CHB21] Paul Crowley, Nathan Huckleberry, and Eric Biggers. Length-preserving encryption with HCTR2. Cryptology ePrint Archive, Paper 2021/1441, 2021. Revised November 2023. https://eprint.iacr.org/2021/1441. 2

[DDG+25] Nilanjan Datta, Avijit Dutta, Shibam Ghosh, Eik List, and Hrithik Nandi. HCTR+: An optimally secure TBC-based accordion mode. *IACR Transactions on Symmetric Cryptology*, 2025(3):183–229, 2025. See also the March 2026 revision of ePrint 2024/2053. https://doi.org/10.46586/tosc.v2025.i3.183-229. 2

[DMMT25] Christoph Dobraunig, Krystian Matusiewicz, Bart Mennink, and Alexander Tereschenko. Efficient instances of docked double decker with AES, and application to authenticated encryption. In *Advances in Cryptology – EUROCRYPT 2025, Proceedings, Part I*, volume 15601 of *Lecture Notes in Computer Science*, pages 62–92. Springer, 2025. https://doi.org/10.1007/978-3-031-91107-1_3. 2

[Hal04] Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In *Progress in Cryptology - INDOCRYPT 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*, pages 315–327. Springer, 2004. https://doi.org/10.1007/978-3-540-30556-9_25. 2, 13

[Hal07] Shai Halevi. Invertible universal hashing and the TET encryption mode. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 412–429. Springer, 2007. https://doi.org/10.1007/978-3-540-74143-5_23. 2

[HKR15] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 15–44. Springer, 2015. Full version available at https://www.cs.ucdavis.edu/~rogaway/aez/rae.pdf. https://doi.org/10.1007/978-3-662-46800-5_2. 2, 3, 6

[HNSS26] Yukihito Hiraga, Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. AESpoly: Symmetric-key cryptographic designs using instruction-level parallelism between AES and polynomial hash. In *2026 IEEE Symposium on Security and Privacy (SP)*, pages 696–715. IEEE Computer Society, May 2026. https://doi.org/10.1109/SP63933.2026.00138. 2, 4, 5, 8, 9, 10

[HR03] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2003. https://doi.org/10.1007/978-3-540-45146-4_28. 2

[HR04] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In *Topics in Cryptology - CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2004. Full version available at https://www.cs.ucdavis.edu/~rogaway/papers/eme.pdf. https://doi.org/10.1007/978-3-540-24660-2_23. 2, 3, 13

[IEE25] IEEE. IEEE standard for wide-block encryption for shared storage media. IEEE Std 1619.2-2025, November 2025. IEEE Standards Association. https://standards.ieee.org/ieee/1619.2/11949/. 13

[Lee24] Byeonghak Lee. A BBB secure accordion mode from HCTR. Presentation at the NIST Workshop on the Requirements for an Accordion Cipher Mode, June 2024. Slides. https://csrc.nist.gov/presentations/2024/a-bbb-secure-accordion-mode-from-hctr. 2

[MF07] David A. McGrew and Scott R. Fluhrer. The security of the extended codebook (XCB) mode of operation. In *Selected Areas in Cryptography, 14th International Workshop, SAC 2007*, volume 4876 of *Lecture Notes in Computer Science*, pages 311–327. Springer, 2007. https://doi.org/10.1007/978-3-540-77360-3_20. 2

[WFW05] Peng Wang, Dengguo Feng, and Wenling Wu. HCTR: A variable-input-length enciphering mode. In *Information Security and Cryptology, First SKLOIS Conference, CISC 2005*, volume 3822 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2005. https://doi.org/10.1007/11599548_15. 2

[WMX+25] Peng Wang, Shuping Mao, Ruozhou Xu, Jiwu Jing, and Yuewu Wang. How to recover the full plaintext of XCB. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025, Proceedings, Part V*, volume 16004 of *Lecture Notes in Computer Science*, pages 230–259. Springer, 2025. https://doi.org/10.1007/978-3-032-01901-1_8. 2

A Experimental Verification

Setup. We tested AESpolyW with AES-128 and a 128-bit hash key. As a baseline, we reproduced one known-answer test vector published by the designers. We then ran 64 trials for each case in Table 2, varying the keys and the input blocks subject to the constraints of each attack. We checked that both original ciphertexts decrypted to their respective inputs in every trial.

