Associated-Data Correlations in Eevee: Deterministic Forgeries against Nonce-Misuse Security

Takanori Isobe[1] and Mostafizar Rahman[2]

[1] The University of Osaka, Suita, Japan
[2] Nanyang Technological University, Singapore, Singapore

Abstract

Eevee's Umbreon and Espeon modes claim online authenticated-encryption security under nonce reuse. We give an independent analysis of associated-data correlations that yield probability-one non-empty-message forgeries with three encryption queries and one verification query. Two AD-only tags reveal a mask difference that translates a ciphertext to another AD value. The basic attacks and their proof diagnosis overlap with independent and concurrent work by Mao et al. Our complementary contribution is a systematic affine treatment of partial final message blocks. For any admissible fixed final-fragment length r, $n - r + 3$ encryption queries suffice; allowing the forged length to be r or $r - h$ for a chosen $1 \leq h < r < n$, while choosing the source plaintext adaptively, reduces this to $n - r + 2$. Both constructions satisfy the padding check and succeed with probability one. They apply to general message lengths, with at least one preceding complete block for Espeon. We also analyze the published short-message Espeon AD omission, validate the published complete-block Umbreon attack using the official ForkSkinny primitive in a specification adapter, and report a separate two-query attack accepted by the unmodified public C modes. Finalization is discussed only as an unproved countermeasure direction.

Keywords: authenticated encryption, nonce misuse, online AE, forkcipher, Eevee, forgery, cryptanalysis

1 Introduction

Eevee [BPA+23b] is a family of three authenticated-encryption with associated-data (AEAD) modes—Umbreon, Jolteon, and Espeon—designed for an IoT-to-cloud transciphering setting. The design goal is unusual and attractive: encryption should remain lightweight on constrained devices while decryption should be highly parallelizable inside secure multi-party computation (MPC). The modes are built from a forkcipher, and the original work gives a unified proof in the online authenticated-encryption (OAE) framework of Hoang et al. [HRRV15]. In particular, Umbreon and Espeon are claimed to retain strong security under nonce reuse, while Jolteon deliberately gives up nonce-misuse security for additional performance.

This paper shows that Eevee's associated-data (AD) processing compromises non-empty-message authentication under its claimed nonce-misuse model. The starting relation is classical: Bellare, Guérin, and Rogaway explicitly describe the same four-message

Emails: takanori.isobe@ist.osaka-u.ac.jp (Takanori Isobe), mrahman454@gmail.com (Mostafizar Rahman)

Date of this document: 2026-10-17.

linear dependency for an unprotected XOR accumulator [BGR95, Section 2]. Eevee's AD-only interface exposes such an accumulator under repeated nonces. The scheme-specific question is how this exposure interacts with the message path and the published security proof. We show that shared AD processing carries the correlation into non-empty-message authentication, and identify where the proof treats marginal uniformity as joint transcript independence.

Figure 1 shows the main attack before we give the full equations. Here $A_i = X_i \| Y$ are distinct two-block AD strings with a shared final block. The two AD-only tags disclose the difference between the AD masks used by the subsequent message query; this is the step that turns a familiar XOR relation into a non-empty-message forgery.

1	**Observe**	$u_0 = P_N(X_0) \oplus Q_{N,1}(Y)$	
AD-only		$u_1 = P_N(X_1) \oplus Q_{N,1}(Y)$	
2	**Cancel**	$\delta = u_0 \oplus u_1 = P_N(X_0) \oplus P_N(X_1)$	
		$\delta = T_A^{(0)}(A_0) \oplus T_A^{(0)}(A_1)$	
3	**Forge**	$\mathsf{Enc}_K(N, A_0, M) = (C, \tau)$	Umbreon : $C^\star = C \oplus \delta$
		$(N, A_1, C^\star, \tau) \mapsto$ accept	Espeon : $C_1^\star = C_1,\ C_2^\star = C_2 \oplus \delta$

Figure 1: The three-encryption, one-verification attack at a single nonce N. Set $A_i = X_i \| Y$ and $u_i = \mathsf{Enc}_K(N, A_i, \varepsilon)$. The common last-block term cancels, exposing the difference of the private message-path masks. Umbreon uses one message block and published Espeon uses two; the tag τ is unchanged. Section 3.2 proves acceptance.

1.1 Our contributions

Our analysis develops Eevee's AD correlations into a systematic treatment of non-empty-message forgeries, including partial final blocks. The basic attacks and proof diagnosis overlap with independent and concurrent work by Mao et al. [MWH+26]; Section 1.2 separates the shared results from the complementary contributions.

1. **Non-empty-message forgeries and partial-block extensions.** Two AD-only queries reveal the difference between the AD masks of two message contexts with the same last AD block. A third encryption query yields a fresh accepted ciphertext under the other AD. The attacks use $q_e = 3$, $q_v = 1$, and $\mu = 3$ and succeed with probability one: one message block for Umbreon and two for the published Espeon pseudocode. For a final fragment of r bits, we also derive probability-one forgeries with $q_e = \mu = n - r + 3$ and $q_v = 1$; the added AD-only queries select a difference that preserves the final padding. Allowing the forged final length to be r or $r - h$ for any $1 \le h < r < n$, with an adaptively chosen source plaintext, saves one query: $q_e = \mu = n - r + 2$. A shared partial final AD fragment also shortens the AD required for either variant.

2. **Proof failure and related attacks.** A four-query AD-only distinguisher has advantage $1 - 2^{-n}$, while three AD-only queries forge with certainty. An affine synthesis extends the correlation to several AD positions. These legal transcripts reveal the original proof's incorrect passage from individual-tag uniformity to joint-transcript independence. We also give a separate nonce-respecting AD-substitution forgery against the published one-block Espeon path.

3. **Implementation validation.** We execute the published Umbreon complete-block attack with the official ForkSkinny C primitive in a specification adapter. The public C

Table 1: Attacks on the published Eevee modes, with exact success probabilities in the ideal-forkcipher model. Here r is the final message fragment's length ($r = n$ is a full block); the Espeon final-fragment forgeries require at least one preceding full message block. "Both" means Umbreon and Espeon. The adaptive-length forgery permits a final length of r or $r - h$ for $1 \leq h < r < n$. NR means nonce-respecting.

Target	Attack type	Section	q_e	q_v	μ	Success/ advantage
Umbreon	Non-empty-message forgery (1 block)	§ 3.2.1	3	1	3	1
Espeon	Non-empty-message forgery (2 blocks)	§ 3.2.2	3	1	3	1
Both	Final-fragment forgery (fixed r)	§ 3.2.3	$n - r + 3$	1	$n - r + 3$	1
Both	Final-fragment forgery (adaptive length)	§ 3.2.3	$n - r + 2$	1	$n - r + 2$	1
Both	AD-only OAE distinguisher	§ 3.1.1	4	0	4	$1 - 2^{-n}$
Both	AD-only forgery	§ 3.1.1	3	1	3	1
Espeon	NR AD-substitution forgery	§ 3.3	1	1	1	1

implementation has a separate conditional-expression issue; its two-query source-level forgeries verify in both Umbreon and Espeon.

The four-message XOR identity is already present in XOR-MAC's design rationale.

Both this work and [MWH+26] independently connect Eevee's legal AD-only interface to non-empty-message forgeries in the original security game. Our additional results explain how affine synthesis preserves the padding of a partial final block and how adaptive length selection saves one encryption query. The public-source consequence is distinct from the attack on the published specification. We discuss finalization only as a possible countermeasure and give no security proof for it.

Table 1 summarizes the attacks. The attacks are structural: no differential, linear, related-key, or reduced-round property of ForkSkinny is used.

