ANIMAGUS: A Provably Secure Tweakable Wide-Block Mode

Gülnihal Öztürk[1], Onur Koçak[2], and Oğuz Yayla[1]

[1]Cryptography, IAM METU, Ankara, Türkiye ,
gulnihal.ozturk_01@metu.edu.tr, oguz@metu.edu.tr
[2]TÜBİTAK, Ankara, Türkiye , onur.kocak@tubitak.gov.tr

February 2026

Abstract

Motivated by the accordion mode approach, we propose a length preserving tweakable wide block mode based on the hash-encrypt-hash paradigm. The construction uses AES and HMAC-SHA3-256 with five purpose separated working keys derived from a single master key, replacing algebraic universal hashing with a cryptographic keyed hash. An encrypted internal seed couples two separately keyed head branches and determines the counter mode keystream. The head layers permit parallel evaluation, while inter-branch dependence is retained even when the tail is empty. The mode supports every bit length within its specified range, starting from $2n$ bits without padding, where n is the block size. Variable length external information is incorporated into a fixed length core tweak through application specific preprocessing. We establish birthday bound tweakable variable-input-length strong pseudorandom permutation security under the stated block cipher, keyed hash, and key derivation assumptions. The analysis covers adaptive encryption and decryption queries, varying message lengths, repeated tweaks, and reused internal values. We also describe derived interfaces for authenticated encryption with associated data, tweakable encryption, and deterministic authenticated encryption, distinguishing their additional security requirements from the core guarantee.

Keywords: Wide block encryption, tweakable encryption, VIL-SPRP, HMAC, birthday bound security.

1 Introduction

Block cipher modes of operation extend block ciphers to variable length inputs and support confidentiality, authentication, storage encryption, and key wrapping. Several modes are standardized in the NIST SP 800-38 series [16–22]. Their security depends on requirements concerning initialization vectors, nonces, message lengths, and per-key data limits. NIST IR 8459 reviews these limitations and recommends further development of block cipher modes [28].

One such direction is the cryptographic accordion, a tweakable variable-input-length strong pseudorandom permutation (VIL-SPRP). The initial proposal [7] and workshop discussions [32] led to the April 2025 *Requirements for Cryptographic Accordions* [8], which distinguish the underlying block cipher, the accordion core, and application specific derived functions. In June 2025, NIST proposed developing HCTR2 variants for birthday bound and beyond birthday bound (BBB) usage with AES, and for a 256-bit block cipher [30]. This framework motivates the study of related wide block constructions.

Wide block encryption has developed mainly through encrypt-mix-encrypt and hash-encrypt-hash approaches. In the former, CMC [25] uses two CBC based passes separated by a mixing step, whereas EME [24] employs ECB based layers that permit parallel processing. Both were introduced by Halevi and Rogaway with tweakable strong pseudorandom permutation (SPRP) guarantees. AEZ [26] builds authenticated encryption from an arbitrary-input-length tweakable cipher, using Feistel components based on reduced round AES. ZCZ [3] follows a related design direction with tweakable block ciphers and achieves BBB security.

The hash-encrypt-hash approach is related to the Luby–Rackoff construction [27] and the work of Naor and Reingold [29] on pseudorandom permutations (PRP) with larger input domains. HCTR, proposed by Wang, Feng, and Wu [33], combines universal hashing and counter mode processing to obtain a tweakable variable-input-length cipher. Related constructions include PEP [5], HCH [4], and TET [23], which vary the hashing and encryption components; TET specifically uses invertible universal hashing.

Shrimpton and Terashima's Protected-IV framework [31] provides a modular construction of variable-input-length tweakable ciphers, including the BBB-secure TCT_2 instantiation. Tweakable HCTR [14], proposed by Dutta and Nandi, uses tweakable block ciphers in an HCTR-style structure, with security depending on the frequency of tweak repetition.

Adiantum [10] instead uses a hash-block cipher-stream cipher-hash (HBSH)

construction, combining a stream cipher for bulk processing with a single block cipher invocation. HCTR2, introduced by Crowley, Huckleberry, and Biggers [11], addresses a specification issue in HCTR and an error in its earlier security analysis. It also improves the security bound, supports variable length tweaks, and uses a single master key.

Dobraunig et al. [13] instantiated the docked double decker framework as ddd-AES, ddd-AES$^+$, and bbb-ddd-AES, combining AES and POLYVAL hashing with parallel processing. The first provides birthday bound security, ddd-AES$^+$ adds variable length tweaks, and bbb-ddd-AES achieves BBB security under a restriction on repeated use of the same tweak.

The AES based ACCOR constructions [15] offer another hash-encrypt-hash approach: ACCOR-S emphasizes efficiency with birthday bound security, while ACCOR-L targets BBB security through a wider head transformation and modified counter mode keying, invoking related-key security in its analysis. HCTR+ [12] instead uses a tweakable block cipher and provides BBB security without restricting tweak repetition.

Chen et al. [9] subsequently proposed two approaches compatible with HCTR2. CHCTR2 cascades two independently keyed HCTR2 instances and achieves $2n/3$-bit multi-user security without an additional tweak repetition restriction, where n is the block size. HCTR2-TwKD uses tweak based key derivation and achieves the same security level when at most $2^{n/3}$ block cipher calls are processed under any single tweak.

Beyond the target security level, the choice of primitives provides another direction for investigating wide block encryption. Our motivation is to examine a construction based on a conventional block cipher and a cryptographic keyed hash, without a separate algebraic universal hashing component. In particular, we seek to combine AES and HMAC-SHA3-256 in a length preserving variable-input-length mode with a provable birthday bound security guarantee. A further design objective is to permit parallel evaluation within each head layer while preserving a reversible dependence between the two head blocks, including when the message has no tail. Establishing security for this arrangement requires accounting for repeated internal values under adaptive encryption and decryption queries, rather than assuming that every primitive invocation receives a fresh input.

To investigate this design direction, we propose **Animagus**, an accordion inspired tweakable wide block mode. Two independently keyed head branches are coupled through an encrypted internal seed, with counter mode processing the tail. The concrete instantiation uses AES and HMAC-SHA3-256 with five purpose separated working keys derived from one master key.

Animagus preserves message length and supports one bit granularity

throughout its specified message range, starting from $2n$ bits, without padding. Working key setup is independent of the tweak. The block cipher calls within each head layer and across counter mode blocks permit parallel evaluation once their inputs are available. The core accepts a fixed length tweak, while variable length external information is handled by application specific preprocessing. We establish birthday bound VIL-SPRP security under the stated primitive assumptions for adaptive encryption and decryption queries, varying lengths, and repeated tweaks, without requiring tweak uniqueness. Derived interfaces for authenticated encryption with associated data (AEAD), tweakable encryption, and deterministic authenticated encryption (DAE) are also described, with their additional security requirements distinguished from the core guarantee. The design is presented as accordion inspired, rather than as satisfying the complete NIST requirements.

Section 2 presents the construction, its parameters, design rationale, and application interfaces. Section 3 provides the security analysis and proof for the core.

2 Our Design

In this paper, we propose Animagus, a new design inspired by the accordion mode approach. The construction is a tweakable wide block mode of operation for variable length inputs and follows the hash-encrypt-hash paradigm. Animagus uses a block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ with an n-bit block size and a k-bit key size, where $\mathcal{K} = \{0,1\}^k$, together with a keyed hash function H that produces a $2n$-bit output. The concrete instantiation studied in this paper employs AES, with $n = 128$ and $k \in \{128, 192, 256\}$, and HMAC-SHA3-256 for H. Thus, the hash output comprises two n-bit parts regardless of the selected AES key length.

The supported message space is

$$\mathcal{M} = \bigcup_{\ell=2n}^{\ell_{\max}} \{0,1\}^{\ell},$$

where ℓ is the message length in bits and $\ell_{\max}$ denotes the maximum supported length. The value $a = \lceil \ell_{\max}/n \rceil$ therefore gives the maximum number of n-bit blocks in a supported message, counting a possible partial final block as one block. The minimum plaintext length is $2n$ bits, and every integer bit length within the supported interval is accepted. Thus, the message length granularity is one bit ($g = 1$), and neither byte alignment nor block alignment is required. For the AES instantiation, $\ell_{\max} = 2^{55}$ bits remains the

maximum format length. This parameter alone does not determine a safe total volume of data to be processed under one key. Messages outside the supported space are rejected.

