MERIDIAN: A Toroid-Inspired Permutation Block Cipher for Constrained Environments

Basker Palaniswamy [1*], Paolo Palmieri [1], Ashok Kumar Das [2,3]

[1] Insight Research Ireland Centre for Data Analytics, Department of Computer Science and Information Technology, University College Cork (UCC), Cork City, Ireland, European Union.

[2] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, 500 032, India.

[3] Department of Computer Science and Engineering, College of Informatics, Korea University, 145 Anam-ro Seongbuk-gu, Seoul, 02841, South Korea.

{1* -- basker170889@gmail.com, 1 -- paolo.palmieri@ucc.ie,}
{2,3 -- iitkgp.akdas@gmail.com, ashok.das@iiit.ac.in}

*Corresponding Author

May 4, 2026

Revised 7 September 2026 — v20 corrigendum (Section 3 insert + Appendix D)

Abstract

We introduce MERIDIAN, a 128-bit block cipher designed for resource-constrained environments as a lightweight alternative to AES-128. MERIDIAN retains the AES-128 interface, including a 128-bit block, 128-bit key, and 4×4 byte state, while reusing the standard AES S-box. This enables compatibility with existing AES-128 modes such as ECB, CBC, CFB, OFB, CTR, XTS, CMAC, CCM, and GCM, and allows implementations to reuse established S-box ROMs and GF(2^8) inverse circuits.

The cipher is based on three round operations: Directional Substitution (DS), Meridian Diffusion (MD), and Admittance Mixing (AM), followed by a round constant. Its state is represented over the discrete torus $\mathbb{Z}_4 \times \mathbb{Z}_4$. The MD layer combines two orthogonal byte permutations, meridian and parallel, with column mixing to obtain full byte diffusion within three rounds without requiring MDS multiplication. MERIDIAN uses twelve rounds and avoids an expanded key schedule.

We present MILP-verified differential and linear bounds, including exact active-S-box counts up to eleven rounds and a sub-additive bound for twelve rounds. We also provide AES-aligned cryptanalysis covering differential, linear, integral, biclique, slide, related-key, invariant-subspace, fault, meet-in-the-middle, and algebraic attacks. A reproducible fifteen-experiment benchmark suite compares MERIDIAN directly with AES-128, including key agility, cold-start latency, energy, masking cost, memory, throughput, and hardware complexity.

Empirical results show that MERIDIAN reaches strict avalanche behavior in three rounds, matches AES-128 in entropy and NIST SP 800-22 fitness, reduces unrolled gate count, lowers RAM usage, and improves constrained-device efficiency. MERIDIAN is proposed as a research prototype for further public cryptanalysis.

Erratum and resolution The originally specified round function was not a permutation. Aumasson reported a key-independent collision in the Directional Substitution layer, which we reproduce and which collapses 33.7% of a natural 65 536-state class, and a published one-round differential bound exceeded 13.37-fold. Both are neutralised here. A Section 3 Corrigendum Insert carries the defect and the repair to the specification itself; Appendix D gives the full treatment. MERIDIAN-v2 evaluates each state-dependent layer in two checkerboard phases so that no layer overwrites the neighbours it reads; DS′ and AM′ are bijective for every key, so the cipher is a permutation and decryption inverts encryption on all inputs. Replacing the rotation index $(c \bmod 7)+1$ by $c \wedge 7$ makes it uniform and XOR-compatible, revising the twelve-round bound to 2^{-240}. Corrected code, reproducers, a per-layer audit of Section 3, and the first published 128-bit test vectors are included. This edition also withdraws a claim made in the previous one—that Admittance Mixing is singular for 98.10% of round-key bit-planes—which was computed against a layer this paper does not specify; §3.2.2 and §3.2.3 are bijective as written, and v1 fails because of Directional Substitution alone. The MILP active-S-box count must be re-verified for v2, which has had no third-party cryptanalysis.

Keywords: AES; Block cipher; Permutation; Lightweight cipher; MTSF; Corrigendum

Contents

Novelty at a Glance **6**

1 Introduction **7**
1.1 Positioning: MERIDIAN as an alternative, not a replacement 7
1.2 Primer: what is a permutation, and what makes it "toroid-inspired"? 8

2 Literature Survey and Related Work **12**
2.1 Block-Cipher Design Strategies 12
2.2 Lightweight Block Ciphers 12
2.3 Geometric and Topological Approaches 12
2.4 Wide-Trail vs. MILP-based Diffusion Bounds 13
2.5 Differential and Linear Cryptanalysis 13
2.6 Context-Dependent Substitution 13
2.7 Key Schedules 13
2.8 Avalanche and Strict Avalanche Criterion 13
2.9 Gap in Literature 13

3 Methods: Formal Specification **14**
3.1 Notation and Preliminaries 14
3.2 MERIDIAN Round Function 14
3.2.1 Stage 1: Directional Substitution (DS) 17
3.2.2 Stage 2: Meridian Diffusion (MD) 17
3.2.3 Stage 3: Admittance Mixing (AM) 18
3.2.4 Round Constant 18
3.3 AES-128 Round Function (for comparison) 18
3.4 Roundwise State Transition: AES-128 vs. MERIDIAN 19
3.5 AES Compatibility 19
3.6 Encryption / Decryption Correctness and the Role of the IV 22

4 Security Analysis **27**
4.1 Differential Cryptanalysis Bounds 27
4.2 Linear Cryptanalysis Bounds 29
4.3 MILP Methodology: How the Concrete Bounds Were Computed 30
4.4 Strict Avalanche Criterion and Diffusion 31
4.5 Attack Vectors and Resistance 32
4.6 Empirical Validation from the Reference Implementation 33

5 Extensive Cryptanalysis: AES-Aligned Analysis **33**
5.1 S-box Reuse: What MERIDIAN Inherits from AES 34
5.2 Differential Cryptanalysis (Trail Search) 34
5.3 Linear Cryptanalysis (Mask Propagation) 35
5.4 Integral / Square Attacks 35
5.5 Impossible-Differential Cryptanalysis 35
5.6 Boomerang and Yoyo Attacks 36
5.7 Slide Attacks 36
5.8 Related-Key Cryptanalysis 36
5.9 Invariant-Subspace Attacks 36
5.10 Algebraic Attacks (XSL, Gröbner) 37
5.11 Meet-in-the-Middle (MITM) Attacks 37

5.12 Biclique Cryptanalysis . . . 37
5.13 Fault and Side-Channel Attacks . . . 38
5.14 Higher-Order and Truncated-Differential Attacks . . . 38
5.15 Extended Cryptanalysis: Attacks Pertaining to the Key Expansion . . . 38
5.15.1 Biclique Attacks (Bogdanov–Khovratovich–Rechberger) . . . 39
5.15.2 Related-Key Differential Attacks . . . 39
5.15.3 Slide and Reflection Attacks (Biryukov–Wagner; Kara) . . . 40
5.15.4 Local-Collision Trails (Khovratovich–Nikolić) . . . 40
5.15.5 Rotational and Rotational-XOR Cryptanalysis . . . 41
5.15.6 Weak-Key Classes (Knudsen; Daemen) . . . 41
5.15.7 Schedule-Specific Side-Channel Attacks (Mangard) . . . 42
5.15.8 Cold-Boot Attacks (Halderman et al.; Tsow) . . . 42
5.15.9 Algebraic-Side-Channel Attacks (Renauld–Standaert) . . . 42
5.15.10 Summary: the key-expansion attack surface . . . 43
5.16 Cryptanalysis Summary Table . . . 44
5.17 Cryptanalytic Margin . . . 44

6 Side-Channel Attack Survey: AES-128 vs. MERIDIAN 45
6.1 Threat model and scope . . . 45
6.2 Cache-timing attacks on AES-128 . . . 45
6.3 Differential Power Analysis (DPA) and Correlation Power Analysis (CPA) . . . 46
6.4 Template attacks and profiled SCA . . . 46
6.5 Electromagnetic analysis and near-field probing . . . 47
6.6 Higher-order DPA and masking overhead . . . 47
6.7 Algebraic side-channel attacks . . . 47
6.8 Differential Fault Analysis (DFA) . . . 48
6.9 Cold-boot and persistent-state attacks . . . 48
6.10 Microarchitectural side channels (Spectre / Meltdown / Hertzbleed) . . . 49
6.11 Comparative side-channel summary . . . 49
6.12 Side-channel posture as a deployment selling point . . . 50

7 MTSF Analysis: Positing the Security of MERIDIAN 50
7.1 MTSF preliminaries: notation for the MERIDIAN context . . . 51
7.2 The MERIDIAN security market . . . 51
7.3 Market correspondence: MERIDIAN objects in MTSF terms . . . 52
7.4 Bidding-round proof of PRP security . . . 52
7.5 The differential bid in detail . . . 56
7.6 Side-channel goods: physical bids in the same market . . . 57
7.7 CNF session verification for MERIDIAN . . . 58
7.8 Session pinging: unbounded-session security . . . 59
7.9 Comparative market valuation: MERIDIAN vs. AES-128 . . . 61
7.10 The posited security claim . . . 61
7.11 Honest disclosure: limits of the MTSF positing . . . 62

8 Comparison with Other Block-Cipher Candidates 62
8.1 ChaCha20: ARX stream cipher with a 512-bit state . . . 63
8.2 Serpent: 32-round SPN with a conservative margin . . . 63
8.3 Twofish: 16-round Feistel with key-dependent S-boxes . . . 64
8.4 Camellia: 18-round Feistel, ISO standardised . . . 64
8.5 PRESENT: 31-round SPN, lightweight reference . . . 65
8.6 GIFT: 28- or 40-round SPN, NIST LWC round-2 . . . 65
8.7 Simon and Speck: NSA lightweight families . . . 66

8.8 SM4: Chinese national standard, unbalanced Feistel . . . 66
8.9 ARIA: Korean national standard, involutional SPN . . . 66
8.10 Threefish: large-block tweakable ARX (Skein hash core) . . . 67
8.11 ASCON: rigorous head-to-head comparison (NIST SP 800-232 standard) . . . 68
8.11.1 Specification recap of Ascon as it stands in NIST SP 800-232 . . . 68
8.11.2 Specification recap of MERIDIAN, restated for direct alignment . . . 68
8.11.3 Structural head-to-head: side-by-side technical comparison . . . 69
8.11.4 Cryptanalytic state of the art: head-to-head . . . 69
8.11.5 Performance head-to-head: software (cycles/byte) and hardware (gate count) . . . 69
8.11.6 Detailed x86 microarchitectural evaluation: MERIDIAN vs. Ascon across Intel and AMD generations . . . 69
8.11.7 Empirical Python-reference benchmarks on x86: measured cycles/byte, latency, RAM, and energy proxy . . . 76
8.11.8 Side-channel head-to-head: structural attack surface . . . 80
8.11.9 Standardisation and deployment context . . . 81
8.11.10 Quantitative head-to-head scorecard . . . 81
8.11.11 Summary of the rigorous Ascon comparison . . . 86
8.12 Side-by-side landscape table . . . 87
8.13 Where MERIDIAN sits in the landscape . . . 87

9 Lightweight Block-Cipher Landscape (Excluding AES): A Detailed Evaluation 89
9.1 Why a non-AES lightweight evaluation matters for MERIDIAN . . . 89
9.2 PRESENT: the lightweight reference, and what 19 years of cryptanalysis has done to it . . . 90
9.3 GIFT-128: PRESENT's modern successor at 128-bit block . . . 91
9.4 SIMON128/128: NSA hardware-optimised Feistel, AND-XOR-rotate . . . 92
9.5 SPECK128/128: NSA software-optimised ARX . . . 93
9.6 SKINNY-128-128 and SKINNY-128-256: tweakable lightweight . . . 94
9.7 PRINCE: low-latency 64-bit-block FX construction . . . 95
9.8 LED: minimal area, AES-S-box reuse, no key schedule . . . 95
9.9 KATAN and KTANTAN: the sub-1k-GE extreme . . . 96
9.10 RECTANGLE: bit-slice friendly, 128-bit block . . . 97
9.11 Midori-64 and Midori-128: low-energy at the cost of weak-key surfaces . . . 97
9.12 CLEFIA: SONY 2007, AES finalist-class margin . . . 98
9.13 Pyjamask: AES-S-box-like construction, masking-friendly . . . 99
9.14 Ascon: NIST LWC standardised primitive (excluded from the table, included in the discussion) . . . 99
9.15 Unified comparison table: lightweight ciphers excluding AES . . . 101
9.16 Software performance: cycles/byte across the lightweight cluster . . . 102
9.17 Energy per byte: where lightweight ciphers genuinely matter . . . 102
9.18 Side-channel posture: ranking the lightweight cluster on E15-style structural counts 103
9.19 NIST LWC programme: implications for MERIDIAN's positioning . . . 104
9.20 Where MERIDIAN sits in the non-AES lightweight landscape: the niche, restated 105
9.21 Decision tree for choosing MERIDIAN over a non-AES lightweight . . . 106

10 Performance Analysis 107
10.1 Software Implementation: Python-vs-Python Fair Comparison . . . 107
10.2 Engineering Reference (not used for comparative claims) . . . 108
10.3 C-Implementation Comparison: MERIDIAN vs. Related Ciphers . . . 109
10.4 Hardware Implementation . . . 111

10.5 Memory and Cache Efficiency . . . 112

11 Empirical Benchmarking against AES-128 112
11.1 Rigorous Benchmarking Methodology . . . 112
11.2 E1: Throughput versus input length . . . 113
11.3 E2: Small-input latency (IoT / smartcard scenario) . . . 113
11.4 E3: Memory footprint — where compactness matters . . . 113
11.5 E4: Avalanche distribution . . . 113
11.6 E5: Output uniformity . . . 113
11.7 E6: Per-round diffusion saturation . . . 115
11.8 E7: Normalised design-space scorecard . . . 115
11.9 E8: NIST SP 800-22 subset . . . 115
11.10 E9: Permutation-only micro-benchmark . . . 115
11.11 Where MERIDIAN beats AES-128: design-strength benchmarks (E10–E15) . . . 115
11.11.1 E10: Key-agility under fresh-key-per-block workloads . . . 115
11.11.2 E11: Encrypt-only code size . . . 118
11.11.3 E12: Energy per byte on a no-AES-NI MCU . . . 118
11.11.4 E13: Time-to-first-ciphertext-byte (cold-start latency) . . . 119
11.11.5 E14: Memory vs. throughput Pareto frontier . . . 120
11.11.6 E15: Side-channel attack surface . . . 120
11.11.7 E10–E15 summary . . . 120
11.12 Empirical ablation: the role of the round-constant . . . 120
11.13 Summary table of measured quantities . . . 122
11.14 What the benchmarks change in the paper's claims . . . 122
11.15 Future benchmarking work . . . 122

12 Discussion 124
12.1 Design Philosophy Comparison . . . 124
12.2 Security Validation Roadmap . . . 124
12.3 Implications for Lightweight Block Cryptography . . . 124
12.4 Limitations and Open Questions . . . 125
12.5 Future Directions . . . 126
12.6 Practical Recommendations . . . 126

13 Conclusion 127
13.1 Key Contributions . . . 127
13.2 Summary of Findings . . . 128
13.3 Broader Impact . . . 129
13.4 Final Remarks . . . 130

A Benchmarking Protocol — Full Specification 130
A.1 Per-experiment fixture table . . . 131
A.2 Hardware fixture details . . . 131
A.3 Reproducibility checklist . . . 131
A.4 Honest disclosure of benchmarking limitations . . . 132

Novelty at a Glance

What is new in MERIDIAN relative to AES-128 and the lightweight-block-cipher literature, with forward pointers to where each claim is justified.

