Revisiting Hash–ECB–Hash Enciphering: Plaintext Recovery Attacks on PEP and TET

Hailun Yan[1,2], Yuexing Yue[1], Yaobin Shen[3], Peng Wang[a,1] and Siwei Sun[1,2]

[1] School of Cryptology, University of Chinese Academy of Sciences, Beijing, China
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing, China
[3] School of Informatics, Xiamen University, Xiamen, China

Abstract

Storage encryption motivates tweakable enciphering schemes, which encrypt entire multi-block data units without length expansion and use a public tweak to identify their context. We revisit PEP (FSE 2006) and TET (CRYPTO 2007), two Hash–ECB–Hash schemes that combine parallel block-cipher calls with invertible mixing, and present classical distinguishing and plaintext-recovery attacks.

For PEP, we exhibit a fixed public field satisfying the published parameter-selection rule whose generated position coefficients repeat. This yields a two-query algebraic distinguisher and recovery of a secret tweak-derived value using one encryption query. For the same instance, we recover target plaintexts over a specified range of complete-block lengths using at most nine encryption queries and one decryption query. Our PEP success bounds assume a uniformly random secret permutation and independently fixed target tweaks and plaintexts.

For TET, we analyze two corrections of a previously reported mismatch between encryption and decryption. Both admit two-query distinguishers on messages with a partial final block. For tails of at least two bits, we obtain deterministic plaintext recovery using two decryption queries in one correction, or two encryption queries and one decryption query in the other. These TET attacks hold for every data permutation and valid parameter choice. The results expose failures of concrete coefficient generation and partial-block processing, and identify requirements for repairing these constructions.

Keywords: Tweakable enciphering schemes, Hash–ECB–Hash, PEP, TET, Distinguishing attacks, Plaintext recovery

1 Introduction

Unlike a block cipher, which operates on one fixed-length block, disk-sector encryption typically handles multi-block data units without expanding their length. A tweakable enciphering scheme (TES) [HR03] provides this length-preserving interface, with a public *tweak* identifying the context, such as a sector address. Liskov, Rivest, and Wagner [LRW02] formalized this additional public input through the *tweakable block cipher* framework, separating its role from that of the secret key. In 2024, NIST held a workshop on the requirements for an *accordion cipher mode*, for which Chen et al. [CDD+24] prepared a

Emails: hailun.yan@ucas.ac.cn (Hailun Yan), yueyuexing24@mails.ucas.ac.cn (Yuexing Yue), yaobin.shen@xmu.edu.cn (Yaobin Shen), p-wang@ucas.ac.cn (Peng Wang), siweisun.isaac@gmail.com (Siwei Sun)
[a]Peng Wang is the corresponding author.

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2026-10-30.

discussion draft outlining requirements for a block-cipher-based tweakable variable-input-length strong pseudorandom permutation (VIL-SPRP).

The standard goal is tweakable strong pseudorandom-permutation (tSPRP) security [HR03, LRW11]: adaptive encryption and decryption queries should not distinguish the scheme from independent uniform permutations for the supported tweak–length domains, together with their inverses. Repeated tweaks are permitted and answered consistently. The goal concerns the entire message, including relations among its blocks. Separately, correctness requires decryption to invert encryption on every supported input. Note, however, that this interface is deterministic for a fixed key and tweak and does not itself provide ciphertext authentication: every supported ciphertext has a preimage. This distinction is relevant to NIST's accordion framework, which separates the core permutation from its derived functions [CDD+25].

Common TES designs can be grouped into three structural approaches according to how block-cipher processing and message-wide mixing are arranged:

- **Encrypt–Mix–Encrypt.** A mixing step separates two block-cipher processing layers, as in Halevi and Rogaway's CMC and EME [HR03, HR04].
- **Hash–CTR–Hash.** Hashing layers surround a central counter-mode processing layer, as in Wang, Feng, and Wu's HCTR [WFW05] and Chakraborty and Sarkar's HCH [CS06a].
- **Hash–ECB–Hash.** The same block cipher is applied independently to each block between invertible keyed hashing layers, as in PEP [CS06b], TET [Hal07a], and HEH [Sar07].

The latter two approaches differ in how their central cipher-processing layer operates.

Our focus is Hash–ECB–Hash, which aims to retain the parallelism of ECB encryption while hiding the equality patterns that ECB exposes when applied directly to plaintext blocks. This approach addresses the problem of efficiently constructing a wide permutation from a narrow block cipher. In its complete-block core, a secret invertible transformation first mixes the message, the same block permutation is then applied to each resulting coordinate, and the inverse of a second mixing transformation produces the ciphertext. The central ECB layer thus operates on mixed internal coordinates, and its calls can be evaluated in parallel [NR]. Despite the term "Hash", the surrounding maps are length-preserving permutations: they use universal-hashing ideas to mix the block vector without discarding the information needed for inversion. These maps may themselves contain additional block-cipher calls.

This structure imposes two distinct mathematical requirements. Invertibility ensures that decryption exists, whereas suitable collision bounds control unintended collisions among central-layer inputs or among central-layer outputs, with repeated queries and inverse queries whose answers are already known handled separately. These bounds must cover positions within a query, across different queries, and in both query directions [NR99, Hal07a]. The parallel layer preserves input equality exactly. A deterministic internal repetition can therefore become an observable relation even when the underlying block permutation is ideal. Whether the final mixing hides or preserves that relation depends on the complete mode; ECB alone does not determine the outcome.

From a general framework to concrete designs. Naor and Reingold [NR99] established a general framework for constructing pseudorandom permutations using universal hashing. Their companion note describes an encryption mode that places invertible keyed mixing around a parallel block-cipher layer [NR]. This framework highlights the role of efficient, invertible, blockwise-universal transformations in extending a block cipher to a wider domain.

At FSE 2006, Chakraborty and Sarkar [CS06b] developed this approach into PEP, a parallelizable, single-key tweakable enciphering mode for complete-block messages. Its mixing layers use public position coefficients and secret polynomial scaling. The number of complete blocks may vary up to a maximum fixed by the public parameters, and the tweak is one block long; PEP does not provide an algorithm for a final partial block. These concrete algebraic choices make the separation of positions depend on an explicit coefficient-generation rule. This is a practical step beyond an abstract hash-family assumption, but it requires the generator to satisfy that assumption for every public parameter choice admitted by the specification.

At CRYPTO 2007, Halevi [Hal07a] explicitly formalized the blockwise-universality condition required by this structure and constructed efficient invertible hashing to design TET. Blockwise universality bounds the probability that two distinct message–position pairs produce equal output blocks, including two positions within the same message; its XOR variant also bounds the probability of every fixed output difference. TET accommodates variable-length tweaks and messages ending in a partial block. Its mixing combines a rank-one update of the identity with a position mask, bringing the cost of mixing and inversion close to that of polynomial evaluation. Section 2.5 presents these hashing properties and their role at both boundaries of the ECB layer. The partial-block algorithm must additionally specify how the tail enters the full-block processing.

Sarkar's HEH [Sar07] uses a different invertible blockwise-universal hash family that removes TET's restrictions on the hashing key and addresses the associated inversion and key-setup costs. He subsequently simplified the invertible hashing in the Naor–Reingold framework and studied polynomial and BRW-polynomial hash instantiations [Sar08]. That work considers different central processing modes; its ECB branch belongs to the structural class considered here.

A later line of work places these constructions in a substitution–permutation-network (SPN) perspective. Cogliati et al. [CDK+18] view the earlier constructions with secret permutations as one-round nonlinear predecessors and study security with public permutations and additional rounds. Cogliati et al. [CEL+21] subsequently proposed CTET+, which reuses one secret permutation across two substitution layers and achieves beyond-birthday-bound security. This use of a single permutation differs from PEP's use of a single block-cipher key for all its operations. Jha and Nandi [JN18] also revisited the abstract complete-block Hash–ECB–Hash composition using the H-technique. These advances clarify proof methods, the distinction between public and secret primitives, and the role of multiple layers. They do not, by themselves, verify the public coefficient generator or partial-block pseudocode of an earlier concrete mode.

Previous Work. Earlier work records both a correctness issue in TET and quantum attacks on PEP and TET. The third-party `tet-heh` implementation documents the mismatch between TET's printed formulas and a correction that preserves printed decryption [ody26]. We credit this record and do not claim the correctness observation as new. Ghosh and Sarkar [GS21] give Simon-based quantum attacks on TET and other schemes; Mao, Wang, and Hu [MWH24] analyze PEP and give a variable-tweak Simon attack on TET. These attacks use quantum queries, whereas our procedures use classical queries and exploit coefficient collisions and tail cancellation.

Classical attacks on related constructions also provide methodological precedents. Nandi [Nan14] shows that an admissible almost-XOR-universal (AXU) hash instantiation can force internal repetitions in abstract POE/POET . This challenges the sufficiency of the abstract hash assumption, but does not establish an attack on the concrete POET instantiations submitted to CAESAR. Liu et al. [LWWY26] revisit the collision conditions for online Hash–ECB–Hash encryption, whose online-permutation target differs from the wide-block target considered here. Bhati and Andreeva's shared-difference attack on

XCB-AES recovers plaintext with one encryption and one decryption and identifies a flaw in its original security proof [BA25]. XCB has a different central architecture and serves here as a methodological comparison.

Our Contributions. We revisit PEP and TET as early concrete designs in the Naor–Reingold Hash–ECB–Hash line. PEP supplies explicit parameters, complete algorithms, and a concrete security analysis; TET formalizes blockwise universality and supplies efficient invertible hashing [CS06b, Hal07a]. Our question is how the framework's security conditions are reflected in the concrete parameters, mixing transformations, and complete encryption and decryption procedures. We use classical chosen-plaintext and chosen-ciphertext queries to the external TES interfaces; the adversary receives neither the secret hashing parameters nor direct oracle access to the underlying block permutation. Our main results (Table 1) are as follows.

Table 1: Main results and scope.

(a) PEP: the same fixed public field instance, $n = 128$.

Attack / goal	Mechanism	Length	Queries	Advantage / success
1: Distinguishing	Repeated position coefficients	$m = 1337$	2 Enc	$1 - O(N^{-1})$
2: First recovery of R	Invertible-power recovery of R	$h = 1340$	1 Enc	$\geq 1 - 1355/(N - 1343)$
3: Plaintext recovery	Anchored auxiliary inversion	$3 \leq m \leq 1338$	9 Enc, 1 Dec	$> 1 - 2^{-109}$

(b) TET: printed correctness and attacks on the two corrections.

Printed TET tail: mismatched inverse coefficients cause round-trip failure (Sec. 4.1).

Attack / goal	Version	Mechanism	Queries	Advantage / success
4: Distinguishing	$\mathsf{TET}_{\mathrm{B}}$	Inverse-path reuse	2 Dec	$1 - (N-1)/(2^{mn+r} - 1)$
	$\mathsf{TET}_{\mathrm{A}}$	Forward-path reuse	2 Enc	$1 - (N-1)/(2^{mn+r} - 1)$
5: Plaintext recovery	$\mathsf{TET}_{\mathrm{B}}$	Target plaintext recovery	2 Dec	1
	$\mathsf{TET}_{\mathrm{A}}$	Target plaintext recovery	2 Enc, 1 Dec	1

Notation. n is the block length, $N = 2^n$, m counts full blocks, and r is the tail length in bits. Enc/Dec count external encryption/decryption queries. The last column gives distinguishing advantage or recovery success probability, according to the attack goal.

PEP conditions. Recovery queries have $h = 1340$ blocks. The bounds assume a uniform secret permutation and include zero-value failures. The first-recovery bound uses an independently fixed tweak and an empty evaluation history; plaintext recovery fixes the target tweak and plaintext independently of the permutation.

TET conditions. The printed encryption and decryption are not mutual inverses, as previously recorded. The length satisfies $n \leq mn + r < N$. Distinguishing requires $1 \leq r < n$; the plaintext-recovery results shown require $2 \leq r < n$. Recovery for $\mathsf{TET}_{\mathrm{A}}$ also requires a valid auxiliary tweak.

- **A counterexample to PEP's coefficient generator.** We give a fixed 128-bit field representation satisfying the published parameter-selection rule whose generator repeats position coefficients at 1337 blocks. An overlooked four-term relation makes two zero-plaintext coordinates enter ECB identically, producing proportional ciphertext coordinates. Attack 1 uses a two-query cross-product test to distinguish this instance with advantage close to one. The counterexample refutes the generator's universal guarantee, rather than establishing an attack for every field representation. Appendix B provides exact arithmetic verification; Section 3.2 accounts for the zero-multiplier branch.
- **Derived-value and plaintext recovery for PEP.** In the same field, the coefficients at positions 5 and 1339 repeat at $h = 1340$ blocks. Attack 2 recovers $R = E_K(T)$ from

a ratio of nonzero ciphertext coordinates. Combining this recovery with an auxiliary decryption under another tweak and a known-preimage anchor, Attack 3 recovers an m-block target plaintext for $3 \le m \le 1338$ using at most nine encryptions and one decryption, each of 1340 blocks. Its success bound assumes a uniform secret permutation and target tweaks and plaintexts fixed independently of that permutation, and accounts for zero values and adaptive calls.