1.2 Independent and concurrent work

Mao, Wang, Han, Guo, Jia, and Chen [MWH+26] independently and concurrently analyzed structural weaknesses in Eevee. Their preprint and this work both obtain the AD-only rectangle distinguisher, the AD-only forgery, the three-encryption non-empty-message forgeries against Umbreon and Espeon, and the nonce-respecting AD substitution on Espeon's published short-message path. Both analyses also identify failures in the original proof's treatment of correlated AD tags and message-domain separation. We retain our derivations to make the present paper self-contained and to establish the starting point for the extensions below.

The complementary results here are the explicit affine synthesis over several AD positions (Proposition 3), the fixed-length partial-final-block construction with $q_e = n-r+3$ (Proposition 4), and the adaptive-length construction with $q_e = n - r + 2$ (Theorem 3). They cover arbitrary numbers of preceding complete message blocks, with at least one required for Espeon, under the stated counter-range conditions. The last AD fragment may also be partial, reducing the AD length. These extensions of the AD-correlation forgery are not covered in the version of [MWH+26] dated 15 October 2026. Their short-message

Table 2: Scope of this paper and the 15 October 2026 version of Mao et al. [MWH+26]. The shared results were obtained independently and concurrently. *Not treated* describes the scope of a paper.

Result	This paper	Mao et al.
AD-only rectangle and forgery	§ 3.1.1	Attacks 2–3
Complete-block AD-correlation forgeries: Umbreon / Espeon	§ 3.2.1–§ 3.2.2	Attacks 4–5
Short-message Espeon AD substitution (NR)	§ 3.3	Attack 7
Affine synthesis over several AD positions	Proposition 3	Not treated explicitly
Partial-final-block AD-correlation forgeries: fixed / adaptive length	Proposition 4; Theorem 3	Not treated
AD-free online-prefix distinguishing	Not treated	Attack 1
Jolteon cross-length forgery (NR)	Not treated	Attack 6

Espeon AD-substitution attack does cover partial messages; that is a different attack and is included in the overlap above.

Their scope is broader in other directions. In particular, [MWH+26] gives an AD-free online-prefix distinguisher and a nonce-respecting cross-length forgery against Jolteon, which are not results of this paper. Table 2 summarizes the comparison. Both works distinguish the published specification from the public C implementation. The final-AD conditional-expression issue recorded in Appendix A is also noted in their implementation analysis. Our appendix gives an explicit two-query source-level construction, with accepting transcripts in our artifact.

1.3 Related work

XOR-MAC and PMAC. Bellare, Guérin, and Rogaway's XOR-MAC analysis explicitly gives the rectangle dependency for an XOR of position-encoded PRF outputs without its protective random-seed or counter term [BGR95, Section 2]. Their constructions use that term to protect the accumulator. Black and Rogaway's PMAC instead applies a final block-cipher evaluation to a checksum incorporating the message blocks and length-dependent treatment [BR02, Figure 1]. These works establish the relevant design background. Our rectangle and affine relations are instances of classical XOR structure; the results here concern Eevee's exposure of that structure under its claimed nonce-reuse guarantees and its interaction with non-empty messages. XOR accumulation itself is compatible with secure authentication when its surrounding construction supplies the required protection.

Eevee was introduced by Bhati et al. [BPA+23b] as an IoT- and MPC-friendly family with OAE security and strong nonce-misuse guarantees for Umbreon and Espeon. The design builds on forkciphers and forkcipher-based AEAD modes [ALP+19, ABPV21]. XOR-based AD accumulation is familiar from prior AEAD design: OCB's HASH XORs offset-separated blockcipher images and requires unique encryption nonces [KR14], while PAEF is proved optimally secure in the nonce-respecting model and its proof explicitly relies on unique tweak–permutation combinations [ALP+19]. Our attack is therefore not a criticism of XOR accumulation in the nonce-respecting setting; it isolates what changes when Eevee claims security after nonce reuse while exposing correlated accumulator values. OAE and nonce-reuse security for online modes were developed by Hoang et al. [HRRV15], and related analyses of SAEF studied nonce-misuse and release-of-unverified-plaintext properties [ABV21, BAV24].

Rahman, Kundu, and Isobe [RKI25] later showed that Eevee and other forkcipher-based AEADs lack key-commitment properties. Their attacks use different keys and explicitly

do not claim to violate Eevee's advertised AEAD security. Our results are orthogonal: all attacks use one key and directly contradict the OAE bounds claimed for Umbreon and Espeon. They therefore affect the core integrity/confidentiality claim rather than an additional robustness notion.

Organization. Section 2 recalls Eevee and its OAE game. Section 3 develops the repeated-nonce attacks and the independent published Espeon specification issue. Section 4 diagnoses the proof, Section 5 reports implementation checks, and Section 6 discusses a possible countermeasure and concludes.

2 Preliminaries and Security Model

We recall the parts of Eevee needed by our attacks and proof analysis. Let F be an n-bit tweakable forkcipher with t-bit tweaks. For a fixed tweak T, branch $b \in \{0,1\}$ is a permutation of $\{0,1\}^n$. We write $F_K^{T,b}(X)$ for a branch output and $F_K^{T,\mathsf{both}}(X)$ for the pair of branch outputs. The original security theorem replaces the forkcipher with independent random tweakable permutations and then proves OAE bounds for the resulting modes [BPA+23c, Theorem 3.1]. We use the published specification in Fig. 9 of that full version. Let $d = t - |N| - 2$, and let $\langle j \rangle_w$ denote the w-bit encoding of an integer $0 \le j < 2^w$. All queries below use legal nonce and input lengths with representable counters. In particular, Espeon requires $|N| \le n - 2$, and Umbreon messages with m ordinary blocks require $m + 1 < 2^{d+1}$.

2.1 Associated-data processing

For a fixed nonce N, associated data is padded and split into blocks. Ordinary full blocks are processed under counter tweaks and XORed into an accumulator. The last AD block uses a tweak that also encodes whether the message is empty. For the two-full-block case $A = X \| Y$, define

$$P_N(X) = F_K^{N\|\langle 5\rangle_d\|00,0}(X), \tag{1}$$

$$Q_{N,b}(Y) = F_K^{N\|\langle 2+b\rangle_d\|00,0}(Y), \tag{2}$$

where $b = 1$ for an AD-only query and $b = 0$ when a non-empty message follows. Then the AD accumulator is exactly

$$T_A^{(b)}(X,Y) = P_N(X) \oplus Q_{N,b}(Y). \tag{3}$$

For an AD-only query, Eevee returns $T_A^{(1)}$ directly as the tag.[3]

The attack only needs the fact that the first-block term $P_N(X)$ is shared between $b = 0$ and $b = 1$. The last-block function may differ.

2.2 OAE games and claimed bounds

The Eevee paper separates OAE confidentiality (*oprpf*) and authenticity (*auth*). In the ideal confidentiality game, a query (N, A, M) receives an online-permutation prefix and an

[3]We inspected the ePrint full version (Report 2023/1361) on 26 September 2026. In its Fig. 9, the AD routine XORs every processed AD-block output into T_A and returns T_A (lines 19–25 of the pseudocode), while Enc then executes "if $|M| = 0$ return T_A" (lines 42–43). The inspected pseudocode has exactly the AD-only behavior modeled in (3); see [BPA+23c].

independent random suffix indexed by (N, A, M). In particular, for $M = \varepsilon$, distinct AD strings under a fixed nonce receive independent random n-bit tags in the ideal world.[4]

We use the split notation (C, τ) for readability, where C denotes the message-length ciphertext part and τ the n-bit tag. In the OAE game these together form the ciphertext returned by the encryption oracle. For an AD-only query, $C = \varepsilon$ and the entire returned ciphertext is the tag τ.