The master key space of Animagus is that of the underlying block cipher. The master key K_M, however, is not used directly for the different cryptographic operations of the construction. Instead, five purpose separated k-bit working keys, K_{head1}, K_{head2}, K_{seed}, K_{ctr}, and K_{hash}, are derived from K_M. The derivation follows the counter mode key derivation function (KDF) specified in NIST SP 800-108 Rev. 1 [6], with HMAC-SHA3-256 as its pseudorandom function (PRF). For each role $u \in \{\text{head1}, \text{head2}, \text{seed}, \text{ctr}, \text{hash}\}$, the corresponding working key is obtained as

$$K_u = \text{MSB}_k \left(H_{K_M} \left(\langle 1 \rangle_{32} \,\|\, \text{Label}_u \,\|\, 0^8 \,\|\, \langle k \rangle_{32} \right) \right).$$

Here, $\text{MSB}_s(X)$ selects the first s bits of X, and $\langle x \rangle_w$ is the unsigned w-bit big-endian representation of x. The labels are the ASCII strings `animagus/head1`, `animagus/head2`, `animagus/seed`, `animagus/ctr`, and `animagus/hash`, respectively, with no implicit terminator. The KDF context is empty, and $k \leq 256$ allows each derivation to be completed in one iteration. These keys are derived once during key setup, rather than independently for each message. The master key is used exclusively for this derivation. We denote the resulting working key tuple by

$$\mathbb{K} = (K_{\text{head1}}, K_{\text{head2}}, K_{\text{seed}}, K_{\text{ctr}}, K_{\text{hash}}).$$

This tuple forms the key input to both encryption and decryption. The working keys are retained for subsequent operations under the same master key. Their derivation is independent of the tweak, so changing T does not require rederiving the working keys. Neither encryption nor decryption repeats the key derivation procedure.

The key K_{head1} is assigned to the block cipher operations on the first branch, and K_{head2} to those on the second branch. The key K_{seed} is used exclusively for the encryption that generates the counter mode seed. K_{ctr} is dedicated to the counter mode keystream, whereas K_{hash} is used for the keyed hash operations. Thus, the two head branches use different block cipher keys. For either head branch, the head transformation, seed generation, and counter mode computations use three different block cipher keys. Since the branches are processed by independently keyed permutations, no symmetry exists between them, and no additional dependency between the branches is required to break such symmetry. The separation of the working keys is computational and is accounted for by the security of the KDF; deterministic

derivation from a single master key does not imply information theoretic independence.

Figure 1 gives a schematic representation of Animagus encryption. In the figure, E denotes the encryption algorithm of the underlying block cipher, H denotes HMAC, and the subscripts indicate the working key used in each operation. We write the plaintext as $P = P_1 \,\|\, P_2 \,\|\, \overline{P}$ and the corresponding ciphertext as $C = C_1 \,\|\, C_2 \,\|\, \overline{C}$. The core takes a fixed length tweak $T \in \{0,1\}^n$. External information, including a nonce and associated data, is converted into T by an application dependent derived function. This function uses K_{hash} on a hash input domain separate from those of the core and truncates its output to n bits. The external input framing and the separation of application types are described after the construction. The following description concerns the operational principles and parameters of the core.

For a variable length plaintext P satisfying the above size constraints, the algorithm separates the input into three components: the two initial branches, P_1 and P_2, each containing n bits, and the remaining segment $\overline{P}$. The minimum message length of $2n$ bits gives $t = |\overline{P}| \geq 0$. When the message contains exactly $2n$ bits, $\overline{P}$ is empty, and the same sequence of operations is followed without a separate short message case.

The first hashing step applies HMAC under K_{hash} to the concatenation $\overline{P} \,\|\, 10 \,\|\, T$, thereby processing T and $\overline{P}$ together. The two-bit string 10 identifies the plaintext tail operation, while 01 identifies the ciphertext tail operation. Both separators are literal two-bit strings, and HMAC processes the exact input bit string and its length. The fixed length of T makes its boundary and the separator position unambiguous. For an empty $\overline{P}$, the input is $10 \,\|\, T$. The resulting $2n$-bit output is divided into the n-bit parts A_1 and A_2. The first part, A_1, is XORed with P_2, and the second part, A_2, is XORed with P_1.

The masked first branch is then encrypted under K_{head1}, giving $F = E_{K_{\text{head1}}}(P_1 \oplus A_2)$, while the masked second branch is encrypted under K_{head2}, giving $B = E_{K_{\text{head2}}}(P_2 \oplus A_1)$. Neither encryption depends on the output of the other, so they can be evaluated in parallel. Their outputs are XORed, and the result is encrypted under the separate key K_{seed} to obtain the counter mode seed:

$$J = E_{K_{\text{seed}}}(F \oplus B).$$

Thus, the seed is obtained by encrypting the XOR of the intermediate head values, rather than using their XOR directly. The generating key is not used by any other operation. The value J is computed within the construction and is neither separately selected nor transmitted as an initialization vector.

The seed also mixes the intermediate head values by XORing J into both branches:

$$F^* = F \oplus J, \qquad B^* = B \oplus J.$$

Because the same value is applied to both branches, their XOR is preserved, namely $F^* \oplus B^* = F \oplus B$. Each mixed value therefore depends on both initial plaintext branches through J, including when the remaining segment is empty.

The keystream for the remaining plaintext segment is generated by counter mode under the separate key K_{ctr}. With $r = \lceil t/n \rceil$, it is defined as

$$L = \mathrm{MSB}_t \left(\|_{i=0}^{r-1} E_{K_{\mathrm{ctr}}} \left(J \oplus \langle i \rangle_n \right) \right).$$

The counter begins at zero, and each counter encoding is XORed with J. The computation generates exactly r block cipher outputs and retains their first t bits. The ciphertext tail is then $\overline{C} = \overline{P} \oplus L$. A partial final plaintext block is consequently handled without extending the message to a multiple of n. For $t = 0$, we have $r = 0$; both L and $\overline{C}$ are empty, and no counter mode block cipher call is performed. Both initial plaintext branches contribute to the internal values used to determine the seed and keystream.

Next, the ciphertext segment $\overline{C}$ is hashed using the same HMAC function under K_{hash}, with input $\overline{C} \,\|\, 01 \,\|\, T$. This input becomes $01 \,\|\, T$ when $\overline{C}$ is empty. Its $2n$-bit output is split into the n-bit parts A_3 and A_4, which incorporate the ciphertext tail information into the encrypted head values. The first mixed value F^* is encrypted under K_{head1} and the result is XORed with A_4 to obtain C_1. Likewise, B^* is encrypted under K_{head2} and the result is XORed with A_3 to obtain C_2:

$$C_1 = E_{K_{\mathrm{head1}}}(F^*) \oplus A_4, \qquad C_2 = E_{K_{\mathrm{head2}}}(B^*) \oplus A_3.$$

The final head encryptions can also be evaluated in parallel, since neither requires the output of the other. The inter branch dependency is supplied by J, rather than by their evaluation order. Finally, C_1, C_2, and $\overline{C}$ are concatenated to form the complete ciphertext $C = C_1 \,\|\, C_2 \,\|\, \overline{C}$, preserving the length: $|C| = |P|$.

The output masks are applied after the final encryptions. As a result, decryption passes $C_1 \oplus A_4$ and $C_2 \oplus A_3$ to the inverse block cipher, rather than C_1 and C_2 alone. Each ciphertext head block is therefore combined with the ciphertext tail and tweak before it enters a permutation. The corresponding input side arrangement masks P_1 and P_2 with A_2 and A_1 before the first encryptions. The security analysis in Section 3 requires this placement.

Decryption uses the inverse head transformations $D_{K_{\text{head1}}} = E^{-1}_{K_{\text{head1}}}$ and $D_{K_{\text{head2}}} = E^{-1}_{K_{\text{head2}}}$ to recover $F^* = D_{K_{\text{head1}}}(C_1 \oplus A_4)$ and $B^* = D_{K_{\text{head2}}}(C_2 \oplus A_3)$. Encrypting their XOR under K_{seed} regenerates J before F and B are known, using the identity $F^* \oplus B^* = F \oplus B$. The intermediate values are recovered as $F = F^* \oplus J$ and $B = B^* \oplus J$, and the keystream is reproduced under K_{ctr}. Thus, seed generation uses encryption in both directions, while recovery of the head values and plaintext branches uses inverse block cipher operations. Encryption and decryption use the same fixed length tweak and derived working keys throughout.