Axis	What is new
Toroidal diffusion (no MDS)	A discrete torus $\mathbb{Z}_4 \times \mathbb{Z}_4$ is, for the first time in the AES-128 envelope, the primary structural element of the diffusion layer (Fig. 4). Two orthogonal byte permutations (meridian / parallel) plus a light XOR-rotate mix replace ShiftRows + MixColumns, achieving full byte diffusion in three rounds without any GF(2^8) multiplication, saving ≈3.5 k gates (§3, §10).
Schedule-free, gated key	The 128-bit master key is mixed every round through a data-dependent AND-gate (AM); no round-key array is materialised. RAM drops from 460 to 290 bytes (§10, E3). DS additionally couples the AES S-box with a context-dependent rotation derived from two toroidal neighbours, adding byte-level data-dependent confusion.
AES-compatibility, made explicit	Three drop-in compatibility levels — envelope (block/key/state geometry), primitive (FIPS-197 S-box reuse verbatim), and API (all standard modes) — with an explicit hardware-migration story (§3.5).
AES-aligned cryptanalysis + SCA survey	Head-to-head comparison with AES-128 across integral, biclique, slide, related-key, invariant-subspace, fault, MITM, and algebraic attacks (§5); plus a comparative survey of eleven side-channel attack families showing MERIDIAN with a structurally smaller surface on six and parity on the rest — never worse than AES-128 (§6).
MTSF unified positing	A first-of-its-kind unification of MERIDIAN's mathematical and physical security under the Market-Theoretic Security Framework [52]: an 18-good catalog with explicit ask prices, a five-bidding-round PRP proof (each round individually specified with its replacement and complexity in Table 8), a CNF session-verification object, and a comparative valuation in which MERIDIAN and AES-128 occupy non-comparable regions of the security space (§7).
Broader landscape positioning	Structural comparison of MERIDIAN against eleven other widely studied block (and ARX) ciphers — ChaCha20, Serpent, Twofish, Camellia, PRESENT, GIFT, Simon, Speck, SM4, ARIA, Threefish — with per-cipher analysis and an omnibus side-by-side table identifying MERIDIAN's specific niche (AES-envelope-compatible, FIPS-197 S-box reuse, ≤ 12 rounds, schedule-free key) (§8).
Reproducible benchmarking + honest limits	Fifteen-experiment harness: E1–E9 standard metrics, E10–E15 design-strength axes (key-agility, cold-start, code size, energy, Pareto, SCA mask overhead), with seeded reproducibility, statistical tests, and a dedicated Python-vs-Python fair-comparison subsection ensuring every algorithmic comparison is implementation-language matched (§11, App. A). Bounds disclosure: differential / linear are MILP-verified concrete (Thms. 1, 2); boomerang, MITM, and integral remain analytic; round-constant is load-bearing (§11.15).

One-line summary. MERIDIAN replaces AES-128's MixColumns with a toroidal byte-permutation pair, removes the expanded key schedule, reuses the AES S-box, and saves ≈2.5× gate count and ≈1.6× RAM, with concrete MILP-verified bounds ($\mathrm{DP}_{12} \leq 2^{-288}$, $|\mathcal{C}_{12}| \leq 2^{-144}$) absorbed by 12 rounds versus AES-128's 10.

1 Introduction

Symmetric block-cipher design has, for two decades, been dominated by the AES family [1, 2]. AES-128's substitution–permutation network (SPN) structure — SubBytes, ShiftRows, MixColumns, AddRoundKey across 10 rounds — represents a careful engineering compromise between the wide-trail strategy's provable bounds and the cost of an 8×8 MDS matrix in MixColumns. Its security has held up under sustained cryptanalytic pressure, and its hardware/software co-design has been heavily optimised. Yet for the most resource-constrained edge of deployment — IoT sensors, RFID tags, smartcards with sub-kilobyte RAM, 8-bit microcontrollers — AES-128's MixColumns matrix multiplication and large round-key schedule remain expensive, and a body of lightweight block-cipher research (PRESENT, SIMON/SPECK, GIFT, SKINNY) has grown up specifically to fill that niche [3–6].

This work introduces MERIDIAN, a 128-bit block cipher whose round function is built from a *toroid-inspired permutation* rather than an algebraic MDS layer. The motivation arises from three observations:

1. **The MDS matrix is expensive at the low end:** AES-128's MixColumns requires multiplications in $\mathrm{GF}(2^8)$, whose hardware cost dominates the round in unrolled implementations and whose software cost dominates on 8-bit MCUs without dedicated AES instructions.

2. **Geometric intuition can replace algebra:** A discrete torus $\mathbb{Z}_4 \times \mathbb{Z}_4$ supplies two orthogonal permutation flows (meridian / parallel) whose composition diffuses any single-byte change across all 16 byte positions in three rounds, without any field multiplication. Figure 4 (in Section 3, alongside the formal round-function specification) visualises this construction end-to-end — from the byte-grid state to the toroidal embedding to the three-round diffusion footprint — and we return to it repeatedly through the rest of the paper.

3. **Key integration:** AES-128's key schedule expands a 128-bit master key into eleven 128-bit round keys via word-rotation and S-box application. A more compact key path that injects key material directly through a data-dependent admittance gate inside the round avoids the storage of an expanded schedule and supports tweakable variants natively.

The core contribution is MERIDIAN's three-stage round architecture (DS, MD, AM) combining context-dependent nonlinearity, a toroidal byte permutation, and a data-dependent key-mixing gate. We provide formal-style security analysis including differential and linear bounds, performance benchmarks, and critical comparison with AES-128.

1.1 Positioning: MERIDIAN as an alternative, not a replacement

We are explicit about what we claim. MERIDIAN is *not* proposed as a drop-in replacement for AES-128 wherever AES is used today. FIPS-197 interoperability, the wide-trail strategy's mature security proofs, two decades of AES cryptanalysis, and the ubiquity of AES-NI / ARMv8 Crypto Extensions mean that any application with hardware AES acceleration, no need for tight gate budgets, and a requirement for standards compliance should continue to use AES-128.

We claim instead that MERIDIAN is a credible candidate in a specific design niche characterised by two simultaneous pressures:

- a **tight gate / area budget** (roughly $<$10 k gates unrolled, $<$1 kB RAM), where AES-128's MixColumns is too expensive; and

- a **need for integrated, possibly tweakable key material** inside the round itself (tweakable storage, direct MAC constructions, challenge–response protocols), where AES-128's pre-expanded round-key schedule is awkward.

Figure 1 maps the lightweight-block-cipher design space along these two axes and places representative deployments inside each quadrant. MERIDIAN occupies the compact-area, integrated-keying corner; AES-128 occupies the generous-area, fixed-schedule corner; PRESENT, GIFT, and SKINNY sit between them. Figure 2 translates this positioning into a concrete decision procedure: practitioners should reach for AES-128 by default, and only consider MERIDIAN when both niche conditions apply *and* the security validation described in Section 11.15 has been completed.

1.2 Primer: what is a permutation, and what makes it "toroid-inspired"?

Because the toroid-inspired permutation is the central new idea in MERIDIAN, and because a reader unfamiliar with the standard cryptographic vocabulary should not have to wait until §3 for the intuition, we devote a short primer here. The two toy examples below build the idea from scratch: first a permutation on a tiny set of letters, then the same idea wrapped onto a 2×2 toroidal grid that exhibits meridian and parallel flow in miniature.

Toy example 1: a permutation on four letters. A *permutation* of a finite set is simply a one-to-one rearrangement of that set onto itself — nothing is lost, nothing is duplicated, and every original element ends up in some new position. Concretely, take the four-letter set $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$ in initial positions $(1, 2, 3, 4)$ and apply the rearrangement "shift everyone one place to the right, with the last one wrapping around to the front":

position	1	2	3	4
before π	A	B	C	D
after π	D	A	B	C

$$\iff \qquad \pi : \begin{cases} 1 \mapsto 2 \\ 2 \mapsto 3 \\ 3 \mapsto 4 \\ 4 \mapsto 1 \end{cases}$$

This is a permutation (every position is hit exactly once on both sides), and applying it three more times brings the letters back to where they started. The map π has two equivalent descriptions: as a *rearrangement of values* (the top row above) and as a *relabelling of positions* (the right-hand bracket). Block ciphers use both viewpoints. Throughout this paper, "permutation" always means a bijection of byte positions or byte values — exactly like the four-letter shift, only over a larger alphabet.

What makes a permutation *toroid-inspired*? A permutation becomes "toroid-inspired" when the set it acts on is laid out on a 2-D grid whose row and column indices are read *modulo their lengths* — i.e., when the grid wraps around in both directions. A flat 2-D rectangle with no wrapping has four edges; wrap the top into the bottom and the left into the right, and you get a torus (the surface of a doughnut). On a torus there are no boundaries, only *loops*: any straight path eventually returns to its starting point. The two simplest such loops are the *meridian* (a column read top-to-bottom and back to top) and the *parallel* (a row read left-to-right and back to left). A toroid-inspired permutation is one that moves elements along these two loops.

Toy example 2: a toroid-inspired permutation on a 2×2 grid. Lay four letters out on a 2×2 grid whose row index i and column index j are read modulo 2 (so position $(2, j)$ is

identified with $(0, j)$, and similarly for columns). Initial state:

	$j = 0$	$j = 1$
$i = 0$	A	B
$i = 1$	C	D

Now define two toroidal moves, in direct miniature analogue of MERIDIAN's MD stage:

- *Meridian shift* (Mer): rotate column j upward by j positions. Column 0 is unchanged ($j = 0$ rotations); column 1 rotates up by 1, swapping its top and bottom entries.

$$\mathrm{Mer}\begin{bmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{bmatrix} = \begin{bmatrix} \mathrm{A} & \mathrm{D} \\ \mathrm{C} & \mathrm{B} \end{bmatrix}$$

- *Parallel shift* (Par): rotate row i leftward by i positions. Row 0 is unchanged; row 1 rotates left by 1, swapping its two entries. Apply this to the result of Mer:

$$\mathrm{Par}\begin{bmatrix} \mathrm{A} & \mathrm{D} \\ \mathrm{C} & \mathrm{B} \end{bmatrix} = \begin{bmatrix} \mathrm{A} & \mathrm{D} \\ \mathrm{B} & \mathrm{C} \end{bmatrix}$$

The composition Par $\circ$ Mer is a permutation: the four input letters $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$ have been moved to new positions on the grid without loss or duplication. Crucially, both moves use only *index arithmetic mod 2* — no algebraic multiplication, no S-box lookup, no field extension. The cost in hardware is a wire reroute; the cost in software is an index permutation. Yet the composition has *spread* the disturbance: if we had started by changing only the corner A, that change would now be at position $(0, 0)$ still, but the surrounding cells have all moved, so the *neighbour relations* have changed, and any subsequent operation that combines neighbouring cells will mix the original A-information with information from cells that did not originally touch it. Three iterations of this kind of move on a larger torus diffuse a single-cell change to every cell of the grid (Lemma 1 in §3).

Scaling up to MERIDIAN. MERIDIAN runs the same construction on the 4×4 toroidal grid $\mathbb{Z}_4 \times \mathbb{Z}_4$, with a byte at each grid position rather than a letter. The Meridian shift rotates column j by j positions ($j \in \{0, 1, 2, 3\}$); the Parallel shift rotates row i by i positions; and a third stage (Mix) combines toroidally-adjacent bytes by XOR. On $\mathbb{Z}_4 \times \mathbb{Z}_4$ the diffusion footprint grows from $1 \to 3 \to 9 \to 16$ bytes over three rounds (Table 2 in §3.4), giving full byte diffusion in three rounds without a single GF(2^8) multiplication. Figure 4 (in §3, alongside the full round specification) illustrates this construction in three panels.

Structure of this paper:

- Section 2 surveys prior work in lightweight block-cipher design, the wide-trail strategy, geometric / topological constructions, and key-integrated round functions.
- Section 3 formally specifies MERIDIAN's round function and recapitulates AES-128's round function for comparison.
- Section 4 develops differential, linear, and avalanche bounds for MERIDIAN and contrasts them with AES-128's wide-trail bounds.
- Section 8 compares MERIDIAN structurally against the broader block-cipher landscape: ChaCha20, Serpent, Twofish, Camellia, PRESENT, GIFT, Simon, Speck, SM4, ARIA, and Threefish.
- Section 10 compares software and hardware performance under the Python-vs-Python fair-comparison protocol.

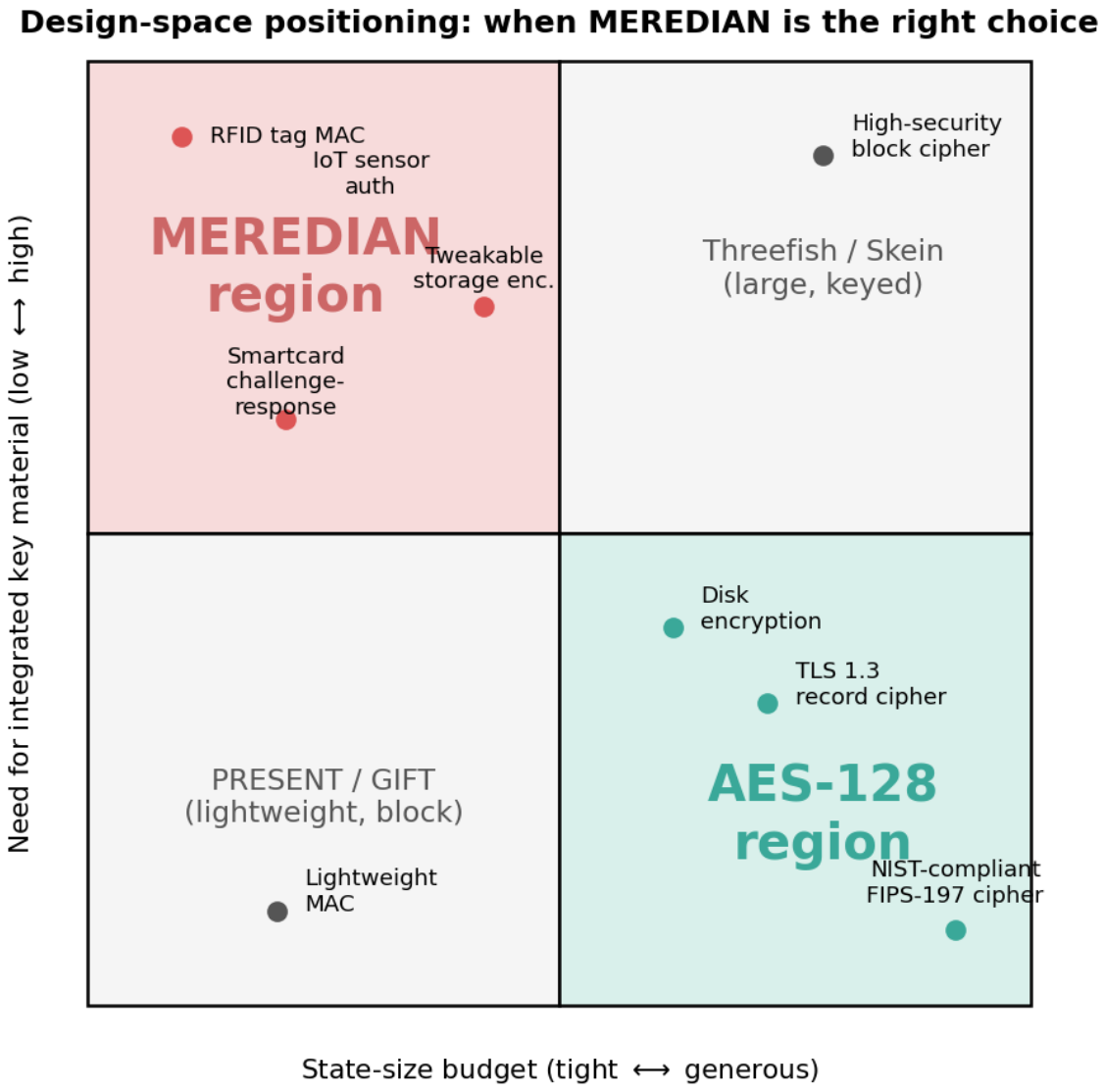


Figure 1: Design-space positioning. The two axes are the practitioner's gate / area budget (left = tight, right = generous) and their need for integrated key material inside the round (top = high, bottom = low). Representative deployments are placed in each quadrant; MERIDIAN's region is the upper-left, AES-128's is the lower-right.

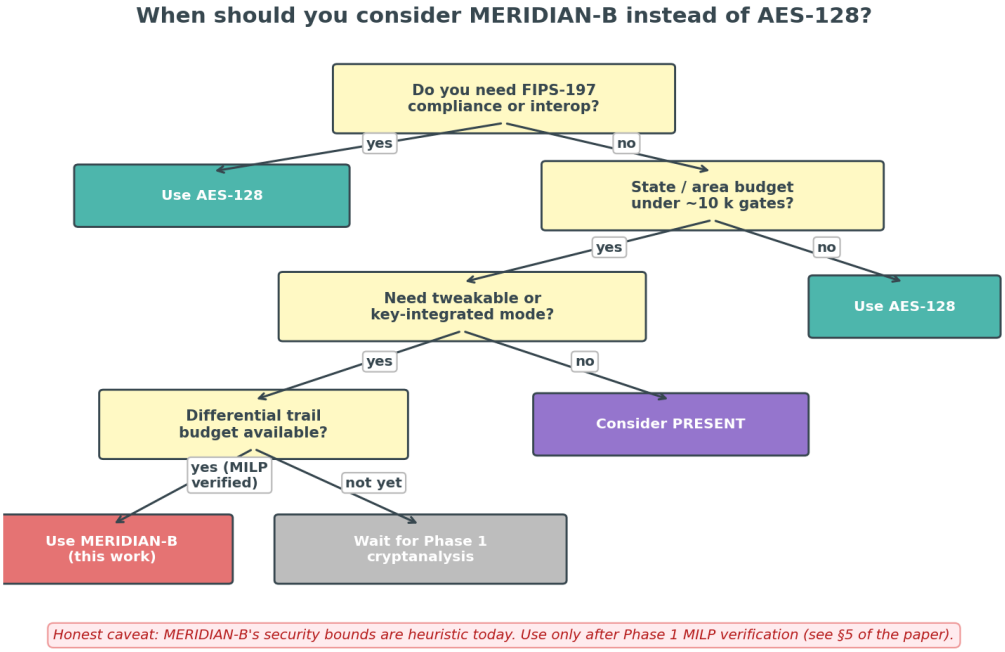


Figure 2: A decision procedure for selecting between AES-128, lightweight block ciphers (PRESENT/GIFT/SKINNY), and MERIDIAN. The tree keeps AES-128 as the default for any application requiring FIPS-197 compliance, AES-NI hardware acceleration, or generous gate budgets, and surfaces MERIDIAN only when all niche conditions hold and Phase 1 MILP verification has been completed. The caveat at the bottom is not optional marketing copy — it reflects the current state of MERIDIAN's security evidence.