For a non-empty message, we distinguish m ordinary full blocks from the final fragment M_*, which has $1 \le r \le n$ bits. Thus $|M| = mn + r$. Figure 2 shows the convention and the internal padding of a partial fragment; a complete final block has $r = n$.

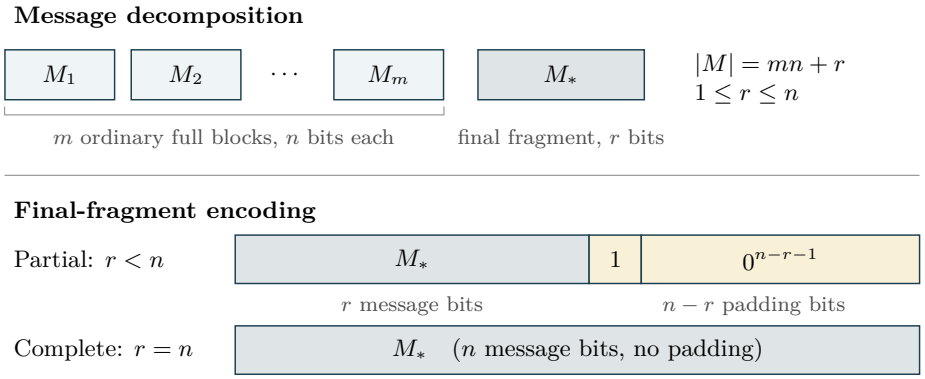

Figure 2: The message-length convention used in Table 1. The ordinary prefix may be empty. The final fragment is the last r message bits; it is a full block when $r = n$. A partial fragment is padded internally to n bits, while its transmitted ciphertext fragment still has r bits. Box widths are schematic.

We write $\mathrm{pre}_r(X)$ for the first r bits of X. For $|X| = r < n$, let $\mathrm{pad}_r(X) = X \| 1 \| 0^{n-r-1}$; for $|X| = n$, let $\mathrm{pad}_n(X) = X$.

Here q_e and q_v count encryption and verification queries. The nonce multiplicity μ is the maximum number of encryption queries using one nonce. The quantities $\sigma_{m,e}$ and $\sigma_{m,v}$ count forkcipher calls induced by the message parts of encryption and verification queries, respectively; AD processing is excluded. An AD-only query contributes zero, and a non-empty message with m ordinary blocks contributes $m + 1$. The original theorem assumes $\mu \le 2^{n-1}$ and $\sigma_{m,e}, \sigma_{m,v} \le 2^{n-2}$; for Espeon, $t \in \{n, 2n\}$. Under these conditions, for an ideal forkcipher the claimed bounds reduce to

$$\mathsf{Adv}^{\mathrm{oprpf}}_{\mathsf{Umbreon}}(\mathcal{A}) \le \frac{3(\sigma_{m,e} + q_e)(\mu - 1)}{2^{n-1}}, \tag{4}$$

$$\mathsf{Adv}^{\mathrm{oprpf}}_{\mathsf{Espeon}}(\mathcal{A}) \le \frac{2q_e(\mu - 1)}{2^n} + \frac{\sigma^2_{m,e}}{2^{t-3}}, \tag{5}$$

The authenticity bound additionally satisfies

$$\mathsf{Adv}^{\mathrm{auth}} \le \mathsf{Adv}^{\mathrm{oprpf}} + q_v / 2^{n-4}.$$

These are the bounds stated in the original analysis [BPA+23c, Theorem 3.1]. The concrete 128-bit examples below satisfy its query limits and violate the bounds by a constant amount. For general message lengths, comparisons with these bounds are restricted to the stated parameter range.

[4] Appendix D of the same full version defines the ideal response as $\pi_{N,A}(M_L) \| f_{N,A,M}$, with a random suffix $f_{N,A,M}$ sampled separately for each tuple (N, A, M). When $M = \varepsilon$, the online-permutation prefix is empty and the suffix consists of exactly the n tag bits. Thus distinct AD values under the same nonce index distinct independent random tags [BPA+23c].

2.3 Attack model and validity of the queries

The following lemma records the three game-syntax facts used throughout the paper; it is included to make clear that the attacks do not rely on an additional misuse notion.

Figure 3 illustrates the nonce multiplicity μ; verification queries are counted separately.

Lemma 1 (Admissibility in the Eevee OAE games)**.** *Under the syntax and OAE games used in the Eevee full version [BPA+23c]:*

(i) an encryption query (N, A, ε) with non-empty AD is legal, and the original proof explicitly treats such AD-only queries;

(ii) for the nonce-misuse adversary used for Umbreon and Espeon, the same encryption nonce may be used repeatedly, with the maximum multiplicity recorded as μ; and

(iii) the original authenticity proof treats a verification tuple with a new (N, A, C) component as a nontrivial forgery candidate. Hence an AD-only target with an unqueried AD value is fresh, and so are the non-empty targets below whose AD and/or ciphertext differ from every encryption response.

Proof. The AEAD syntax in the original paper allows binary strings as AD and message inputs, subject to the legal-length conditions above; the concrete algorithms reject the both-empty input. Its OAE proof explicitly discusses the case "message length is zero, i.e., only AD is queried" and separately analyzes nonce-misusing adversaries whose nonces may repeat up to μ times. In the integrity proof, Case 1 is precisely a verification query for which the tuple consisting of nonce, AD, and the message-length ciphertext portion is new relative to the encryption transcript. These are exactly the query patterns used below. □

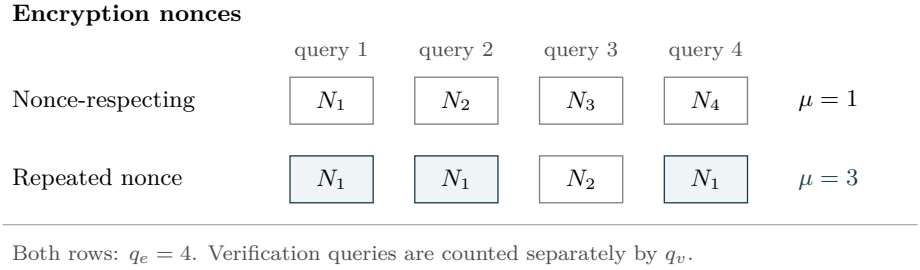

Figure 3: Query counts and nonce multiplicity. Distinct symbols denote distinct nonce values. The maximum number of encryption queries sharing one nonce is μ; it need not equal the total encryption-query count q_e.

All repeated-nonce attacks below use one key. The headline non-empty attacks use three encryption queries under a single nonce; the partial-fragment extensions state their larger query counts explicitly. We never use release of unverified plaintext, related keys, primitive queries, faults, or side channels. The only nonce-respecting attack is the independent Espeon specification break in Section 3.3. The analogous repeated-nonce correlations also occur in Jolteon, but we do not count them as a break because Jolteon deliberately does not claim nonce-misuse security.

3 Attacks against Eevee

We first isolate the AD relation exposed under nonce reuse, then transfer it to non-empty-message authentication. The final subsection treats a separate nonce-respecting failure of the published Espeon pseudocode.

3.1 The Exposed AD Relation

We first establish the relation visible in AD-only tags; its transfer to non-empty-message authentication is the main result of Section 3.2. The XOR relation itself predates Eevee.

3.1.1 Rectangle correlations in AD-only tags

We first instantiate the classical XOR-accumulator relation [BGR95, Section 2] in Eevee's AD-only interface, before treating message processing.

Lemma 2 (Rectangle relation)**.** *Fix a key K, nonce N, and the AD-only processing class. Let $X_0 \neq X_1$ and $Y_0 \neq Y_1$ be n-bit strings, and let*

$$\tau_{ij} = \mathsf{Enc}_K(N, X_i \| Y_j, \varepsilon), \qquad i, j \in \{0, 1\}.$$

Then

$$\tau_{00} \oplus \tau_{01} \oplus \tau_{10} \oplus \tau_{11} = 0^n. \tag{6}$$

The equality is deterministic and holds for every forkcipher instance.