The construction is expressed in terms of an n-bit block cipher, but the concrete instantiation examined in this paper is AES with HMAC-SHA3-256. Other block sizes would require corresponding primitive choices, parameter specifications, and security analysis. These instantiations remain outside the scope of this work.

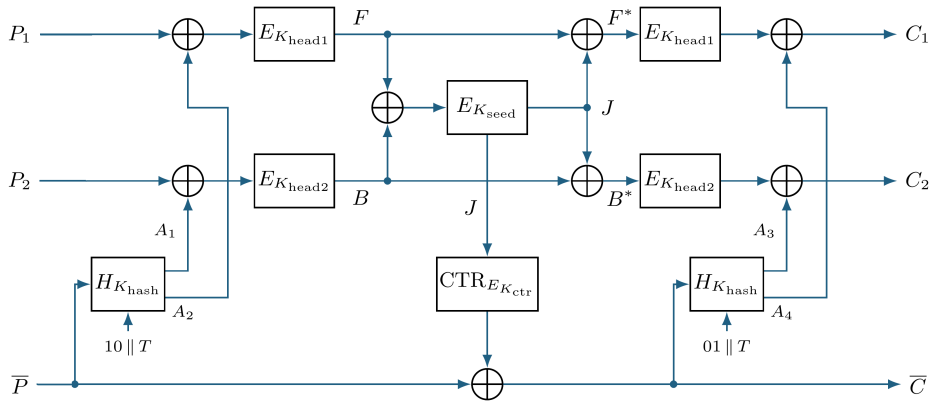

Figure 1: Animagus Encryption Algorithm

Algorithm 1 Enc$(\mathbb{K}, \mathbf{T}, P)$

$P_1 \leftarrow \mathrm{MSB}_n(P)$
$P_2 \leftarrow \mathrm{MSB}_n(P \ll n)$
$\overline{P} \leftarrow \mathrm{LSB}_{|P|-2n}(P)$
$X_1 \leftarrow H_{K_{\mathrm{hash}}}(\overline{P} \,\|\, 10 \,\|\, T)$
$A_1 \leftarrow \mathrm{MSB}_n(X_1)$
$A_2 \leftarrow \mathrm{LSB}_n(X_1)$
$F \leftarrow E_{K_{\mathrm{head1}}}(P_1 \oplus A_2)$
$B \leftarrow E_{K_{\mathrm{head2}}}(P_2 \oplus A_1)$
$J \leftarrow E_{K_{\mathrm{seed}}}(F \oplus B)$
$F^* \leftarrow F \oplus J$
$B^* \leftarrow B \oplus J$
$L \leftarrow \mathrm{CTR}_{E_{K_{\mathrm{ctr}}}}(J, |\overline{P}|)$
$\overline{C} \leftarrow \overline{P} \oplus L$
$X_2 \leftarrow H_{K_{\mathrm{hash}}}(\overline{C} \,\|\, 01 \,\|\, T)$
$A_3 \leftarrow \mathrm{MSB}_n(X_2)$
$A_4 \leftarrow \mathrm{LSB}_n(X_2)$
$C_1 \leftarrow E_{K_{\mathrm{head1}}}(F^*) \oplus A_4$
$C_2 \leftarrow E_{K_{\mathrm{head2}}}(B^*) \oplus A_3$
$C \leftarrow C_1 \,\|\, C_2 \,\|\, \overline{C}$
return C

Algorithm 2 Dec$(\mathbb{K}, \mathbf{T}, C)$

$C_1 \leftarrow \mathrm{MSB}_n(C)$
$C_2 \leftarrow \mathrm{MSB}_n(C \ll n)$
$\overline{C} \leftarrow \mathrm{LSB}_{|C|-2n}(C)$
$\overline{X_1} \leftarrow H_{K_{\mathrm{hash}}}(\overline{C} \,\|\, 01 \,\|\, T)$
$\overline{A_1} \leftarrow \mathrm{MSB}_n(\overline{X_1})$
$\overline{A_2} \leftarrow \mathrm{LSB}_n(\overline{X_1})$
$\overline{F}^* \leftarrow D_{K_{\mathrm{head1}}}(C_1 \oplus \overline{A_2})$
$\overline{B}^* \leftarrow D_{K_{\mathrm{head2}}}(C_2 \oplus \overline{A_1})$
$\overline{J} \leftarrow E_{K_{\mathrm{seed}}}(\overline{F}^* \oplus \overline{B}^*)$
$\overline{F} \leftarrow \overline{F}^* \oplus \overline{J}$
$\overline{B} \leftarrow \overline{B}^* \oplus \overline{J}$
$\overline{L} \leftarrow \mathrm{CTR}_{E_{K_{\mathrm{ctr}}}}(\overline{J}, |\overline{C}|)$
$\overline{P} \leftarrow \overline{C} \oplus \overline{L}$
$\overline{X_2} \leftarrow H_{K_{\mathrm{hash}}}(\overline{P} \,\|\, 10 \,\|\, T)$
$\overline{A_3} \leftarrow \mathrm{MSB}_n(\overline{X_2})$
$\overline{A_4} \leftarrow \mathrm{LSB}_n(\overline{X_2})$
$P_1 \leftarrow D_{K_{\mathrm{head1}}}(\overline{F}) \oplus \overline{A_4}$
$P_2 \leftarrow D_{K_{\mathrm{head2}}}(\overline{B}) \oplus \overline{A_3}$
$P \leftarrow P_1 \,\|\, P_2 \,\|\, \overline{P}$
return P

2.1 Design Rationale

Our design employs the hash-encrypt-hash structure exemplified by HCTR2 [11], while drawing on the accordion mode approach. Combining a two-block head with counter mode tail processing preserves message length and accommodates partial final blocks without padding. A key departure from constructions based on algebraic universal hashing is the use of HMAC-SHA3-256. Its pseudorandom function security, rather than collision resistance alone, supports the analysis. This choice avoids a separate finite field multiplication layer for hashing, although it may introduce additional hashing cost.

The two head branches are coupled through $J = E_{K_{\mathrm{seed}}}(F \oplus B)$, which supplies both the counter seed and the mixing value. Both initial head blocks therefore influence the mixed values even when the tail is empty. The identity $F^* \oplus B^* = F \oplus B$ permits decryption to regenerate J before recovering the plaintext tail. The two encryptions within each head layer can be evaluated in parallel. Purpose separated keys distinguish the head

branches, seed generation, counter processing, and hashing; each head key is reused within its own branch, as accounted for in the proof.

Plaintext side masks precede the first head encryptions, whereas ciphertext side masks follow the final encryptions and are removed before the inverse calls during decryption. This placement incorporates the relevant tail and tweak into the submitted head blocks before they enter a permutation. The core achieves birthday bound VIL-SPRP security under the stated primitive assumptions. Authentication is provided through derived functions that prepend and verify fixed authentication data, not by the second head branch alone; these interfaces require separate security analysis.

The specified parameters also meet selected key size and message length requirements in NIST IR 8552 [8]: 256-bit master key support, one bit message granularity, a minimum input length of $2n$ bits, and an AES message format limit exceeding 2^{48} bits. These parameter choices do not imply full conformity with the accordion requirements or establish implementation performance.

Table 1, adapted from [15] and extended with additional related schemes, compares encrypt-mix-encrypt and hash-encrypt-hash constructions. For $m = \lceil |P|/n \rceil$, including a possible partial final block, Animagus requires four head calls, one seed call, $m-2$ counter calls, and two keyed hash evaluations, giving $(m+3)\mathrm{BC} + 2\mathrm{HF}$. Key setup and application level tweak derivation are excluded. HF denotes a complete keyed hash evaluation, whose cost depends on input length. The final row reports the simplified statistical bound $3q^2a^2/2^n$; the primitive security terms appear in the full theorem in Section 4.

2.2 Applications of Animagus

Animagus can be used within application specific derived functions for Authenticated Encryption with Associated Data, Tweakable Encryption, and Deterministic Authenticated Encryption. These functions leave the core encryption and decryption procedures unchanged, while specifying how external information is incorporated into the tweak and whether fixed authentication data is included in the plaintext. The security of each application depends on that of the core construction and requires analysis of its corresponding derived function.