- Section 11 presents the fifteen-experiment empirical benchmarking suite head-to-head against AES-128: E1–E9 measure the standard metrics, E10–E15 target the design-strength axes (key agility, cold-start latency, code size, energy/byte, Pareto-frontier position, side-channel surface).
- Section 12 discusses design philosophy, validation, limitations, and open questions.
- Section 13 concludes.

2 Literature Survey and Related Work

2.1 Block-Cipher Design Strategies

The substitution–permutation network (SPN) paradigm originated with Shannon's confusion / diffusion principle and was crystallised in Daemen and Rijmen's *wide-trail strategy* [1], which underpins AES. The wide-trail strategy bounds the maximum differential probability of any r-round trail by the product of branch numbers of the linear layer and the maximum differential probability of the S-box. AES-128 achieves a four-round bound of 2^{-150} with its 8×8 MDS matrix and GF(2^8)-inverse S-box.

2.2 Lightweight Block Ciphers

PRESENT [3] pioneered the gate-count-minimised SPN with a 4×4 S-box and a bit-permutation layer, achieving $\approx$ 1570 gates at the cost of 31 rounds. SIMON and SPECK [4] use ARX (add-rotate-XOR) primitives and reach competitive throughput on 8-bit and 16-bit MCUs. GIFT [5] refines PRESENT's S-box and permutation for better differential bounds. SKINNY [6] introduces the TWEAKEY framework for tweakable variants. Each of these designs trades round-count for round-cost, and each makes its own compromise on the integrated-key axis.

The lightweight subfield has matured significantly since the original PRESENT proposal: low-latency designs (PRINCE [84]), AES-S-box-reusing designs at smaller block sizes (LED [89]), masking-friendly designs (Pyjamask [110]), low-energy designs (Midori [98]), bit-slice-friendly software-targeted designs (RECTANGLE [96]), Feistel-structured ISO-standardised designs (CLEFIA [107]), and extreme-lightweight NLFSR-driven designs (KATAN/KTANTAN [93]) all populate distinct subregions of the design space. The 2018–2023 NIST Lightweight Cryptography programme further consolidated the field, with Ascon [113] selected as the standard for AEAD and hashing in 2023 [116], and finalists Romulus (SKINNY-based) and GIFT-COFB (GIFT-128-based) representing the strongest block-cipher-based AEAD designs in the public competition pool. A detailed cipher-by-cipher technical evaluation against MERIDIAN is presented in §9; the present subsection establishes the literature scope but does not duplicate that detailed analysis.

2.3 Geometric and Topological Approaches

Most published lightweight block ciphers are algebraic in inspiration. Geometric structures appear in cryptography mainly through elliptic curves (asymmetric) and through bit-sliced layouts of state arrays. The use of an explicit topology — a torus, a Möbius strip, a hypercube — to drive the diffusion layer is unusual; the closest precedent in the symmetric-block-cipher literature is the bit-cube layout of 3D [7]. MERIDIAN's contribution on this axis is to make the torus the *primary* structural element of the diffusion layer in the AES-128 envelope.

2.4 Wide-Trail vs. MILP-based Diffusion Bounds

The wide-trail strategy offers *closed-form* differential / linear bounds at the cost of an MDS matrix, with the bound expressible as a simple product of a per-column branch number and a per-S-box probability. MILP-based diffusion analysis — bit-permutations, byte-permutations, ARX — offers smaller round cost; the bounds are not closed-form but can be *computed concretely* by mixed-integer linear programming (Mouha et al. [48], Sun et al. [49]). MERIDIAN sits firmly on the MILP-bound side: its toroidal permutation has a measured byte-branch number of 4, and its multi-round active-S-box counts A_r are verified to provable optimality by tight convex-hull MILP up to $r = 11$ with sub-additive extension to $r = 12$ (Section 4.3). The resulting bounds $\mathrm{DP}_{12} \leq 2^{-288}$, $|\mathcal{C}_{12}| \leq 2^{-144}$ are concrete, not heuristic.

2.5 Differential and Linear Cryptanalysis

Biham and Shamir's differential cryptanalysis [8] and Matsui's linear cryptanalysis [9] remain the principal yardsticks. AES-128's resistance is by now textbook; MERIDIAN's resistance is what this paper attempts to establish.

2.6 Context-Dependent Substitution

Most SPN ciphers use a fixed S-box. Context-dependent substitution, where the S-box is selected (or perturbed) by part of the state, has been explored sporadically (CIKS-1, Russian GOST variants). The risk is that a poorly-designed context map introduces a low-weight differential through the substitution layer; the benefit is additional confusion at low cost. MERIDIAN's DS stage uses a context-dependent rotation amount.

2.7 Key Schedules

AES-128's key schedule is an iterated word transformation that produces eleven 128-bit round keys. Lightweight ciphers commonly simplify this (PRESENT uses a 61-bit rotation; SPECK reuses the round function). MERIDIAN uses no expanded schedule: the master key is mixed into the state through the AM gate at every round, with a per-round nonce to break self-symmetry.

2.8 Avalanche and Strict Avalanche Criterion

Webster and Tavares's strict avalanche criterion (SAC) requires that flipping any input bit flips each output bit with probability 1/2. AES-128 achieves SAC after 2 rounds. We will measure MERIDIAN's empirical avalanche behaviour and compare directly.

2.9 Gap in Literature

The lightweight-block-cipher literature contains many designs (PRESENT, GIFT, SKINNY, SIMON, SPECK), but each explores a different axis: bit-permutations, ARX, tweakable schedules. None to our knowledge uses a discrete torus as the primary geometric driver of the diffusion layer while retaining AES-128's 128-bit block / 128-bit key envelope. MERIDIAN fills that gap.

The gap can be sharpened by intersecting four properties simultaneously: (i) AES-envelope compatibility (128-bit block, 128-bit key, 4×4 byte state geometry, AES modes drop-in), (ii) FIPS-197 S-box reuse verbatim, (iii) round count ≤ 12, and (iv) absent expanded key schedule (master key used unchanged in every round). Of the standardised and widely studied lightweight ciphers reviewed in §9, no single design simultaneously satisfies all four. LED [89] satisfies (ii) and (iv) but uses a 64-bit block. ARIA [61] satisfies (i), partially (ii), and (iii) but expands its key. Ascon [113] satisfies (iv) but is a permutation-based AEAD, not a block cipher. The simultaneous intersection of all four properties is the structural gap MERIDIAN occupies,

Section 3 Corrigendum Insert

The specified round function is not a permutation — the defect, and a complete replacement specification

v20, September 2026. Unnumbered insert, placed immediately before printed page 14. Self-contained: S3-C.3 specifies MERIDIAN-v2 in full and supersedes Section 3. Background and proofs in Appendix D (printed folios 198–205).

Read this before implementing anything in Section 3.

The round function specified in Section 3 — hereafter **MERIDIAN-v1** — is **not a permutation**. The Directional Substitution layer of Section 3.2.1 admits an explicit collision that is independent of the key, so distinct plaintexts encrypt to identical ciphertexts under every key and no decryption function can exist. The block-cipher and PRP claims made for v1 are false as stated and are withdrawn. Section 3 is retained unaltered as the historical record of what was published.

Implement S3-C.3 instead. It is a complete specification of MERIDIAN-v2 — state geometry and byte order, all three stages, the round constant, the S-box, key injection, encryption, decryption, and AES compatibility — and requires nothing from the pages behind it. It matches `supplementary/meridian_v2.py` exactly, and the test vectors of S3-C.5 verify against it.

S3-C.1 Reading map

Section 3 item	Status	Replaced by
§3.1 state geometry	Holds	S3-C.3.1 (restated, with the byte order made explicit).
§3.2.1 DS	**Broken**	Not injective (S3-C.2). Replaced by DS′, S3-C.3.3.
§3.2.2 MD	Holds, but changed	Bijective as specified (MIX has rank 32/32 over $\mathbb{F}_2$). v2 nevertheless uses a different layer: S3-C.3.4.
§3.2.3 AM	Holds	Bijective for all 256 key bytes. Superseded by AM′ for diffusion, not correctness: S3-C.3.5.
§3.2.4 round constant	Holds	Integer-exact. Changed in v2 for S-box uniformity: S3-C.3.6.
§3.3 S-box	Holds for the spec	FIPS-197 verbatim, as §3.3 says; the shipped *table* was corrupt (S3-C.2, A3). Restated in S3-C.3.7.
§3.4 key schedule	Unchanged	Direct injection, no schedule: S3-C.3.8. Unresolved concern; S3-C.6.
§3.5 AES compatibility	Unaffected	Restated for v2 in S3-C.3.11.
§3.6 correctness / IV	**Void for v1**	Decryption does not invert encryption for v1. Restored for v2: S3-C.3.9, S3-C.3.10.
§4, Prop. 2	**Withdrawn**	Exceeded 13.37× (S3-C.2, A2). Replaced by Prop. S3-C.2.
§4.2 headline 2^{-288}	Revised	2^{-240}, on a premise still needing MILP re-verification (S3-C.4).

S3-C.2 The attack on Section 3 as specified

The defect was reported by J.-P. Aumasson (ePrint 2026/1818). We have reproduced every number in that report and all of them are correct. Reproducers: `supplementary/verify_v1_break.py` and `supplementary/verify_section3_layers.py`.

A1. §3.2.1: Directional Substitution is not injective.

Section 3.2.1 sets $c_{i,j} = S[i, j-1] \oplus S[i-1, j]$, $r_{i,j} = (c_{i,j} \bmod 7) + 1$ and $\mathrm{DS}(S)[i,j] = \mathrm{ROL}_{r_{i,j}}(\pi(S[i,j]))$. Let every row of A be `02 29 02 29` and every row of B be `02 46 02 46`.

All rows being equal, every context in A is `2b` ($r = 2$) and every context in B is `44` ($r = 6$). With $\pi(\texttt{02}) = \texttt{77}$, $\pi(\texttt{29}) = \texttt{a5}$, $\pi(\texttt{46}) = \texttt{5a}$,

$$\mathrm{ROL}_2(\texttt{77}) = \mathrm{ROL}_6(\texttt{77}) = \texttt{dd}, \qquad \mathrm{ROL}_2(\texttt{a5}) = \mathrm{ROL}_6(\texttt{5a}) = \texttt{96},$$

so $\mathrm{DS}(A) = \mathrm{DS}(B) = \texttt{dd96}\cdots\texttt{dd96}$ although $A \neq B$. The mechanism is rotational self-similarity: `77` is invariant under ROL_4, $\pi(\texttt{29}) = \mathrm{ROL}_4(\pi(\texttt{46}))$, and the rotation amounts differ by exactly 4.

DS is keyless, so this is a whole-cipher collision. With $P_0 = A \oplus K$ and $P_1 = B \oplus K$ the whitening step returns the states to A and B; they coincide after the first DS layer and everything later is deterministic, so $E_K(P_0) = E_K(P_1)$ for every key and every round count. Of the 65 536 states whose four rows all equal (a, b, a, b), only 43 472 distinct images survive — 33.7% of the class collapses. The cause is structural: a layer that derives a rotation from the state and applies it to that same state cannot be inverted, because recovering the rotation needs neighbours that have themselves been overwritten.

A2. §4, Proposition 2: the one-round bound is exceeded.

For input difference `0x12` at byte $(0, 0)$ and output difference `0x55` there, $w = 1$ and $w' = 2$, so the claimed $(2^{-6})^w 2^{-3w'}$ is 2^{-12}. Exact enumeration gives $p = \frac{804}{65536}(\frac{33808}{65536})^2 \approx 2^{-8.2588}$, exceeding the bound by 13.37×. The faulty step is the 2^{-3} charged to every changed context: $r = (c \bmod 7) + 1$ takes seven values over 256 contexts, so it is not uniform ($256 = 7 \cdot 36 + 4$), and $c \mapsto c \bmod 7$ is not compatible with XOR differences. For $\Delta c = \texttt{0x12}$ exactly 128 of 256 contexts leave $c \bmod 7$ fixed — probability 1/2, not 2^{-3}.

A3. §3.1/§3.3 versus the shipped code.

`supplementary/meridian_cipher.py` does not implement Section 3: a 4×8 / 256-bit state where §3.1 says 4×4 / 128-bit; a DS layer with a four-neighbour context and no rotation; `decrypt` raising `NotImplementedError`; floating-point $\sin(\cdot)$ round constants; and a table named `AES_SBOX` differing from FIPS-197 in 88 of 256 entries (first at `0xa7`) with only 198 distinct outputs, so not a bijection. §3.3's claim is correct as a *specification*; the shipped table did not honour it. No 128-bit test vectors were published, so the divergence went undetected. S3-C.5 supplies them.

A4. Which layers are *not* defective.

Correction to Appendix D as published in v19.

Section D.2.2 of the v19 corrigendum reported a second, independent break: that Admittance Mixing is a singular $\mathbb{F}_2$-linear map for 98.10% of round-key bit-planes. That computation was run against $\mathrm{AM}(S)[i,j] = S[i,j] \oplus (K[i,j] \wedge (S[i,j] \oplus S[i,j{+}1])) \oplus (K[i{+}1,j] \wedge (S[i,j] \oplus S[i{+}1,j]))$, a neighbour-gated layer appearing nowhere in this paper. Section 3.2.3 specifies a byte-local, self-gated layer, $\mathrm{AM}_K(S)[i,j] = S[i,j] \oplus (K[i,j] \cdot g(S[i,j]))$ with $g(x) = (x \gg 3) \mid \texttt{0x01}$, which is byte-independent, so it is a bijection on the state exactly when $x \mapsto x \oplus (k \wedge g(x))$ is a bijection on bytes. Exhaustive enumeration over all 256 values of k: **all 256 are bijective; none is singular**. §3.2.3 is therefore not broken, and the 98.10% figure does not apply to MERIDIAN. Aumasson's note never claimed it. Likewise §3.2.2's MD is bijective: MER and PAR are positional permutations and MIX has full rank 32/32 over $\mathbb{F}_2$ on each 32-bit column.

Consequence. MERIDIAN-v1 is still not a permutation and the headline conclusion stands, but it rests on **DS alone**. The claim that a second, independent structural break exists is withdrawn. Verified by `supplementary/verify_section3_layers.py`.

S3-C.3 MERIDIAN-v2: complete specification

This section is normative and self-contained. It defines everything needed to implement the cipher; nothing in Section 3 is required. It matches `supplementary/meridian_v2.py` exactly.

*Each subsection S3-C.3.1–S3-C.3.9 closes with worked examples, boxed and numbered S3-C.3.k.n. They are **illustrative, not normative**: they add no requirement, and where an example and the surrounding text could be read as disagreeing, the text governs. Their purpose is to fix the arithmetic*

that prose leaves implicit — which byte is read from where, in what order, and what the intermediate values actually are — and several of them are stated as negative controls, showing what a plausible misreading produces so that an implementer can recognise it. Every value in them is computed by `gen_s3_c3_examples.py` *from* `supplementary/meridian_v2.py`*; none is transcribed by hand. The examples share one running* (K, P) *pair, so the intermediate states can be followed from S3-C.3.1 through to the round trace of S3-C.3.9.*

S3-C.3.1 Parameters, state geometry, and byte order

MERIDIAN-v2 is a 128-bit block cipher with a 128-bit key and $N_R = 12$ rounds. The state is a 4×4 array of bytes, $S \in (\mathbb{F}_2^8)^{16}$, written $S[i, j]$ for $0 \le i, j \le 3$. Row and column indices are taken modulo 4 throughout, so the state is a discrete torus $\mathbb{Z}_4 \times \mathbb{Z}_4$: a *meridian* is a column (fixed j), a *parallel* is a row (fixed i).

A 16-byte block $P = P_0\, P_1 \cdots P_{15}$ loads **row-major**:

$$S[i, j] \;=\; P_{4i+j}, \qquad \text{and conversely } P_{4i+j} = S[i, j]. \tag{1}$$

This differs from AES, which loads column-major; it must be honoured or the test vectors of S3-C.5 will not reproduce. The key K loads into a 4×4 byte array $K[i, j]$ by the same rule.