Proof. By (3) with $b = 1$, $\tau_{ij} = P_N(X_i) \oplus Q_{N,1}(Y_j)$. Every term appears exactly twice in the XOR of the four tags, hence cancels. □

Proposition 1 (Four-query OAE distinguisher)**.** *Within the nonce-misuse OAE game of Lemma 1, four AD-only encryption queries suffice to distinguish real Eevee from the OAE ideal world with advantage $1 - 2^{-n}$.*

Proof. The adversary makes the four queries of Lemma 2 and outputs real iff their XOR is zero. In the real world this always occurs. In the OAE ideal world, the four AD-only tags are independent uniform n-bit strings because the four (N, A, M) tuples are distinct. Their XOR is zero with probability 2^{-n}. □

For Umbreon, $q_e = 4$, $\mu = 4$, and $\sigma_{m,e} = 0$, so (4) gives at most $36/2^{n-1}$ in the ideal-forkcipher model. For Espeon, (5) gives $24/2^n$. Both are negligible, while Proposition 1 gives advantage $1 - 2^{-n}$.

Proposition 2 (Three-query AD-only forgery)**.** *Three legal AD-only encryption queries and one verification query produce a fresh valid AD-only ciphertext/tag pair with probability one.*

Proof. Query (X_0, Y_0), (X_0, Y_1), and (X_1, Y_0) to obtain $\tau_{00}, \tau_{01}, \tau_{10}$. Set

$$\tau^\star = \tau_{00} \oplus \tau_{01} \oplus \tau_{10}.$$

By Lemma 2, $\tau^\star = \tau_{11}$. Therefore $(N, X_1 \| Y_1, C = \varepsilon, \tau^\star)$ is valid and was not returned by the encryption oracle. □

The proposition applies algebraically to all three modes if nonce reuse is permitted. It is a direct contradiction only for modes whose security claim covers nonce misuse, namely Umbreon and Espeon.

3.1.2 Affine synthesis over several AD positions

The rectangle is the two-coordinate instance of a broader property.

Proposition 3 (Affine synthesis)**.** *Fix a nonce, AD length and padding class, and the final AD block. Let $k \geq 2$ ordinary AD positions vary independently between baseline values $A_i^{(0)}$ and alternatives $A_i^{(1)}$, with $A_i^{(0)} \neq A_i^{(1)}$ at every varying position. A baseline query plus one singleton-change query per position, for $k+1$ encryption queries in total and nonce multiplicity $\mu = k+1$, determines the valid AD-only tags of all 2^k combinations. Consequently, any unqueried combination can be forged with probability one.*

Proof. Because every ordinary AD block is processed under a position-specific tweak and all outputs are XORed, the AD-only tag has the form

$$\tau(v_1, \ldots, v_k) = R \oplus \bigoplus_{i=1}^{k} P_i(A_i^{(v_i)}),$$

where R contains all fixed-block contributions. Let $\tau_0 = \tau(0, \ldots, 0)$ and let τ_i be the tag when only position i is switched to its alternative. Direct cancellation gives

$$\tau(v_1, \ldots, v_k) = \tau_0 \oplus \bigoplus_{i:v_i=1} (\tau_0 \oplus \tau_i). \tag{7}$$

Thus the $k+1$ observed tags determine the other $2^k - k - 1$ tags exactly. For $k \geq 2$ at least one of these combinations was not queried, and outputting its synthesized tag is a valid authenticity forgery. □

This shows that the issue is not a special four-point collision but an affine code embedded in the repeated-nonce AD transcript.

3.2 Cross-Purpose Non-Empty-Message Forgeries

The AD-only correlation also yields a normal non-empty-message forgery.

Choose two full two-block AD strings sharing the last block,

$$A_0 = X_0 \| Y, \qquad A_1 = X_1 \| Y.$$

First make the two AD-only queries

$$u_i = \mathsf{Enc}_K(N, A_i, \varepsilon) = T_A^{(1)}(X_i, Y), \quad i \in \{0, 1\},$$

and define $\delta = u_0 \oplus u_1$. From (3),

$$\delta = P_N(X_0) \oplus P_N(X_1) = T_A^{(0)}(X_0, Y) \oplus T_A^{(0)}(X_1, Y). \tag{8}$$

Thus AD-only queries reveal the exact difference between the AD masks used by non-empty-message queries, despite the dedicated final-AD tweak for the two cases. Moreover, because $X_0 \neq X_1$ and P_N is a permutation, $\delta \neq 0$.

3.2.1 Umbreon

Consider a full one-block message $M \in \{0,1\}^n$. In Umbreon there are no ordinary message-loop calls; immediately before the final forkcipher call, the mode XORs the AD accumulator into the final message block. Therefore, for AD A_0 the final primitive input is

$$Z = M \oplus T_A^{(0)}(A_0).$$

Let the final forkcipher return (τ, R), where τ is the tag branch and R the other branch. The transmitted final ciphertext is $C = R \oplus M$.

Theorem 1 (Non-empty forgery for Umbreon)**.** *There is an authenticity adversary against* Umbreon *using* $q_e = 3$*,* $q_v = 1$*,* $\mu = 3$*, and* $\sigma_{m,e} = 1$ *that succeeds with probability one.*

Proof. Obtain u_0, u_1 and δ as above. Query (N, A_0, M) and receive (C, τ). Output

$$(N, A_1, C^\star, \tau), \qquad C^\star = C \oplus \delta.$$

During verification under A_1, the recovered plaintext is $M^\star = M \oplus \delta$. Equation (8) gives

$$M^\star \oplus T_A^{(0)}(A_1) = M \oplus T_A^{(0)}(A_0) = Z.$$

Hence the final forkcipher input, tag branch, and paired branch output are exactly the same as in the queried encryption. Since $C^\star = R \oplus M^\star$, the final consistency check succeeds. □

The claimed theorem would upper bound the same adversary, in the ideal-forkcipher model, by

$$\frac{3(1+3)(3-1)}{2^{n-1}} + \frac{1}{2^{n-4}} = \frac{64}{2^n},$$

which equals 2^{-122} for $n = 128$, versus actual success probability one.

3.2.2 Published Espeon: two message blocks

The published Espeon pseudocode omits T_A from the final call only when the message has no ordinary block before the final block. With two full message blocks $M_1 \| M_2$, however, the first ordinary block explicitly absorbs the AD state. This is enough to turn the same AD-mask difference into a probability-one forgery entirely within the published specification.

For AD A_0, the first message ciphertext is

$$C_1 = F_K^{T_1,0}\big(M_1 \oplus T_A^{(0)}(A_0)\big),$$

where T_1 is determined by the nonce and the first-message domain encoding. The checksum before the final call is $\Delta = M_1$. The final tweak depends on the already produced ciphertext prefix (for two blocks, on the fixed C_0 and on C_1), and the final primitive input is

$$Z = M_2 \oplus M_1.$$

Let the final forkcipher call return (τ, R), so the transmitted final ciphertext block is $C_2 = R \oplus M_2$.

Theorem 2 (Non-empty forgery for published Espeon)**.** *There is an authenticity adversary against the published* Espeon *mode using* $q_e = 3$*,* $q_v = 1$*,* $\mu = 3$*, and* $\sigma_{m,e} = 2$ *that succeeds with probability one.*

Proof. Use the two AD-only queries on $A_0 = X_0 \| Y$ and $A_1 = X_1 \| Y$ to obtain δ as in (8). Query the two-block message $M_1 \| M_2$ under A_0, obtaining $(C_1 \| C_2, \tau)$. Submit under A_1 the forged ciphertext

$$C_1^\star = C_1, \qquad C_2^\star = C_2 \oplus \delta,$$

with the same tag τ.