Here, M denotes the application message, whereas P denotes the input passed to the Animagus core. The derived functions use the working keys defined above. Although the core takes a fixed length n-bit tweak, external inputs such as a nonce, associated data, or application context may have

Table 1: Summary of Previous Works

Design	Security	Cipher	Hash	Operation Count
CMC	$7q^2a^2/2^n$	BC		$(2m+1)$BC
EME	$7q^2a^2/2^n$	BC		$(2m+2)$BC
AEZ	$13q^2a^2/2^{128}$	BC		mBC
ZCZ	$3q^2a^2/2^{2n+1}$	TBC	Polynomial	$(2m+6+m/n)$TBC+ 4HF
HCTR	$4.5q^2a^2/2^n$	BC	Polynomial	mBC + 2HF
PEP	$3q^2a^2/2^n$	BC	Polynomial	$(m+5)$BC + $(4m-6)FM$
HCH	$7q^2a^2/2^n$	BC	Polynomial	$(m+3)$BC + $2(m-1)$FM + 2HF
TET	$1.5q^2a^2/2^n$	BC	Invertible Block-wise Universal	$(m+1)$BC + $2m$FM
TCT_2	$qa^2/2^n + q^3a^3/2^{2n}$	BC	Polynomial, AU/AXU	$(6m+6)$BC + $(7m+7)$HF
THCTR	$2qa/2^n + 2q^2a/2^{n+s}$	TBC	n-bit keyed AXU, $(n+s)$-bit keyed pAXU	mTBC + 3HF
ADIANTUM	$q^2a/2^{116}$	BC	Polynomial, ϵ-ΔU	BC+SC+FM+FA+ 2HF
HCTR2	$1.5q^2a^2/2^n$	BC	Polynomial, AXU	mBC + 2HF
ddd-AES	$q^2a^2/2^n$	BC	Universal	$(m+2)$BC + $2m$HF
ddd-AES^+	$q^2a^2/2^n$	BC	Universal	$(m+t+1)$BC+$2m$HF
bbb-ddd-AES	$q^3a^3/2^{2n}$	BC	Universal	$(2m+4)$BC + $2m$HF
HCTR+	$q^2a/2^{2n} + 1/2^n$	TBC	$2n$-bit keyed AXU	$(m+2)$TBC + 2HF
ACCOR-S	$4.5q^2a^2/2^{128}$	BC	Polynomial, AXU	mBC + 2HF
ACCOR-L	$qa/2^{124}$	BC	Polynomial, AXU	$(m+1)$BC + 4FM + 2HF
CHCTR2	$q^3a^3/2^{2n}$	BC	Universal	$2m$BC + 3HF
HCTR2-TwKD	$q^3a^3/2^{2n}$	BC	Universal	mBC + 2HF + KDF
ANIMAGUS	$3q^2a^2/2^n$	BC	Cryptographic	$(m+3)$BC + 2HF

BC: Block Cipher, TBC: Tweakable Block Cipher, SC: Stream Cipher, HF: Hash Function, AU: Almost Universal, AXU: Almost XOR Universal, pAXU: partial AXU, ΔU: Almost Δ Universal, FM: Field Multiplication, FA: Field Addition, n: block size, a: maximum message length in blocks, q: number of queries, s: tweak length after hashing, m: number of blocks of a specific message

variable lengths. A common derived function, denoted as MakeTweak, processes these inputs under K_{hash}.

An application identifier d is included in the tweak derivation input to

distinguish the intended interfaces. The assignments $d = 1$, $d = 2$, and $d = 3$ correspond to nonce based authenticated encryption, confidentiality only tweakable encryption, and deterministic authenticated encryption, respectively. Its value is fixed by the invoked application interface. For a nonce N and associated data AD, with $|N|, |AD| < 2^{64}$, the encoding and tweak derivation are given by

$$W(d, N, AD) = \langle d \rangle_8 \,\|\, \langle |N| \rangle_{64} \,\|\, N \,\|\, AD,$$
$$\mathrm{MakeTweak}_{K_{\mathrm{hash}}}(d, N, AD) = \mathrm{MSB}_n \left(H_{K_{\mathrm{hash}}} \left(W(d, N, AD) \,\|\, 00 \,\|\, 0^n \right) \right).$$

The integer fields follow the fixed width big-endian encoding defined previously, and ε denotes the empty bit string. The nonce length field specifies where N ends and AD begins: after the indicated number of nonce bits, all remaining bits belong to AD. Consequently, distinct valid external input tuples have distinct encodings.

The suffix $00 \,\|\, 0^n$ places tweak derivation in a separate input domain from the core hashing operations, whose inputs are $\overline{P} \,\|\, 10 \,\|\, T$ and $\overline{C} \,\|\, 01 \,\|\, T$. In each case, exactly n bits follow the two-bit role code. The three hash input domains are therefore disjoint, also when $\overline{P}$ and $\overline{C}$ are empty. The public string 0^n fixes the position of the role code; it is not transmitted as message padding. The identifier d further separates the derivation inputs for different applications. These separations apply to the hash inputs and do not exclude collisions between the resulting truncated tweaks.

For the AES instantiation, H is HMAC-SHA3-256, and the tweak is obtained from the first $n = 128$ output bits. With the same working key, identical external inputs yield the same tweak. When MakeTweak is used, the receiver can therefore recompute T from the corresponding external information. The counter seed J is regenerated internally during core decryption, so neither the derived tweak nor the seed needs to be transmitted separately.

2.2.1 Authenticated Encryption with Associated Data

An Authenticated Encryption with Associated Data (AEAD) scheme combines message confidentiality with integrity for the message and its associated data. In this application, the nonce and associated data are incorporated into the tweak, and fixed authentication data is prepended to the message. This uses the encode-then-encipher approach discussed in [8] without modifying the Animagus core.

The authentication data (AUD) is a fixed public n-bit value defined as $AUD = 0^n$. The core input is formed as $P = AUD \,\|\, M$, and encryption

uses

$$T = \mathrm{MakeTweak}_{K_{\mathrm{hash}}}(1, N, AD),$$
$$C = \mathrm{Enc}(\mathbb{K}, T, AUD \,\|\, M).$$

The nonce and associated data lengths satisfy $1 \leq |N| < 2^{64}$ and $0 \leq |AD| < 2^{64}$, respectively. Thus, associated data may be empty. The application supplies N and should avoid nonce repetition under the same master key. The core VIL-SPRP analysis itself permits repeated tweaks. This does not replace the separate analysis required to establish nonce misuse resistance for the authenticated interface.

The minimum core input length of $2n$ bits gives the application message range

$$n \leq |M| \leq \ell_{\max} - n.$$

For AES, the minimum application message length is therefore 128 bits. The ciphertext has length $|C| = |M| + n$; its additional n bits come from the authentication data included in the core plaintext. No separate HMAC tag is appended. Messages outside the supported range are rejected, rather than extended by padding.

On decryption, the receiver recomputes T from the supplied nonce and associated data. After checking the input lengths, it applies core decryption and parses the result as

$$V \,\|\, M = \mathrm{Dec}(\mathbb{K}, T, C), \qquad |V| = n.$$

The recovered prefix V is checked against AUD. If they match, the prefix is removed and M is returned; otherwise, decryption returns $\perp$. This comparison is performed in constant time, and no part of the recovered application message is released until verification succeeds.

The receiver requires the same nonce and associated data as those used for encryption. Their transport encoding is outside the core ciphertext format and is not counted in $|C| = |M| + n$.

2.2.2 Tweakable Encryption

A tweakable encryption scheme uses an additional input, the tweak, to specify the encryption context. This interface is intended to provide message confidentiality without an authentication check. Accordingly, no dedicated authentication data is prepended, and the application message is passed directly to the Animagus core.

An external tweak of exactly n bits can be supplied directly to the core. Alternatively, a variable length tweak or context V, with $|V| < 2^{64}$, is converted by the common derived function:

$$T = \text{MakeTweak}_{K_{\text{hash}}}(2, \varepsilon, V).$$

The nonce field is empty in this case, and V is placed in the external context field. Encryption computes

$$C = \text{Enc}(\mathbb{K}, T, M),$$

for $2n \leq |M| \leq \ell_{\max}$. Decryption uses the same tweak and working keys. The absence of added authentication data means that the ciphertext has the same length as the application message.