- **Path reuse and plaintext recovery in two TET corrections.** We analyze two inverse-consistent variants: $\mathsf{TET}_{\mathrm{B}}$ preserves printed encryption and corrects decryption, whereas $\mathsf{TET}_{\mathrm{A}}$ preserves printed decryption and adjusts encryption. Section 4.1 specifies both mappings. In either version, public differences cancel at a mixing boundary, making distinct inputs reuse internal permutation paths. Attack 4 gives two-query distinguishers for $1 \le r < n$, where r is the final partial-block length in bits. Attack 5 recovers target plaintexts for $2 \le r < n$, using two decryptions for $\mathsf{TET}_{\mathrm{B}}$, or two encryptions under an auxiliary tweak and one decryption for $\mathsf{TET}_{\mathrm{A}}$. These relations hold for every data permutation and valid parameter choice, and no recovery decryption submits the target itself.

We give explicit query procedures and observable tests, derive exact ideal-world probabilities, and distinguish real-world identities from the probability bounds needed for PEP's exceptional branch. Together, these results examine public-parameter validation and tail integration at both permutation boundaries.

Organization. Section 2 introduces notation and the constructions. Sections 3 and 4 analyze PEP and TET; Section 5 discusses repair requirements, and Section 6 concludes. Appendices A–D give construction procedures, public-parameter verifications, probability proofs, and further attacks and extensions.

2 Preliminaries

2.1 Notation and Definitions

- $n \ge 2$ is the block size in bits of the underlying block cipher, $N = 2^n$.
- $\mathbb{F} = \mathbb{F}_{2^n}$ is the field of N elements, identified with n-bit strings through a fixed binary basis. Write $\mathbb{F}^\times = \mathbb{F} \setminus \{0\}$. Addition in $\mathbb{F}$, also written $\oplus$, is bitwise XOR; products, powers, and quotients denote field operations, with division defined only for nonzero denominators.
- M and C denote complete plaintext and ciphertext strings of the same bit length $L \ge n$. The integer $m = \lfloor L/n \rfloor$ counts their full blocks; $\boldsymbol{M}, \boldsymbol{C} \in \mathbb{F}^m$ are the corresponding block vectors. The vectors $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{F}^m$ are the inputs and outputs of the central block-cipher layer.
- Bold vectors are columns, and T denotes transpose. Unless stated otherwise, coordinate indices and sums range over $1, \ldots, m$.
- $T \in \mathcal{T}$ is a public tweak, and $K \in \mathcal{K}$ denotes the scheme's secret key material. Each mode specifies its tweak space and key components below.
- $E_K : \mathbb{F} \to \mathbb{F}$ and E_K^{-1} denote block encryption and decryption. The map $E_K^{\otimes m}$ applies the same permutation to every coordinate; it is also denoted by ECB_{E_K} in the figures.

- $H^{\text{in}}_{K,T}, H^{\text{out}}_{K,T} : \mathbb{F}^m \to \mathbb{F}^m$ are the invertible mixing maps entering the central primitive from the encryption and decryption sides, respectively. Their dependence on the message length is implicit.
- $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ denotes a tweakable enciphering scheme (TES). For each key and tweak, Enc^T_K is a length-preserving permutation on each supported message length, with inverse Dec^T_K.
- π denotes a uniformly sampled secret permutation on $\mathbb{F}$ in the ideal-primitive analysis. In contrast, Π^T denotes the ideal message permutation at tweak T, sampled independently for each supported tweak–length domain.
- $\mathsf{bin}_n(u)$ is the n-bit binary encoding of an integer $0 \le u < N$, with the most significant bit first and leading zeroes retained. The same string may be viewed as a field element. For strings, $|z|$ is bit length, $u \parallel v$ is concatenation, and 0^k is a string of k zeroes.

2.2 Hash–ECB–Hash

The construction of Naor and Reingold [NR99] surrounds coordinatewise encryption with two efficiently invertible keyed maps. For a fixed tweak and complete-block length, the construction in Figure 1 and its inverse are

$$\begin{aligned} \mathsf{Enc}^T_K &= (H^{\text{out}}_{K,T})^{-1} \circ E_K^{\otimes m} \circ H^{\text{in}}_{K,T}, \\ \mathsf{Dec}^T_K &= (H^{\text{in}}_{K,T})^{-1} \circ (E_K^{-1})^{\otimes m} \circ H^{\text{out}}_{K,T}. \end{aligned} \tag{1}$$

The tweak notation accommodates PEP and TET; the original generic mode is untweaked. Here "hash" means invertible mixing, without compression.

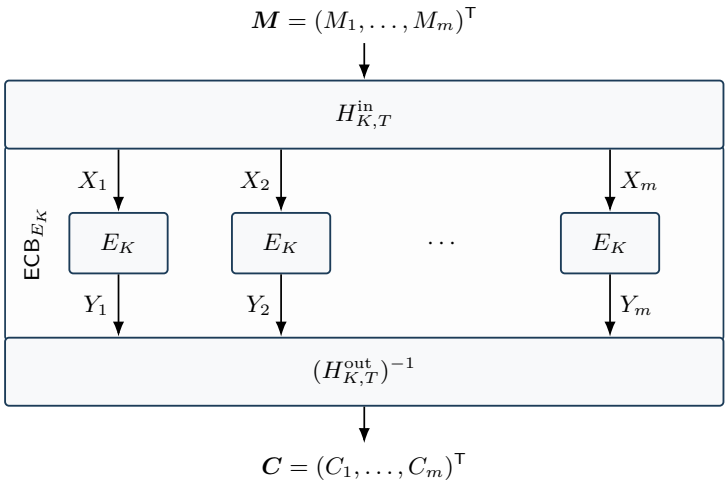

Figure 1: The common complete-block structure. The drawing represents $m \ge 3$; for $m = 1, 2$ the ECB layer has exactly m boxes. The two mixing layers may differ and may contain additional block-cipher calls. The picture asserts no independence between their parameters and the ECB primitive. TET's partial-block branch adds the path in Figure 4.

2.3 PEP

PEP [CS06b] is a tweakable enciphering scheme constructed from a block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. The key space of PEP is the same as that of the underlying block

cipher E, and the tweak space is $\mathcal{T} = \{0,1\}^n$. The message space consists of all binary strings of length mn. In this paper, we consider $3 \le m \le m_{\max} < N$, where $m_{\max}$ is the maximum supported block count.

PEP represents $\mathbb{F}$ as $\mathbb{F}_2[x]/(\rho(x))$, where ρ is a public primitive polynomial of degree n, and sets $\alpha = x \bmod \rho$. The element $\sum_{k=0}^{n-1} z_k \alpha^k$ is encoded by the integer $\sum_k z_k 2^k$ in n bits. The public position coefficients $\mathbf{p} = (p_{m,1}, \ldots, p_{m,m})^{\mathsf{T}}$ are required to be pairwise distinct and satisfy $\sum_i p_{m,i} = 0$.

For each tweak and length, derive

$$R = E_K(T), \qquad a = E_K(R + \mathsf{bin}_n(m)), \qquad b = E_K(\alpha a). \tag{2}$$

Define the diagonal scaling matrix D_R and, for $z \in \mathbb{F}$, the map F_z by

$$D_R = \mathrm{diag}(1, R, \ldots, R^{m-1}), \qquad F_z(\boldsymbol{v}) = \boldsymbol{v} + \mathbf{p} E_K\left(z + \sum_{j=1}^{m} v_j\right). \tag{3}$$

Encryption follows the three stages in Figure 2, adapted from Chakraborty and Sarkar's algorithm [CS06b]. The figure uses $\mathbf{a}$ for our coefficient vector $\mathbf{p}$. Every formula applies to $1 \le i \le m$, and all cipher calls use K.

1. **Input mixing: D_R followed by F_a.**

$$V_i = R^{i-1} M_i, \qquad \mu = E_K\left(a + \sum_{j=1}^{m} V_j\right), \qquad X_i = V_i + p_{m,i}\mu. \tag{4a}$$

2. **Parallel encryption.**

$$Y_i = E_K(X_i). \tag{4b}$$

3. **Output mixing: F_b followed by D_R.**

$$\nu = E_K\left(b + \sum_{j=1}^{m} Y_j\right), \qquad Z_i = Y_i + p_{m,i}\nu, \qquad C_i = R^{i-1} Z_i. \tag{4c}$$

Combining the three stages gives

$$\mathsf{Enc}_K^T = D_R \circ F_b \circ E_K^{\otimes m} \circ F_a \circ D_R. \tag{5}$$

Since $\sum_i p_{m,i} = 0$, F_z preserves the coordinate sum and satisfies $F_z^{-1} = F_z$. For $R \neq 0$, the inverse scaling is $D_R^{-1} = \mathrm{diag}(1, R^{-1}, \ldots, R^{-(m-1)})$. Reversing the encryption layers therefore gives

$$\mathsf{Dec}_K^T = D_R^{-1} \circ F_a \circ (E_K^{-1})^{\otimes m} \circ F_b \circ D_R^{-1}. \tag{6}$$

Algorithm 1 in Appendix A records these operations. Decryption recomputes the same parameters (R, a, b); preservation of the coordinate sum lets it recover both mixing values. The zero-sum condition and $R \neq 0$ suffice for invertibility, whereas coefficient distinctness is a separate security requirement.

Remark 1 (Zero multiplier). The original specification [CS06c] leaves $R = 0$ undefined for decryption. We state exact attack probabilities for the natural evaluation of the forward formulas, and then give bounds valid for any completion that changes only this exceptional branch. At $R = 0$, we evaluate the factor R^0 as 1. That forward evaluation itself need not be a permutation. In an ideal underlying permutation model and for a fixed tweak, the exception has probability N^{-1}. For a concrete block-cipher family, define $\delta_T = \Pr_K[E_K(T) = 0]$, where K is a uniformly sampled PEP key and T is fixed; this probability is not assumed to equal N^{-1}.

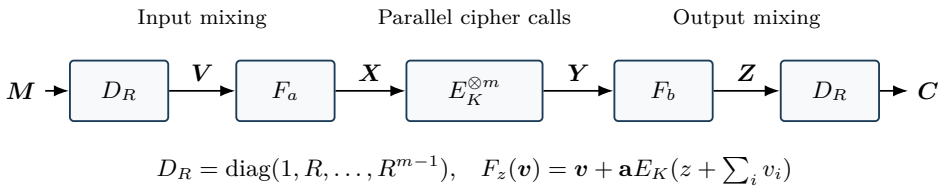

Figure 2: PEP encryption for $m \geq 3$.

2.4 TET

TET [Hal07a] fixes a public primitive element $\alpha \in \mathbb{F}$ and uses a parameter-derivation key K_1 and a separate data key K_2; write $E_K = E_{K_2}$ below. Fix a legal tweak T and a supported length $n \leq L = mn + r < N$, where $0 \leq r < n$. For $r > 0$, write the plaintext and ciphertext as $(\boldsymbol{M}, p)$ and $(\boldsymbol{C}, c)$, with $p, c \in \{0,1\}^r$; for $r = 0$, omit the tails.

Parameter preparation derives the secret scalars $\tau, \beta \in \mathbb{F}$. Choose τ as the first candidate $\mathrm{PRF}_{K_1}(0, j)$, $j = 0, 1, \ldots$, for which $\lambda_m = 1 + \tau + \cdots + \tau^m \neq 0$, and set $\beta = \mathrm{PRF}_{K_1}(L, T)$. The PRF outputs a field element, with integer arguments encoded as n-bit strings. Both directions derive the same parameters from the input length. Thus τ is fixed for a given key and full-block count, but values selected for different counts need not be independent; β depends on both the length and the tweak. Algorithm 2 in Appendix A gives the parameter preparation and scalar mixing procedures.

Let I be the $m \times m$ identity matrix and $\mathbf{1}$ the all-one column. Define the mixing matrices and mask vector by

$$\begin{aligned} v_m &= (\tau^m, \ldots, \tau), & \boldsymbol{b} &= (\beta, \alpha\beta, \ldots, \alpha^{m-1}\beta)^{\mathsf{T}}, \\ B_m &= I + \mathbf{1}v_m, & A_m &= B_m^{-1} = I + \lambda_m^{-1}\mathbf{1}v_m. \end{aligned} \tag{7}$$

The descending powers follow the algorithm's reverse-order Horner evaluation. We use B_m for Halevi's full matrix M_τ and λ_m for his σ; A_m is its inverse, not his rank-one matrix A_τ. These matrices satisfy

$$B_m A_m = A_m B_m = I, \qquad B_m\mathbf{1} = \lambda_m\mathbf{1}, \qquad A_m\mathbf{1} = \lambda_m^{-1}\mathbf{1}.$$

For any vector $\boldsymbol{z}$, Horner's rule computes $s = \sum_{i=1}^{m} \tau^{m-i+1} z_i$ using $s_0 = 0$, $s_i = \tau(s_{i-1} + z_i)$, and $s = s_m$. Hence mixing needs no general matrix multiplication:

$$B_m\boldsymbol{z} = \boldsymbol{z} + s\mathbf{1}, \qquad A_m\boldsymbol{z} = \boldsymbol{z} + \lambda_m^{-1}s\mathbf{1}.$$

For complete blocks ($r = 0$), encryption follows three stages:

1. **Input mixing.**
$$\boldsymbol{X} = A_m\boldsymbol{M} + \boldsymbol{b}. \tag{8a}$$

2. **Parallel encryption.**
$$Y_i = E_K(X_i) \qquad (1 \leq i \leq m). \tag{8b}$$

3. **Output mixing.**
$$\boldsymbol{C} = A_m(\boldsymbol{Y} + \boldsymbol{b}). \tag{8c}$$

Decryption reverses these stages using the same parameters:

$$\boldsymbol{Y} = B_m\boldsymbol{C} + \boldsymbol{b}, \qquad \boldsymbol{X} = (E_K^{-1})^{\otimes m}(\boldsymbol{Y}), \qquad \boldsymbol{M} = B_m(\boldsymbol{X} + \boldsymbol{b}). \tag{9}$$

For a partial tail ($1 \le r < n$), define the public injection $d(z) = z \parallel 1 \parallel 0^{n-r-1}$ and let pref_r return the first r bits of a block. Figure 3 depicts three stages, which we denote by $\mathcal{L}_f : (\boldsymbol{M}, p) \mapsto (\boldsymbol{X}, p)$ for input mixing, $\mathcal{P}_K : (\boldsymbol{X}, p) \mapsto (\boldsymbol{Y}, c)$ for the central cipher-and-tail transformation, and $\mathcal{R}_g : (\boldsymbol{Y}, c) \mapsto (\boldsymbol{C}, c)$ for output mixing. Here $f, g \in \mathbb{F}$ are the coefficients of the padded-tail terms at the input and output boundaries, respectively. The printed encryption uses $f = g = 1$ and is given by

$$\begin{aligned} &\boldsymbol{X} = A_m \boldsymbol{M} + d(p)\mathbf{1} + \boldsymbol{b}, \\ &Y_i = E_K(X_i) \quad (i < m), \qquad U = E_K(X_m), \qquad Y_m = E_K(U), \\ &c = p + \mathsf{pref}_r(U), \qquad \boldsymbol{C} = A_m(\boldsymbol{Y} + \boldsymbol{b}) + d(c)\mathbf{1}. \end{aligned} \tag{10}$$

The padded tails are added after division by λ_m.