In verification under A_1, inversion of the unchanged C_1 yields the same forkcipher input as in the source query, hence the recovered first plaintext block is

$$M_1^\star = M_1 \oplus \delta,$$

because $T_A^{(0)}(A_0) \oplus T_A^{(0)}(A_1) = \delta$. Consequently $C_1^\star = C_1$, so the final tweak is unchanged. The checksum is now $\Delta^\star = M_1 \oplus \delta$. Inverting the unchanged tag therefore recovers the same final input $Z = M_2 \oplus M_1$, and the final plaintext block becomes

$$M_2^\star = Z \oplus \Delta^\star = M_2 \oplus \delta.$$

The paired branch output R is unchanged, and

$$C_2^\star = C_2 \oplus \delta = R \oplus (M_2 \oplus \delta) = R \oplus M_2^\star,$$

so the final consistency check accepts. Since P_N is a permutation and $X_0 \neq X_1$, $\delta \neq 0$; thus the forged ciphertext is distinct from the queried source ciphertext even before accounting for the changed AD. □

For $\sigma_{m,e} = 2$, the claimed Espeon bound gives

$$\mathsf{Adv}^{\mathrm{auth}} \leq \frac{28}{2^n} + \frac{4}{2^{t-3}},$$

which is negligible for the intended parameter ranges, while the actual success probability is one. In particular, for $t = n$ this is $60/2^n$.

3.2.3 Long messages and partial final blocks

The previous forgeries also work when more ordinary full message blocks precede a complete final block. The only ciphertext edit remains an XOR on the final block. For a partial final block, the same edit must preserve its canonical padding; the affine AD relation lets us select a mask that does so.

Proposition 4 (Non-empty forgeries for every final-block length)**.** *Fix an integer $1 \leq r \leq n$ and write $s = n - r$. Suppose the d-bit AD counter can represent $s + 5$ without wraparound. Restrict all inputs to the legal domain of Section 2; in particular, require $m + 1 < 2^{d+1}$ for* Umbreon*. The published* Umbreon *mode admits a probability-one forgery of any message with $m \geq 0$ ordinary full blocks followed by an r-bit final fragment. The published* Espeon *mode admits the same for $m \geq 1$. Both use*

$$q_e = \mu = s + 3, \qquad q_v = 1, \qquad \sigma_{m,e} = m + 1,$$

and AD of $(s + 2)n$ bits in each query. Only the last ciphertext fragment may change; the tag is preserved.

Proof. Use $s + 1$ ordinary full AD positions, followed by a common final full AD block Y. Let $A_{\mathbf{0}}$ be the baseline AD and A_i change only ordinary position i, for $1 \leq i \leq s + 1$. Make the $s + 2$ AD-only queries on these values, receiving tags $u_0, u_1, \ldots, u_{s+1}$, and set $d_i = u_0 \oplus u_i$. Position-specific XOR accumulation implies, for any $\boldsymbol{v} \in \{0, 1\}^{s+1}$,

$$T_A^{(0)}(A_{\boldsymbol{v}}) \oplus T_A^{(0)}(A_{\mathbf{0}}) = \bigoplus_{i:v_i=1} d_i. \tag{9}$$

Indeed, the last-block term is the same within each message purpose and cancels; the remaining per-position differences are identical in the AD-only and non-empty-message domains. This identity does not assume independence among the d_i.

Write $\mathrm{suf}_s(x)$ for the final s bits of x. The $s + 1$ vectors $\mathrm{suf}_s(d_i)$ in $\mathbb{F}_2^s$ are linearly dependent. Gaussian elimination therefore finds a nonzero $\boldsymbol{v}$ such that

$$\delta = \bigoplus_{i:v_i=1} d_i, \qquad \mathrm{suf}_s(\delta) = 0^s.$$

For $s = 0$, take any one position. Query one non-empty message encryption under $A_{\mathbf{0}}$ to obtain $(C_1 \| \cdots \| C_m \| C_*, \tau)$. Submit $(C_1 \| \cdots \| C_m \| C_*^\star, \tau)$ under $A_{\boldsymbol{v}}$, where

$$C_*^\star = C_* \oplus \mathrm{pre}_r(\delta).$$

For Umbreon, retaining each ordinary ciphertext makes the first recovered message block differ by δ and leaves every subsequent message input and internal child state unchanged. Its checksum differs by δ: the first incoming AD state is XORed once, while all later states coincide. This also holds for $m = 0$, when the incoming AD state is XORed immediately before the final call. For Espeon, retaining the first ordinary ciphertext makes its recovered message block differ by δ; all later ordinary message blocks and the ciphertext-history-dependent final tweak stay unchanged, while the checksum differs by δ. Consequently, inversion of the unchanged tag in either mode recovers the original final primitive input and a final padded plaintext that differs by δ.

For $r < n$, the canonical padding is $M_* \| 1 \| 0^{s-1}$. Since $\mathrm{suf}_s(\delta) = 0^s$, its required suffix survives. The paired branch output is unchanged, so the r transmitted ciphertext bits differ by exactly $\mathrm{pre}_r(\delta)$ and satisfy the verification check. For $r = n$ there is no padding suffix to check. Finally $A_{\boldsymbol{v}} \neq A_{\mathbf{0}}$, making the verification tuple fresh even if the full δ happens to vanish. The ordinary AD counters run from 5 to $s + 5$, giving the stated no-wraparound condition. □

The fixed last AD block Y can also be replaced by any common nonempty partial final AD fragment, with the same canonical padding in every query. Only differences at the ordinary full-block positions enter (9). Thus the AD length in the theorem may alternatively be $(s + 1)n + \ell$ bits for $1 \leq \ell < n$.

Theorem 3 (Adaptive final length saves one AD query)**.** *Let $2 \leq r < n$, fix any $1 \leq h < r$, and write $s = n - r$. Suppose the d-bit AD counter represents $s + 4$ without wraparound. Restrict all inputs to the legal domain of Section 2; in particular, require $m + 1 < 2^{d+1}$ for Umbreon. For any $m \geq 0$ in published Umbreon and any $m \geq 1$ in published Espeon, a source encryption of m ordinary full message blocks followed by an r-bit fragment yields a fresh forgery with the same tag and a final ciphertext fragment of either r or $r - h$ bits. The choice of length is made after observing the AD-only responses. The adversary also chooses the last h bits of the source plaintext after observing those responses and before making the non-empty-message encryption query. The remaining source plaintext bits may be fixed in advance. The attack succeeds with certainty and uses*

$$q_e = \mu = s + 2, \qquad q_v = 1, \qquad \sigma_{m,e} = m + 1.$$

All AD queries can have $sn + \ell$ bits for any fixed $1 \leq \ell \leq n$.

Proof. Use s independently varying ordinary full AD positions and one common nonempty last AD fragment of ℓ bits. Query the baseline and each of the s singleton variations in AD-only mode, obtaining differences $d_i = u_0 \oplus u_i$. Their XOR over any selected nonempty subset is again the exact difference δ between the corresponding message-purpose AD hashes. Consider the s suffix vectors $\mathrm{suf}_s(d_i) \in \mathbb{F}_2^s$.

If these vectors are linearly dependent, select a nonempty subset with $\mathrm{suf}_s(\delta) = 0^s$. If they are independent, they form a basis of $\mathbb{F}_2^s$; select the unique nonempty subset with $\mathrm{suf}_s(\delta) = 10^{s-1}$. Either way the destination AD differs from the baseline. After learning δ, choose the source final fragment M_* so that its last h bits equal $10^{h-1} \oplus \delta[r - h \ldots r - 1]$, where bits are numbered from zero at the left. Query its encryption under the baseline AD. Let $W = \mathrm{pad}_r(M_*) \oplus \delta$.