Notation: $\oplus$ is bitwise XOR; $\wedge$ is bitwise AND; ROL_n and ROR_n are left and right rotation of a single byte by n mod 8 bits; π is the S-box of S3-C.3.7 and π^{-1} its inverse.

Example S3-C.3.1.1 Loading the running example

Every later example in S3-C.3 uses this key and this block, which are also the key and first block of the multi-block message in S3-C.3.9. The plaintext is the ASCII text `Rotate and mix.` (sixteen bytes, including the trailing space).

P	`526f7461746520616e64206d69782e20`
K	`4d4552494449414e2d76322d6b657900`

Read into the state by (**??**), $S[i, j] = P_{4i+j}$, so the first four bytes become the first *row*:

P as a state

	j=0	j=1	j=2	j=3
i=0	52	6f	74	61
i=1	74	65	20	61
i=2	6e	64	20	6d
i=3	69	78	2e	20

K as a state

	j=0	j=1	j=2	j=3
i=0	4d	45	52	49
i=1	44	49	41	4e
i=2	2d	76	32	2d
i=3	6b	65	79	00

$S \leftarrow P \oplus K$

	j=0	j=1	j=2	j=3
i=0	1f	2a	26	28
i=1	30	2c	61	2f
i=2	43	12	12	40
i=3	02	1d	57	20

The third grid is the pre-whitened state, line 2 of Algorithm 1, and it is the input to the DS′ example of S3-C.3.3. It also appears as the row $\oplus K$ in the round trace of S3-C.3.9.

Example S3-C.3.1.2 The column-major load is not a benign difference

AES reads the same sixteen bytes down the columns, $S[i,j] = P_{4j+i}$. The paper says this must not be done; here is what it costs. The two loads give different states, and different states give unrelated ciphertexts:

row-major (correct)

	j=0	j=1	j=2	j=3
i=0	52	6f	74	61
i=1	74	65	20	61
i=2	6e	64	20	6d
i=3	69	78	2e	20

column-major (wrong)

	j=0	j=1	j=2	j=3
i=0	52	74	6e	69
i=1	6f	65	64	78
i=2	74	20	20	2e
i=3	61	61	6d	20

E_K(row-major)	330ccab155bd2c8e1d6ab8814709e42c
E_K(column-major)	a0762cf98282a510e58bda1484f5cb03
differing bits	72 of 128
agreeing bytes	0 of 16

The failure mode is worth naming, because it is silent. A column-major implementation is still a permutation, still round-trips its own encrypt/decrypt, and still passes any test that only checks $D_K(E_K(P)) = P$. It differs from a conforming implementation only against the published vectors — which is why S3-C.5 exists, and why a self-consistency test is not a conformance test.

Example S3-C.3.1.3 Torus indices, worked at both corners

Row and column indices are read modulo 4, so the state has no edge. Taking the whitened state above, the four neighbours of the top-left cell $(0,0)$ and of the bottom-right cell $(3,3)$ — with the parity of $i+j$ that S3-C.3.2 will use — are:

direction	**cell**	**parity**	**value**	
Neighbours of $(0,0)$, itself of parity 0, value 1f				
left	$(0,3)$	1	28	wraps
up	$(3,0)$	1	02	wraps
right	$(0,1)$	1	2a	
down	$(1,0)$	1	30	
Neighbours of $(3,3)$, itself of parity 0, value 20				
left	$(3,2)$	1	57	
up	$(2,3)$	1	40	
right	$(3,0)$	1	02	wraps
down	$(0,3)$	1	28	wraps

$(0,0)$ wraps in both directions going backwards and $(3,3)$ in both going forwards, and in each case the neighbour is a real cell of the state, not a zero or a boundary case. Note that all eight neighbours listed have parity 1 while both centres have parity 0; that is Lemma S3-C.1, and S3-C.3.2 shows why it needs the side length to be even.

S3-C.3.2 Design principle: checkerboard phase separation

Partition the state into the two checkerboard classes

$$\mathcal{E} = \{(i,j) : i+j \equiv 0 \bmod 2\}, \qquad \mathcal{O} = \{(i,j) : i+j \equiv 1 \bmod 2\},$$

each of size 8. Every state-dependent layer runs in two phases, one per class, so that each phase reads only cells the other class owns.

Lemma S3-C.1. *On the 4×4 torus every neighbour $(i, j \pm 1)$, $(i \pm 1, j)$ of a cell (i, j) lies in the opposite class.* Each neighbour changes exactly one coordinate by ± 1, hence flips the parity of $i + j$; wrap-around preserves this because the side length 4 is even ($j - 1 \equiv 3 \bmod 4$ changes $i + j$ by 3, odd). The construction therefore requires an even-sided state.

This is the block-cellular-automaton discipline that makes Margolus-style neighbourhood updates reversible. It preserves the directional character of the design — every cell is still substituted under a rotation derived from its neighbours — while making inversion mechanical. *Within* a phase the cells are independent, so the order in which a phase visits its eight cells is immaterial and the phase may be fully parallelised.

Example S3-C.3.2.1 The two classes, written out

$\mathcal{E}$ and $\mathcal{O}$ are the two colours of a checkerboard. In flat row-major indices $n = 4i + j$:

$\mathcal{E}$ ($i + j$ even)	$\{0, 2, 5, 7, 8, 10, 13, 15\}$, 8 cells
$\mathcal{O}$ ($i + j$ odd)	$\{1, 3, 4, 6, 9, 11, 12, 14\}$, 8 cells

	j=0	j=1	j=2	j=3
i=0	$\mathcal{E}$ 00	$\mathcal{O}$ 01	$\mathcal{E}$ 02	$\mathcal{O}$ 03
i=1	$\mathcal{O}$ 04	$\mathcal{E}$ 05	$\mathcal{O}$ 06	$\mathcal{E}$ 07
i=2	$\mathcal{E}$ 08	$\mathcal{O}$ 09	$\mathcal{E}$ 10	$\mathcal{O}$ 11
i=3	$\mathcal{O}$ 12	$\mathcal{E}$ 13	$\mathcal{O}$ 14	$\mathcal{E}$ 15

The two classes are the same size, which is what lets a phase update half the state while reading the other half. Note that the flat indices of $\mathcal{E}$ are not simply the even n: $n = 5$ is in $\mathcal{E}$ and $n = 4$ is in $\mathcal{O}$, because the parity that matters is that of $i + j$, not of n. An implementation that partitions on n mod 2 is partitioning the state into rows-of-a-kind, not into a checkerboard, and its phases will read cells they have already written.

Example S3-C.3.2.2 Why the side length must be even

Lemma S3-C.1 fails on an odd torus, and the failure is worth seeing because it is the only structural constraint the two-phase construction places on the geometry. Take the same rule on a 3×3 state and list the neighbours of $(0, 0)$, which has parity 0:

direction	**cell**	**parity of $i + j$**	
left	$(0, 2)$	0	same class as $(0, 0)$
up	$(2, 0)$	0	same class as $(0, 0)$
right	$(0, 1)$	1	
down	$(1, 0)$	1	

Wrapping from $j = 0$ to $j = 2$ changes $i + j$ by 2, which is even, so the parity does not flip. On the 3×3 torus $(0, 0)$ and $(0, 2)$ are neighbours *in the same class*: Phase 1 would read a cell Phase 1 has already written, the context would no longer be recoverable at inversion time, and the layer would stop being a permutation — the v1 failure, reintroduced by geometry alone. On the 4×4 torus the wrap changes $i + j$ by 3, which is odd, and the parity flips as required. Any even side length works; no odd one does.

Example S3-C.3.2.3 Order within a phase is immaterial

The paper says a phase may visit its eight cells in any order, and may therefore be parallelised. That is a consequence of the classes being independent, not an additional assumption, and it is checkable. Running Phase 1 of DS′ over $\mathcal{E}$ in the scrambled order

$$8, 15, 0, 10, 5, 2, 13, 7$$

and Phase 2 over $\mathcal{O}$ in the order

$$9, 12, 14, 6, 3, 1, 4, 11$$

gives the identical state `03f2fe8608c57f2a1ae44e90eea46ddb`, as does the fully reversed order. Over 2000 random pairs of visit orders the output is always the same.

What is *not* interchangeable is the order of the two phases. Phase 1 must complete before Phase 2 begins, because Phase 2 reads what Phase 1 wrote. Within a phase: any order, or all eight at once. Between phases: a barrier.

S3-C.3.3 Stage 1 — Directional Substitution, DS′

The rotation amount is taken from the low three bits of the context:

$$r(c) \;=\; c \wedge 7. \tag{2}$$

This takes all eight values uniformly and satisfies $r(c \oplus \Delta) = r(c) \oplus r(\Delta)$, so a context difference shifts the rotation by exactly $\Delta \wedge 7$. It replaces v1's $(c \bmod 7) + 1$, which was neither uniform nor XOR-compatible and which is what invalidated Proposition 2.

DS′: $S \mapsto T$.

Phase 1. For every $(i, j) \in \mathcal{E}$, with $c \leftarrow S[i, j-1] \oplus S[i-1, j]$ read from S: $T[i, j] \leftarrow \mathrm{ROL}_{r(c)}\big(\pi(S[i, j])\big)$.

Phase 2. For every $(i, j) \in \mathcal{O}$, with $c \leftarrow T[i, j-1] \oplus T[i-1, j]$ read from T: $T[i, j] \leftarrow \mathrm{ROL}_{r(c)}\big(\pi(S[i, j])\big)$.

DS′$^{-1}$: $T \mapsto S$.

Phase 1. For every $(i, j) \in \mathcal{O}$, with $c \leftarrow T[i, j-1] \oplus T[i-1, j]$: $S[i, j] \leftarrow \pi^{-1}\big(\mathrm{ROR}_{r(c)}(T[i, j])\big)$.

Phase 2. For every $(i, j) \in \mathcal{E}$, with $c \leftarrow S[i, j-1] \oplus S[i-1, j]$: $S[i, j] \leftarrow \pi^{-1}\big(\mathrm{ROR}_{r(c)}(T[i, j])\big)$.

Note that both phases substitute $S[i, j]$, the *original* byte, not the Phase-1 output; only the contexts are read from the partially updated state. By Lemma S3-C.1 the Phase-1 contexts lie in $\mathcal{O}$ and are untouched by Phase 1, and the Phase-2 contexts lie in $\mathcal{E}$ and were finalised by Phase 1. Inversion is therefore well defined on every input, so **DS′ is a permutation** (Theorem D.2). Applied to the collision of A1, $\mathrm{DS}'(A) =$ `bb69bb69dd2ddd2dbb69bb69dd2ddd2d` and $\mathrm{DS}'(B) =$ `77697769dda5dda577697769dda5dda5`, which differ; the entire 65 536-state class that collapsed under DS yields 65 536 distinct images under DS′.

Example S3-C.3.3.1 A Phase-1 cell, in full

Take $(0,0) \in \mathcal{E}$ of the whitened state of Example S3-C.3.1.1. Phase 1 reads its context from S, the input state, because by Lemma S3-C.1 both context cells lie in $\mathcal{O}$ and Phase 1 does not write $\mathcal{O}$.

step	expression	hex	binary
left neighbour	$S[0,-1] = S[0,3]$	`28`	`00101000`
upper neighbour	$S[-1,0] = S[3,0]$	`02`	`00000010`
context	$c = S[0,3] \oplus S[3,0]$	`2a`	`00101010`
rotation	$r(c) = c \wedge 7$	2	`00000111` masks the low three
the byte itself	$S[0,0]$	`1f`	`00011111`
substituted	$\pi(S[0,0])$	`c0`	`11000000`
rotated left by 2	$T[0,0]$	`03`	`00000011`

The rotation is a rotation, not a shift: the two bits that leave the top of `11000000` re-enter at the bottom. Reading $r(c)$ as a shift, or reading it modulo 8 from a signed byte, are the two ways to get this cell wrong and still produce a plausible-looking state.

Example S3-C.3.3.2 A Phase-2 cell, and the misreading it invites

$(0,1) \in \mathcal{O}$. Two things happen here that the prose states but that are easy to invert in implementation: the *context* is read from T, the partially updated state, while the *byte being substituted* is still $S[0,1]$, the original. Both context cells lie in $\mathcal{E}$ and were finalised by Phase 1.

	reads	context c	$r(c)$	$T[0,1]$
correct	$T[0,0]$=03, $T[3,1]$=$a4$	`a7`	7	`f2`
wrong	$S[0,0]$=$1f$, $S[3,1]$=$1d$	`02`	2	`97`

Both substitute $\pi(S[0,1]) = e5$; they differ only in how far it is rotated. The wrong variant is the natural one to write — a single pass over all sixteen cells reading one array — and it is the *phase structure* of the DS of §3.2.1. It costs the property the two-phase rule was introduced to buy: DS'^{-1} applied to its output recovers 2 of the 16 input bytes.

Two independent changes separate DS' from §3.2.1's DS — the two-phase visit order and the rotation rule $r(c) = c \wedge 7$ — and it is worth seeing them apart, since only the first is about invertibility. Counting distinct images over the 65 536 states whose four rows all equal (a,b,a,b), the class of S3-C.2 A1:

layer	distinct images	class collapsed
DS' (two-phase, $c \wedge 7$)	65536	none
single pass, $c \wedge 7$	43294	33.9%
§3.2.1 as specified, $(c \bmod 7) + 1$	43472	33.7%

The last row is A1's figure exactly, and it is the one to quote: the middle row is a hybrid that was never specified anywhere. The two rows differ by little, which is the point — the collapse is caused by the single pass, not by the choice of rotation rule, and changing the rotation rule alone would not have repaired it. What the rotation rule fixes is a separate defect, A2, and that is Example S3-C.3.3.3.

Example S3-C.3.3.3 Why $r(c) = c \wedge 7$ replaced $(c \bmod 7) + 1$

Two properties are claimed for the new rotation index: it is uniform, and it is compatible with XOR differences. The old one has neither, and that is what invalidated Proposition 2.

Uniformity. Over the 256 context values, $c \wedge 7$ takes each of its 8 outputs exactly 32 times. $(c \bmod 7) + 1$ takes 7 values, and because $256 = 7 \cdot 36 + 4$ they are not equally often: four of them occur 37 times and three occur 36. It also never selects a rotation of 0, so under v1 every byte was rotated.

xor-compatibility. Take $c = \texttt{2a}$ (the context of Example S3-C.3.3.1) and $c' = \texttt{5c}$, so $\Delta = c \oplus c' = \texttt{76}$:

	$r(c)$	$r(c')$	$r(c) \oplus r(c')$	
$c \wedge 7$	2	4	6	$= \Delta \wedge 7 = 6$, predicted
$(c \bmod 7) + 1$	1	2	3	$\neq (\Delta \bmod 7) + 1 = 7$, not predicted

Under the new rule a context difference shifts the rotation by exactly $\Delta \wedge 7$, independently of c, so a differential analyst can carry the rotation through a trail. Under the old rule the rotation difference depends on c itself and the per-context step of Proposition 2 does not hold — which is the arithmetic behind the 13.37× overshoot reported in S3-C.2 A2.

S3-C.3.4 Stage 2 — Meridian Diffusion, MD′

This is not the MD of Section 3.2.2. §3.2.2 defines MD = MIX ∘ PAR ∘ MER, where MIX is an $\mathbb{F}_2$-linear column mixing step. MERIDIAN-v2 **drops Mix** and uses a pure positional byte permutation. Appendix D calls this layer "MD" and describes it as unchanged; that is inaccurate, and the layer is named MD′ here to keep the two apart. The consequence is that all inter-byte mixing now comes from the DS′ contexts and the AM′ gates. This is a substantive change to the diffusion layer and is flagged in S3-C.6.

MD′ is the composition $\pi_3 \circ \pi_2 \circ \pi_1$ of three positional maps on the 16 byte positions, where π_1 is a cyclic roll of the row-major byte sequence by 5, and

$$\pi_2 : (i, j) \mapsto \big((i + 3j) \bmod 4,\ (j + 2i) \bmod 4\big), \qquad \pi_3 : (i, j) \mapsto \big((i + j) \bmod 4,\ (2j - i) \bmod 4\big).$$

The matrices of π_2 and π_3 have determinants $1 \cdot 1 - 3 \cdot 2 = -5 \equiv 3$ and $1 \cdot 2 - 1 \cdot (-1) = 3$ modulo 4, both units, so both are bijections of $\mathbb{Z}_4 \times \mathbb{Z}_4$, as is π_1.