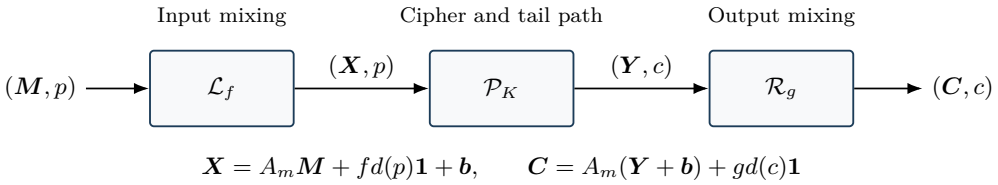

Figure 3: TET encryption.

The central path uses $m + 1$ cipher calls, and its bridge U supplies the tail mask. Its inverse recovers $U = E_K^{-1}(Y_m)$ and $X_m = E_K^{-1}(U)$. Figure 4 shows this last-coordinate path.

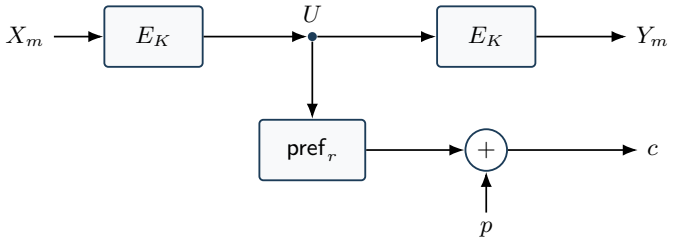

Figure 4: The last full coordinate in TET's partial-block branch. The bridge U is both a permutation output and a permutation input. The remaining coordinates use one ECB call each.

2.5 Blockwise Universality and the Composition Argument

Invertible mixing makes the Hash–ECB–Hash construction decryptable, but does not by itself hide equality between blocks. Because the central layer applies the same permutation to every coordinate, equal inputs produce equal outputs. The relevant security condition [NR99] therefore concerns individual coordinates of the mixing output, rather than only the whole vector.

Blockwise collision conditions. Halevi's definitions [Hal07a] consider $m \ge 1$ and a keyed family $H : \mathcal{K}_H \times \mathbb{F}^m \to \mathbb{F}^m$, with a uniformly sampled key $\kappa \in \mathcal{K}_H$. The family is ϵ-*blockwise universal*, for $0 < \epsilon < 1$, if every fixed pair $(\boldsymbol{z}, i) \ne (\boldsymbol{z}', j)$ satisfies

$$\Pr_{\kappa}[H_\kappa(\boldsymbol{z})_i = H_\kappa(\boldsymbol{z}')_j] \le \epsilon. \tag{11}$$

Here $\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{F}^m$ and $1 \le i, j \le m$. The stronger *ϵ-XOR-blockwise-universal* condition requires, for every fixed $\Delta \in \mathbb{F}$,

$$\Pr_{\kappa}[H_\kappa(\boldsymbol{z})_i + H_\kappa(\boldsymbol{z}')_j = \Delta] \le \epsilon. \tag{12}$$

The vectors, positions, and offset are chosen independently of the key; setting $\Delta = 0$ recovers Equation (11). Crucially, both definitions include two different positions of the *same* message. For example, the identity map is invertible but maps the zero vector to equal blocks at every position, so it fails blockwise universality for $m \ge 2$. Conversely, a collision bound does not guarantee that every keyed map is invertible.

Encryption and decryption boundaries. Equation (1) identifies the two maps whose outputs enter the central primitive: H^{in} on encryption and H^{out} on decryption. The latter is the inverse of the output layer in the forward diagram. A security proof must control collisions at both entry boundaries, including interactions with previous primitive inputs and outputs. Invertibility alone does not transfer a universality bound from a family to its inverse. In the abstract independent-primitive setting, the secret random block permutation π is independent of the joint hashing parameters; the two hashing layers need not have mutually independent keys. PEP, which uses the same block cipher inside its mixing maps and in its ECB layer, requires a separate analysis of these dependencies.

Halevi's efficient affine construction. Halevi's blockwise polynomial evaluation (BPE) construction [Hal07a] combines a summary of the entire message with a position-dependent mask. Using the parameters of Section 2.4, put

$$s_\tau(\boldsymbol{z}) = \sum_{j=1}^{m} \tau^{m-j+1} z_j, \qquad H_{\tau,\beta}(\boldsymbol{z})_i = z_i + s_\tau(\boldsymbol{z}) + \alpha^{i-1}\beta. \tag{13}$$

This is precisely $B_m\boldsymbol{z} + \boldsymbol{b}$: a linear map followed by a translation, hence an affine map. The summary can be evaluated by Horner's rule in m field multiplications, and then added to all blocks. When $\lambda_m \neq 0$, the inverse linear map has the same form, $A_m\boldsymbol{w} = \boldsymbol{w} + \lambda_m^{-1} s_\tau(\boldsymbol{w})\mathbf{1}$; thus inversion also takes linear work in m, after parameter preparation. If $\lambda_m = 0$, then $B_m\mathbf{1} = 0$, so B_m is not invertible.

For the collision experiment, let $\mathcal{S}_m = \{t \in \mathbb{F} : 1 + t + \cdots + t^m \neq 0\}$ and $g_m = N - |\mathcal{S}_m|$. Fix $1 \le m \le N - 3$, sample τ uniformly from $\mathcal{S}_m$, and sample β independently uniformly from $\mathbb{F}$. For either H or H^{-1}, Halevi [Hal07a] proves that any fixed XOR difference occurs with probability N^{-1} when $i \neq j$, and with probability at most $m/(N - g_m)$ when $i = j$ and $\boldsymbol{z} \neq \boldsymbol{z}'$. For different positions, the difference contains the nonzero coefficient $(\alpha^{i-1} + \alpha^{j-1})\beta$; the uniform mask therefore gives the N^{-1} bound. At the same position, the masks cancel and the event reduces to a nonzero polynomial in τ of degree at most m, which has at most m roots. Thus the affine mask supplies position separation, while polynomial evaluation controls collisions between distinct messages at the same position.

Scope of these conditions. These bounds concern fixed inputs at one block count and a specified key distribution. They do not by themselves prove security for adaptive encryption and decryption queries: a proof must account for the query transcript, repeated or already answered queries, and sampling a permutation without replacement. Likewise, complete-block bounds do not automatically cover partial tails, and a length-dependent parameter selection rule does not imply independence across lengths. In particular, a polynomial root bound applies only after the collision polynomial has been shown to be nonzero. These are the conditions to check when analyzing PEP's concrete coefficients and TET's tail processing.

2.6 Security Model

The original PEP [CS06b] and TET [Hal07a] papers use the classical tweakable strong pseudorandom-permutation (tSPRP) experiment adopted here. An adversary adaptively selects direction, tweak, and query string, and receives either $(\mathsf{Enc}_K^T, \mathsf{Dec}_K^T)$ or $(\Pi^T, (\Pi^T)^{-1})$. The ideal permutations are uniform and independent for distinct tweak–length domains; the two directions share the same permutation within a domain. With an absolute-value convention, the advantage is

$$\mathsf{Adv}_{\mathcal{E}}^{\mathrm{tSPRP}}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{\mathsf{Enc},\mathsf{Dec}} = 1] - \Pr[\mathcal{A}^{\Pi,\Pi^{-1}} = 1]\right|. \tag{14}$$

Repeated tweaks are permitted. The adversary has no separate oracle for the secret block permutation or hashing parameters. A correctness failure of a printed pair is recorded separately from a tSPRP attack on an inverse-consistent scheme. For target plaintext recovery, decryption excludes the target tweak–ciphertext tuple itself; additional restrictions are stated with the attack.

3 PEP: Coefficient Collisions and Plaintext Recovery

In this section, we show how a defect in PEP's public coefficient generator leads to distinguishing and plaintext-recovery attacks. We first give a fixed public parameter choice that satisfies the generator's stated requirements but produces repeated coefficients. We then show that these repetitions create a ciphertext relation detectable with two encryption queries. Finally, we use a related coefficient collision to recover block-cipher values and, with one auxiliary decryption, a target plaintext. The attacks use only PEP's external interfaces under one key; they require neither a weakness in the underlying block cipher nor direct access to it.

3.1 A Counterexample to the Coefficient Generator

The generator's parameter requirements do not suffice to guarantee distinct position coefficients. Chakraborty and Sarkar choose a primitive degree-n polynomial ρ with no trinomial multiple of degree below $m_{\max}$, and claim that their construction then produces an allowed sequence for every supported block count [CS06b, Sec. 2.1, Prop. 1]. We exhibit a counterexample with $n = 128$, $N = 2^{128}$, and $m_{\max} = 1342$. Appendix B specifies the fixed polynomial (37), verifies its primitivity and the trinomial exclusion by exact arithmetic, and records all three branches of the coefficient generator.

The counterexample arises from a four-term relation that the trinomial condition does not exclude. For $\alpha = x \bmod \rho$, the verified identities are

$$1 + \alpha^u + \alpha^v \neq 0 \quad (0 < u < v \leq 1341), \qquad \alpha^{891} = 1 + \alpha + \alpha^2 + \alpha^3,$$

where u, v are integer exponents. Multiplying the second identity by $1 + \alpha$ gives

$$\alpha^{891} + \alpha^{892} = 1 + \alpha^4. \tag{15}$$

For $m = 1337 = 3 \cdot 445 + 2$, the generator sets its fifth coefficient to $1 + \alpha^4$ and its last coefficient to $\alpha^4(\alpha^{887} + \alpha^{888})$. Consequently,

$$p_{1337,5} = p_{1337,1337} = 1 + \alpha^4 = \texttt{0x11}. \tag{16}$$

The resulting vector has 1337 entries but only 1336 distinct values. Its sum is zero, and none of its entries is 0 or 1: the defect is the repeated coefficient, not a zero coefficient or a failed checksum. The collision persists after correcting the printed remainder-index bound to give exactly m coefficients, as explained in Appendix B.

The distinctness proof overlooks the exceptional closing terms in two generator branches. In the comparison of two binomials, Proposition 1 treats both as adjacent-power terms $\alpha^e + \alpha^{e+1}$ [CS06b, pp. 299–300]. This reasoning does not cover the closing terms $1 + \alpha^3$ and $1 + \alpha^4$. In our instance,

$$\rho(x) \mid x^{892} + x^{891} + x^4 + 1,$$

which is consistent with the required exclusion of low-degree trinomial multiples. Thus the generator can satisfy its polynomial-selection rule while violating the distinctness condition it is intended to ensure.

This counterexample concerns a fixed, admitted public parameter choice. The attacker does not change the field polynomial of an implementation during a query. The generated vector fails the allowed-sequence definition, although its zero sum still ensures invertibility whenever $R \neq 0$. The result therefore refutes the generator's universal guarantee; it does not show that every choice of public polynomial, or every actually distinct zero-sum coefficient sequence, is insecure.

3.2 From Coefficient Collisions to a Distinguishing Attack

A repeated coefficient produces a relation that survives all three encryption stages. Suppose $p_{m,i} = p_{m,j}$ for $i < j$, and choose $M_i = M_j = 0$. The input mixing and ECB stages in (4) then give

$$V_i = V_j = 0, \qquad X_i = X_j, \qquad Y_i = Y_j.$$

The unscaled outputs are also equal because both positions receive the same output mask. Writing their common value as $W = Y_i + p_{m,i}\nu$, we obtain

$$(C_i, C_j) = (R^{i-1}W, R^{j-1}W). \tag{17}$$

The unknown W may vary between queries, but $R = E_K(T)$ is shared by queries under the same key and tweak. A cross-product test eliminates both unknowns without dividing by a potentially zero ciphertext block.

Attack 1 (Two-query PEP distinguisher)**.** Two chosen plaintexts suffice to test this relation in the fixed instance of Section 3.1. Set $m = 1337$, fix any tweak T, and make the following encryption queries under the same key:

1. Encrypt $\boldsymbol{M} = (0, \ldots, 0)^\mathsf{T}$ and $\boldsymbol{M}' = (1, 0, \ldots, 0)^\mathsf{T}$ to obtain $\boldsymbol{C}$ and $\boldsymbol{C}'$.
2. Output 1 if and only if $C_5 C'_{1337} = C_{1337} C'_5$.