When $\mathrm{suf}_s(\delta) = 0^s$, W has the same canonical r-bit-fragment padding as the source; take $r' = r$. In the other case, the source padding 10^{s-1} cancels in the last s bits of W, while our choice of M_* makes $W[r - h \ldots r - 1] = 10^{h-1}$; hence W has the canonical

$(r-h)$-bit-fragment padding, and we take $r' = r - h \geq 1$. Keep every ordinary ciphertext block and the tag, and set the last ciphertext fragment to

$$C_*^\star = \mathrm{pre}_{r'}(C_*) \oplus \mathrm{pre}_{r'}(\delta).$$

As in Proposition 4, decryption under the changed AD alters the first ordinary plaintext block (if any) and the final padded plaintext by δ while preserving the internal forkcipher inputs and ordinary ciphertext. Both final fragments are partial, so the published final tweak uses the same padding flag even if $r' \neq r$; the ciphertext-history part of the Espeon tweak also stays fixed. The new final ciphertext is the transmitted prefix of the unchanged paired branch XORed with W, and the padding check succeeds. The largest ordinary AD counter is $s + 4$. □

For $n = 128$ and a 15-byte final fragment, $r = 120$ and $s = 8$. The fixed-length attack of Proposition 4 needs ten AD-only queries on 160-byte AD and one message encryption: $q_e = \mu = 11$. For a 15-byte Umbreon message, the published ideal-forkcipher authenticity bound would be only $736/2^{128}$, versus actual success probability one. For a 16-byte full block followed by a 15-byte final fragment in Espeon, the claimed bound at $t = n = 128$ would be $268/2^{128}$. This extends the same cross-purpose bridge to a partial final block.

Theorem 3 reduces this instance to eight ordinary AD positions plus a common one-byte final AD fragment, namely 129-byte AD. Nine AD-only queries and one message encryption give $q_e = \mu = 10$, $q_v = 1$. Taking $h = 8$, the forged final fragment has 15 or 14 bytes, selected from the observed AD-only tags. The corresponding published bounds would be $610/2^{128}$ for Umbreon and $228/2^{128}$ for Espeon when $t = n = 128$, whereas the forgery still succeeds with probability one. For bit-string interfaces, taking $r = n - 1$ and $h = 1$ already gives a three-encryption, one-verification forgery with a final fragment of either $n - 1$ or $n - 2$ bits.

Public-source difference. The public microcontroller source injects T_A into the final block even when there is no preceding ordinary message block. Thus the one-block source path differs from the published pseudocode. The specification-level attacks in Section 3.2 do not depend on this implementation difference. Additional source observations are deferred to Appendix A.

3.3 One-Block AD Omission in Published Espeon

The published pseudocode has an additional issue that is independent of the repeated-nonce attacks above. Let $0 < |M| \leq n$. In the notation of the pseudocode, the number of ordinary full message blocks before the final block is $m = 0$. The Espeon-specific use of T_A occurs only in the loop handling the first ordinary message block. Because the loop is skipped, T_A never enters the final primitive input. The final tweak for $m = 0$ depends on N, the constant counter value, and the padding class, but not on AD. Consequently the final input, tag, and ciphertext are all independent of A.

Theorem 4 (Nonce-respecting AD substitution in published Espeon)**.** *For the published Espeon pseudocode and any $0 < |M| \leq n$, one encryption query and one verification query suffice to forge with probability one, without nonce reuse.*

Proof. Choose $A \neq A'$ and query (N, A, M) to obtain (C, τ). As summarized in Fig. 4, the one-block encryption path does not use T_A; the identical pair (C, τ) is valid under (N, A', M). Therefore (N, A', C, τ) is a fresh valid verification query. Only one encryption query uses N, so $\mu = 1$. □

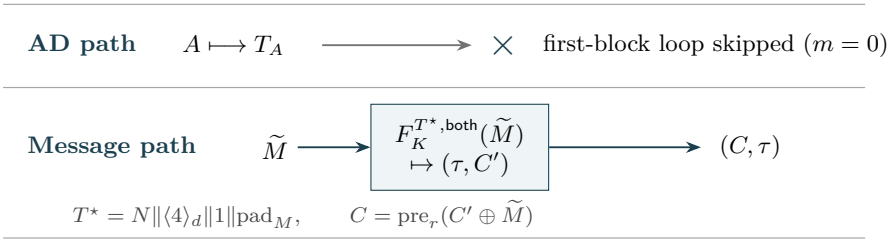

Figure 4: The one-block path of published Espeon (Fig. 9 of the original full version [BPA+23c]). Here $r = |M|$, $\widetilde{M} = \mathrm{pad}_r(M)$, and pad_M is 1 for a full block and 0 for a partial block. AD reaches only the first ordinary message-block loop. That loop is skipped for $0 < |M| \leq n$, leaving neither the final input nor its tweak dependent on A. The public C implementation has a different message path.

This contradicts even the nonce-respecting part of the published theorem. In the ideal-forkcipher model with $q_e = q_v = 1$, $\mu = 1$, and $\sigma_{m,e} = 1$, the published bound is at most $2^{-(t-3)} + 2^{-(n-4)}$, whereas the attack succeeds with probability one.

The public microcontroller source differs from the published pseudocode at this point: immediately before the final Espeon forkcipher call it XORs the AD state into the final message block. Thus Theorem 4 is specifically a break of the published specification. This discrepancy does not affect Theorem 2, which already gives a probability-one non-empty-message forgery for the published two-block path.

4 Proof Diagnosis

Mao et al. [MWH+26] independently identified the proof failures discussed here. We include the diagnosis to explain the shared attacks and the joint-transcript perspective underlying our partial-block extensions; see Section 1.2 for the relationship between the two works.

The central proof issue is a mismatch between marginal and joint distributions. In the AD-only case, the Eevee proof observes that the tag is a sum of outputs of domain-separated random permutations and bounds the statistical distance between an individual permutation output and a random-function output. This reasoning can justify that a single new tag is close to uniform under appropriate conditions. It does not imply that a sequence of tags generated from related AD inputs is jointly independent.

Figure 5 illustrates this probabilistic distinction using two abstract bits with identical marginals but different joint laws.

The four variables in Lemma 2 make the distinction explicit. Each τ_{ij} can be individually uniform. Nevertheless,

$$H(\tau_{11} \mid \tau_{00}, \tau_{01}, \tau_{10}) = 0,$$

because the fourth tag is determined by the first three, whereas in the OAE ideal world the conditional entropy is n. Equivalently, the proof controls marginals while the adversary distinguishes the joint transcript.

The issue is not the marginal PRP/PRF statistical-distance calculation underlying Eq. (7) in Appendix C of the Eevee full version [BPA+23c] in isolation, but its use to bound the joint adaptive transcript. That calculation controls the distribution of an individual AD tag, whereas the OAE adversary observes several correlated tags on related AD inputs. The deterministic rectangle therefore survives even if every one-dimensional marginal is uniform. Consequently, that equation, when used as a per-query substitute for joint transcript independence in the subsequent union/max argument, does not upper-bound the adversary constructed here.

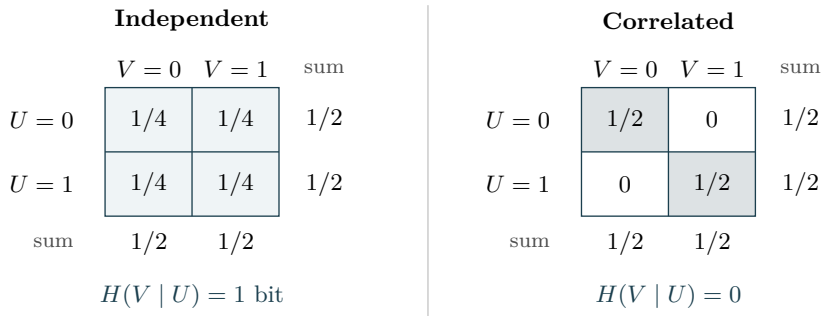

Figure 5: A two-bit illustration of marginal uniformity versus joint independence. Both tables have the same uniform row and column marginals. On the right, however, observing U determines V. An individual-output uniformity argument cannot distinguish these two joint distributions.