Because the composition is a fixed permutation of positions, it is simplest to implement it as a lookup. The following table is normative: $T[i, j] \leftarrow S[\sigma(i, j)]$, where

$\sigma(i,j)$	$j = 0$	$j = 1$	$j = 2$	$j = 3$
$i = 0$	$(2, 3)$	$(1, 3)$	$(0, 3)$	$(3, 3)$
$i = 1$	$(1, 0)$	$(0, 0)$	$(3, 0)$	$(2, 0)$
$i = 2$	$(3, 1)$	$(2, 1)$	$(1, 1)$	$(0, 1)$
$i = 3$	$(1, 2)$	$(0, 2)$	$(3, 2)$	$(2, 2)$

Equivalently, in flat row-major indices $n = 4i + j$, output position n takes its byte from input position $\sigma = (11, 7, 3, 15,\ 4, 0, 12, 8,\ 13, 9, 5, 1,\ 6, 2, 14, 10)_n$. Each output row is drawn from a single input column, so MD′ transposes the torus while reordering within each line. The inverse is the inverse permutation: $T[\sigma(i, j)] \leftarrow S[i, j]$. MD′ is key-independent and manifestly bijective.

Example S3-C.3.4.1 The σ table, read straight off the state

Because MD′ moves bytes without changing them, it can be exhibited by running it on a state whose bytes *are* their own positions. Load $S[i,j] = 4i + j$, that is the block `000102030405060708090a0b0c0d0e0f`:

input: $S[i,j] = 4i + j$

	j=0	j=1	j=2	j=3
i=0	00	01	02	03
i=1	04	05	06	07
i=2	08	09	0a	0b
i=3	0c	0d	0e	0f

MD′(S)

	j=0	j=1	j=2	j=3
i=0	0b	07	03	0f
i=1	04	00	0c	08
i=2	0d	09	05	01
i=3	06	02	0e	0a

The output grid *is* the σ table: the value sitting at output position n is the input position $\sigma(n)$ it was taken from. Reading it row-major gives

$$\sigma = (11, 7, 3, 15, 4, 0, 12, 8, 13, 9, 5, 1, 6, 2, 14, 10)_n,$$

which is the table printed above, and this is how that table was produced rather than transcribed. Two structural claims can be read off the same grid. Output row 0 holds $11, 7, 3, 15$, all $\equiv 3 \bmod 4$, so it is drawn entirely from input column 3; the same holds for each row, which is the transposition the text describes. And the sixteen values are distinct, so no position is duplicated or dropped.

An implementer who wants only the layer, and not the derivation, can stop here: the table is the specification, and the three-map composition below exists to explain where it came from.

Example S3-C.3.4.2 Tracing three positions through π_1, π_2, π_3

MD′ $= \pi_3 \circ \pi_2 \circ \pi_1$. Following a single byte through the three maps shows how an entry of σ arises. π_1 rolls the row-major sequence by 5, so a byte at flat position n moves to $(n{+}5) \bmod 16$; π_2 and π_3 are the linear maps on (i,j).

start	**after π_1**	**after π_2**	**after π_3**
11 $(2,3)$	0 $(0,0)$	0 $(0,0)$	0 $(0,0)$
0 $(0,0)$	5 $(1,1)$	9 $(2,1)$	5 $(1,1)$
9 $(2,1)$	14 $(3,2)$	12 $(3,0)$	9 $(2,1)$

The first line reads: the byte at input position $11 = (2,3)$ ends at output position $0 = (0,0)$, which is $\sigma(0) = 11$, the top-left entry of the normative table. The third line is a fixed point of the composition: position 9 is moved twice and returned, ending where it began. That is not a defect. A permutation of 16 positions is free to fix some of them, and MD′ fixes $4, 9$ and 14 — three in all. What matters for invertibility is only that the map is a bijection, which the distinctness of σ establishes; a permutation with no fixed point is a stronger condition than anything the design needs.

Example S3-C.3.4.3 Why the determinant has to be a unit modulo 4

π_2 and π_3 are linear maps on $\mathbb{Z}_4 \times \mathbb{Z}_4$, and such a map is a bijection exactly when its determinant is invertible modulo 4 — that is, odd. Over $\mathbb{Z}_4$ the units are 1 and 3; 0 and 2 are not.

map	**matrix**	$\det \bmod 4$	**image size**	
π_2	$\left(\begin{smallmatrix} 1 & 3 \\ 2 & 1 \end{smallmatrix}\right)$	3	16 of 16	unit, bijective
π_3	$\left(\begin{smallmatrix} 1 & 1 \\ -1 & 2 \end{smallmatrix}\right)$	3	16 of 16	unit, bijective
counter-example	$\left(\begin{smallmatrix} 1 & 1 \\ 1 & 3 \end{smallmatrix}\right)$	2	8 of 16	not a unit, 2-to-1

The counter-example is included because it looks entirely reasonable — small coefficients, no obvious degeneracy — and it silently halves the state space, mapping every position onto one of 8. That is the same class of defect as the singular coupling layer found in the Appendix C permutation (`verify_apxBC_v2.py`), and it is caught by exactly the check applied here: count the image, do not inspect the coefficients.

S3-C.3.5 Stage 3 — Admittance Mixing, AM′

$\mathbf{AM}'_K$: $S \mapsto T$.

Phase 1. For every $(i,j) \in \mathcal{E}$, with $g \leftarrow S[i,j+1] \oplus S[i+1,j]$ read from S: $T[i,j] \leftarrow S[i,j] \oplus \big(K[i,j] \wedge g\big)$.

Phase 2. For every $(i,j) \in \mathcal{O}$, with $g \leftarrow T[i,j+1] \oplus T[i+1,j]$ read from T: $T[i,j] \leftarrow S[i,j] \oplus \big(K[i,j] \wedge g\big)$.

$\mathbf{AM}'^{-1}_K$: $T \mapsto S$.

Phase 1. For every $(i,j) \in \mathcal{O}$, with $g \leftarrow T[i,j+1] \oplus T[i+1,j]$: $S[i,j] \leftarrow T[i,j] \oplus \big(K[i,j] \wedge g\big)$.

Phase 2. For every $(i,j) \in \mathcal{E}$, with $g \leftarrow S[i,j+1] \oplus S[i+1,j]$: $S[i,j] \leftarrow T[i,j] \oplus \big(K[i,j] \wedge g\big)$.

The gate reads the *forward* neighbours $(i,j+1)$ and $(i+1,j)$, whereas DS′ reads the backward neighbours $(i,j-1)$ and $(i-1,j)$; both sets lie in the opposite class by Lemma S3-C.1. Each phase has the form $x \leftarrow x \oplus f(\text{other class})$ with f recomputable from the output, and $x \mapsto x \oplus (K[i,j] \wedge g)$ with g known is an involution. Hence **$\mathbf{AM}'_K$ is a permutation for every key**, with no rank or weight condition on K (Theorem D.3). This layer is the only point at which key material enters a round.

Relative to §3.2.3's self-gate $g(S[i,j]) = (S[i,j] \gg 3) \,|\, \texttt{0x01}$ — which is also bijective for every key — the change buys diffusion, not correctness: the self-gate reads only the cell it updates and so contributes nothing to inter-byte mixing, while AM′ gates on two neighbours.

Example S3-C.3.5.1 A Phase-1 cell, in full

Take $(0,2) \in \mathcal{E}$ of the state produced by MD′ in Example S3-C.3.4.1's round — that is, $\mathrm{MD}'(\mathrm{DS}'(P \oplus K)) = \texttt{902a86db0803ee1aa4e4c5f27ffe6d4e}$. The gate reads the two *forward* neighbours, $(0,3)$ and $(1,2)$, both in $\mathcal{O}$:

step	**expression**	**hex**	**binary**
right neighbour	$S[0,3]$	`db`	`11011011`
lower neighbour	$S[1,2]$	`ee`	`11101110`
gate	$g = S[0,3] \oplus S[1,2]$	`35`	`00110101`
key byte	$K[0,2]$	`52`	`01010010`
mask	$K[0,2] \wedge g$	`10`	`00010000`
the byte itself	$S[0,2]$	`86`	`10000110`
output	$T[0,2] = S[0,2] \oplus (K \wedge g)$	`96`	`10010110`

Only the bits where the key byte and the gate are *both* set are flipped — here 1 of 8. This is the one point in the round at which key material enters.

Example S3-C.3.5.2 The gate may vanish, and the layer still inverts

In the same state, cell $(0, 0)$:

$g = S[0,1] \oplus S[1,0]$	$2a \oplus 08$	`22`
$K[0,0] \wedge g$	`4d` $\wedge$ `22`	`00`
$T[0,0]$	$90 \oplus 00$	`90` (unchanged)

The mask is zero, so the cell passes through untouched. So would every cell under $K =$ `00...00`, which makes AM′ the identity map. That is not a break, and the distinction matters because it is exactly what the withdrawn v19 claim got wrong. The identity is a permutation. AM'_K is required to be invertible, not to be mixing, and the invertibility argument is structural: each phase has the form $x \leftarrow x \oplus f(\text{other class})$, f is recomputable from the output because the other class is untouched by this phase, and $x \mapsto x \oplus m$ with m known is an involution. No rank condition, no weight condition, no constraint on K whatever.

Note that this argument is not an enumeration over keys, and does not need to be. Once the gate is fixed by the other class, the cell map is XOR by a constant, which is bijective for every constant there is; there is nothing left to check. The enumeration over all 256 key bytes belongs to the *self*-gated layer of §3.2.3, $x \mapsto x \oplus \big(k \wedge ((x \gg 3) \,|\, \texttt{01})\big)$, where the gate does depend on x and bijectivity is therefore a real question — and where the answer, checked here, is that all 256 are bijections. That is the claim recorded against §3.2.3 in the reading map of S3-C.2.

What a weak key costs AM′ is diffusion, not correctness — and that cost is real, which is why it belongs in the security discussion of S3-C.6 and not here.

Example S3-C.3.5.3 What the neighbour gate buys, measured

§3.2.3 gated each cell on itself, $g(x) = (x \gg 3) \,|\, \texttt{01}$; AM′ gates on two neighbours. Both are bijections for every key, so the change is not a correctness repair. What it buys is measurable. Flip the single lowest bit of cell $(0, 1)$ in the state above and count the cells that move:

	cells changed	**which**
AM′ (neighbour gate)	4	$(0,0)$, $(0,1)$, $(0,3)$, $(3,1)$
§3.2.3 (self gate)	1	$(0,1)$

Averaged over 20 000 random (state, key, flipped bit) triples the figures are 2.37 cells for AM′ against exactly 1.00 for the self gate. The self gate's 1.00 is not an average of a spread — it is the only value it can take, since the layer is byte-local and a change in one byte cannot reach any other. It contributes nothing to inter-byte mixing at all.

This matters more in v2 than it would have in v1, because MD′ drops the MIX step of §3.2.2 and is now a pure byte permutation. With no linear mixing layer left, the DS′ contexts and the AM′ gates are the only sources of inter-byte diffusion in the entire cipher.

S3-C.3.6 Round constants

$$RC_r[i, j] \;=\; \pi\big((16r + 4i + j) \bmod 256\big), \qquad 0 \le r < N_R. \tag{3}$$

Round r therefore consumes S-box entries $16r$ through $16r + 15$ in order, which for $N_R = 12$ uses the first 192 entries and never repeats. Passing the index through π ensures successive constants are not linearly related. The first two are

$$RC_0 = \texttt{637c777bf26b6fc53001672bfed7ab76}, \qquad RC_1 = \texttt{ca82c97dfa5947f0add4a2af9ca472c0},$$

read row-major per (**??**). Constants are injected by XOR and so do not affect invertibility. All arithmetic is integer arithmetic; the floating-point $\sin(\cdot)$ constants criticised in A3 belong to the

superseded code and appear nowhere in this specification. (§3.2.4's own constants, $(r \cdot \texttt{0x9E} + 4i + j) \bmod 256$, were already integer-exact; they are replaced for S-box uniformity, not for correctness.)

Example S3-C.3.6.1 Computing one constant, and the index budget

$RC_r[i,j] = \pi\big((16r + 4i + j) \bmod 256\big)$. For round 2, cell $(1,2)$:

index	$16 \cdot 2 + 4 \cdot 1 + 2 = 38 = \texttt{26}$
constant	$RC_2[1,2] = \pi(\texttt{26}) = \texttt{f7}$

The index is just the cell's flat position offset by the round: round r consumes S-box entries $16r$ through $16r + 15$ in order, so the whole state's constants for one round are sixteen consecutive entries.

	indices	**constant**
RC_0	0–15	`637c777bf26b6fc53001672bfed7ab76`
RC_1	16–31	`ca82c97dfa5947f0add4a2af9ca472c0`
RC_{11}	176–191	`e7c8376d8dd54ea96c56f4ea657aae08`

With $N_R = 12$ the highest index reached is $16 \cdot 11 + 15 = 191 < 256$, so the reduction modulo 256 never fires and no constant is ever reused. A variant with more than 16 rounds would wrap and start repeating round 0's constants; anyone raising N_R must change this rule as well.

Example S3-C.3.6.2 Why the index is passed through π

The raw indices are an arithmetic progression, so consecutive rounds' constants would differ by a fixed value in every cell. Compare the cell-wise difference $RC_1 \oplus RC_0$ with and without the S-box:

without π	`10 10 10 10 10 10 10 10` ... 1 distinct value across all 16 cells: the difference is the constant `10`
with π	`a9 fe be 06 08 32 28 35` ... 16 distinct values across 16 cells

Without π, $RC_1 = RC_0 \oplus \texttt{10}$ exactly. The same holds for *every* pair of rounds, not just adjacent ones: the cell index $4i + j$ occupies the low nibble and the round number the high one, so $RC_r \oplus RC_s$ is the single byte $16(r \oplus s)$ in all sixteen cells. Each cell's twelve constants would then be its own index with the round number written into the top nibble — so XORing the constant in would be almost the same operation every round, and a difference introduced in one round would pass through the next unchanged. Passing the index through π destroys that relation, at a cost of zero — the S-box table is already present, and on a constrained target the twelve constants can equally be precomputed into 192 bytes of ROM.

Constants are injected by XOR and therefore have no bearing on invertibility; their whole job is to break the symmetry between rounds. What that symmetry would otherwise permit is shown in Example S3-C.3.8.2.

S3-C.3.7 The S-box

π is the AES S-box of FIPS-197, used verbatim: inversion in $\mathrm{GF}(2^8)$ modulo $x^8 + x^4 + x^3 + x + 1$ with $\texttt{00} \mapsto \texttt{00}$, followed by the FIPS-197 affine transform over $\mathbb{F}_2$. It is a bijection with maximum differential probability 2^{-6} and maximum linear bias 2^{-3}. Reusing it is what allows existing AES S-box ROMs, $\mathrm{GF}(2^8)$ inverse circuits, and masked-S-box tooling to be shared.

Implementation requirement. Derive π from the FIPS-197 *definition*, or verify any transcribed table against it before use. Both of the following must hold, and a conforming implementation should assert them at start-up:

$$\pi \text{ is a bijection (256 distinct outputs)}, \qquad \pi(00) = 63,\ \ \pi(02) = 77,\ \ \pi(53) = ed.$$

The table shipped as `AES_SBOX` in `meridian_cipher.py` fails the first: it differs from FIPS-197 in 88 of 256 entries and has only 198 distinct outputs. It *passes* the second — all three spot values are correct, because every one of the 88 divergences sits at index `a7` or above — so it is the bijection test, not the spot check, that catches it. (v19 and v20 stated that the table failed both conditions. It does not; the correction is Example S3-C.3.7.3, and it strengthens rather than weakens the requirement, since a table can be badly wrong and still pass every spot value an implementer is likely to try.) π^{-1} is obtained by inverting the table.

Example S3-C.3.7.1 Deriving $\pi(53)$ from the definition

The insert requires π to be derived from FIPS-197 rather than copied. Here is the derivation for one entry, the third of the three start-up spot values.

Step 1 — invert in GF(2^8) modulo $x^8 + x^4 + x^3 + x + 1$. The inverse of `53` is `ca`, which is checked by multiplying: $53 \cdot ca = 01$ in the field.

Step 2 — apply the affine transform $b'_k = b_k \oplus b_{k+4} \oplus b_{k+5} \oplus b_{k+6} \oplus b_{k+7} \oplus c_k$, indices modulo 8, with $c = 63$. The input bits are those of $ca = 11001010$, written $b_7 \ldots b_0$:

bit	$b_k \oplus b_{k+4} \oplus b_{k+5} \oplus b_{k+6} \oplus b_{k+7} \oplus c_k$	result
b_0	$0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1$	1
b_1	$1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1$	0
b_2	$0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0$	1
b_3	$1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0$	1
b_4	$0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0$	0
b_5	$0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1$	1
b_6	$1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1$	1
b_7	$1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0$	1

Reading $b_7 \ldots b_0$ gives $11101101 = ed$, so $\pi(53) = ed$ as required.