Splicing distinguishers. For the center-difference attack (Section 3.1), we encrypted two five-block messages that differed only in the center block under a common four-block tweak. For the cross-tweak attack (Section 3.3), we encrypted the same five-block message under two distinct tweaks. With a common left tweak and distinct one-block right tweaks, all corresponding noncenter ciphertext blocks differed. To exercise the equality branch, we used two-block right tweaks chosen with knowledge of the key so that the two inputs to the block cipher in the right-tweak hash were exchanged. Their output XOR was unchanged. All corresponding noncenter ciphertext blocks then agreed, and the distinguisher accepted at the comparison step.

In the trials that produced a splice, we copied the first two right-side ciphertext blocks from the second response to the first and decrypted at the base tweak. We checked that the splice was fresh at that tweak and that exactly the two selected plaintext blocks changed. We also checked that the base center mask and the checksums V_l, V_r, U_r were preserved and that re-encrypting the recovered plaintext reproduced the splice.

EtE forgeries. For each encoding in Section 4.1, we encrypted the same payload twice, with fixed associated data and distinct nonces. The appended encoding used seven blocks, with the zero block at position 7. The inserted and prepended encodings used five blocks, with the zero block at position 2 or 1, respectively. We spliced the first two right-side ciphertext blocks and submitted the result at the base tweak. In every trial the decoder accepted and returned a different payload whose first block matched that of the original. As a control, we flipped one bit of the redundancy block in the recovered encoding before decoding. All 192 modified encodings were rejected.

Table 2: Observed results for the AESpolyW splicing and EtE forgery tests.

Case	Observed outcome	Passed
Center-difference splice	Only the two selected blocks changed	64/64
Cross-tweak, all different	Only the two selected blocks changed	64/64
Cross-tweak, all equal	Accepted at the comparison step	64/64
EtE with appended zero	Fresh ciphertext accepted	64/64
EtE with inserted zero	Fresh ciphertext accepted	64/64
EtE with prepended zero	Fresh ciphertext accepted	64/64

All 384 trials gave the predicted outcome. Of these, 320 produced and verified a splice, and 64 exercised the cross-tweak equality branch.

B AESpolyW Pseudocode

In Algorithm 1, $\rho_r = |M_r| = |C_r|$ and $t_r = |T_r|$ are bit lengths. A partition arrow specifies either the bit lengths of the parts or a block size. With a specified block size, the final block may be shorter if the input length is not a multiple of that size. The algorithm passes full blocks to XE^*, EX^*, PHASH, and XOR_n. In XCTR, $\mathsf{str}_n(i)$ encodes the counter i in n bits. The AES-128 instantiation tested here uses little-endian byte order, placing the least significant byte first. The operation $\mathsf{msb}_k(D)$ returns the leftmost k bits of D.

Algorithm 1 AESpolyW

Encryption AESpolyW.E$[E_K, H_H](T, M)$

$\Delta_0 \leftarrow E_K(0^n)$; $\Delta_1 \leftarrow E_K(1^n)$
$\rho_r \leftarrow n\lfloor 0.5(|M|_n - 1)\rfloor$; $m_r = \lfloor 0.5(|M|_n - 1)\rfloor$; $t_r \leftarrow n\lfloor 0.5|T|_n\rfloor$
$(M_0, M_l, M_r) \xleftarrow{n,\,|M|-n-\rho_r,\,\rho_r} M$
$(T_l, T_r) \xleftarrow{|T|-t_r,\ t_r} T$
$S \leftarrow \mathsf{PHASH}[E_K](\Delta_0, T_r)$; $U_l \leftarrow H_H(T_l, M_l)$
$Y_1 \leftarrow \mathsf{XE}^*[E_K](\Delta_1, M_r)$; $U_r \leftarrow \mathsf{XOR}_n(Y_1 \| S)$
$X_0 \leftarrow U_l \oplus U_r \oplus M_0$; $N \leftarrow E_K(X_0)$
$C_l \leftarrow \mathsf{XCTR}[E_K](N, M_l)$; $X_3 \leftarrow Y_1 \oplus \mathsf{Copy}_{m_r}(N)$
$Y_0 \leftarrow E_K(N)$; $V_l \leftarrow H_H(T_l, C_l)$
$V_r \leftarrow \mathsf{XOR}_n(X_3 \| S)$; $C_r \leftarrow \mathsf{EX}^*[E_K](\Delta_1, X_3)$
$C_0 \leftarrow V_l \oplus V_r \oplus Y_0$; $C \leftarrow C_0 \| C_l \| C_r$
return C