There is a second, closely related inference in the original proof. Appendix C, Observation 1 argues that, when the current encryption query has a non-empty message, an earlier AD-only query does not help because the `noM` bit domain-separates the final AD call [BPA+23c]. Our bridge attack gives an explicit counterexample to that conclusion. For $A_0 = X_0 \| Y$ and $A_1 = X_1 \| Y$ with the same final block Y, the AD-only difference

$$u_0 \oplus u_1 = P_N(X_0) \oplus P_N(X_1)$$

cancels the domain-separated final-block contribution $Q_{N,1}(Y)$. The same ordinary-block difference appears in the non-empty accumulators because $Q_{N,0}(Y)$ also cancels between A_0 and A_1. Hence the final-AD domain separation changes the last-block term but does not prevent an AD-only transcript from revealing exactly the mask difference needed by the non-empty forgery in Section 3.2.

The integrity proof additionally assumes that a new (N, A, C) verification query contains a fresh checksum component. The nonce-respecting Espeon counterexample in Section 3.3 shows that this premise is false for the one-block pseudocode path, where the final checksum does not contain AD at all.

5 Implementation Validation

The reproducibility artifact models each forkcipher tweak as a pair of random permutations on an 8-bit state. It checks the rectangle, AD-only forgery, affine synthesis, both non-empty-message forgeries, and the independent one-block `Espeon` substitution. These are executable checks of exact algebra; the security claims follow from the proofs above under their stated parameter conditions. Figure 6 shows how the two official C backend experiments distinguish the published Fig. 9 mode from the public source implementation's different AD behavior.

Published Umbreon attack with the official primitive. We implemented the two-full-AD-block and one-full-message-block path of the published Fig. 9, using the unmodified official ForkSkinny-128-256 C primitive at commit `a8bb073` and the upstream Umbreon wrapper's exact key/tweak schedule and branch wiring. Each of 32 fixed trials made two AD-only encryptions and one message encryption, constructed the target of Theorem 1, and accepted its fresh verification under the second AD. The recovered plaintext equaled $M \oplus \delta$ in all 32 trials. An independent wiring control compared one-block empty-AD encryptions and decryptions to the unmodified upstream Umbreon mode in all 32 trials.

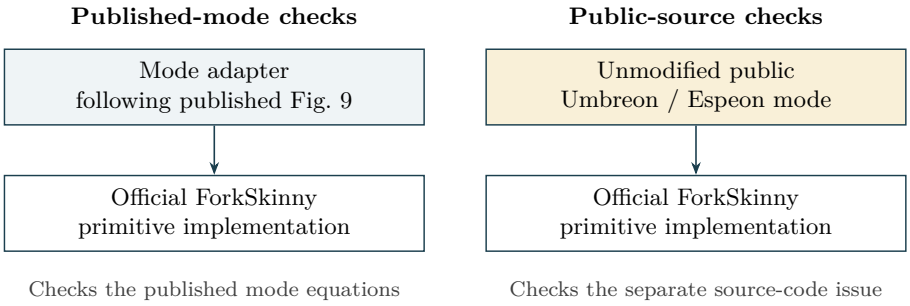

Figure 6: The two implementation-validation scopes. Both use official ForkSkinny code, but their mode layers differ. The published-mode adapters and the unmodified public modes therefore support different claims.

For nonempty AD, this harness follows the published Fig. 9 counter/tweak rules, whereas the upstream C mode has the distinct conditional behavior recorded in Appendix A. Thus these trials validate the paper-level attack with the official primitive, not acceptance by the unmodified upstream mode for these 32-byte AD inputs. The full transcript, source, and reproduction command are in the artifact.

Unmodified public source modes. The separate source-level two-query attack of Appendix A was run against the unmodified Umbreon-ForkSkinny-128-256 and Espeon-ForkSkinny-128-256 C modes with the official C backend at the same pinned commit. Both verifiers accepted (`verify_rc=0`); ciphertexts, tags, and recovered plaintexts are in our artifact. This source result relies on a C conditional-expression issue and does not replace the three-query published-mode demonstration.

Availability and scope of supporting material. Supporting code and saved transcripts are available from the authors on request. The source package's `artifact/` directory contains the basic ideal-model program, the complete-block official-primitive harness, the source-mode harness, and their saved outputs and reproduction commands. The upstream implementation is identified by the commit above and is not redistributed in the package. The archived executable checks reported here cover the basic attacks. The partial-final-block and adaptive-length guarantees are established by Proposition 4 and Theorem 3; no experimental counts for those extensions are claimed in this version.

6 Countermeasure and Conclusion

Possible countermeasure. One direction is to stop returning the raw XOR AD accumulator as the tag. A nonce- and purpose-separated final primitive evaluation could map it to

$$U_{N,b}(A) = F_K^{T_{\mathrm{fin}}(N,b),0}(T_A^{(b)}(A)), \qquad b \in \{0,1\},$$

where the reserved finalization tweaks for AD-only ($b = 1$) and message processing ($b = 0$) are disjoint from the ordinary AD and message tweaks. Publishing $U_{N,1}(A)$ instead of $T_A^{(1)}(A)$ would prevent the two observed AD-only tags from directly disclosing the raw difference $T_A^{(0)}(A_0) \oplus T_A^{(0)}(A_1)$ used in our attack.

This observation addresses the displayed bridge, not every security case. *We do not prove security of this modification.* A complete construction must specify all message lengths, padding and tweak domains, and ensure that each Espeon path binds AD. A full nonce-misuse OAE proof would need to analyze the joint AD-only/message transcript and

inverse verification. The public C conditional expression is a separate issue that also needs correction. No performance or backward-compatibility guarantee is claimed for the sketch.

Mao et al. [MWH+26] also discuss requirements for a repair. Their AD-free online-prefix distinguisher and Jolteon cross-length forgery expose additional issues that an AD-finalization change alone does not address. A revised Eevee-like family would therefore need to reconcile its message processing with the intended online ideal object and bind message length as well as AD. Neither their repair discussion nor the sketch here supplies a fully specified construction with a complete security proof. Designing and proving such a construction while retaining the IoT and MPC goals remains a direction for future work.

Conclusion. The published Umbreon and Espeon modes admit probability-one non-empty-message forgeries under their claimed nonce-misuse security model. The basic AD-correlation attacks and the associated proof failures were obtained independently and concurrently in this work and [MWH+26]. Building on the classical XOR relation [BGR95], our affine treatment extends the forgery to partial final blocks while preserving valid padding. The fixed-length construction uses $n-r+3$ encryption queries; adaptive selection between lengths r and $r-h$ saves one query. Both accommodate preceding complete blocks under the stated conditions. Official-primitive and public-source experiments have different scopes, as specified in Section 5. A complete, proved repair remains open in both works.

Acknowledgments

We thank Shuping Mao, Peng Wang, Jiadong Han, Tingting Guo, Yan Jia, and Ying Chen, the authors of [MWH+26], for carefully reading our manuscript and for their thoughtful and constructive correspondence. Their feedback helped us clarify the overlap and complementary contributions of our independent and concurrent studies.