Both messages have zero blocks at the repeated-coefficient positions. Equation (17) therefore makes the test pass for every $R \neq 0$, regardless of the block permutation. It also passes at $R = 0$ under the natural forward evaluation. The adversary uses no decryption query and does not recover any internal value.

Advantage. The ideal acceptance probability is small because two distinct messages must receive two distinct uniformly random ciphertexts. Its exact value is

$$p_2 = \frac{N^{-1} + N^{-2} - N^{-3} - N^{-m}}{1 - N^{-m}}, \qquad N = 2^{128}, \quad m = 1337. \tag{18}$$

To see this, first sample two independent uniform m-block vectors. The test asks whether a 2×2 matrix formed from their selected coordinates is singular, which has probability $N^{-1} + N^{-2} - N^{-3}$. Equality of the two complete vectors has probability N^{-m} and always passes the test. Conditioning on distinct complete outputs gives (18).

The natural forward evaluation has acceptance gap $1 - p_2 > 1 - N^{-1} - N^{-2}$. For any completion changing only the $R = 0$ branch, the distinguishing advantage in the ideal underlying-permutation model is at least $1 - N^{-1} - p_2$. Only one N^{-1} term is needed because the two real queries share the same tweak-derived multiplier. As noted in Section 2.3, the natural forward evaluation at $R = 0$ need not itself define a permutation.

The failed collision-proof premise. The generator defect removes the nonzero coefficient on which the intra-query collision argument relies. For two positions, input mixing gives

$$X_i + X_j = R^{i-1}M_i + R^{j-1}M_j + (p_{m,i} + p_{m,j})\mu.$$

Chakraborty and Sarkar [CS06c, Appendix A.2, intra-query case 2] use $p_{m,i} + p_{m,j} \neq 0$ to show that multiplying a uniform mixing value by this coefficient preserves uniformity. At our repeated pair, this coefficient is zero; setting both plaintext blocks to zero makes $X_i = X_j$ with probability one. The failure therefore occurs when the generator is claimed to supply distinct coefficients, before a collision bound requiring that property can be applied.

3.3 Derived-Value Recovery and Plaintext Recovery

The same mechanism also exposes values of the secret block permutation and enables target plaintext recovery through auxiliary decryption. Write $\pi = E_K$, retain the fixed polynomial (37), and use auxiliary block count $h = 1340$. The algebra below holds for a fixed permutation whenever the stated nonzero checks succeed; the probability bounds separately assume a uniformly sampled secret permutation. Recovering $\pi(x)$ means learning one input–output pair, not recovering the secret key.

Recovering a block-cipher value. At length h, a different separation between repeated coefficients makes the ciphertext ratio determine R uniquely. Put $c_i = p_{h,i}$. The same generator branch, now with $t = 446$, gives

$$c_5 = c_{1339} = 1 + \alpha^4, \qquad \gcd(1334, N - 1) = 1. \tag{19}$$

Indeed, the 444th trailing binomial occurs at position $5 + 2 \cdot 445 + 444 = 1339$ and equals $\alpha^{891} + \alpha^{892}$. Coprimality follows from $1334 = 2 \cdot 23 \cdot 29$ and the factorization of $N - 1$ in Appendix B.

Attack 2 (PEP derived-value recovery)**.** Define a recovery procedure $\mathcal{R}(x)$ that uses $x \in \mathbb{F}$ as its query tweak. Maintain a cache of permutation pairs learned by the attack; these pairs are exact until the first error. If the cache contains $\pi(x)$, return that value. Otherwise, choose independent uniform $u, v \in \mathbb{F}$ and query

$$\boldsymbol{C} = \mathsf{Enc}_K^x(u, v, 0, \ldots, 0), \qquad |\boldsymbol{C}| = hn.$$

If $C_5 = 0$ or $C_{1339} = 0$, report failure. Otherwise, return and cache

$$\widehat{\pi}(x) = (C_{1339}/C_5)^e, \qquad e = 1334^{-1} \bmod (N - 1). \tag{20}$$

Each uncached recovery makes one encryption query and is attempted only once.

For $R = \pi(x) \neq 0$, equation (17) gives $(C_5, C_{1339}) = (R^4 W, R^{1338} W)$; hence a nonzero ratio is R^{1334}, and (20) returns R exactly.

Randomizing the first two plaintext blocks bounds the chance that this recovery fails, without assuming that W is independent and uniform. For an x fixed independently of a uniform π, start the first recovery with an empty internal evaluation history and cache,

before any query or target generation. Its probability of failure or an incorrect return is at most

$$\frac{1355}{N-1343} < 2^{-117}. \tag{21}$$

This bound counts $R = 0$ as failure and permits any completion confined to that branch. For $R \neq 0$, the sum $S = u + Rv$ is uniform, and v remains uniform conditional on S. Moreover, $c_3 = \alpha^2 + \alpha^3$ occurs only once, allowing the third ECB output to be sampled last in the analysis. Appendix C uses these facts to control internal repetitions and $W = 0$. With prior internal evaluations or adaptive tweaks, its conditional bounds must be used instead of reusing the first-recovery bound directly.

Batch inversion through auxiliary decryption. One decryption under a different tweak reveals up to $h-2$ selected inverse values. For a target tweak T^*, fix $T_0 = T^* + 1 \neq T^*$ and use $\mathcal{R}$ or the cache to obtain

$$R_0 = \pi(T_0), \qquad a_0 = \pi(R_0 + \mathsf{bin}_n(h)), \qquad b_0 = \pi(\alpha a_0).$$

Stop if a recovery fails or $R_0 = 0$. Given values $y_1, \ldots, y_m$ with $m \leq h-2$, choose an h-block vector $\boldsymbol{Y}^{\mathrm{aux}}$ with $Y_1^{\mathrm{aux}} = R_0$, $Y_2^{\mathrm{aux}} = 0$, and $Y_{i+2}^{\mathrm{aux}} = y_i$ for $1 \leq i \leq m$; fill unused positions with R_0. Recover $\nu_0 = \pi(b_0 + \sum_i Y_i^{\mathrm{aux}})$, and submit

$$C_i^{\mathrm{aux}} = R_0^{i-1}(Y_i^{\mathrm{aux}} + c_i\nu_0), \qquad \boldsymbol{M}^{\mathrm{aux}} = \mathsf{Dec}_K^{T_0}(\boldsymbol{C}^{\mathrm{aux}}). \tag{22}$$

The zero-sum property of (c_i) makes decryption recompute the same ν_0, so its ECB layer inverts exactly the selected values.

The first coordinate determines the input-mixing value. Since $\pi^{-1}(R_0) = T_0$ and $c_1 = 1 + \alpha \neq 0$, compute

$$\mu_0 = (M_1^{\mathrm{aux}} + T_0)/c_1, \qquad X_i^{\mathrm{aux}} = R_0^{i-1}M_i^{\mathrm{aux}} + c_i\mu_0 = \pi^{-1}(Y_i^{\mathrm{aux}}). \tag{23}$$

Cache all the recovered permutation pairs. In particular, the second coordinate reveals $x_0 = \pi^{-1}(0)$. Any later request for $\pi(x_0)$ returns zero from the cache, avoiding a recovery query under this known zero-multiplier tweak. The remaining selected coordinates supply $\pi^{-1}(y_1), \ldots, \pi^{-1}(y_m)$.

Attack 3 (PEP target plaintext recovery)**.** Combining forward recovery with batch inversion reverses the target's three encryption stages. Let $(T^*, \boldsymbol{C}^*)$ be an m-block target with $3 \leq m \leq h-2 = 1338$. First recover its parameters $R = \pi(T^*)$, $a = \pi(R + \mathsf{bin}_n(m))$, and $b = \pi(\alpha a)$; stop if $R = 0$ or any recovery fails. Then perform the following operations in order:

1. Undo the output mixing: compute Z_i and recover ν in (24), giving the target ECB outputs Y_i^*.
2. Apply the auxiliary decryption above with $y_i = Y_i^*$, giving $X_i^* = X_{i+2}^{\mathrm{aux}}$.
3. Recover μ and undo the input mixing to obtain M_i^*.

The corresponding formulas are

$$\begin{aligned} Z_i &= R^{-(i-1)}C_i^*, \quad \nu = \pi\left(b + \sum_i Z_i\right), \qquad Y_i^* = Z_i + p_{m,i}\nu, \\ X_i^* &= \pi^{-1}(Y_i^*), \qquad \mu = \pi\left(a + \sum_i X_i^*\right), \quad M_i^* = R^{-(i-1)}(X_i^* + p_{m,i}\mu). \end{aligned} \tag{24}$$

All forward values come from $\mathcal{R}$ or the cache, and all target inverse values come from the one auxiliary decryption at $T_0 \neq T^*$. In particular, ν is recovered before batch inversion and μ after it.

Query complexity and success probability. The attack uses at most nine h-block encryptions and one h-block decryption. Five forward values are needed for the target (R, a, b, ν, μ), three for auxiliary parameter preparation (R_0, a_0, b_0), and one for the auxiliary output mixing (ν_0). Thus the auxiliary queries contain at most $10h$ message blocks in total. Offline work requires $O(m + h + q_R n)$ field multiplications or squarings and $O(q_R + 1)$ field inversions, where $q_R = 9$.

For a uniform secret permutation, the probability of correct target plaintext recovery is at least $1 - \varepsilon(9, S_*) > 1 - 2^{-109}$, where

$$S_* = 10(h + 5) + m + 5, \qquad \varepsilon(q_R, S_*) = \frac{q_R(3S_* + h + 5) + S_*}{N - S_*}. \tag{25}$$

Here the target tweak and plaintext are fixed independently of the secret permutation before its ciphertext is generated. Each auxiliary query uses at most $h + 5$ primitive calls; target generation contributes another $m + 5$ calls to S_* but is not an adversarial auxiliary query. Appendix C proves the conditional failure bound and its adaptive composition, including the histories before and after the zero preimage becomes known.

The recovery guarantee retains the original restrictions on exceptional branches and oracle access. Every recovery is attempted once, and all divisors and negative-power arguments are checked for nonzero before use. Zero multipliers and incorrect recoveries count as failures; no unique preimage is claimed for the noninjective natural forward evaluation at $R = 0$. Any completion confined to that branch leaves the normal-prefix analysis unchanged. The probability bound assumes a uniform secret permutation; transferring it to a concrete block cipher requires the corresponding PRP/SPRP reduction loss. This is chosen-ciphertext plaintext recovery through PEP interfaces. Appendix D.1 records the further distinguishers and extensions to other supported lengths.

4 TET: Partial-Block Correctness and Plaintext Recovery

In this section, we analyze TET's treatment of a final partial block. We first explain why the printed encryption and decryption algorithms are not inverses, and define two inverse-consistent corrections. We then show that either correction permits two distinct queries to reuse the same internal cipher path, yielding a distinguisher and revealing a secret mixing scalar. Finally, we use this scalar relation to recover a target plaintext without submitting the target ciphertext for decryption.

4.1 Correctness of the Printed Algorithms and Two Corrections

The printed algorithms disagree on the scaling required to invert the tail additions. We use the specification of Section 2.4, fix a legal tweak and length $n \le L = mn + r < N$ with $1 \le r < n$, and retain its notation $A_m, B_m, \lambda_m, \boldsymbol{b}$ and d. The identities $B_m\mathbf{1} = \lambda_m\mathbf{1}$ and $A_m\mathbf{1} = \lambda_m^{-1}\mathbf{1}$ determine the inverse tail coefficients.

A common description of the tail placements. Two scalars $f, g \in \mathbb{F}$ make the distinction between the versions explicit. They are fixed within a domain, may depend on the hashing parameters, and are identical in both directions; they are neither adversarial inputs nor fresh randomness. Replace the two outer boundaries of (10) by

$$\begin{aligned} \boldsymbol{X} &= A_m\boldsymbol{M} + f\,d(p)\mathbf{1} + \boldsymbol{b}, \\ \boldsymbol{C} &= A_m(\boldsymbol{Y} + \boldsymbol{b}) + g\,d(c)\mathbf{1}. \end{aligned} \tag{26}$$

The central cipher path and $c = p + \mathsf{pref}_r(U)$ remain unchanged. The actual inverse of these boundaries is

$$\begin{aligned} \boldsymbol{Y} &= B_m \boldsymbol{C} + \lambda_m g\, d(c)\mathbf{1} + \boldsymbol{b}, \\ \boldsymbol{M} &= B_m(\boldsymbol{X} + \boldsymbol{b}) + \lambda_m f\, d(p)\mathbf{1}. \end{aligned} \tag{27}$$

Between these inverse boundaries, recover the central state by

$$\begin{aligned} U &= E_K^{-1}(Y_m), & X_m &= E_K^{-1}(U), \\ X_i &= E_K^{-1}(Y_i) \quad (i < m), & p &= c + \mathsf{pref}_r(U). \end{aligned} \tag{28}$$

Thus a tail added with coefficient g after output mixing requires coefficient $\lambda_m g$ at the inverse entry; the same reasoning forces $\lambda_m f$ at the inverse exit.