Computing T from a variable length context requires one HMAC evaluation, with processing cost determined by the context length. Changing the external context does not require rederiving the working key tuple. The core hashing and message processing operations are still performed for the resulting tweak. This preprocessing does not add an integrity check: decryption maps a supported ciphertext to a plaintext without checking authentication data. The interface must therefore not provide an unauthenticated decryption bypass for either authenticated interface under the same key instance.

2.2.3 Deterministic Authenticated Encryption

A Deterministic Authenticated Encryption (DAE) scheme combines confidentiality and message authentication with deterministic operation. Unlike the nonce based interface, it accepts no nonce and introduces no fresh randomness during encryption. The fixed length internal tweak is nevertheless retained in the Animagus construction.

Neither a nonce nor associated data is supplied to this interface. Its internal tweak is fixed as

$$T = \text{MakeTweak}_{K_{\text{hash}}}(3, \varepsilon, \varepsilon).$$

For a given key instance, this value is constant and can be computed once during initialization. Encryption and decryption use this same tweak; the application interface accepts only the key and the message or ciphertext, respectively.

As in the AEAD interface, the fixed value $AUD = 0^n$ is prepended to the application message. Encryption is defined by

$$C = \text{Enc}(\mathbb{K}, T, AUD \,\|\, M).$$

The permitted message lengths and ciphertext expansion are also unchanged from the AEAD interface:

$$n \le |M| \le \ell_{\max} - n, \qquad |C| = |M| + n.$$

Decryption applies the core algorithm and checks the first n recovered bits against AUD. The application message is returned only upon a successful comparison; otherwise, the output is $\perp$. The same input length checks and constant time comparison are used, and unverified plaintext is not released.

Under a fixed key, encrypting the same application message again yields the same ciphertext. This reflects the deterministic nature of the interface, not the presence of a hidden nonce or the omission of the internal tweak.

3 Security Analysis of the Design

The analysis follows the concrete security approach of Bellare et al. [1]. The security goal is formulated in the tweakable enciphering framework of Halevi and Rogaway [25]. A tweakable enciphering scheme defines a length preserving permutation for every fixed key and tweak. In the ideal experiment, an independent uniform permutation $\Pi_{T,\ell}$ on $\{0,1\}^\ell$ is selected for each supported tweak length pair (T, ℓ), and its two directions are answered consistently. For Animagus, the message and tweak spaces are

$$\mathcal{M} = \bigcup_{\ell=2n}^{\ell_{\max}} \{0,1\}^\ell, \qquad \mathcal{T} = \{0,1\}^n.$$

The tweakable variable-input-length strong pseudorandom permutation advantage is defined as

$$\mathrm{Adv}^{\text{VIL-SPRP}}_{\text{Animagus}}(\mathcal{A}) := \Big| \Pr[\mathcal{A}^{\mathrm{Enc}_K, \mathrm{Dec}_K} \Rightarrow 1] - \Pr[\mathcal{A}^{\Pi, \Pi^{-1}} \Rightarrow 1] \Big|. \qquad (1)$$

In the first probability, a master key is sampled uniformly and $K \leftarrow \mathrm{Derive}(K_M)$ is computed once. The second probability includes the sampling of the ideal permutation family. The adversary may interleave adaptive encryption and decryption queries, reuse tweaks and message components, and choose different supported lengths. Only complete repeated queries and exact inverse queries whose answers are already known are omitted. This convention is imposed for every possible oracle transcript, including transcripts produced by the intermediate games. In particular, all nonredundant queries may use the same tweak. The bound in Theorem 1 applies under the stated aggregate

resource limits and does not impose a separate restriction on the number of queries made under any one tweak.

The working key tuple derived from the master key, as specified in Section 3, is

$$K = (K_{\text{head1}}, K_{\text{head2}}, K_{\text{seed}}, K_{\text{ctr}}, K_{\text{hash}}).$$

Let $K^\star$ denote five independently sampled uniform working keys of their prescribed lengths. The key separation advantage is

$$\text{Adv}^{\text{sep}}_{\text{KDF}}(\mathcal{B}) := \left|\Pr[\mathcal{B}(\text{Derive}(K_M)) = 1] - \Pr[\mathcal{B}(K^\star) = 1]\right|.$$

This is a computational assumption on the complete derived tuple. It does not assert information theoretic independence of keys derived from the same master key. All working keys remain fixed throughout the experiment.

Let q and ρ be fixed, execution wide upper bounds on the numbers of construction queries and counter evaluations. For a transcript with $Q \le q$ queries, put

$$t_j = \ell_j - 2n, \qquad r_j = \left\lceil \frac{t_j}{n} \right\rceil, \qquad \sigma_{\text{tr}} = \sum_{j=1}^{Q} \left\lceil \frac{\ell_j}{n} \right\rceil.$$

The actual counter complexity is $\rho_{\text{tr}} = \sum_j r_j = \sigma_{\text{tr}} - 2Q \le \rho$. Empty tails have $t_j = r_j = 0$ and contribute no counter calls; empty concatenations and zero-bit prefixes are interpreted as ε. Each query makes two calls under each head key, one seed call, and two keyed hash evaluations.

Theorem 1. *Let E be the underlying n-bit block cipher and let H be the $2n$-bit output keyed hash function. Let $\mathcal{A}$ be a classical adversary with budgets q and ρ as above. Assume that each query satisfies $r_j \le 2^n$, so that its counter encodings are distinct. There exist adversaries $\mathcal{B}_{\text{KDF}}, \mathcal{B}_{\text{h1}}, \mathcal{B}_{\text{h2}}, \mathcal{B}_{\text{s}}, \mathcal{B}_{\text{c}}$, and $\mathcal{B}_H$ such that*

$$\text{Adv}^{\text{VIL-SPRP}}_{\text{Animagus}}(\mathcal{A}) \le \epsilon_{\text{prim}} + \frac{9q^2 + \rho^2}{2^n} + \frac{\binom{q}{2}}{2^{2n}}, \tag{2}$$

where

$$\begin{aligned} \epsilon_{\text{prim}} = {} & \text{Adv}^{\text{sep}}_{\text{KDF}}(\mathcal{B}_{\text{KDF}}) + \text{Adv}^{\text{SPRP}}_{E}(\mathcal{B}_{\text{h1}}) + \text{Adv}^{\text{SPRP}}_{E}(\mathcal{B}_{\text{h2}}) \\ & + \text{Adv}^{\text{PRP}}_{E}(\mathcal{B}_{\text{s}}) + \text{Adv}^{\text{PRP}}_{E}(\mathcal{B}_{\text{c}}) + \text{Adv}^{\text{PRF}}_{H}(\mathcal{B}_H). \end{aligned} \tag{3}$$

The primitive query bounds are $2q, 2q, q, \rho$, and $2q$, respectively. Only the head reductions require inverse access. The simulations include the remaining

construction operations and table maintenance; the total length of the hash inputs in a transcript is $2\sum_{j=1}^{Q}(t_j + n + 2)$ *bits. Alternatively, under an execution wide message block budget* σ*, the statistical part can be bounded directly by* $3\sigma^2/2^n$*.*

Proof. The proof is organized using the code based game playing framework of Bellare and Rogaway [2]. Computational game transitions are justified by primitive reductions; the exact-to-relaxed transition uses the identical until bad principle. We then apply a response first reordering and bound the resulting collision events. The final comparison with an ideal tweakable permutation follows CMC [25]. The treatment of repeated head values below is specific to Animagus. Write $p_i = \Pr[\mathcal{A}^{G_i} \Rightarrow 1]$.

Game 0. This is the real experiment, in which $\mathcal{A}$ interacts with the Animagus encryption and decryption algorithms specified in Section 3. The working key tuple is derived once from a uniformly sampled master key and remains fixed throughout the interaction.

Game 1. Replace the derived tuple by $K^\star$. By definition,

$$|p_0 - p_1| \le \mathrm{Adv}^{\mathrm{sep}}_{\mathrm{KDF}}(\mathcal{B}_{\mathrm{KDF}}).$$

The independent working keys are not conditioned to be distinct.