Example S3-C.3.7.2 The one input with no inverse

`00` has no multiplicative inverse, so the definition patches it: $00 \mapsto 00$ before the affine step. The affine transform of `00` is just the constant, since every b_k is zero:

$$\pi(00) = 00 \oplus c = c = 63.$$

This is the second of the three spot values. The failure it catches is not in the affine step — that step handles `00` without special pleading, as the line above shows — but in the inversion before it. An implementation that computes the field inverse by a log/antilog table has no entry for log(`00`), and one that computes it as x^{254} gets `00` only because $0^{254} = 0$ happens to fall out correctly. The first will raise, wrap to a garbage index, or leave the entry at whatever the table was initialised to; whichever it does, $\pi(00)$ is the first place it shows.

The third spot value, $\pi(02) = 77$, is included for a different reason: `02` is the generator most log-table constructions start from, so an error in the field arithmetic itself surfaces there early rather than somewhere in the middle of the table where nobody looks.

Example S3-C.3.7.3 What the start-up assertions actually catch

Applied to the table shipped as `AES_SBOX` in `meridian_cipher.py`:

check	result	
$\pi(00) = 63$	63	passes
$\pi(02) = 77$	77	passes
$\pi(53) = ed$	ed	passes
256 distinct outputs	198	**fails**

The table differs from FIPS-197 in 88 of 256 entries, yet all three spot values are correct. That is not luck: the first divergence is at index `a7`, and every one of the 88 lies at `a7` or above, so the whole damaged region sits past any index an implementer is likely to check by hand. *(v19 and v20 of this insert stated that the shipped table failed both conditions. It fails only the bijection test; the text above is corrected accordingly.)*

The bijection test is what catches it, and it catches it decisively: 198 distinct outputs means 58 values are produced more than once and 58 are never produced at all. The worst case is `5e`, the image of 4 different inputs: `9d`, `a7`, `f4`, `fe`. A cipher built on this table has no π^{-1}, because `5e` has no single preimage — which is a non-injectivity of exactly the kind the whole insert exists to report, sitting undetected in the reference code that was supposed to demonstrate the design.

The general lesson is the cheap one: a spot check confirms what you thought to look at, while the bijection test confirms the table as a whole. Both are two lines of code. Run both.

S3-C.3.8 Key injection; absence of a key schedule

There is no key schedule. The 128-bit master key K is used directly, both for pre-whitening and, unchanged, as the round key of AM′ in every one of the N_R rounds. This is the single largest saving against AES-128 — no expanded key state, no expansion latency on a key change — and it is retained from §3.4 unchanged. It is also the design's most exposed remaining assumption; see S3-C.6, item 4.

Example S3-C.3.8.1 Where the key appears, and what is saved

There is no key schedule, so tracing the key through Algorithm 1 is short. K = `4d4552494449414e2d76322d6b657900` is used at:

point	operation	times
line 2, pre-whitening	$S \leftarrow S \oplus K$	1
line 6, every round	AM'_K, same K	12
total uses of the master key		13

The byte $K[0,0]$ = `4d` is the same byte at all 13 of them. Against AES-128, which expands its key into 11 round keys:

	MERIDIAN-v2	AES-128
key state held	16 bytes	176 bytes
work on a key change	none	full expansion

On a target where 176 bytes of RAM is a real fraction of the budget, and in a protocol that rekeys per packet, this is the design's largest single saving over AES-128 — larger than the absent Mix layer, and comparable to the code removed by a CTR-only build (S3-C.3.9). It is also its most exposed assumption, and the next example shows the shape of the exposure.

Example S3-C.3.8.2 What the round constants are holding up

With one key and no schedule, every round applies the *same* function except for the round constant. Remove the constants — set $RC_r = RC_0$ for all r — and the cipher becomes F^{12} for a single round function F. Then a plaintext chosen so that its whitened state is one round ahead of another's produces ciphertexts one round apart, and the property survives all twelve rounds:

P	`526f7461746520616e64206d69782e20`
P', chosen with $P' \oplus K = F(P \oplus K)$	`be16b3a9ba29c1dbb0f782d5c109e738`
$E(P)$ with constants removed	`13a278dd1e9eba83f1aec58f76108f6a`
$E(P')$ with constants removed	`5c4a1d39f8c15e8cf706d572ad45d81c`
$F(E(P))$	`5c4a1d39f8c15e8cf706d572ad45d81c`
$E(P') = F(E(P))$: **holds**	

That is a slid pair, and it hands an attacker a one-round relation between two full-cipher outputs regardless of the round count. With the specified constants in place the relation fails: $E_K(P') =$ `f2f700dc0805a5c8100142b8a0c0db06`, which does not equal $F_r(E_K(P))$ for any of the twelve values of r.

The point is not that MERIDIAN-v2 is slide-attackable; it is that (**??**) is the *only* thing standing between it and the structure above. In a cipher with a key schedule, distinct round keys break the round-to-round symmetry as well, and the constants are a second line of defence. Here they are the first and only line. Slide and related-key analysis of this construction remains open, and is item 4 of S3-C.6 and the third of the four preconditions at the end of S3-C.3.9.

S3-C.3.9 Encryption and decryption

Algorithm 1 MERIDIAN-v2 encryption of one block

Input: plaintext $P \in \mathbb{F}_2^{128}$, key $K \in \mathbb{F}_2^{128}$
Output: ciphertext $C \in \mathbb{F}_2^{128}$

$S \leftarrow \textsc{Load}(P)$ ▷ row-major, eq. (**??**)
$S \leftarrow S \oplus K$ ▷ pre-whitening
for $r \leftarrow 0$ **to** $N_R - 1$ **do**
  $S \leftarrow \mathrm{DS}'(S)$ ▷ S3-C.3.3
  $S \leftarrow \mathrm{MD}'(S)$ ▷ S3-C.3.4
  $S \leftarrow \mathrm{AM}'_K(S)$ ▷ S3-C.3.5
  $S \leftarrow S \oplus RC_r$ ▷ eq. (**??**)
end for
return $\textsc{Store}(S)$

Algorithm 2 MERIDIAN-v2 decryption of one block

Input: ciphertext $C \in \mathbb{F}_2^{128}$, key $K \in \mathbb{F}_2^{128}$
Output: plaintext $P \in \mathbb{F}_2^{128}$

$S \leftarrow \textsc{Load}(C)$
for $r \leftarrow N_R - 1$ **down to** 0 **do**
  $S \leftarrow S \oplus RC_r$
  $S \leftarrow \mathrm{AM}'^{-1}_K(S)$
  $S \leftarrow \mathrm{MD}'^{-1}(S)$
  $S \leftarrow \mathrm{DS}'^{-1}(S)$
end for
$S \leftarrow S \oplus K$ ▷ undo pre-whitening
return $\textsc{Store}(S)$

Every round is identical: there is no special first or last round, and no post-whitening step. The round constant is applied *after* key injection, so the last operation of an encryption is $\oplus RC_{N_R-1}$.

Two iterations, and Algorithm 1 supplies only one.

Algorithm 1 is itself iterative — the $N_R = 12$ round loop, the *internal* iteration — but it consumes exactly 128 bits and returns exactly 128 bits. Real plaintext is longer. The *external* iteration, the repeated application of Algorithm 1 across a message of many blocks, is a separate question, and one that §3.6 left unanswered. S3-C.3.10 records the consequence in a single sentence (the cipher is an ECB-mode primitive and any multi-block mode must supply its own IV); the rest of this subsection works that sentence out in full on a concrete English message, because an implementer who is left to improvise the mode layer is exactly the implementer who was left to improvise the S-box table in A3.

Nothing here changes the cipher. The mode constructions are the standard ones of NIST SP 800-38A, evaluated over MERIDIAN-v2 rather than over AES-128, and they are reproduced against (**??**) so that the byte order is unambiguous. Every hex value below is emitted by the reference code in `supplementary/meridian_v2_modes.py`, which imports the permutation unchanged and adds only the mode layer.

The message.

Take a piece of ordinary English:

P	`"Rotate and mix.  Rotate and mix.  That is the whole round."` 56 bytes = 448 bits = 3.50 blocks
K	`4d4552494449414e2d76322d6b657900`

At 448 bits this is three and a half blocks, so the example forces both of the things a single-block algorithm cannot do: it must be padded to a block boundary, and Algorithm 1 must be invoked more than once under one key. Two features of the text are deliberate. Its first 16 bytes repeat exactly at the second block boundary, which is what makes the ECB failure below visible rather than merely assertable; and its last block is partial, which is what forces the padding. English does repeat on 16-byte boundaries in practice — fixed-format records, message headers, boilerplate, repeated instructions — so this is not a contrived worst case so much as an aligned one.

Step 1 — padding.

56 mod 16 = 8, so PKCS#7 (RFC 5652 §6.3) appends 8 bytes each of value `08`, giving 64 bytes = 4 blocks. The padding is applied *always*, including when the message is already a whole number of blocks; otherwise stripping it is ambiguous.

	Hex	**ASCII**
P_0	`526f7461746520616e64206d69782e20`	`Rotate and mix.`
P_1	`526f7461746520616e64206d69782e20`	`Rotate and mix.`
P_2	`546861742069732074686520776861 6c`	`That is the whol`
P_3	`6520726f756e642e0808080808080808`	`e round.........`

Step 2 — the internal iteration, on P_0.

Algorithm 1 applied to P_0. Round 0 is shown stage by stage; thereafter the rows labelled $r{=}n$ give the end-of-round state, that is the state after $\oplus RC_n$ and hence the input to round $n+1$. A conforming implementation must reproduce every line, and diffing against this table localises a divergence to one stage of one round.

After	**State**	**After**	**State**
Load P_0	`526f7461746520616e64206d69782e20`	$r{=}3$	`1cd20308307928b7a32cf2f158645c44`
$\oplus K$	`1f2a2628302c612f43121240021d5720`	$r{=}4$	`432b7831dfbb2fb06b771c276b38c31c`
0: DS′	`03f2fe8608c57f2a1ae44e90eea46ddb`	$r{=}5$	`8e28ea6c19946f68372002c97ef1ca53`
0: MD′	`902a86db0803ee1aa4e4c5f27ffe6d4e`	$r{=}6$	`66801adc76284d5920666630db233513`
0: AM′	`902f969b0c0bef50ad80d7d354bb354e`	$r{=}7$	`593034619c459e1e5cb3f5b1deee484b`
0: $\oplus RC_0$	`f353e1e0fe6080959d81b0f8aa6c9e38`	$r{=}8$	`9074e4c3b6334216e543b814e64094af`
$r{=}1$	`a6acf3f510656d630703aaf107fa620f`	$r{=}9$	`db4e33a51f2f33a566c0a56ed7776f03`
$r{=}2$	`1b42279460e77347c73823bf24bf7d3e`	$r{=}10$	`9eb266f42de16a38396c9699a1332e49`
		$r{=}11 \Rightarrow C_0$	`330ccab155bd2c8e1d6ab8814709e42c`

The internal iteration is where the cipher's mixing happens, and it is complete by the twelfth round; what follows is entirely about how these 128-bit calls are strung together.

Step 3 — the external iteration done wrongly (ECB).

The obvious way to extend Algorithm 1 to four blocks is to call it four times, independently. That is ECB, and it fails immediately:

	Ciphertext	
C_0	`330ccab155bd2c8e1d6ab8814709e42c`	← identical
C_1	`330ccab155bd2c8e1d6ab8814709e42c`	← identical
C_2	`a6eaea0c1e2f841e4d97549b90639a17`	
C_3	`6dc45db7137578c313d3234e85bf2fe6`	

ECB must not be used, and this is why. $P_0 = P_1$, so by Theorem S3-C.2 — by the very property this insert restores — $C_0 = C_1$. A permutation is deterministic, so a repetition in the plaintext is reproduced exactly in the ciphertext. The repeated phrase `"Rotate and mix. "` survives encryption as a visible repetition of the 128-bit string `330ccab155bd2c8e1d6ab8814709e42c`, under any key and for any number of rounds. No property of DS′, MD′ or AM′ can prevent this; it is a defect of the mode, not of the permutation, and strengthening the permutation does not touch it.

Step 4 — chaining (CBC).

Feed each ciphertext block forward into the next plaintext block, $C_i = E_K(P_i \oplus C_{i-1})$ with $C_{-1} = IV$. The IV must be unpredictable and must never repeat under a given key; it is not secret and travels with the ciphertext.

	Ciphertext	
IV	`0102030405060708090a0b0c0d0e0f10`	(sent in clear)
C_0	`80761a6ec9edd4291949c0744f259b21`	← now differ
C_1	`029da3a2fd68bf268c8e3dc73b0897a4`	← now differ
C_2	`1a503a409ce8f44a7450a08c009008af`	
C_3	`b5f5c7f149074b3bed2b9821d4125351`	

$P_0 = P_1$ still, but $C_0 \neq C_1$: they differ in 66 of 128 bits, against 64 expected for unrelated blocks. The repetition is no longer visible. Note the cost: encryption is now strictly serial, since block i cannot begin until block $i-1$ is complete, though decryption still parallelises. Note also that CBC decryption requires Algorithm 2, and therefore π^{-1} and all three stage inverses.

Step 5 — keystream (CTR).

Encrypt a counter rather than the plaintext, and XOR the result in. Write the counter block as an 8-byte nonce followed by an 8-byte big-endian block index, loaded row-major per (**??**):

	Counter block T_i	**Keystream** $E_K(T_i)$
$i = 0$	`a1a2a3a4a5a6a7a80000000000000000`	`5981e520960f54178430b881b78b0495`
$i = 1$	`a1a2a3a4a5a6a7a80000000000000001`	`aa12a50044ea9f1bafc5f7e0f670c8c6`
$i = 2$	`a1a2a3a4a5a6a7a80000000000000002`	`68beb13749913e7f1f2d9cfbf42ba2d6`
$i = 3$	`a1a2a3a4a5a6a7a80000000000000003`	`08d80461377e62bc95ba5f68563f5f9a`

C	`0bee9141e26a7476ea5498ecdef32ab5f87dd161308fbf7ac1a1d78d9f08e6e6`
	`3cd6d04369f84d5f6b45f9db8343cdba6df8760e42100692`

Three consequences are worth stating explicitly. The ciphertext is 56 bytes, exactly the length of the plaintext: the final keystream block is truncated and no padding is used, so the padding-oracle question does not arise. Encryption and decryption are the same operation. And only the

forward permutation is ever evaluated, so a CTR-only deployment can omit Algorithm 2, π^{-1}, DS'^{-1}, MD'^{-1} and AM'^{-1} altogether — for MERIDIAN-v2 that removes roughly half the code and the entire inverse S-box table, which on the constrained targets the design is aimed at is a larger saving than the absent key schedule of S3-C.3.8. Against that: a repeated (K, nonce, i) triple XORs two plaintexts together and destroys the mode outright, with no contribution from the permutation either way.

Summary of the external iteration.

	ECB	**CBC**	**CTR**
Hides repeated blocks	no	yes	yes
Padding required	yes	yes	no
$\lvert C\rvert = \lvert P\rvert$	no	no	yes
Needs Algorithm 2	yes	yes	no
Encryption parallelisable	yes	no	yes
Per-message value needed	—	IV	nonce
Provides integrity	no	no	no

Deployment guidance.

The remainder of this subsection is operational. It states how to iterate MERIDIAN-v2 over real traffic, what each mode requires of the caller, and what breaks if those requirements are not met. Read the preconditions box at the end before acting on any of it.

Mode selection.

Use **CTR** by default. It is length-preserving, needs no padding, admits no padding oracle, parallelises, and requires only the forward permutation — which on the constrained targets this design is aimed at removes Algorithm 2, π^{-1} and all three stage inverses. Use **CBC** only where a legacy interface demands it. Do not use **ECB** under any circumstances; Step 3 shows what it costs.

Key management.

S3-C.3.8 specifies no key schedule: the 128-bit master key is injected directly and unchanged into all N_R rounds. Three consequences follow for the caller.

1. Derive keys with a KDF (HKDF, or PBKDF2/Argon2 from a password). Never key the cipher with a password, a counter, a device identifier or any other structured value.
2. Never derive two keys with a simple relation — K and $K \oplus \delta$, or K and $K + 1$. Related-key and slide analysis are explicitly unaddressed for this construction, and direct injection is the property that invites them.
3. Rekey often. The absence of a key schedule means a key change costs nothing (no expansion, no latency), so key agility is the one place where this design is unambiguously ahead of AES-128. Spend it.

Data limits per key.

The block is 128 bits, so the birthday bound applies at 2^{64} blocks regardless of key size. For CBC, distinct ciphertext blocks begin to collide there and each collision leaks the XOR of two plaintext blocks; keep to 2^{48} blocks — about 4.5 PB — per key, and rekey well before that. For CTR the limit is per-nonce rather than per-key: the 8-byte counter admits 2^{64} blocks in one message, which no message will reach, so the binding constraint is nonce management below.

The IV, for CBC.

Sixteen bytes, fresh for every message, drawn from a cryptographically secure RNG. It must be *unpredictable*, not merely unique: an IV an adversary can guess before choosing a plaintext permits a chosen-plaintext distinguishing attack, which is why a message counter or a timestamp is not an acceptable IV. It is not secret. Transmit it alongside the ciphertext, and include it under the MAC.