The printed mismatch. Halevi's Figure 2 [Hal07a] divides each encryption mixing summary by λ_m before adding the padded tail, giving $f = g = 1$, but uses coefficient 1 at both decryption boundaries. For example, the first internal vector reconstructed by the printed decryption is

$$\widehat{\boldsymbol{Y}} = B_m \boldsymbol{C} + d(c)\mathbf{1} + \boldsymbol{b} = \boldsymbol{Y} + (\lambda_m + 1)d(c)\mathbf{1}. \tag{29}$$

It therefore need not equal the original $\boldsymbol{Y}$. This correctness issue was already identified in the third-party `tet-heh` implementation note [ody26], which moves both encryption tail additions inside the division.

A small-domain example shows that the mismatch can change the external decryption result. In $\mathbb{F}_4 = \mathbb{F}_2[x]/(x^2 + x + 1)$, let $\omega = x \bmod (x^2 + x + 1)$ and encode $0 = 00$, $1 = 01$, $\omega = 10$, and $\omega^2 = 11$. Take $n = 2$, $m = 1$, $r = 1$, $\tau = \alpha = \omega$, $\lambda_m = \omega^2$, $\beta = 0$, and the identity data permutation. Then the printed algorithms give

$$00 \parallel 0 \xrightarrow{\mathsf{Enc}_{\text{printed}}} 11 \parallel 0 \xrightarrow{\mathsf{Dec}_{\text{printed}}} 01 \parallel 1.$$

These parameters are realizable under the original derivation: with $T = 00$ and $L = 3$, take a derivation permutation that swaps 0 and 1 and fixes ω, ω^2. This example establishes a functional error; the attacks below do not rely on the identity permutation.

Two inverse-consistent corrections. Preserving either printed direction determines a corresponding correction to the other direction. Table 2 specifies the versions $\mathsf{TET}_{\mathrm{B}}$ and $\mathsf{TET}_{\mathrm{A}}$ introduced in Section 1. The former preserves printed encryption; the latter preserves printed decryption. Both corrections are permutations on the complete

Table 2: Two corrections of TET's partial-block branch.

Version	f	g	Direction preserved
$\mathsf{TET}_{\mathrm{B}}$	1	1	Printed encryption; multiply both inverse tail terms by λ_m.
$\mathsf{TET}_{\mathrm{A}}$	λ_m^{-1}	λ_m^{-1}	Printed decryption; divide both encryption tail terms by λ_m.

message, including its tail. The displayed inverses establish that the three maps $\mathcal{L}_f$, $\mathcal{P}_K$, and $\mathcal{R}_g$ introduced in Section 2.4 are bijections on $\mathbb{F}^m \times \{0,1\}^r$, with

$$\mathsf{Enc} = \mathcal{R}_g \circ \mathcal{P}_K \circ \mathcal{L}_f, \qquad \mathsf{Dec} = \mathcal{L}_f^{-1} \circ \mathcal{P}_K^{-1} \circ \mathcal{R}_g^{-1}. \tag{30}$$

The output is the concatenation of the full blocks and the tail. For $r = 0$, omitting all tail terms and using ordinary ECB gives (8). Algorithm 3 in Appendix A gives the execution order for the two corrected variants, using the shared preparation in Algorithm 2.

Correctness must precede a security argument that treats encryption and decryption as inverse oracles. In particular, removing an inverse query whose answer is already determined by a previous query presupposes this property. Such a step cannot justify the noninverse printed pair, and an analysis of either correction must use its corrected coefficients.

4.2 Path-Reuse Distinguishers and Mixing-Scalar Recovery

Two queries suffice to distinguish either corrected variant and recover one of its mixing scalars. The queries introduce a common difference across all full blocks and a matching tail difference, so that the two changes cancel at the entry to the central cipher path. Fix the same key, tweak T, and legal length $L = mn + r$ throughout this subsection, where $m \geq 1$ and $1 \leq r < n$. Choose $0 \neq \delta \in \{0,1\}^r$ and put $\Delta = \delta \parallel 0^{n-r} \in \mathbb{F}$. Then

$$d(z + \delta) + d(z) = \Delta.$$

The adversary knows the field representation, n, m, r, α, and the version's algorithm, but not τ, β, the cipher key, or U. In particular, the query differences below require no secret value.

Attack 4 (TET path-reuse distinguishers)**.** The two corrections admit the same response test, in opposite query directions:

1. $\mathsf{TET}_{\mathrm{B}}$**: two decryptions.** Choose any $(\boldsymbol{C}, c)$ and decrypt it and $(\boldsymbol{C} + \Delta\mathbf{1}, c + \delta)$, obtaining $(\boldsymbol{M}, p)$ and $(\boldsymbol{M}', p')$. Accept if $p' + p = \delta$ and all differences $M'_i + M_i$ are equal and nonzero.

2. $\mathsf{TET}_{\mathrm{A}}$**: two encryptions.** Choose any $(\boldsymbol{M}, p)$ and encrypt it and $(\boldsymbol{M} + \Delta\mathbf{1}, p + \delta)$, obtaining $(\boldsymbol{C}, c)$ and $(\boldsymbol{C}', c')$. Accept if $c' + c = \delta$ and all differences $C'_i + C_i$ are equal and nonzero.

Why the internal path is reused. In $\mathsf{TET}_{\mathrm{B}}$, the selected ciphertext difference leaves $\boldsymbol{Y}$ unchanged because the first line of (27) changes by

$$B_m(\Delta\mathbf{1}) + \lambda_m \Delta\mathbf{1} = 0.$$

Consequently, every inverse-cipher input is reused, and $\boldsymbol{X}, U$ remain unchanged. The tail equation gives $p' + p = \delta$, and the inverse exit gives $\boldsymbol{M}' + \boldsymbol{M} = \lambda_m \Delta\mathbf{1}$. In $\mathsf{TET}_{\mathrm{A}}$, the plaintext difference instead cancels at the forward entry:

$$A_m(\Delta\mathbf{1}) + \lambda_m^{-1} \Delta\mathbf{1} = 0.$$

Now $\boldsymbol{X}, U, \boldsymbol{Y}$ remain unchanged, giving $c' + c = \delta$ and $\boldsymbol{C}' + \boldsymbol{C} = \lambda_m^{-1}\Delta\mathbf{1}$. Since $\lambda_m \neq 0$ and $\Delta \neq 0$, both response tests always accept, for every data permutation and valid parameter choice.

Advantage and recovered scalar. The ideal acceptance probability is obtained by counting complete responses, not merely internal collisions. The external queries are distinct because $\delta \neq 0$. Given the first response from an ideal permutation or its inverse, the second is uniform over the remaining $2^{mn+r} - 1$ values. Exactly $N - 1$ of them have the prescribed tail difference and a common nonzero full-block difference. Hence

$$\mathsf{Adv} = 1 - \frac{N - 1}{2^{mn+r} - 1}. \tag{31}$$

In the real scheme, dividing any full-block difference by Δ gives

$$\mathsf{TET}_{\mathrm{B}}: \quad \lambda_m = \frac{M_i' + M_i}{\Delta}, \qquad \mathsf{TET}_{\mathrm{A}}: \quad \kappa = \lambda_m^{-1} = \frac{C_i' + C_i}{\Delta}. \tag{32}$$

This recovers a mixing scalar, not the cipher key. Each query has $mn + r$ bits, or $m + 1$ rounded-up message blocks, and the test and scalar extraction take $O(m)$ field operations. These attacks do not apply to $r = 0$.

Why invertibility does not prevent the attack. The collision occurs only in the full-block projection of an augmented mixing map. For example, $\mathcal{L}_f$ retains p as part of its output, so two different tails can give the same $\boldsymbol{X}$ without giving the same pair $(\boldsymbol{X}, p)$. Thus the preceding cancellations are consistent with the bijectivity established in Section 4.1. The repeated bridge U then carries the external tail difference through unchanged. A permutation property for the complete-block core alone does not control this partial-block interaction.

The XOR-blockwise-universality condition introduced by Halevi [Hal07b] and recalled in Section 2.5 cannot be applied unchanged to these corrected boundaries. Its inputs and comparison offset must be fixed independently of the secret hash key, whereas the tail term here contains λ_m^{-1} on the $\mathsf{TET}_{\mathrm{A}}$ encryption side or λ_m on the $\mathsf{TET}_{\mathrm{B}}$ decryption side, both dependent on τ. The resulting collision expression vanishes identically for the chosen differences, instead of defining a nonzero polynomial to which a root bound applies. Appendix D.2 gives the corresponding criterion for general f, g and discusses the sampling of the bridge value.

4.3 Target Plaintext Recovery

The same path-reuse relations recover an entire target plaintext with two or three oracle queries. Fix a target $(T^*, \boldsymbol{C}^*, c^*)$ whose plaintext is $(\boldsymbol{M}^*, p^*)$, with $m \geq 1$, $2 \leq r < n$, and a legal length $L = mn + r$. Every query below uses the same key and m, r. For $\mathsf{TET}_{\mathrm{A}}$, also require an available valid auxiliary tweak $T_0 \neq T^*$. Parameter preparation fixes λ_m across tweaks with the same key and full-block count, even though their masks β may differ.

Attack 5 (TET target plaintext recovery)**.** $\mathsf{TET}_{\mathrm{B}}$**: two modified-target decryptions.** Two distinct tail differences both reveal the scalar and permit recovery of the target. Choose distinct nonzero $\delta_1, \delta_2 \in \{0,1\}^r$, set $\Delta_j = \delta_j \,\|\, 0^{n-r}$, and obtain

$$\mathsf{Dec}_K^{T^*}(\boldsymbol{C}^* + \Delta_j \mathbf{1}, c^* + \delta_j) = (\boldsymbol{M}^{(j)}, p^{(j)}), \qquad j = 1, 2.$$

At any coordinate $1 \leq i \leq m$, compute

$$\begin{aligned} \lambda_m &= \frac{M_i^{(1)} + M_i^{(2)}}{\Delta_1 + \Delta_2}, \\ \boldsymbol{M}^* &= \boldsymbol{M}^{(1)} + \lambda_m \Delta_1 \mathbf{1}, \quad p^* = p^{(1)} + \delta_1. \end{aligned} \tag{33}$$

$\mathsf{TET}_{\mathrm{A}}$**: calibration followed by one decryption.** Two encryptions under T_0 first recover $\kappa = \lambda_m^{-1}$ using Attack 4. Choose $0 \neq \delta \in \{0,1\}^r$ and $\Delta = \delta \,\|\, 0^{n-r}$, then query and recover as follows:

$$\begin{aligned} \mathsf{Dec}_K^{T^*}(\boldsymbol{C}^* + \kappa\Delta\mathbf{1}, c^* + \delta) &= (\widetilde{\boldsymbol{M}}, \widetilde{p}), \\ (\boldsymbol{M}^*, p^*) &= (\widetilde{\boldsymbol{M}} + \Delta\mathbf{1}, \widetilde{p} + \delta). \end{aligned} \tag{34}$$

Why recovery succeeds. For $\mathsf{TET}_{\mathrm{B}}$, the decryption argument of Section 4.2 gives, whenever $(\boldsymbol{M}, p) = \mathsf{Dec}_K^T(\boldsymbol{C}, c)$,

$$\mathsf{Dec}_K^T(\boldsymbol{C} + \Delta\mathbf{1}, c + \delta) = (\boldsymbol{M} + \lambda_m \Delta\mathbf{1}, p + \delta). \tag{35}$$

Adding the two queried responses cancels $\boldsymbol{M}^*$ and yields $\lambda_m(\Delta_1 + \Delta_2)\mathbf{1}$, proving (33). The restriction $r \geq 2$ ensures that two distinct nonzero tail differences are available.

For $\mathsf{TET}_{\mathrm{A}}$, the chosen ciphertext difference changes the inverse entry by

$$B_m(\lambda_m^{-1}\Delta\mathbf{1}) + \lambda_m\lambda_m^{-1}\Delta\mathbf{1} = 0.$$

Thus $\boldsymbol{Y}, \boldsymbol{X}, U$ are reused, the plaintext tail changes by δ, and the full-block vector changes by $\lambda_m f\Delta\mathbf{1} = \Delta\mathbf{1}$. Calibration transfers the scalar between tweaks without requiring their masks to coincide.

Success, query restrictions, and extensions. Both procedures recover the target with probability one for every data permutation, valid parameter set, and target in the stated range. $\mathsf{TET}_{\mathrm{B}}$ uses two decryptions; $\mathsf{TET}_{\mathrm{A}}$ uses two encryptions and one decryption. All target-derived decryption queries differ from the target because their tail differences are nonzero, and calibration uses another tweak. The only divisors are Δ or $\Delta_1 + \Delta_2$, which are nonzero; individual zero response blocks cause no difficulty. Each query has $mn+r$ bits, or $m+1$ rounded-up message blocks, and recovery requires $O(m)$ field operations. These are chosen-ciphertext plaintext-recovery attacks on the TES interface, which excludes decryption of the target itself but has no additional authentication or rejection rule. Their success does not require an ideal-permutation assumption.

Appendix D.2 covers the one-bit-tail case and multiple targets after one calibration. In particular, a one-bit tail requires three decryptions for $\mathsf{TET}_{\mathrm{B}}$, while $\mathsf{TET}_{\mathrm{A}}$ still uses two encryptions and one decryption. Reusing a scalar is restricted to a fixed full-block count; no transfer across different m is claimed.

5 Discussion and Requirements for Repair

In this section, we discuss the requirements for repairing the defects identified above. For PEP, we consider coefficient validation and the zero-multiplier branch; for TET, we examine an asymmetric placement of the tail terms. Establishing the security of a modified scheme still requires a complete proof.