For the AES/HMAC-SHA3-256 derivation specified in Section 3, the five PRF inputs have distinct role labels. Replacing H_{K_M} by a random function therefore produces five independent uniform k-bit working keys after truncation. Thus the key separation term is bounded by an HMAC-SHA3-256 PRF advantage with five oracle queries.

Game 2. Replace the two head cipher instances by independent uniform permutations π_1, π_2 on $\{0,1\}^n$. Each permutation is retained for both calls on its branch, in every query and in both directions. Successive SPRP hybrids give

$$|p_1 - p_2| \le \mathrm{Adv}^{\mathrm{SPRP}}_{E}(\mathcal{B}_{\mathrm{h1}}) + \mathrm{Adv}^{\mathrm{SPRP}}_{E}(\mathcal{B}_{\mathrm{h2}}).$$

Each reduction makes at most $2q$ forward or inverse queries. In a hybrid, the other roles are simulated using independently sampled keys or previously introduced ideal primitives. This uses the unconditioned key distribution and requires no separate key collision term.

Game 3. Replace the seed and counter instances by independent uniform permutations π_{s} and π_{c}, independent of π_1, π_2. Both are evaluated only in the forward direction, so

$$|p_2 - p_3| \le \mathrm{Adv}^{\mathrm{PRP}}_{E}(\mathcal{B}_{\mathrm{s}}) + \mathrm{Adv}^{\mathrm{PRP}}_{E}(\mathcal{B}_{\mathrm{c}}).$$

The respective query budgets are q and ρ.

Game 4. Replace $H_{K_{\text{hash}}}$ by a consistent random function $\mathcal{R} : \{0,1\}^* \to \{0,1\}^{2n}$. Then

$$|p_3 - p_4| \le \mathrm{Adv}_H^{\mathrm{PRF}}(\mathcal{B}_H).$$

The encodings $(\bar{P}, T) \mapsto \bar{P} \parallel 10 \parallel T$ and $(\bar{C}, T) \mapsto \bar{C} \parallel 01 \parallel T$ are injective and have disjoint images, including for empty tails. Repeated arguments reuse the same value of $\mathcal{R}$ and do not themselves set a flag.

Game 5. Represent $\pi_1, \pi_2, \pi_{\mathrm{s}}, \pi_{\mathrm{c}}$ by exact lazy sampling. Initialize $\mathsf{bad} = \mathsf{false}$ once before the interaction. For each permutation, maintain a partial mapping and its domain and range sets $\mathcal{D}_i, \mathcal{R}_i$, initially empty. A forward call at X samples a uniform candidate Y. If $X \in \mathcal{D}_i$, set bad and return the stored image; otherwise, if $Y \in \mathcal{R}_i$, set bad and replace Y by a uniform unused image. Insert the resulting mapping. Inverse calls exchange the domain and range roles. These corrections give the exact permutation distribution.

The simulator also retains previously computed head values in four consistency tables:

$$\begin{aligned} \mathsf{P}_1[T, \bar{P}, P_1] &= F, \quad & \mathsf{P}_2[T, \bar{P}, P_2] &= B, \\ \mathsf{C}_1[T, \bar{C}, C_1] &= F^*, \quad & \mathsf{C}_2[T, \bar{C}, C_2] &= B^*. \end{aligned}$$

An existing submitted side entry is reused without another primitive call or another insertion into the domain and range sets. Otherwise, the appropriate forward or inverse call creates the entry. The seed, counter, and output operations are then performed, and the returned side entries are stored. Every table index includes the tweak, complete tail, and relevant head block; the table name specifies its side and branch.

This reuse is necessary. For example, fixing $T, \bar{P}, P_1$ while varying P_2 leaves $F = \pi_1(P_1 \oplus A_2)$ unchanged. Reusing its table entry is an exact simulation, not a freshness assumption. Distinct indices can still give equal primitive arguments, which are checked by the sampler. The tables are proof simulation devices, not additional implementation requirements. Since all lookups agree with the partial permutations, $p_4 = p_5$.

Throughout the proof, A_1, A_2 denote the plaintext side masks and A_3, A_4 denote the ciphertext side masks, independently of the query direction. In the local decryption notation of Algorithm 2, they correspond to $\overline{A}_3, \overline{A}_4, \overline{A}_1, \overline{A}_2$, respectively.

For each query, the hash masks under the idealized primitives satisfy

$$A_1 \parallel A_2 = \mathcal{R}(\bar{P} \parallel 10 \parallel T), \qquad A_3 \parallel A_4 = \mathcal{R}(\bar{C} \parallel 01 \parallel T).$$

The seed and mixing operations retain the relations

$$J = \pi_{\mathrm{s}}(F \oplus B), \qquad F^* = F \oplus J, \qquad B^* = B \oplus J.$$

To identify the collision cases, abbreviate the masked boundary points by

$$I_1 = P_1 \oplus A_2, \quad I_2 = P_2 \oplus A_1, \qquad O_1 = C_1 \oplus A_4, \quad O_2 = C_2 \oplus A_3.$$

The primitive relations and corresponding domain/range entry forms are

Permutation	Relations	Domain forms	Range forms
π_1	$I_1 \mapsto F,\ F^* \mapsto O_1$	$I_1,\ F^*$	$F,\ O_1$
π_2	$I_2 \mapsto B,\ B^* \mapsto O_2$	$I_2,\ B^*$	$B,\ O_2$
π_{s}	$s \mapsto J$	s	J
π_{c}	$J \oplus \langle i \rangle_n \mapsto W_i$	$J \oplus \langle i \rangle_n$	W_i

where $s = F \oplus B = F^* \oplus B^*$ and W_i is a full counter output block. The table specifies possible entry forms, not additional insertions for values already obtained by a consistency lookup.

Game 6. Remove the corrections made after setting bad in Game 5. Every executed primitive call returns a fresh uniform candidate, while head value lookups and the random function table remain consistent. Before each domain or range insertion, flag a repeated value and retain its occurrence for counting. A consistency lookup makes no insertion; intersections between different sets are not flagged. The games are identical until the first flag, so [2] gives

$$|p_5 - p_6| \le \Pr[G_6 \text{ sets } \mathsf{bad}]. \tag{4}$$

We bound this probability separately for the head, seed, and counter permutations. All probabilities below refer to Game 6.

Response distribution. Fix $\mathcal{R}$, the preceding internal state, the current query, its submitted side head values, and the seed candidate J. On encryption, the final head candidates O_1, O_2 and the full counter output string W are independent and uniform. The change of variables $(O_1, O_2, W) \leftrightarrow (C_1, C_2, \bar{C}, \omega)$ is bijective, with inverse

$$O_1 = C_1 \oplus A_4, \quad O_2 = C_2 \oplus A_3, \qquad W = (\bar{P} \oplus \bar{C}) \parallel \omega,$$

where ω contains the unused $nr - t$ bits of W. The ciphertext tail determines A_3, A_4. Decryption has the corresponding inverse $I_1 = P_1 \oplus A_2$, $I_2 = P_2 \oplus A_1$,

with the same expression for W. Thus every response is uniform, including when $t = r = 0$ and $W = \omega = \varepsilon$.

The public transcript can therefore be sampled first. Conditional on it, the random function entries, the independent draws η for new submitted side head entries, the seed candidates J_j, and the unused suffixes remain independent uniform base variables. Each stored head value is an expression $\eta_\alpha \oplus \bigoplus_{h \in S} J_h$; reuse preserves this expression and a returned value adds the current J_j. The head values themselves need not be independent.

We charge only the first flag and write $\mathsf{B}.$ for its indicated collision class. A repeated returned side index already causes a head collision: for example, $(T, \bar{C}, C_1)$ repeats O_1 in $\mathcal{R}_1$. Hence symbolic comparisons need only public prefixes τ preceding such a repetition, including the current query. This restriction involves only public indices, not hidden collision events. Fresh draws are instead analyzed using their pre-draw internal history $\mathcal{H}$. Neither argument conditions on $\neg\mathsf{bad}$.

Head collisions. The entry forms in Game 5 give three cases, according to how the later entry is obtained.

If it is a fresh candidate U, then for any earlier value v in the same set, $\Pr[U = v \mid \mathcal{H}] = 2^{-n}$. This covers new submitted side head values and final forward or inverse candidates, including the first repeated returned side index. If it is a mixed value, encryption gives, for example,

$$\Pr[F_j^* = v \mid \mathcal{H}] = \Pr[J_j = F_j \oplus v \mid \mathcal{H}] = 2^{-n},$$

since F_j and v are fixed before J_j is drawn. Decryption uses $F_j = F_j^* \oplus J_j$; the second branch is identical. Earlier entries in the relevant head set do not contain the current J_j.