Decryption AESpolyW.D$[E_K, E_K^{-1}, H_H](T, C)$

$\Delta_0 \leftarrow E_K(0^n)$; $\Delta_1 \leftarrow E_K(1^n)$
$\rho_r \leftarrow n\lfloor 0.5(|C|_n - 1)\rfloor$; $m_r = \lfloor 0.5(|C|_n - 1)\rfloor$; $t_r \leftarrow n\lfloor 0.5|T|_n\rfloor$
$(C_0, C_l, C_r) \xleftarrow{n,\ |C|-n-\rho_r,\ \rho_r} C$
$(T_l, T_r) \xleftarrow{|T|-t_r,\ t_r} T$
$S \leftarrow \mathsf{PHASH}[E_K](\Delta_0, T_r)$; $V_l \leftarrow H_H(T_l, C_l)$
$X_3 \leftarrow \mathsf{XE}^*[E_K^{-1}](\Delta_1, C_r)$; $V_r \leftarrow \mathsf{XOR}_n(X_3 \| S)$
$Y_0 \leftarrow V_l \oplus V_r \oplus C_0$; $N \leftarrow E_K^{-1}(Y_0)$
$M_l \leftarrow \mathsf{XCTR}[E_K](N, C_l)$; $Y_1 \leftarrow X_3 \oplus \mathsf{Copy}_{m_r}(N)$
$M_r \leftarrow \mathsf{EX}^*[E_K^{-1}](\Delta_1, Y_1)$; $X_0 \leftarrow E_K^{-1}(N)$
$U_l \leftarrow H_H(T_l, M_l)$; $U_r \leftarrow \mathsf{XOR}_n(Y_1 \| S)$
$M_0 \leftarrow U_l \oplus U_r \oplus X_0$
$M \leftarrow M_0 \| M_l \| M_r$
return M

Subroutine $\mathsf{XCTR}[E_K](N, D)$

For $i \in [|D|_n]$ **do** $Y_{2,i} \leftarrow E_K(N \oplus \mathsf{str}_n(i))$ **end for**
$Y_2 \leftarrow \mathsf{msb}_{|D|}(Y_{2,1} \| \cdots \| Y_{2,|D|_n})$
$D' \leftarrow D \oplus Y_2$
return D'

Subroutine $\mathsf{XE}^*[E_K](\Delta, D)$

if $D = \varepsilon$ **then return** ε **end if**
$D_1, \ldots, D_{|D|_n} \xleftarrow{n} D$; $D' \leftarrow \varepsilon$
For $i \in [|D|_n]$ **do** $D' \leftarrow D' \| E_K(h_\Delta(i) \oplus D_i)$ **end for**
return D'

Subroutine $\mathsf{EX}^*[E_K](\Delta, D)$

if $D = \varepsilon$ **then return** ε **end if**
$D_1, \ldots, D_{|D|_n} \xleftarrow{n} D$; $D' \leftarrow \varepsilon$
For $i \in [|D|_n]$ **do** $D' \leftarrow D' \| \big(h_\Delta(i) \oplus E_K(D_i)\big)$ **end for**
return D'

Subroutine $\mathsf{PHASH}[E_K](\Delta, D)$

38: **if** $D = \varepsilon$ **then return** 0^n **end if**
39: $D' \leftarrow \mathsf{XE}^*[E_K](\Delta, D)$; $S \leftarrow \mathsf{XOR}_n(D')$
40: **return** S

Subroutine $\mathsf{Copy}_c(N)$

41: $Z \leftarrow \varepsilon$
42: **For** $i \in [c]$ **do** $Z \leftarrow Z \| N$ **end for**
43: **return** Z

Subroutine $\mathsf{XOR}_n(D)$

44: $D_1, \ldots, D_{|D|_n} \xleftarrow{n} D$
45: **return** $\bigoplus_{i \in [|D|_n]} D_i$