5.1 PEP: Coefficient Validation and the Zero Multiplier

A repaired public generator must satisfy the coefficient requirements for every supported parameter choice. Its analysis must cover all comparisons between coefficient types, including the exceptional closing terms. For a fixed public field and a declared finite set of supported lengths, distinctness and the zero-sum property can be verified exhaustively; a general generator requires a proof covering all prescribed parameters. Checking only the checksum or excluding trinomial multiples does not address the relation in (15). Admissible sequences should also exclude zero coefficients and account separately for forced internal coincidences when a coefficient equals one.

The zero-multiplier branch requires separate treatment. The multiplier must be defined and invertible for every key and tweak. Replacing $R = 0$ by a nonzero value still requires an analysis of cross-tweak dependencies and of the replacement's effect on the primitive distribution.

A complete security proof must additionally account for the reuse of the same cipher in the mixing and ECB layers, together with adaptive forward and inverse queries.

5.2 TET: Tail Placement and Remaining Security Requirements

An asymmetric placement of the tail terms provides one candidate modification. Specifically, set $f = 1$ and $g = \lambda_m^{-1}$ in (26), and denote the resulting variant by $\mathsf{TET}_{\mathrm{M}}$. Its encryption and decryption boundaries are

$$\begin{aligned} \boldsymbol{X} &= A_m \boldsymbol{M} + d(p)\mathbf{1} + \boldsymbol{b}, \quad & \boldsymbol{C} &= A_m(\boldsymbol{Y} + \boldsymbol{b} + d(c)\mathbf{1}), \\ \boldsymbol{Y} &= B_m \boldsymbol{C} + d(c)\mathbf{1} + \boldsymbol{b}, \quad & \boldsymbol{M} &= B_m(\boldsymbol{X} + \boldsymbol{b}) + \lambda_m d(p)\mathbf{1}. \end{aligned} \tag{36}$$

The first row gives the encryption boundaries and the second gives their inverses, using $B_m = A_m^{-1}$ and $B_m\mathbf{1} = \lambda_m\mathbf{1}$. The central path remains the one in (10), with inverse (28).

This modification changes the cancellation conditions at the two boundaries and is proposed only as a direction for repair. A security proof still requires nonzero collision bounds under an explicitly specified joint parameter distribution. It must also account for adaptive queries and bridge values, and provide reductions to the underlying cipher and parameter-derivation function. We do not claim a complete security proof for $\mathsf{TET}_{\mathrm{M}}$.

6 Conclusion

We exhibit a permitted PEP field for which the proposed generator repeats position coefficients. The resulting relation gives a two-query distinguisher and target plaintext recovery with at most nine encryptions and one decryption in the main range. Its probability analysis accounts for dependent calls and zero values. For TET, we distinguish the previously recorded correctness mismatch from attacks on two inverse-consistent corrections. Public tail differences reuse the central cipher path, giving deterministic recovery with two or three auxiliary queries. These cases show why invertibility, position separation, and partial-block handling must be verified for the actual maps and parameter dependencies. Establishing a secure replacement requires a separate proof.

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, *CRYPTO 2025, Part V*, volume 16004 of *LNCS*, pages 172–199. Springer, Cham, August 2025. doi:10.1007/978-3-032-01901-1_6.

[CDD+24] Yu Long Chen, Michael Davidson, Morris Dworkin, Jinkeon Kang, John Kelsey, Yu Sasaki, Meltem Sönmez Turan, Donghoon Chang, Nicky Mouha, and Alyssa Thompson. Proposal of requirements for an accordion mode: Discussion draft for the NIST accordion mode workshop 2024. Discussion draft, National Institute of Standards and Technology, April 2024. Issued April 10, 2024. URL: https://csrc.nist.gov/pubs/other/2024/04/10/proposal-of-requirements-for-an-accordion-mode-dis/iprd.

[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 Internal Report 8552, National Institute of Standards and Technology, April 2025. URL: https://csrc.nist.gov/pubs/ir/8552/final, doi:10.6028/NIST.IR.8552.

[CDK+18] Benoît Cogliati, Yevgeniy Dodis, Jonathan Katz, Jooyoung Lee, John P. Steinberger, Aishwarya Thiruvengadam, and Zhe Zhang. Provable security of (tweakable) block ciphers based on substitution-permutation networks. In

Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 722–753. Springer, Cham, August 2018. doi:10.1007/978-3-319-96884-1_24.

[CEL+21] Benoît Cogliati, Jordan Ethan, Virginie Lallemand, ByeongHak Lee, Jooyoung Lee, and Marine Minier. CTET+: A beyond-birthday-bound secure tweakable enciphering scheme using a single pseudorandom permutation. *IACR Trans. Symm. Cryptol.*, 2021(4):1–35, 2021. doi:10.46586/tosc.v2021.i4.1-35.

[CS06a] Debrup Chakraborty and Palash Sarkar. HCH: A new tweakable enciphering scheme using the hash-encrypt-hash approach. In Rana Barua and Tanja Lange, editors, *INDOCRYPT 2006*, volume 4329 of *LNCS*, pages 287–302. Springer, Berlin, Heidelberg, December 2006. doi:10.1007/11941378_21.

[CS06b] Debrup Chakraborty and Palash Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. In Matthew J. B. Robshaw, editor, *FSE 2006*, volume 4047 of *LNCS*, pages 293–309. Springer, Berlin, Heidelberg, March 2006. doi:10.1007/11799313_19.

[CS06c] Debrup Chakraborty and Palash Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. Cryptology ePrint Archive, Report 2006/275, 2006. URL: https://eprint.iacr.org/2006/275.

[GS21] Sebati Ghosh and Palash Sarkar. Breaking tweakable enciphering schemes using Simon's algorithm. *DCC*, 89(8):1907–1926, 2021. doi:10.1007/s10623-021-00893-5.

[Hal07a] Shai Halevi. Invertible universal hashing and the TET encryption mode. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 412–429. Springer, Berlin, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5_23.

[Hal07b] Shai Halevi. Invertible universal hashing and the TET encryption mode. Cryptology ePrint Archive, Report 2007/014, 2007. URL: https://eprint.iacr.org/2007/014.

[HR03] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 482–499. Springer, Berlin, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_28.

[HR04] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In Tatsuaki Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 292–304. Springer, Berlin, Heidelberg, February 2004. doi:10.1007/978-3-540-24660-2_23.

[JN18] Ashwin Jha and Mridul Nandi. A survey on applications of H-technique: Revisiting security analysis of PRP and PRF. Cryptology ePrint Archive, Report 2018/1130, 2018. URL: https://eprint.iacr.org/2018/1130.

[LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, Berlin, Heidelberg, August 2002. doi:10.1007/3-540-45708-9_3.

[LRW11] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. *Journal of Cryptology*, 24(3):588–613, July 2011. doi:10.1007/s00145-010-9073-y.

[LWWY26] Gang Liu, Peng Wang, Rong Wei, and Dingfeng Ye. Revisiting construction of online cipher in Hash-ECB-Hash structure. *Journal of Cyber Security*, 11(1):227–242, 2026. In Chinese. URL: https://jcs.iie.ac.cn/xxaqxb/ch/reader/view_abstract.aspx?file_no=20260114&flag=1&journal_id=xxaqxb, doi:10.19363/J.cnki.cn10-1380/tn.2026.01.14.

[MWH24] Shuping Mao, Peng Wang, and Lei Hu. Study on tweakable enciphering schemes against Simon's quantum algorithm. *Journal of Cyber Security*, 9(2):96–105, 2024. In Chinese. URL: https://jcs.iie.ac.cn/xxaqxben/ch/reader/view_abstract.aspx?file_no=20240208&flag=1&journal_id=xxaqxben, doi:10.19363/J.cnki.cn10-1380/tn.2024.03.08.

[Nan14] Mridul Nandi. Forging attacks on two authenticated encryption schemes COBRA and POET. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 126–140. Springer, Berlin, Heidelberg, December 2014. doi:10.1007/978-3-662-45611-8_7.

[NR] Moni Naor and Omer Reingold. A pseudo-random encryption mode. Unpublished manuscript. Undated. URL: https://omereingold.wordpress.com/wp-content/uploads/2014/10/nr_mode.pdf.

[NR99] Moni Naor and Omer Reingold. On the construction of pseudorandom permutations: Luby-Rackoff revisited. *Journal of Cryptology*, 12(1):29–66, January 1999. doi:10.1007/PL00003817.

[ody26] odysseia06. TET and HEH. GitHub source repository, 2026. Commit 5ad74b65399539f4a80fda6263a371ee7cb2d651, README section "An erratum in TET's Figure 2"; accessed October 22, 2026. URL: https://github.com/odysseia06/tet-heh/tree/5ad74b65399539f4a80fda6263a371ee7cb2d651.

[Sar07] Palash Sarkar. Improving upon the TET mode of operation. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC 07*, volume 4817 of *LNCS*, pages 180–192. Springer, Berlin, Heidelberg, November 2007. doi:10.1007/978-3-540-76788-6_15.

[Sar08] Palash Sarkar. Efficient tweakable enciphering schemes from (block-wise) universal hash functions. Cryptology ePrint Archive, Report 2008/004, 2008. URL: https://eprint.iacr.org/2008/004.

[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, Beijing, China, December 15–17, 2005, Proceedings*, volume 3822 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2005. doi:10.1007/11599548_15.

A Pseudocode for the analyzed branches

The following procedures collect the complete execution order for the PEP complete-block branch and the two inverse-consistent TET tail variants. They use the encodings of Section 2.1; addition is field addition on full blocks and XOR on tails. The notation $\mathsf{len}(\boldsymbol{z})$ counts vector coordinates, whereas $|p|$ counts tail bits. The procedures derive m, L from their inputs. Statements run in order, and an indexed assignment is made at every indicated coordinate; an empty index range makes no assignments. The primitive calls inside these construction algorithms are unavailable directly to the adversary, which uses only the external interfaces described in the attacks.

A.1 PEP complete blocks

Algorithm 1 expands (4) and (6) into coordinate operations. $\mathsf{Coefficients}(m)$ is exactly the three-branch public generator in Appendix B; it computes the prescribed vector, including repeated values, without adding a distinctness rejection step. All calls use the same E_K; a, b are derived scalars and $p_{m,i}$ are public coefficients. The generated coefficients have sum zero, which makes the two mixing values recomputable during decryption. At $R = 0$, encryption here means the natural forward evaluation with $R^0 = 1$; the marker $\mathsf{undefined}$ records the missing decryption branch of the original specification. The completion conventions and success bounds remain those of Section 3.

Algorithm 1 PEP parameter preparation and complete-block encryption/decryption

Require: $3 \le m \le m_{\max} < N$, $T \in \mathbb{F}$; parse the input as m full blocks
function PEPSetup(K, T, m)
  $(p_{m,1}, \ldots, p_{m,m}) \leftarrow \mathsf{Coefficients}(m)$ ▷ Appendix B
  $R \leftarrow E_K(T)$; $a \leftarrow E_K(R + \mathsf{bin}_n(m))$; $b \leftarrow E_K(\alpha a)$
  return $(R, a, b, (p_{m,i})_{i=1}^{m})$
end function
procedure PEPEncrypt$(K, T, \boldsymbol{M})$
  $m \leftarrow \mathsf{len}(\boldsymbol{M})$; $(R, a, b, (p_{m,i})) \leftarrow$ PEPSetup(K, T, m)
  $V_i \leftarrow R^{i-1} M_i \quad (1 \le i \le m)$
  $\mu \leftarrow E_K(a + \sum_{i=1}^{m} V_i)$
  $X_i \leftarrow V_i + p_{m,i}\mu$; $Y_i \leftarrow E_K(X_i) \quad (1 \le i \le m)$
  $\nu \leftarrow E_K(b + \sum_{i=1}^{m} Y_i)$
  $C_i \leftarrow R^{i-1}(Y_i + p_{m,i}\nu) \quad (1 \le i \le m)$
  return $C_1 \,\|\, \cdots \,\|\, C_m$
end procedure
procedure PEPDecrypt$(K, T, \boldsymbol{C})$
  $m \leftarrow \mathsf{len}(\boldsymbol{C})$; $(R, a, b, (p_{m,i})) \leftarrow$ PEPSetup(K, T, m)
  if $R = 0$ **then**
    return $\mathsf{undefined}$
  end if
  $Z_i \leftarrow R^{-(i-1)} C_i \quad (1 \le i \le m)$
  $\nu \leftarrow E_K(b + \sum_{i=1}^{m} Z_i)$
  $Y_i \leftarrow Z_i + p_{m,i}\nu$; $X_i \leftarrow E_K^{-1}(Y_i) \quad (1 \le i \le m)$
  $\mu \leftarrow E_K(a + \sum_{i=1}^{m} X_i)$
  $M_i \leftarrow R^{-(i-1)}(X_i + p_{m,i}\mu) \quad (1 \le i \le m)$
  return $M_1 \,\|\, \cdots \,\|\, M_m$
end procedure

A.2 TET parameter preparation and the tail branches

Algorithms 2 and 3 use the parameter preparation of Section 2.4 and the two corrections of Section 4.1, on the supported domain $n \le L < N$ with public primitive element α. The secret key includes the derivation component K_1 and the data-cipher key denoted by K in E_K; the PRF pair encodings are those of the original specification. The shared candidate sequence restarts at $j = 0$ for each call, so the selected τ, λ_m depend on the key and m, while β depends on L, T. For a vector of dimension m, $\mathsf{Mix}(\tau, \boldsymbol{z}, 1) = B_m \boldsymbol{z}$ and $\mathsf{Mix}(\tau, \boldsymbol{z}, \lambda_m^{-1}) = A_m \boldsymbol{z}$. The public version v selects $\mathsf{TET}_{\mathrm{A}}$ or $\mathsf{TET}_{\mathrm{B}}$ and is the same in both directions.