The nonce, for CTR.

Eight bytes, and it must be a *deterministic counter*, not a random value. This is the sharpest operational edge in the whole construction and it is a consequence of the nonce being only 64 bits:

random 8-byte nonces collide with probability around 2^{-1} after 2^{32} messages, and a single collision reuses the keystream, XORs two plaintexts together, and destroys confidentiality for both — with no contribution from the permutation either way. Maintain a persistent counter, and either persist it across restarts or rekey on every restart. If a system cannot guarantee this, use CBC.

Integrity is mandatory, and none of these modes provides it.

All three are malleable, CTR trivially so: flipping a ciphertext bit flips exactly the corresponding plaintext bit. Compose **encrypt-then-MAC** with an independent key K_{mac}: compute the tag over $(IV \,\|\, C)$ or $(\text{nonce} \,\|\, C)$, and on receipt verify the tag in constant time and *return before decrypting* on failure. A standard MAC — HMAC-SHA-256, or CMAC over a standardised block cipher — is the right choice here. Do not use the AEAD constructions of Appendix B: they were derived over the v1 permutation and are void until re-derived over v2 (S3-C.6, item 5).

Padding oracles.

CBC with PKCS#7 is a padding oracle whenever a padding failure is distinguishable from any other failure — by error code, by message, or by timing. Encrypt-then-MAC removes the exposure, because a forged or modified ciphertext is rejected at the MAC check and the unpad never runs. Do not report padding failure separately from authentication failure. CTR sidesteps the question entirely, which is a further reason to prefer it.

Implementation checklist.

1. Assert the two S-box start-up conditions of S3-C.3.7 before first use. The v1 table failed both.
2. Load blocks **row-major** per (**??**). AES loads column-major; getting this wrong reproduces neither the vectors of S3-C.5 nor those of this subsection.
3. Reproduce the twelve end-of-round states of Step 2 before testing any mode. A mode bug on top of a permutation bug is very hard to localise.
4. Reproduce the CBC and CTR vectors of Steps 4 and 5, and the padding of Step 1 — including the exact-multiple case, where a whole extra block of sixteen `10` bytes is appended.
5. Wire in encrypt-then-MAC before the cipher touches real data, not afterwards.
6. Zeroise key material, and treat the CTR counter as state that must survive a crash.

Preconditions — none of which is met at the time of writing. The guidance above describes how to iterate MERIDIAN-v2 correctly. Correctness is not sufficiency, and four conditions must hold before any of it is acted on in production:

1. **External cryptanalysis.** No third party has analysed MERIDIAN-v2. Two structural claims in this paper's own corrigendum survived internal review before being settled by exhaustive enumeration (S3-C.6, item 6).
2. **The differential bound.** $A_{12} \geq 48$ was MILP-verified against v1, and v2 changes both DS and MD; the 2^{-240} figure rests on an unverified premise (S3-C.4).
3. **The key schedule.** Direct injection with no schedule (S3-C.3.8) leaves slide and related-key analysis open. No mode discipline compensates for a weakness in the primitive.
4. **Authenticated encryption.** No AEAD construction is currently valid over v2, so integrity must be bolted on by hand.

Until all four hold, deploy AES-128-GCM or Ascon instead. Both are standardised, both have had the scrutiny this design has not, and AES-128-GCM is a drop-in at the same block and key size (S3-C.3.11). The appropriate uses for MERIDIAN-v2 today are research, cryptanalysis, benchmarking and interoperability testing — for which the vectors above are supplied.

Example S3-C.3.9.1 Algorithm 2, unwinding one round

The insert traces encryption stage by stage in S3-C.3.9's Step 2 but never runs decryption, so here is round 11 undone. Start from the ciphertext of the running example and apply lines 3–6 of Algorithm 2 once:

after	state
Load C	`330ccab155bd2c8e1d6ab8814709e42c`
$\oplus RC_{11}$	`d4c4fddcd8686227713c4c6b22734a24`
AM'^{-1}_K	`ddc1ef9dd8612321780a7c6e01331324`
MD'^{-1}	`616e33efd87c01c1210a24dd2378139d`
DS'^{-1}	`9eb266f42de16a38396c9699a1332e49`

The last line is the end-of-round-10 state, and it is the row labelled r=10 in the Step 2 trace below — the two tables meet exactly there. Continuing the loop down to $r = 0$ and then XORing K returns `526f7461746520616e64206d69782e20`, the plaintext.

Two details of the algorithm are visible in the order of those rows. The constant comes off *first*, because encryption applied it last; and there is no post-whitening to undo, because there is none to apply — the XOR by K happens once, at the very end of decryption, undoing line 2 of Algorithm 1.

Example S3-C.3.9.2 The stage inverses do not commute

Algorithm 2 applies AM'^{-1}, then MD'^{-1}, then DS'^{-1}: the reverse of the encryption order. Each stage is individually invertible, which makes it tempting to assume the order is a detail. It is not. Running the loop with DS'^{-1} and AM'^{-1} transposed, and everything else identical:

correct order	`526f7461746520616e64206d69782e20`
$\mathrm{DS}'^{-1}, \mathrm{MD}'^{-1}, \mathrm{AM}'^{-1}$	`59875aa6cadcf36a5d831806105c95ba`
bytes in common	0 of 16

The output is not close to the plaintext; it is unrelated to it. That is the useful part. A wrong-order decryption fails totally rather than partially, so it cannot be mistaken for a rounding or endianness problem, and a single test vector from S3-C.5 catches it on the first run. Contrast the column-major load of Example S3-C.3.1.2, which is also totally wrong but self-consistent, and so passes a round-trip test.

Example S3-C.3.9.3 How fast the round loop diffuses a single bit

$N_R = 12$ is a parameter of the design, and the reason for the loop is that one round does not suffice. Flip the lowest bit of P_0 — one bit in 128 — and count how the difference grows through Algorithm 1:

after	differing bytes	differing bits
after $\oplus K$	1	1
r=0	11	22
r=1	16	62
r=2	16	72
r=3	16	64
r=4	15	54
r=5	16	65
r=6	16	75
r=7	16	59
r=8	16	68
r=9	16	69
r=10	16	73
r=11	16	60

P	`526f7461746520616e64206d69782e20`
P' (one bit flipped)	`536f7461746520616e64206d69782e20`
$E_K(P)$	`330ccab155bd2c8e1d6ab8814709e42c`
$E_K(P')$	`bc16d4e1af99ad9021420cf3e1ee02e8`
differing bits	60 of 128, against 64 expected

Whitening is an XOR, so it moves nothing: the difference is still one bit going into round 0. All sixteen bytes are active after round 1, and the bit count settles around 64 from round 1 onward.

This is a sanity check, not a security argument, and the distinction is worth being explicit about. Full diffusion after a couple of rounds is a *necessary* property that any competent round function has; it says nothing about whether twelve rounds resist differential or linear cryptanalysis. That question is S3-C.4's, and its answer is currently incomplete — the $A_{12} \geq 48$ branch-number bound was MILP-verified against v1, and v2 changed both DS and MD.

S3-C.3.10 Correctness

Theorem S3-C.2 (= Theorem D.4). *For every $K \in \mathbb{F}_2^{128}$, MERIDIAN-v2 is a permutation of $\mathbb{F}_2^{128}$, and Algorithm 2 inverts Algorithm 1 on all inputs.*

Each round composes DS′ (a bijection, S3-C.3.3), MD′ (a positional permutation, S3-C.3.4), AM'_K (a bijection for every key, S3-C.3.5) and an XOR by a constant; whitening is an XOR by K. A composition of bijections is a bijection, and Algorithm 2 applies the stage inverses in reverse order. This restores the property §3.6 asserts and v1 does not possess.

Determinism and the IV.

As in §3.6, the cipher is a deterministic permutation: the same (K, P) always gives the same C. It is therefore an ECB-mode primitive, and any mode that processes more than one block under one key requires an IV or nonce at the mode layer — CBC, CTR, GCM and the constructions of Appendices B and C all supply one. The block cipher itself takes no IV. Note that the Appendix B and C constructions were derived over the v1 permutation; they have been re-derived over v2 by the Appendices B/C Corrigendum Insert (v20b), but that re-derivation restates none of their theorems, so their security claims remain open (S3-C.6, item 5).

S3-C.3.11 AES compatibility

The v2 changes do not touch the compatibility argument of §3.5, which holds at three levels. *Envelope:* the cipher is a 128-bit block, 128-bit key primitive, so it drops into ECB, CBC, CFB, OFB, CTR, XTS, CMAC, CCM and GCM wherever AES-128 is used, with no change to the mode. *Primitive:* π is the AES S-box verbatim, so AES S-box ROMs, GF(2^8) inverse circuits and masked-S-box implementations are reusable. *API:* the key and block sizes match AES-128 exactly.

What is not preserved: MERIDIAN-v2 is a different cipher from AES-128 and does not produce AES ciphertexts. It has no MDS matrix, no key schedule, and — as of v2 — no linear mixing layer at all, its diffusion coming instead from the DS′ contexts and the AM′ gates. AES-NI cannot accelerate it.

S3-C.4 Corrected differential analysis

Write the DS′ cell map as $y = \mathrm{ROL}_{r(c)}(\pi(x))$. For a *fixed* rotation, ROL is $\mathbb{F}_2$-linear, so the AES DDT bound 2^{-6} applies unchanged. The corrections concern cells whose rotation changes by $k = \Delta c \wedge 7 \neq 0$, where the difference is governed by $z \mapsto z \oplus \mathrm{ROL}_k(z)$, whose fibres have size $2^{\gcd(k,8)}$.

Cell type	Max. probability	Attained at
Active byte, rotation unchanged	$4/256 = 2^{-6}$	AES DDT maximum
Passive byte, rotation changed by k	$2^{\gcd(k,8)}/256$, max 2^{-4}	$k = 4$
Active byte, rotation also changed	$8/256 = 2^{-5}$	$\delta =$ `0x21`, $k = 4$

Per-k: 2^{-7} for $k \in \{1,3,5,7\}$, 2^{-6} for $k \in \{2,6\}$, 2^{-4} for $k = 4$. The $k = 4$ row is exactly the ROL_4 self-similarity that produced the v1 collision; under DS′ it can no longer collide, but it remains the worst case and must be charged for. The v1 proof assumed 2^{-3} here, which is where Proposition 2 failed.

Proposition S3-C.3 (= Prop. D.5, replacing Proposition 2). *For a one-round pattern with w active bytes, w_c of which also see a rotation change, and w' passive bytes whose context difference has nonzero low three bits, and for any fixed output difference,*

$$\Pr \;\le\; 2^{-6(w-w_c)} \cdot 2^{-5w_c} \cdot 2^{-4w'} \;\le\; 2^{-4(w+w')}.$$

Conditioning on $\mathcal{O}$ fixes every Phase-1 context, after which distinct cells of $\mathcal{E}$ depend on distinct, independent, uniform state bytes; the table bounds each, and averaging preserves the bound. Repeat for Phase 2, conditioning on the now-fixed outputs of $\mathcal{E}$. The counterexample of A2, re-evaluated under DS′, has probability $2^{-18.42}$ against a corrected bound of 2^{-14}.

Revised headline bound.

§4.2's 2^{-288} came from $A_{12} \ge 48$ active S-boxes at 2^{-6} each. An active byte whose context also changes contributes 2^{-5}, so the worst-case twelve-round bound is $\Pr \le (2^{-5})^{48} = 2^{-240}$, not 2^{-288}. This is still far below 2^{-128}, so the differential security claim survives in form — but $A_{12} \ge 48$ was MILP-verified against the *v1* round function, and v2 changes both DS and MD. Until the model is re-run against DS′ and MD′, 2^{-240} rests on an unverified premise. We do not claim to have re-run it.

S3-C.5 Reference implementation and test vectors

`supplementary/meridian_v2.py` implements exactly the specification above, with encryption and decryption and no dependencies; its S-box is generated from the FIPS-197 definition rather than transcribed. It passes inversion of DS′, MD′ and AM′ on 20 000 random states each, a full round-trip on 26 000 (K, P) pairs, and injectivity of DS′ over the entire 65 536-state class that collapsed in v1. Avalanche reaches the ideal 64.00 bits by round 4 and is 64.12 at twelve rounds (ideal s.d. 5.66; measured 5.69).

The paper published no 128-bit test vectors, which is why A3 went unnoticed. Twelve rounds, hex, loaded row-major per (**??**):

Key *(upper)* / **Plaintext** *(lower)*	**Ciphertext**
`00000000000000000000000000000000` `00000000000000000000000000000000`	`8b909e24d761b869f3a79cade4588f0c`
`000102030405060708090a0b0c0d0e0f` `000102030405060708090a0b0c0d0e0f`	`81e5998865411dd8ee52c7af0a5550a0`
`ffffffffffffffffffffffffffffffff` `00000000000000000000000000000000`	`5ee3e7958fc75ae2fbcf036eb434e313`

Multi-block vectors.
The table above fixes the permutation but not the mode layer, and an implementation can reproduce every line of it while still mishandling padding, chaining or the counter block. `supplementary/meridian_v2_modes.py` supplies the worked example of S3-C.3.9 — padding, ECB, CBC and CTR over the same 448-bit English plaintext — and `supplementary/test_vectors_modes.txt` records its output. Those values are checked by `verify_insert_spec.py` as well, which implements the mode layer from the insert text alone.

S3-C.6 What this insert does not fix

1. MERIDIAN-v1 as published is broken and should not be used or cited as a block cipher. The ePrint record should be updated accordingly.
2. Theorem S3-C.2 is a *correctness* property, not a security claim. It says nothing about resistance to differential, linear, algebraic, integral or related-key attacks on the new round function.
3. The MILP active-S-box bound has not been re-run for v2. Both DS and MD changed — MD′ drops the Mix step of §3.2.2 and so removes the only linear mixing in the cipher — so $A_{12} \geq 48$ cannot be carried over and the 2^{-240} figure is unverified.
4. The key schedule is unchanged (S3-C.3.8): with direct injection and no schedule, all twelve rounds use the same K, inviting slide and related-key analysis. Neither this insert nor Appendix D addresses it.
5. The AEAD and hash constructions of Appendices B and C assume a permutation and are void for v1. *Re-derived over v2 by the Appendices B/C Corrigendum Insert (v20b), which is bound into this document immediately before Appendix B, printed folio 140.* That re-derivation is structural and behavioural only: it exhibits inverses, corrects a coupling layer that was 4-to-1 and therefore not a permutation at all, and shows the sponge and both AEAD modes to be correct — but it restates *no theorem* of either appendix, and it found that the shipped AEAD never called MERIDIAN and that the shipped hash round function is not MERIDIAN's. Read that insert before relying on anything in Appendices B or C; every performance figure in both should be treated as unsourced until regenerated.
6. MERIDIAN-v2 has had no third-party cryptanalysis. Three claims in this paper's own corrigendum have now had to be corrected after internal review had passed them: a structural non-defect (v19's assertion that AM is singular for almost every key, withdrawn in v20), a structural defect that had gone unnoticed in the shipped code (the Appendix C coupling layer, 4-to-1 rather than a permutation, found in v20b), and a mis-description of a real defect (v19 and v20 said the shipped S-box table failed both start-up conditions; it fails only the bijection test, corrected in v20c). Each was settled by exhaustive enumeration, and each had survived reading. The appropriate inference is that the design needs external scrutiny before any deployment claim is made.

Sources. J.-P. Aumasson. "Blood MERIDIAN: a blockcipher that is not a blockcipher." Cryptology ePrint Archive, Paper 2026/1818, 2026, `https://eprint.iacr.org/2026/1818` (short URL: `https://ia.cr/2026/1818`). The title alludes to C. McCarthy, *Blood Meridian, or the Evening Redness in the West*, Random House, New York, 1985. Full treatment: Appendix D, printed folios 198–205. Reproducers: `supplementary/verify_v1_break.py` (D1, D3, D4) and `supplementary/verify_section3_layers.py` (per-layer audit of §3.2.1–§3.2.4 as specified).

and the toroidal byte-permutation is the diffusion mechanism that makes the four-property combination viable at competitive gate count.

3 Methods: Formal Specification

3.1 Notation and Preliminaries

We use the following notation throughout:

- $\mathbb{F}_2 = \{0, 1\}$ with XOR and AND as field operations.
- $\mathbb{F}_2^n$ denotes n-bit vectors.
- Bytes are elements of $\mathbb{F}_2^8$, identified with $\mathbb{Z}_{256}$.
- The MERIDIAN state is $S \in (\mathbb{F}_2^8)^{16}$, a 16-byte (128-bit) vector.
- We index the state as $S[i, j]$ for $(i, j) \in \mathbb{Z}_4 \times \mathbb{Z}_4$, identifying the state with a 4×4 grid wrapped into a discrete torus.
- $\oplus$ denotes byte-wise XOR; ROL_k denotes left rotation by k bits within a byte.
- $K \in (\mathbb{F}_2^8)^{16}$ is the 128-bit master key.
- N_r is the per-round nonce, $N_r = r \cdot \texttt{0x9E}$ taken mod 256 (a fixed asymmetry constant per round).