References

[ABPV21] Elena Andreeva, Amit Singh Bhati, Bart Preneel, and Damian Vizár. 1, 2, 3, fork: Counter mode variants based on a generalized forkcipher. *IACR Transactions on Symmetric Cryptology*, 2021(3):1–35, 2021. doi:10.46586/tosc.v2021.i3.1-35.

[ABV21] Elena Andreeva, Amit Singh Bhati, and Damian Vizár. Nonce-misuse security of the SAEF authenticated encryption mode. In *Selected Areas in Cryptography – SAC 2020*, volume 12804 of *LNCS*, pages 512–534. Springer, 2021. doi:10.1007/978-3-030-81652-0_20.

[ALP+19] Elena Andreeva, Virginie Lallemand, Antoon Purnal, Reza Reyhanitabar, Arnab Roy, and Damian Vizár. Forkcipher: A new primitive for authenticated encryption of very short messages. In *ASIACRYPT 2019*, pages 153–182. Springer, 2019.

[BAV24] Amit Singh Bhati, Elena Andreeva, and Damian Vizár. OAE-RUP: A strong online AEAD security notion and its application to SAEF. In *Security and Cryptography for Networks – SCN 2024*, volume 14974 of *LNCS*, pages 117–139. Springer, 2024. doi:10.1007/978-3-031-71073-5_6.

[BGR95] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In *Advances in Cryptology – CRYPTO 1995*, volume 963 of *LNCS*, pages 15–28. Springer, 1995. Full version, October 1995, Section 2. URL: https://web.cs.ucdavis.edu/~rogaway/papers/xormacs.pdf.

[BPA+23a] Amit Singh Bhati, Erik Pohle, Aysajan Abidin, Elena Andreeva, and Bart Preneel. Eevee: Software and MPC artifact. GitHub repository, KULeuven-COSIC/Eevee,

2023. Source inspected at commit a8bb073 (main tip observed 26 September 2026). URL: https://github.com/KULeuven-COSIC/Eevee/tree/a8bb073.

[BPA+23b] Amit Singh Bhati, Erik Pohle, Aysajan Abidin, Elena Andreeva, and Bart Preneel. Let's go Eevee! a friendly and suitable family of AEAD modes for IoT-to-cloud secure computation. In *ACM CCS 2023*, pages 2546–2560. ACM, 2023. doi:10.1145/3576915.3623091.

[BPA+23c] Amit Singh Bhati, Erik Pohle, Aysajan Abidin, Elena Andreeva, and Bart Preneel. Let's go Eevee! a friendly and suitable family of AEAD modes for IoT-to-cloud secure computation. Cryptology ePrint Archive, Report 2023/1361, 2023. Full version inspected 26 September 2026. URL: https://eprint.iacr.org/2023/1361.

[BR02] John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 384–397. Springer, 2002. Full version, 15 February 2002, Figure 1. URL: https://www.cs.ucdavis.edu/~rogaway/ocb/pmac.pdf.

[HRRV15] Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online authenticated-encryption and its nonce-reuse misuse-resistance. In *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 493–517. Springer, 2015.

[KR14] Ted Krovetz and Phillip Rogaway. The OCB authenticated-encryption algorithm. Technical Report RFC 7253, RFC Editor, 2014. URL: https://www.rfc-editor.org/rfc/rfc7253, doi:10.17487/RFC7253.

[MWH+26] Shuping Mao, Peng Wang, Jiadong Han, Tingting Guo, Yan Jia, and Ying Chen. Structural cryptanalysis of the Eevee AEAD family. Cryptology ePrint Archive, Paper 2026/2496, 2026. Version dated 15 October 2026. URL: https://eprint.iacr.org/2026/2496.

[RKI25] Mostafizar Rahman, Samir Kundu, and Takanori Isobe. On the key-commitment properties of forkcipher-based AEADs. *IACR Communications in Cryptology*, 1(4), 2025. doi:10.62056/ayfhp2fgx.

A Source Audit of the Public Artifact

This appendix records a source-level issue in the public microcontroller code accompanying the Eevee paper [BPA+23a]. It is deliberately separated from the core cryptanalysis: none of Sections 3.1, 3.2, and 3.3 depends on this implementation observation. We inspected public repository commit `a8bb073` (commit title `Added conference version`, the `main` tip observed on 26 September 2026). In the commit-pinned file `software-microcontroller/eevee-forkskinny/umbreon_core.c`, source line 43 assigns the last tweak byte by the C expression

$$\texttt{no_m + (complete_block) ? 0 : 2.}$$

Because addition binds more tightly than the conditional operator, C parses this as

$$(\texttt{no_m} + \texttt{complete_block})\ ?\ 0 : 2.$$

Hence for a complete final AD block the assigned value is zero for both $\texttt{no_m} = 0$ and $\texttt{no_m} = 1$. The commit-pinned file `software-microcontroller/eevee-forkskinny/espeon.c` contains the analogous 128-384-path expression at source line 81,

$$\texttt{(no_m << 2) + (complete_block) ? 0 : 0xc,}$$

and contains `block_xor128(m_in, t_a)` at source line 188, confirming that the public one-final-block path injects the AD state before the final forkcipher call. The same file contains the analogous final-AD conditional in its 128-256 path. Thus, when the final AD block is complete, the last-AD tweak does not distinguish AD-only processing from AD

processing followed by a message. The same precedence pattern also appears in Jolteon, although Jolteon does not claim nonce-misuse security.

For nonempty AD A ending in a complete block, let $u = \mathsf{Enc}_K(N, A, \varepsilon)$ in the artifact. The collapsed domain separation implies that the AD mask used by a non-empty query under the same (N, A) is exactly u, not merely related by an unknown final-block term.

The final-AD conditional-operator issue is also recorded independently in Section 6.1 of Mao et al. [MWH+26]. The construction below states its two-query consequence explicitly.

Source-level reproducibility. The commit-pinned host file `software-microcontroller/Makefile` selects `EEVEE_FORKSKINNY_BACKEND_C` at source line 18 for the host build. The conditional expressions above are in the mode-level Umbreon/Espeon source, so their C parsing is independent of the primitive backend. Our supplementary material contains the exact source locations, a minimal C precedence check, and a harness built against the authors' unmodified repository at commit `a8bb073`. With its official ForkSkinny C backend, the source-level two-query constructions were accepted by both Umbreon-ForkSkinny-128-256 and Espeon-ForkSkinny-128-256 verification (`verify_rc=0` in each case); the complete transcripts are included in the artifact. The result below is a source-level corollary of the displayed C expressions and mode equations; the ideal-forkcipher core results do not depend on this source experiment.

Proposition 5 (Source-level two-query consequence)**.** *For the public Umbreon source, and for the public Espeon source that injects T_A into the final message block, the displayed final-AD tweak expressions imply a two-encryption-query, one-verification-query non-empty-message forgery under a repeated nonce whenever the chosen nonempty AD ends in a complete block.*

Proof. First query (N, A, ε) to learn $u = T_A(A)$. Next query a full one-block message $M \in \{0,1\}^n$ with empty AD, (N, ε, M), obtaining (C, τ). In this source query the final input is M. For the target query with AD A, choose $M^\star = M \oplus u$. Then $M^\star \oplus T_A(A) = M$, so the final primitive input and tag remain unchanged. Replacing the last ciphertext block by $C^\star = C \oplus u$ preserves the paired-branch consistency relation. Thus $(N, A, C^\star, \tau)$ verifies. □

This issue is separate from the paper-level repeated-nonce correlation: even if the C precedence error is fixed, Section 3.1 still applies to the specified XOR-separable AD construction.

The reproduction artifact contains the random-permutation model, both official-C harnesses, accepting transcripts, and the source-level conditional-expression check. The published-mode harness applies the Fig. 9 AD rule, while the source-mode harness invokes the unchanged public C modes. The ideal-forkcipher attacks are independent of these C results.