Algorithm 2 TET parameter preparation and scalar mixing

Require: $n \le L = mn + r < N$, $0 \le r < n$, $v \in \{\mathrm{A}, \mathrm{B}\}$
function TETSetup(K_1, T, L, v)
  $m \leftarrow \lfloor L/n \rfloor$; $j \leftarrow 0$
  repeat
    $\tau \leftarrow \mathrm{PRF}_{K_1}(0, j)$; $\lambda_m \leftarrow 1 + \sum_{k=1}^{m} \tau^k$; $j \leftarrow j + 1$
  until $\lambda_m \neq 0$
  $\beta \leftarrow \mathrm{PRF}_{K_1}(L, T)$; $b_i \leftarrow \alpha^{i-1}\beta \quad (1 \le i \le m)$
  $(f, g) \leftarrow (1, 1)$; **if** $v = \mathrm{A}$ **then** $(f, g) \leftarrow (\lambda_m^{-1}, \lambda_m^{-1})$
  return $(\tau, \lambda_m, \boldsymbol{b}, f, g)$
end function
function Mix$(\tau, \boldsymbol{z}, \theta)$
  $m \leftarrow \mathsf{len}(\boldsymbol{z})$; $s \leftarrow 0$
  for $i = 1, \ldots, m$ **do**
    $s \leftarrow \tau(s + z_i)$
  end for
  return $\boldsymbol{z} + \theta s \mathbf{1}$
end function

In Algorithm 3, $d(z) = z \parallel 1 \parallel 0^{n-r-1}$ and $\mathsf{pref}_r(U)$ is the first r bits of U; $p, c \in \{0,1\}^r$. As in Table 2, $v = \mathrm{B}$ preserves printed encryption and $v = \mathrm{A}$ preserves printed decryption; the inverse tail coefficients are $\lambda_m g, \lambda_m f$. For $r = 0$, omit the tails and all their terms, replace the central path by $Y_i = E_K(X_i)$ or $X_i = E_K^{-1}(Y_i)$ at every coordinate, and return only the full-block vector. This gives exactly (8), with the same parameter preparation.

Algorithm 3 The partial-block branches of $\mathsf{TET}_{\mathrm{A}}$ and $\mathsf{TET}_{\mathrm{B}}$

Require: $n \le L = mn + r < N$, $1 \le r < n$, $v \in \{\mathrm{A}, \mathrm{B}\}$; legal T
procedure TETEncrypt$(K, T, (\boldsymbol{M}, p), v)$
  $m \leftarrow \mathsf{len}(\boldsymbol{M})$; $(\tau, \lambda_m, \boldsymbol{b}, f, g) \leftarrow$ TETSetup$(K_1, T, nm + |p|, v)$
  $\boldsymbol{X} \leftarrow$ Mix$(\tau, \boldsymbol{M}, \lambda_m^{-1}) + f\, d(p)\mathbf{1} + \boldsymbol{b}$
  $Y_i \leftarrow E_K(X_i) \quad (1 \le i < m)$
  $U \leftarrow E_K(X_m)$; $Y_m \leftarrow E_K(U)$; $c \leftarrow p + \mathsf{pref}_r(U)$
  $\boldsymbol{C} \leftarrow$ Mix$(\tau, \boldsymbol{Y} + \boldsymbol{b}, \lambda_m^{-1}) + g\, d(c)\mathbf{1}$
  return $C_1 \parallel \cdots \parallel C_m \parallel c$
end procedure
procedure TETDecrypt$(K, T, (\boldsymbol{C}, c), v)$
  $m \leftarrow \mathsf{len}(\boldsymbol{C})$; $(\tau, \lambda_m, \boldsymbol{b}, f, g) \leftarrow$ TETSetup$(K_1, T, nm + |c|, v)$
  $\boldsymbol{Y} \leftarrow$ Mix$(\tau, \boldsymbol{C}, 1) + \lambda_m g\, d(c)\mathbf{1} + \boldsymbol{b}$
  $X_i \leftarrow E_K^{-1}(Y_i) \quad (1 \le i < m)$
  $U \leftarrow E_K^{-1}(Y_m)$; $X_m \leftarrow E_K^{-1}(U)$; $p \leftarrow c + \mathsf{pref}_r(U)$
  $\boldsymbol{M} \leftarrow$ Mix$(\tau, \boldsymbol{X} + \boldsymbol{b}, 1) + \lambda_m f\, d(p)\mathbf{1}$
  return $M_1 \parallel \cdots \parallel M_m \parallel p$
end procedure

B PEP Coefficients and Verification of the Public Parameters

This appendix verifies the fixed public parameters used in Section 3.1 by exact arithmetic. We specify the polynomial and coefficient generator, establish primitivity, and exhaustively check the required trinomial exclusion. These checks establish that the polynomial satisfies the original selection rule while its generated coefficients contain the repeated values used in our attacks. Set $N = 2^{128}$ and encode a binary polynomial by taking integer bit k as its coefficient of x^k; $\mathrm{enc}(\rho)$ includes the leading x^{128} term. Computations initially take place in the quotient ring $\mathbb{F}_2[x]/(\rho)$; the multiplicative-order verification below establishes that it is a field. This representation is independent of the byte-field arithmetic inside AES.

B.1 The fixed public parameter and the relevant branch

The polynomial is encoded by

$$\mathrm{enc}(\rho) = \texttt{0x1650d5e3f398e280a74a282037882005d}. \tag{37}$$

Integer bit k is the coefficient of x^k, including the leading term. For every integer $k \geq 0$, define the length-$3k$ base sequence by

$$u_{3k,i} = \alpha^i \quad (1 \leq i \leq 2k), \qquad u_{3k,2k+j} = \alpha^{2j-1} + \alpha^{2j} \quad (1 \leq j \leq k). \tag{38}$$

Write $u_{3k} = (u_{3k,1}, \ldots, u_{3k,3k})$ and set $u_0 = ()$. For an integer block count $3 \leq m \leq m_{\max} < N$, write $t = \lfloor m/3 \rfloor$. For the attacked branch $m \bmod 3 = 2$, equivalently $m = 3t + 2$, Chakraborty and Sarkar's generator is

$$\begin{aligned}(p_{m,1}, \ldots, p_{m,5}) &= (1+\alpha, \alpha+\alpha^2, \alpha^2+\alpha^3, \alpha^3+\alpha^4, \alpha^4+1),\\ p_{m,5+i} &= \alpha^4 u_{3(t-1),i} \qquad (1 \leq i \leq m-5).\end{aligned} \tag{39}$$

The displayed range uses the coherent remainder length $m - 5$; the other residue classes and indexing clarification appear below.

B.2 Verification of Primitivity

Primitivity follows by verifying that the residue class $\alpha = x \bmod \rho$ has multiplicative order $N - 1$. The complete prime factorization of $N - 1$ is

$$N - 1 = 3 \cdot 5 \cdot 17 \cdot 257 \cdot 641 \cdot 65537 \cdot 274177 \cdot 6700417 \cdot 67280421310721.$$

Each factor is prime, and their product is $N - 1$. Polynomial modular exponentiation gives $\alpha^{N-1} = 1$ and the following nonidentity residues. In this table, ℓ ranges over the distinct prime divisors of $N-1$; each residue uses the field-element encoding defined above, without an explicit hexadecimal prefix. Leading zeroes are immaterial.

Prime ℓ	$\alpha^{(N-1)/\ell}$ in hexadecimal
3	`24285d7a5331313db484ef3e43e7c442`
5	`599b028e45f3c0b424bd502b1c048375`
17	`c970e0e9f5bd979a7bb4f21f963ce8f7`
257	`938ea61f289210265d13cafc96d7c14e`
641	`c4cec6314e209a6813b8653a425ec878`
65537	`6e8f1c3db6ddd2efcd50157dec2467e3`
274177	`38561dd31fe47851b395834fb149ba01`
6700417	`8741b8f581926ed4bf16c8cab614c789`
67280421310721	`25a664322e06ceb3b8fc8057befc37d2`

These tests prove that the residue class of x has order $N-1$. They also establish irreducibility without assuming it: the quotient ring has N elements, and α already generates $N-1$ distinct units. Consequently every nonzero element is a unit, the quotient is a field, and ρ is primitive.

B.3 Trinomial exclusion and the three generation branches

The indices i, j in this paragraph are integer exponents, not block values or derived masks. The trinomial search computes every α^j for $0 \le j \le 1341$. For each j, an exact dictionary lookup tests whether $1+\alpha^j$ equals any preceding nonconstant power α^i, $0 < i < j$. There are no matches. This exhausts all normalized trinomials $1+x^i+x^j$ in the stated range; it is not a sampling experiment. Since $\rho(0)=1$, a general trinomial multiple can be divided by its lowest power of x without changing divisibility. Thus there is no trinomial multiple of degree less than 1342, and choosing $m_{\max} = 1342$ meets Chakraborty and Sarkar's parameter rule. The attack length 1337 lies inside this permitted domain.

The other two generator branches [CS06b, CS06c] use the same base sequence. The notation u_{3k} and its entries are those of Equation (38), where k is a nonnegative integer. At length $m = 3t$, the sequence is exactly u_{3t}. At length $3t+1$, its four-element prefix is $(1+\alpha, \alpha+\alpha^2, \alpha^2+\alpha^3, \alpha^3+1)$, followed by $\alpha^3 u_{3(t-1)}$. At length $3t+2$, the five-element prefix and the remainder are those in Equation (39). The original construction prints the remainder's index upper bound as m. The coherent bounds are $m-4$ and $m-5$, respectively, because the remainder has length $3(t-1)$. We clarify this evident indexing error before testing the generator; the collision survives that clarification. For the smallest branch lengths the remainder is empty.

Direct reduction gives

$$\alpha^{891} = 1+\alpha+\alpha^2+\alpha^3 = \texttt{0x0f}, \qquad \alpha^{892} = \texttt{0x1e}.$$

Therefore $p_{1337,5} = p_{1337,1337} = 1+\alpha^4 = \texttt{0x11}$. The generated vector has 1337 entries and 1336 distinct values, contains neither zero nor one, and has XOR sum zero. PEP with this vector remains invertible when $R = E_K(T) \neq 0$, although it fails the claimed position-separation property. The associated four-term relation is compatible with the verified exclusion of trinomials. These verifications concern the exhibited fixed public instance; they do not establish that other public polynomials fail.

The exact checks can be reproduced with `verification/check_pep.py`. They include primality checks for the listed factors, all modular-power residues, exhaustive trinomial exclusion, and the coefficient identities at lengths 1337 and 1340. The script also checks the uniqueness of c_3 needed in Appendix C. Its separate functional attack tests check formulas and query counts; they do not establish the adaptive probability bound by experiment.

C Probability and composition proofs for PEP recovery

This appendix fixes $N = 2^{128}$, auxiliary block count $h = 1340$, and the public field of Appendix B. All probabilities concern a uniform secret permutation and the adversary's coins. The internal table records every evaluated permutation input–output pair, even when the adversary does not know that pair. The adversary's cache stores only pairs obtained through the attack procedures. These two objects are distinct.

C.1 Coefficient properties needed by a recovery query

For $h = 3 \cdot 446 + 2$, the base-sequence parameter in (39) is 445. The scaled monomials are $\alpha^5, \ldots, \alpha^{894}$, and the scaled binomials are $\alpha^{2j+3} + \alpha^{2j+4}$ for $1 \le j \le 445$. At $j = 444$, the position is $5 + 890 + 444 = 1339$, and the value is $\alpha^{891} + \alpha^{892} = 1 + \alpha^4 = c_5$. Each basic

triple sums to zero, as does the five-entry prefix, so $\sum_{i=1}^{h} c_i = 0$. Also, $c_1 = 1 + \alpha \neq 0$ and $c_3 = \alpha^2 + \alpha^3 \neq 0$.

In particular, c_3 occurs only once in the entire vector. The other adjacent binomials start at exponents $0, 1, 3$ or $5, 7, \ldots, 893$. Equality with $\alpha^2(1+\alpha)$, after cancelling nonzero $1 + \alpha$, would contradict the order $N - 1$ of α. Equality of a monomial with c_3 would give a trinomial multiple of degree below 1342, contradicting the parameter verification in Appendix B. Finally, the difference between the closing term $1 + \alpha^4$ and c_3 is a nonzero polynomial of degree four, which cannot be divisible by the degree-128 polynomial ρ. These coefficient properties follow from the verified parameters and the algebraic relations above.

C.2 Conditional failure bound for one query

Lemma 1. *Consider a recovery query not already answered by the cache. After deriving its parameters R, a, b, fix the complete preceding internal history and its permutation table, containing B distinct input–output pairs, and assume $R \neq 0$. The adversary then chooses independent uniform u, v. If $B + h < N$, the conditional probability of failure or an incorrect return is at most*

$$f(B) = \frac{B}{N} + \frac{B+2}{N-B} + \frac{2}{N} + \frac{B+h}{N-B-h+1} + \frac{1}{N-B-h}. \tag{40}$$

The preceding table may include forward or inverse evaluations, and the current tweak may depend on the entire past transcript.

Proof. Let $\mathcal{U}$ be the table's input set. Parameter derivation does not depend on the newly chosen u, v, so these coins may be sampled after the parameters. Put $S = u + Rv$. For $R \neq 0$, the pair (S, v) is uniform and independent, and $\mu = \pi(a+S)$. Exclude the following events in order; each bound conditions only on the preceding table and steps, not on success of all future steps.