It remains to consider a new submitted masked point, such as $I_{1,j}$. Put $u_j = \bar{P}_j \parallel 10 \parallel T_j$. Against an earlier masked point in $\mathcal{D}_1$,

$$\Pr[I_{1,j} = I_{1,h} \mid \tau] = \begin{cases} 2^{-n}, & u_j \neq u_h, \\ 0, & u_j = u_h. \end{cases}$$

In the first case the mask halves are independent. In the second they coincide, but $P_{1,j} \neq P_{1,h}$, since an identical table index would have been reused without insertion. Against an internal value $F_h^* = \eta_\alpha \oplus \bigoplus_{a \in S} J_a$, the equality instead requires

$$\eta_\alpha = P_{1,j} \oplus A_{2,j} \oplus \bigoplus_{a \in S} J_a.$$

The right hand side does not contain η_α, so its probability is 2^{-n}. Decryption exchanges I_1, F^* with O_1, F and uses the ciphertext side hash; branch 2

exchanges them with I_2, B^* or O_2, B. These cases include both calls within one query and all four combinations of query directions. There are at most $2q$ insertion attempts in each of the four head sets, giving

$$\Pr[\mathsf{B}_{\mathrm{h}}] \le \frac{4\binom{2q}{2}}{2^n}. \tag{5}$$

Seed input collisions. Unlike J_j, the input s_j is computed from possibly reused head values. We must therefore exclude a forced identity $s_j \equiv s_h$ before assigning probability 2^{-n} to their numerical equality. Here $\equiv$ denotes equality of formal XOR expressions, not of their sampled values.

Distinct entries in one branch have distinct expressions: a new submitted entry contains a fresh η, and a new returned entry contains the current J_j, absent from all older entries. We prove the following invariant by induction over queries. For pairs consisting of one entry from each branch on the same side, two distinct pairs have identical XORs only when they are the submitted and returned pairs of one processed query. The invariant includes pairs whose entries have different tweaks or tails.

Adding new submitted entries preserves the invariant because their η variables are fresh. Let x, y be the current submitted values and $x' = x \oplus J_j$, $y' = y \oplus J_j$ the new returned values. Nonredundancy excludes a submitted pair that is either side of an earlier query, so by induction $x \oplus y$ has no alternative old pair representation. A pair containing exactly one new value cannot agree with an old pair, since J_j does not cancel. Two pairs sharing a new entry agree only if their other entries agree. The remaining comparison is

$$x' \oplus v \equiv u \oplus y' \quad \Longleftrightarrow \quad x \oplus y \equiv u \oplus v,$$

with u, v old returned side entries. Induction would make the submitted pair one side of an earlier query, contradicting nonredundancy. Finally, $x' \oplus y' = x \oplus y$ gives precisely the current query's two pairs. This proves the invariant in both directions.

Consequently, $s_j \not\equiv s_h$ for $h < j$: otherwise the later submitted pair would repeat either side of query h. Their difference contains an independent base variable with coefficient one. Therefore,

$$\Pr[s_j = s_h \mid \tau] = 2^{-n}, \qquad \Pr[\mathsf{B}_{\mathrm{s,in}}] \le \frac{\binom{q}{2}}{2^n}. \tag{6}$$

The second inequality counts the at most q seed input insertions; the separate seed key excludes comparisons with head or counter inputs.

Seed output collisions. Every J_j is a fresh uniform candidate in Game 6, even when its input has appeared before. Hence $\Pr[J_j = J_h \mid \mathcal{H}] = 2^{-n}$ for $h < j$, and

$$\Pr[\mathsf{B}_{\mathrm{s,out}}] \le \frac{\binom{q}{2}}{2^n}. \tag{7}$$

Counter input collisions. Write $Z_{j,i} = J_j \oplus \langle i \rangle_n$. Distinct indices within one query give distinct inputs deterministically. For $h < j$, however,

$$\Pr[Z_{j,i} = Z_{h,k} \mid \mathcal{H}] = \Pr[J_j = Z_{h,k} \oplus \langle i \rangle_n \mid \mathcal{H}] = 2^{-n}.$$

Since the length is fixed before sampling J_j, query j has $r_j \sum_{h<j} r_h$ comparisons with earlier counter inputs. Thus

$$\Pr[\mathsf{B}_{\mathrm{c,in}}] \le \frac{1}{2^n} \mathbb{E}\left[\sum_{j=1}^{Q} r_j \sum_{h<j} r_h\right] \le \frac{\binom{\rho}{2}}{2^n}. \tag{8}$$

Counter output collisions. Each full candidate $W_{j,i}$ is uniform and independent of earlier counter outputs at its sampling time. A prescribed equality therefore has probability 2^{-n}, including within query pairs, and

$$\Pr[\mathsf{B}_{\mathrm{c,out}}] \le \frac{\binom{\rho}{2}}{2^n}. \tag{9}$$

For a partial final block, ω completes the full n-bit candidate; equal visible prefixes alone are not range collisions. A query with $t_j = r_j = 0$ contributes no counter entries.

These classes cover every possible first flag. Count each comparison when its later entry is generated, using $\mathcal{H}$ for fresh draw estimates and τ for the symbolic cases. This also covers adaptive lengths and stopping times. Keeping the actual counts Q and ρ_{tr} when summing the preceding pairwise bounds yields

$$\begin{aligned} \Pr[G_6 \text{ sets } \mathsf{bad}] &\le \frac{1}{2^n} \mathbb{E}_{G_6}\left[4\binom{2Q}{2} + 2\binom{Q}{2} + 2\binom{\rho_{\mathrm{tr}}}{2}\right] \\ &\le \frac{\mathbb{E}_{G_6}[9Q^2 + \rho_{\mathrm{tr}}^2]}{2^n} \le \frac{9q^2 + \rho^2}{2^n}. \end{aligned} \tag{10}$$

The three contributions are the four head sets, the two seed sets, and the two counter sets, respectively; no independence between collision events is required.

Game 7. Finally, replace the two sided random response experiment by the ideal family $\{\Pi_{T,\ell}\}$. We give the comparison explicitly, following the random response analysis of Halevi and Rogaway [25]. Couple the experiments

until their first inconsistent response. If a_j mappings are already defined for the current tweak length pair, a nonredundant query has a new argument on the queried side. Its uniform response is an already used opposite endpoint with probability

$$\frac{a_j}{2^{\ell_j}} \le \frac{j-1}{2^{2n}}.$$

If this occurs, replace that response in the ideal experiment by a uniform unused endpoint; otherwise retain it. This gives the correct permutation distribution in either direction. Summing over at most q queries gives

$$|p_6 - p_7| \le \sum_{j=1}^{q} \frac{j-1}{2^{2n}} = \frac{\binom{q}{2}}{2^{2n}}.$$

This comparison retains decryption access and is not a forward only PRP/PRF switch. Combining the game transitions and Equation (10) proves Equation (2).

For the alternative bound, suppose every execution satisfies $\sigma_{\mathrm{tr}} \le \sigma$. Since $\rho_{\mathrm{tr}} = \sigma_{\mathrm{tr}} - 2Q$ and $0 \le Q \le \sigma_{\mathrm{tr}}/2$, convexity gives

$$9Q^2 + \rho_{\mathrm{tr}}^2 = 9Q^2 + (\sigma_{\mathrm{tr}} - 2Q)^2 \le \frac{9}{4}\sigma_{\mathrm{tr}}^2 \le \frac{9}{4}\sigma^2.$$

Applying this inside the expectation in Equation (10), and using at most $\lfloor \sigma/2 \rfloor$ queries in the final comparison, bounds the statistical contribution by

$$\frac{9\sigma^2}{4 \cdot 2^n} + \frac{\sigma^2}{2^{2n+3}} \le \frac{3\sigma^2}{2^n}.$$

The estimate uses the joint relation between query and counter counts in each execution, rather than independent maxima for those counts. □

Security level. When the primitive advantages are negligible, the theorem establishes birthday bound VIL-SPRP security with statistical growth $O(\sigma^2/2^n)$. The minimum length case also gives a birthday scale distinguisher. Fix T and P_1, and encrypt distinct P_2 values in messages of length $2n$. The input mask and F are fixed; B, $F \oplus B$, J, and $F \oplus J$ are all distinct across these queries. Since the ciphertext tail is empty, A_4 is fixed and C_1 never repeats. For an ideal $2n$-bit permutation, the probability of a repeated first block after $q \le 2^n$ such queries is

$$1 - \prod_{j=0}^{q-1} \frac{2^{2n} - j2^n}{2^{2n} - j}.$$

This is nonnegligible at q of order $2^{n/2}$, showing tightness of the birthday security scale for fixed size messages, without asserting that every term in the upper bound is tight.

Scope of the theorem. The theorem concerns the Animagus core specified in Section 3, with an externally specified n-bit tweak. The derived application interfaces additionally employ tweak derivation and, where applicable, authentication redundancy. Their security requires a separate treatment of derived tweak collisions and verification queries and does not follow from the core theorem alone.

4 Conclusion

In this paper, we proposed and analyzed Animagus, an accordion inspired tweakable wide block mode based on the hash-encrypt-hash paradigm. The construction explores the use of a conventional block cipher together with a cryptographic keyed hash, without a separate algebraic universal hashing component. It preserves message length and supports every bit length within its specified range, starting from $2n$ bits, without requiring padding.

The concrete instantiation uses AES and HMAC-SHA3-256 with five purpose separated working keys derived from a single master key. The working keys are independent of the tweak in their derivation and need not be rederived when the tweak changes. An encrypted internal seed couples the two head branches and determines the counter mode keystream. This arrangement permits parallel evaluation within each head layer and retains inter branch dependence when the message has no tail. The hashing component is analyzed under a pseudorandom function assumption, rather than relying on collision resistance alone.

We established birthday bound VIL-SPRP security under the stated block cipher, keyed hash, and key derivation assumptions. The proof covers adaptive encryption and decryption queries, varying message lengths, and repeated tweaks. It explicitly accounts for dependencies arising from reused head values and from using each head key in both layers of its branch. Thus, the analysis supports the proposed arrangement without assuming that every internal primitive invocation receives a fresh input.

Application specific derived functions incorporate variable length external information into the fixed length core tweak and define interfaces for authenticated encryption with associated data, tweakable encryption, and deterministic authenticated encryption. The authenticated interfaces retain

the core transformations and introduce authentication through fixed redundancy and verification. Their security requires additional analysis of derived tweak collisions and verification queries beyond the core guarantee. Further work may evaluate implementation performance and establish concrete security bounds for these authenticated interfaces.

Acknowledgement

We would like to thank Jonathan Katz and Eik List for their comments on this manuscript.

References

[1] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 394–403, 1997.

[2] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[3] Ritam Bhaumik, Eik List, and Mridul Nandi. Zcz–achieving n-bit sprp security with a minimal number of tweakable-block-cipher calls. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 336–366. Springer, 2018.

[4] Debrup Chakraborty and Palash Sarkar. Hch: A new tweakable enciphering scheme using the hash-encrypt-hash approach. In *International Conference on Cryptology in India*, pages 287–302. Springer, 2006.

[5] Debrup Chakraborty and Palash Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. In *International Workshop on Fast Software Encryption*, pages 293–309. Springer, 2006.

[6] Lily Chen. Recommendation for key derivation using pseudorandom functions, February 2024.

[7] Yu Long Chen, Michael Davidson, Morris Dworkin, Jinkeon Kang, John Kelsey, Yu Sasaki, Meltem Sönmez Turan, Donghoon Chang, Nicky Mouha, MD Largo, et al. Proposal of requirements for an accordion mode. In *Discussion Draft for the NIST Accordion Mode Workshop 2024*, 2024.

[8] 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. 2025.

[9] Yu Long Chen, Yukihito Hiraga, Nicky Mouha, Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Beyond-birthday-bound security with HCTR2: Cascaded construction and tweak-based key derivation. Cryptology ePrint Archive, Paper 2026/085, 2026.

[10] Paul Crowley and Eric Biggers. {Adiantum}: length-preserving encryption for entry-level processors. *Cryptology ePrint Archive*, 2018.

[11] Paul Crowley, Nathan Huckleberry, and Eric Biggers. Length-preserving encryption with HCTR2. Cryptology ePrint Archive, Paper 2021/1441, 2021.

[12] Nilanjan Datta, Avijit Dutta, Shibam Ghosh, Eik List, and Hrithik Nandi. HCTR+: An optimally secure TBC-based accordion mode. Cryptology ePrint Archive, Paper 2024/2053, 2024.

[13] Christoph Dobraunig, Krystian Matusiewicz, Bart Mennink, and Alexander Tereschenko. Efficient instances of docked double decker with AES, and application to authenticated encryption. Cryptology ePrint Archive, Paper 2024/084, 2024.

[14] Avijit Dutta and Mridul Nandi. Tweakable HCTR: A BBB secure tweakable enciphering scheme. Cryptology ePrint Archive, Paper 2019/1324, 2019.

[15] Hieu Nguyen Duy, Pablo García Fernández, Aleksei UDOVENKO, and Alexei BIRYUKOV. Accordion mode based on hash-encrypt-hash. FNR - Fonds National de la Recherche, 2024. Trust and Security.

[16] Morris Dworkin. Recommendation for block cipher modes of operation: Methods and techniques, nist sp 800-38a, 2001. `https://https://doi.org/10.6028/NIST.SP.800-38A`.

[17] Morris Dworkin. Recommendation for block cipher modes of operation: Methods for key wrapping, nist sp 800-38f, 2001. `https://doi.org/10.6028/NIST.SP.800-38F`.

[18] Morris Dworkin. Recommendation for block cipher modes of operation: The ccm mode for authentication and confidentiality, nist sp 800-38c, 2004. `https://doi.org/10.6028/NIST.SP.800-38C`.

[19] Morris Dworkin. Recommendation for block cipher modes of operation: The cmac mode for authentication, nist sp 800-38b, 2005. `https://doi.org/10.6028/NIST.SP.800-38B`.

[20] Morris Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac, nist sp 800-38d, 2007. `https://doi.org/10.6028/NIST.SP.800-38D`.

[21] Morris Dworkin. Recommendation for block cipher modes of operation: The xts-aes mode for confidentiality on storage devices, nist sp 800-38e, https://doi.org/10.6028/nist.sp.800-38e, 2010. https://doi.org/10.6028/NIST.SP.800-38E.

[22] Morris Dworkin. Recommendation for block cipher modes of operation: Methods for format-preserving encryption, nist sp 800-38g, 2016. https://doi.org/10.6028/NIST.SP.800-38G.

[23] Shai Halevi. Invertible universal hashing and the tet encryption mode. In *Annual International Cryptology Conference*, pages 412–429. Springer, 2007.

[24] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. Cryptology ePrint Archive, Paper 2003/147, 2003.

[25] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. Cryptology ePrint Archive, Paper 2003/148, 2003.

[26] Viet Tung Hoang, PR Ted Krovetz, and Philip Rogaway. Aez v5: Authenticated encryption by enciphering., 2017. https://www.cs.ucdavis.edu/âĹijrogaway/aez/aez.pdf.

[27] Michael Luby and Charles Rackoff. How to construct pseudo-random permutations from pseudo-random functions (abstract). In *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, page 447. Springer, 1985.

[28] Nicky Mouha and Morris Dworkin. Report on the block cipher modes of operation in the nist sp 800-38 series. *NIST IR*, 8459, 2024.

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

[30] National Institute of Standards and Technology. PRE-DRAFT call for comments: NIST launches development of cryptographic accordions. Initial Preliminary Draft Announcement for NIST SP 800-197A, June 2025. Published June 6, 2025.

[31] Thomas Shrimpton and R. Seth Terashima. A modular framework for building variable-input-length tweakable ciphers. In Kazue Sako and

Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, pages 405–423, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[32] Alyssa Thompson and Meltem Sonmez Turan. Nist workshop on the requirements for an accordion cipher mode 2024: Workshop report. *NIST IR*, 2024.

[33] Peng Wang, Dengguo Feng, and Wenling Wu. Hctr: A variable-input-length enciphering mode. In *International Conference on Information Security and Cryptology*, pages 175–188. Springer, 2005.