First, $z = a + S$ lies in $\mathcal{U}$ with probability B/N. If z is fresh, μ is uniform over $N - B$ unused outputs. The event $\mu = 0$ or $X_3 = c_3\mu \in \mathcal{U} \cup \{z\}$ excludes at most $B + 2$ outputs, giving the second term. After fixing S, μ, the value v remains uniform, and

$$X_1 = S + Rv + c_1\mu, \qquad X_2 = Rv + c_2\mu.$$

Each equality $X_3 = X_1$ or $X_3 = X_2$ fixes at most one v, giving the third term $2/N$. At every other zero-plaintext coordinate, $X_i = c_i\mu$; uniqueness of c_3 and $\mu \neq 0$ ensure that none equals X_3.

Thus X_3 is a fresh ECB input occurring exactly once. All ECB inputs are fixed before their evaluation, so $\pi(X_3)$ can be sampled last. Evaluate the other ECB coordinates first, and let B' be the actual number of distinct input–output pairs then in the internal table. Since $X_5 = X_{1339}$, we have $B' \leq B + 1 + (h-2) = B + h - 1$. Given this extended table, Y_3 is uniform over exactly $N - B'$ unused outputs. The final mixing input is $w = b + \sum_{i=1}^{h} Y_i$, in which Y_3 occurs exactly once. The event that w lies in the current input set or equals X_3 excludes at most $B' + 1$ values of Y_3, giving the fourth term. If w is fresh, $\nu = \pi(w)$ is uniform over at least $N - B - h$ unused outputs. Since $c_5 \neq 0$, the equality $W = Y_5 + c_5\nu = 0$ fixes at most one ν, giving the last term. Outside these events, R, W are nonzero, and (20) returns the correct value. The argument never treats W itself as an independent uniform value. □

Fix x independently of π and start the first recovery with an empty internal table and cache, before any query or target generation. Then $\Pr[\pi(x) = 0] = 1/N$. Let Fail_1 denote failure or an incorrect return in this first recovery. After parameter derivation, $B \leq 3$;

apply the conditional bound to each complete derivation history with $R \neq 0$ and average, obtaining

$$\begin{aligned} \Pr[\mathsf{Fail}_1] &\leq \frac{1}{N} + \left(1 - \frac{1}{N}\right) f(3) \leq \frac{1}{N} + f(3) \\ &\leq \frac{1355}{N - 1343} < 2^{-117}. \end{aligned} \tag{41}$$

Repeated internal evaluations only reduce the table size; the three parameter inputs need not be distinct. This numerical bound applies only to the stated first recovery from an empty table; with a preceding history, use $f(B)$ or the composition bound below.

C.3 Adaptive calls, the zero cache, and resource accounting

The first-input bound cannot simply be multiplied by the number of calls. Later tweaks depend on earlier responses, and knowledge of $\pi^{-1}(0)$ permits deliberately choosing a zero-multiplier tweak. We prove (25) and explain how the attack handles this issue.

Consider plaintext recovery in Section 3.3, with $q_R = 9$ and S_* as in (25). For recovery, fix the target tweak and plaintext independently of the secret permutation, then generate the target ciphertext. Expand target generation and auxiliary queries into primitive calls. For the analysis, consider only the normal prefix before the first incorrect recovery, recovery failure, or zero event defined below. Every cached pair is correct throughout this prefix. Each normal h-block encryption or decryption uses at most $h+5$ primitive calls, regardless of how many calls reuse existing entries.

Before the second inverse ECB coordinate of the first auxiliary decryption, all new table entries arise from forward evaluations. Its first inverse coordinate is $\pi^{-1}(R_0) = T_0$, already in the table; the second explicitly evaluates $x_0 = \pi^{-1}(0)$. The auxiliary query is atomic: its entire response arrives before any next recovery starts. Equation (23) lets the adversary cache $(x_0, 0)$ from this response. Every later request for $\pi(x_0)$ is answered from that cache. Let Z be the event that, before the first incorrect or failed recovery, a fresh forward call first reaches output zero before this explicit inverse step. This is the only additional zero event to count. If $S_* < N$ is a deterministic bound on primitive calls in the normal prefix, each such fresh evaluation reaches zero with conditional probability at most $1/(N - S_*)$ given its past history. Thus

$$\Pr[Z] \leq \frac{S_*}{N - S_*}. \tag{42}$$

This accounting includes a zero target multiplier for multiblock messages. Along the prefix before Z or any incorrect or failed recovery, every recovery query actually executed has a nonzero multiplier. After the first inverse exposure of zero, even new entries from other inverse evaluations cannot create an unknown zero-output point.

Number the at most q_R uncached recoveries in execution order. For recovery j, let $\mathcal{H}_j$ be the complete internal history after deriving its parameters R, a, b but before sampling its u, v, including all existing table entries, preceding query responses, and coins already used. Write B_j for its table size. Let G_j mean that this recovery is reached normally: no earlier Z, incorrect recovery, or recovery failure has occurred, the current parameters have been derived, and $R \neq 0$. Assign dummy histories to unreached indices and set their G_j to false. The event G_j is determined entirely by $\mathcal{H}_j$: correctness of earlier recoveries can be checked against the internal table, since the actual $R = \pi(x)$ is evaluated during each recovery's parameter derivation.

Given any such complete past history, the unevaluated permutation is still uniform over all extensions of the table. Past normality is determined by that history and imposes no further constraint on unspecified permutation values. The current u, v are independent of $\mathcal{H}_j$, and parameter derivation does not depend on them, so they may be sampled at

this point and Lemma 1 applied. Let F_j mean that this complete recovery query fails or returns an incorrect value. On every history satisfying G_j, using $B_j + h \le S_*$ and $B_j \le S_*$ gives

$$\Pr[F_j \mid \mathcal{H}_j] \le f(B_j) \le \frac{3S_* + h + 5}{N - S_*}. \tag{43}$$

Outside Z and all incorrect or failed recoveries, the anchor formulas recover every auxiliary inverse value exactly, the cache remains correct, and target plaintext recovery is exact. Thus, writing Fail for failure or an incorrect output of the overall procedure, a first-failure union bound and conditional expectation give

$$\begin{aligned} \Pr[\mathsf{Fail}] &\le \Pr[Z] + \sum_{j=1}^{q_R} \mathbb{E}\big[\mathbf{1}_{G_j} \Pr[F_j \mid \mathcal{H}_j]\big] \\ &\le \frac{S_* + q_R(3S_* + h + 5)}{N - S_*} = \varepsilon(q_R, S_*). \end{aligned} \tag{44}$$

Each application conditions only on the complete past history and normal arrival, never on success of all future recoveries or on the global event Z^c. Cache correctness is used only along the normal prefix; arbitrary zero-branch completions do not affect the preceding normal evaluations.

In the retained range, $S_* \le 14793$, so $\varepsilon(9, S_*) \le 426309/(N - 14793) < 2^{-109}$.

D Additional attacks and extensions

The main text uses representative attacks to explain the principal defects. For reasons of space, this appendix only outlines additional attacks and conclusions from the same mechanisms, without expanding their algorithms or proofs. PEP uses the fixed public instance in (37), $N = 2^{128}$, and $h = 1340$; recovery bounds still assume a uniform secret permutation, with target tweaks and plaintexts fixed independently of it. Write $\varepsilon(q, S)$ for the same function as in (25), evaluated at the corresponding total-call budget $S < N$; q_R, q_D bound the auxiliary encryption and decryption counts. TET uses the two inverse-consistent corrections, valid parameters, and $n \le mn + r < N$; the noninverse printed pair is not treated as a TES.

D.1 PEP

A one-query subgroup test. For $m = 1337$, encrypt only the all-zero message and set $A = C_5$, $B = C_{1337}$. Repeated coefficients force the nonzero ratio $B/A = R^{1332}$ into the cube subgroup, since $\gcd(1332, N - 1) = 3$. Accept only if $A = B = 0$, or if $A, B \ne 0$ and $(B/A)^{(N-1)/3} = 1$, avoiding division by zero. The ideal acceptance probability is $p_1 = [1 + (N - 1)^2/3]/N^2$. The natural forward computation always passes; for any completion confined to $R = 0$, the uniform-permutation advantage is at least $1 - N^{-1} - p_1$, approximately $2/3$. The cost is one 1337-block encryption.

A five-query cross-length distinguisher. Using the recovery procedure in Section 3.3, at most four h-block encryptions obtain the four permutation values needed to predict the single-block ciphertext for an independently fixed (T, M). One further single-block encryption tests the prediction, giving a classical CPA distinguisher with at most five encryptions and $4h + 1$ message blocks. Its advantage is at least $1 - \varepsilon(4, 5384) - N^{-1}$, where $\varepsilon(4, 5384) = 75372/(N - 5384)$. The same approach can simulate other supported lengths different from h: the real domains share one permutation, whereas ideal TES length domains are independent. The attack requires the auxiliary length and chosen tweaks under the same key, and stops on recovery failure.

Plaintext recovery at other supported lengths. Anchored inversion in Section 3.3 handles $h-2=1338$ target values per batch. Two batches cover longer targets; Chakraborty and Sarkar's respective special branches [CS06b, CS06c] can be inverted to handle one- and two-block targets, covering all $1 \le m \le 1342$. Outside the main-text range, the bounds are 7 encryptions and 1 decryption for $m=1$; 9 encryptions and 1 decryption for $m=2$; and 10 encryptions and 2 decryptions for $1339 \le m \le 1342$. Every auxiliary query has h blocks, decryption uses a different tweak from the target, and the respective failure probabilities are below 2^{-110}, 2^{-109}, and 2^{-108}. They follow by substituting $S=(q_R+q_D)(h+5)+s_{\mathrm{tar}}(m)$ into $\varepsilon(q_R,S)$, with $s_{\mathrm{tar}}(1)=4$, $s_{\mathrm{tar}}(2)=7$, and $s_{\mathrm{tar}}(m)=m+5$ otherwise. Nonzero checks, caching of the first zero preimage, and stopping on failure remain in force. No unique preimage is claimed for a noninjective zero-multiplier branch, and this is not encryption-only plaintext recovery.

D.2 TET

Plaintext recovery with a one-bit tail. For $r=1$, first calibrate the scalar using Section 4.2 under a valid auxiliary tweak $T_0 \neq T^*$. Then make one target-derived decryption with nonzero tail difference and undo the known difference. $\mathsf{TET}_{\mathrm{B}}$ needs two calibration decryptions and one target-derived decryption, totaling three decryptions; $\mathsf{TET}_{\mathrm{A}}$ needs two calibration encryptions and one decryption. Both recover the target with probability one for every data permutation and valid parameters, without querying its decryption. The unique nonzero tail difference suffices; individual zero response blocks cause no difficulty.

Multiple targets after one calibration. For J targets under the same key and m, r, where $1 \le r < n$, require pairwise distinct target tweaks and a valid calibration tweak outside their set. One calibration followed by one derived decryption per target costs at most $J+2$ decryptions for $\mathsf{TET}_{\mathrm{B}}$, or two encryptions and J decryptions for $\mathsf{TET}_{\mathrm{A}}$, recovering every target with probability one. Nonzero tail differences and these tweak conditions ensure that no decryption submits any target tuple. Scalar reuse is restricted to a fixed full-block count; no propagation across m is claimed. In these two extensions each query has $mn+r$ bits, or $m+1$ rounded-up message blocks, and recovery takes $O(m)$ field operations per target.

A general criterion for public cancellation. In (26), if $k=\lambda_m f$ is publicly computable, choose plaintext difference $(k\Delta\mathbf{1},\delta)$; if g is publicly computable, choose ciphertext difference $(g\Delta\mathbf{1},\delta)$, with $\delta \neq 0$ and $\Delta = \delta \parallel 0^{n-r}$. The corresponding two encryptions or two decryptions cancel the entry difference. Testing for tail difference δ and a common full-block difference gives advantage $1-N/(2^{mn+r}-1)$. If the surviving scalar (g forward, $\lambda_m f$ inverse) is guaranteed nonzero, additionally requiring a nonzero common difference replaces the numerator by $N-1$. Dividing the common difference by nonzero Δ also reveals that scalar. $\mathsf{TET}_{\mathrm{A}}$ and $\mathsf{TET}_{\mathrm{B}}$ are the respective cases $\lambda_m f=1$ and $g=1$; failing this criterion does not establish security.

Sampling the bridge value. A sampling argument for a partial-block response must distinguish uniformity of the response from uniformity of its internal representation. For fixed parameters and input tail p, consider the map

$$(\boldsymbol{Y},U) \longmapsto (\boldsymbol{C},c), \qquad c = p + \mathsf{pref}_r(U), \quad \boldsymbol{C} = A_m(\boldsymbol{Y}+\boldsymbol{b}) + g\,d(c)\mathbf{1}.$$

This maps $mn+n$ bits to $mn+r$ bits and is balanced, not bijective. For each fixed response, there are exactly 2^{n-r} choices of U with prefix $p+c$, and exactly one corresponding $\boldsymbol{Y}$, obtained from the first line of (27). Independent uniform sampling of $(\boldsymbol{Y},U)$ therefore

produces a uniform response in this sampling view. It does not imply that these variables are independent in a real cipher execution. Once a plaintext–ciphertext transcript is fixed, the first r bits of U are fixed by $p + c$; the bridge cannot at the same time be treated as a fresh uniform n-bit value. A security proof must justify the passage between these sampling views.