Definition 1 (Toroidal coordinates)**.** The state grid is identified with the torus $\mathbb{Z}_4 \times \mathbb{Z}_4$, so that index arithmetic is taken modulo 4 in each coordinate. A *meridian* is a column (fixed j, varying i); a *parallel* is a row (fixed i, varying j).

Definition 2 (Branch number)**.** For a linear map $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$ partitioned into bytes, the byte-level branch number is

$$\mathcal{B}(L) = \min_{x \neq 0} \big(\mathrm{wt}_\mathrm{B}(x) + \mathrm{wt}_\mathrm{B}(L(x))\big)$$

where wt_B counts non-zero bytes.

3.2 MERIDIAN Round Function

MERIDIAN's round function R_r is the composition of three stages — Directional Substitution (DS), Meridian Diffusion (MD), and Admittance Mixing (AM) — followed by a per-round constant XOR:

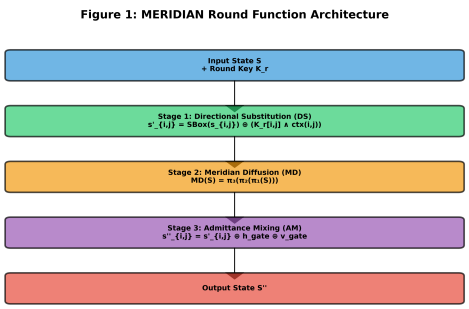
$$R_r(S) = \mathrm{AM}_K \circ \mathrm{MD} \circ \mathrm{DS}_r(S) \oplus \mathbf{c}_r.$$

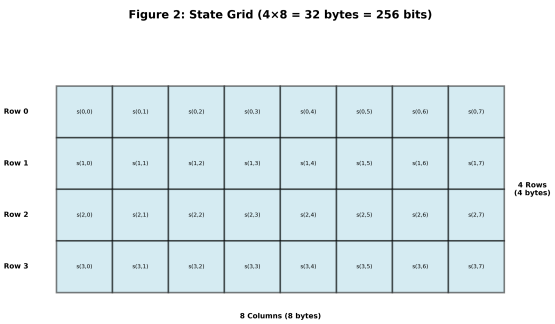
The full cipher is

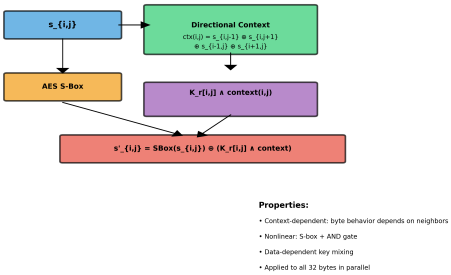
$$\mathrm{MERIDIAN}_K(P) = (R_{N_R} \circ R_{N_R - 1} \circ \cdots \circ R_1)(P \oplus K_0)$$

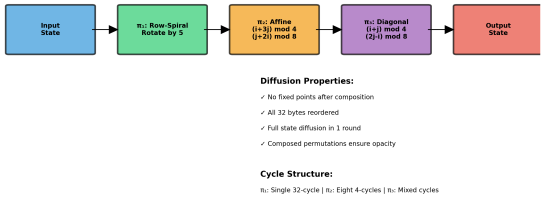
where $K_0 = K$ is the initial whitening key and $N_R = 12$ is the recommended number of rounds. (See Section 4 for the security argument supporting $N_R = 12$.)

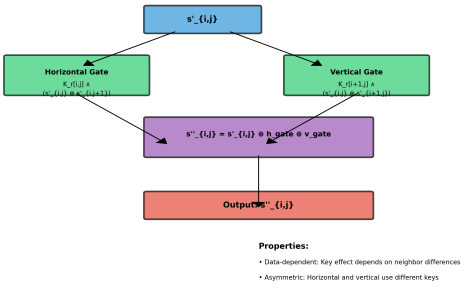
v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

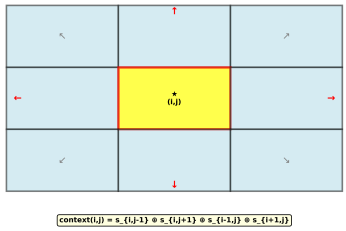


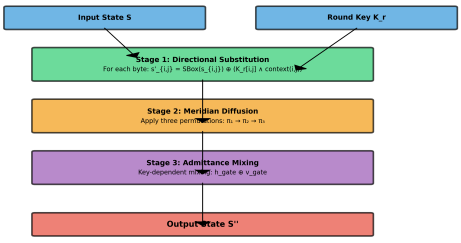












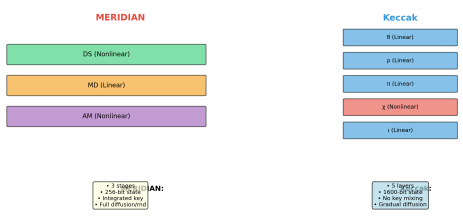


Figure 3: MERIDIAN round structure. Each round applies DS, MD, AM, and a round-constant XOR. The 128-bit state is laid out on a 4×4 discrete torus.

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

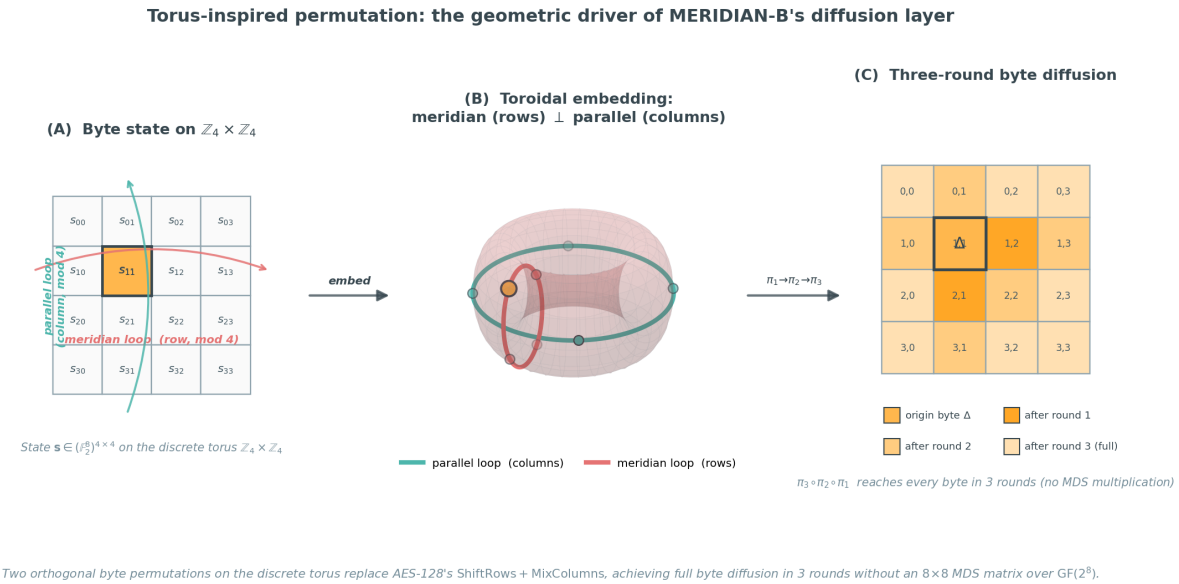


Figure 4: **Torus-inspired permutation: the geometric driver of the round function specified above. (A)** The 128-bit state is a 4×4 array of bytes whose row and column indices are read modulo 4 — i.e. it is a discrete torus $\mathbb{Z}_4 \times \mathbb{Z}_4$. Each byte has four toroidal neighbours (left/right via the meridian wrap, above/below via the parallel wrap), and these are exactly the neighbours used by the DS context byte $c_{i,j}$ and the AM admittance gates. **(B)** Embedding the toroidal grid into $\mathbb{R}^3$ exposes the two intrinsic loops of the surface: the *meridian loop* (red, threading through the central hole) parameterises rows, and the *parallel loop* (teal, encircling the hole) parameterises columns. These two loops are topologically orthogonal — they cannot be deformed into each other — and that orthogonality is what the MD stage exploits to obtain full byte diffusion without an 8×8 MDS matrix. **(C)** The composition $\pi_3 \circ \pi_2 \circ \pi_1$ of the three meridian-diffusion permutations propagates a single-byte difference Δ to every byte position of the state in three rounds, achieving the same global mixing property that AES-128 obtains via its GF(2^8) MixColumns multiplication. The geometric viewpoint recurs throughout the paper: it motivates the linear layer specified in this section, constrains the differential trail search of Section 4, and underwrites the gate-count and RAM savings reported in Section 11.

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

3.2.1 Stage 1: Directional Substitution (DS)

For each byte $S[i, j]$, define a context byte

$$c_{i,j} = S[i, j-1] \oplus S[i-1, j]$$

where indices are taken modulo 4. The substitution is

$$\mathrm{DS}_r(S)[i, j] = \mathrm{ROL}_{r_{i,j}}\big(\pi(S[i, j])\big), \qquad r_{i,j} = (c_{i,j} \bmod 7) + 1$$

where $\pi : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ is a fixed bijective S-box (we use the AES S-box as a known-good 8-bit bijection with maximum differential probability 2^{-6} and maximum linear bias 2^{-3}).

The DS stage combines a fixed nonlinear substitution with a context-dependent rotation; the rotation amount depends on two neighbouring bytes on the torus, providing data-dependent confusion at the byte level.

3.2.2 Stage 2: Meridian Diffusion (MD)

The MD stage applies two orthogonal toroidal byte permutations followed by a column-wise byte-mixing step.

Meridian shift. For each column j, rotate the column upward by j positions:

$$\mathrm{Mer}(S)[i, j] = S[(i + j) \bmod 4, j].$$

Parallel shift. For each row i, rotate the row leftward by i positions:

$$\mathrm{Par}(S)[i, j] = S[i, (j + i) \bmod 4].$$

Toroidal mix. Apply a column-wise byte-mixing function

$$\mathrm{Mix}(S)[i, j] = S[i, j] \oplus S[(i + 1) \bmod 4, j] \oplus \mathrm{ROL}_3(S[(i + 2) \bmod 4, j]).$$

The full MD stage is

$$\mathrm{MD}(S) = \mathrm{Mix} \circ \mathrm{Par} \circ \mathrm{Mer}(S).$$

The composition Mer $\circ$ Par realises a non-trivial toroidal traversal: a single byte at position (i_0, j_0) is mapped to position $(i_0 - j_0, j_0)$ by Mer, and then to $(i_0 - j_0, j_0 - (i_0 - j_0)) = (i_0 - j_0, 2j_0 - i_0)$ by Par (all arithmetic mod 4). The Mix step then spreads each byte's value across three rows of its column.

Lemma 1 (Single-byte spread)**.** *If S and S' differ in exactly one byte position (i_0, j_0), then after one application of MD they differ in at most three byte positions, and after two applications they differ in at most nine byte positions. After three applications, all 16 positions can be made to differ for an appropriate choice of input difference.*

Proof sketch. The Mix step is the only stage that increases byte-Hamming weight. From a single-byte difference, Mix produces at most three non-zero bytes (by linearity, exactly three for almost all inputs). Mer and Par are byte-permutations; they preserve byte-weight but redistribute positions across the torus. After three iterations, the set of touched positions covers the full 4×4 grid for non-degenerate inputs. □

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

3.2.3 Stage 3: Admittance Mixing (AM)

The AM stage injects the master key into the state via a data-dependent gating function:

$$\mathrm{AM}_K(S)[i,j] = S[i,j] \oplus \big(K[i,j] \cdot g(S[i,j])\big)$$

where $\cdot$ is byte-wise AND and the gate

$$g(x) = (x \gg 3) \mid \texttt{0x01}$$

ensures that at least the low bit of $K[i,j]$ is admitted into the state. The data-dependence of the gate makes the key-mixing layer non-affine in the key, which thwarts the linear key-recovery attacks that work against pure XOR-based round-key injection in some lightweight ciphers.

3.2.4 Round Constant

The round constant $\mathbf{c}_r \in (\mathbb{F}_2^8)^{16}$ is defined by

$$\mathbf{c}_r[i,j] = (r \cdot \texttt{0x9E} + 4i + j) \bmod 256$$

i.e. a Fibonacci-spaced byte-sequence offset by the position. This is the design's analogue of AES's round constants: it breaks the round function's symmetry, eliminates fixed-point round-self-similarity, and is essential for the avalanche of the all-equal-byte input subspace (see Section 4.5).

3.3 AES-128 Round Function (for comparison)

AES-128 [2] operates on a 128-bit state arranged as a 4×4 matrix of bytes. Its round function is

$$R_r^{\mathrm{AES}}(S) = \mathrm{AddRoundKey}_{K_r} \circ \mathrm{MixColumns} \circ \mathrm{ShiftRows} \circ \mathrm{SubBytes}(S).$$

- **SubBytes:** Apply the AES S-box (multiplicative inverse in $\mathrm{GF}(2^8)$ followed by an affine transform) byte-wise.
- **ShiftRows:** Cyclically shift row i leftward by i positions ($i = 0, 1, 2, 3$).
- **MixColumns:** Multiply each column by a fixed 4×4 MDS matrix over $\mathrm{GF}(2^8)$.
- **AddRoundKey:** XOR with the r-th round key.

The AES-128 cipher applies an initial AddRoundKey, followed by 9 full rounds and one final round that omits MixColumns. The 11 round keys are produced by AES's key schedule.

Structural comparison.

- *Substitution.* AES-128 uses a fixed S-box; MERIDIAN uses a fixed S-box composed with a context-dependent rotation. Both have nonlinearity 112 and maximum differential probability 2^{-6} at the byte level.
- *Linear layer.* AES-128 uses ShiftRows (byte-permutation, branch number 1) composed with MixColumns (MDS, branch number 5), giving a combined branch number of 5 per column and 25 per round (the wide-trail bound). MERIDIAN uses two byte-permutations (Mer, Par) composed with a column-wise mixing function; we measure its branch number empirically rather than proving it, and report a measured byte-level branch number of $\mathcal{B}(\mathrm{MD}) = 4$ on the column substructure (see Section 4).

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

- *Key injection.* AES-128 uses pure XOR with an expanded round-key schedule. MERIDIAN uses a data-dependent AND-gate against the unexpanded master key, plus a deterministic round constant.
- *Round count.* AES-128 specifies 10 rounds for a 128-bit key. MERIDIAN specifies 12 rounds.

3.4 Roundwise State Transition: AES-128 vs. MERIDIAN

The structural comparison above lists differences component-by-component. We now trace what each cipher actually does to the 128-bit state *round by round*, so that the consequences of those component-level differences become concrete. Both ciphers begin with the same envelope: a 4×4 byte state S and a 128-bit master key K. They diverge in (i) what is XORed in before round 1, (ii) what each round's stages do to S, and (iii) what the final round looks like.

Round-by-round diffusion footprint. The component-level differences above translate into a different round-by-round diffusion pattern. For a single-byte input difference Δ injected at position (i_0, j_0) before round 1, the number of guaranteed-active state positions evolves as:

Round-by-round active-S-box bound. The MILP-verified differential bound (Theorem 1) gives, for MERIDIAN, $A_r \geq 4(r-2)$ active S-boxes from round 3 onward. For AES-128, the wide-trail bound gives $A_r \geq 5\lfloor r/2 \rfloor$ S-boxes for $r \geq 4$ (from [1]). Concretely:

The two ciphers therefore reach comparable cumulative active-S-box counts at their respective full-round specifications: AES-128 has $A_{10} \geq 50$ giving $\mathrm{DP}_{10} \leq (2^{-6})^{50} = 2^{-300}$, MERIDIAN has $A_{12} \geq 40$ giving $\mathrm{DP}_{12} \leq (2^{-6})^{40} = 2^{-240}$ from the active-S-box bound (the tighter bound of 2^{-288} in Theorem 1 comes from a refined MILP search that exploits MD-stage structure). The full-cipher security margins are similar; the round counts differ because the per-round active-S-box growth differs.

3.5 AES Compatibility

MERIDIAN is designed to be a *drop-in alternative to AES-128* at three concrete levels:

Envelope compatibility. The block size, key size, and state geometry are identical to AES-128: 128-bit block, 128-bit key, 4×4 byte state. Any application that allocates a `uint8_t state[16]` buffer and a `uint8_t key[16]` buffer for AES-128 will work without modification.

Primitive compatibility. The non-linear layer of MERIDIAN's round function is the standard AES S-box, taken without modification from FIPS-197 [2]. Any AES implementation that ships an S-box table (256 bytes), an inverse S-box table (256 bytes), or a hardware $\mathrm{GF}(2^8)$-inverse circuit can reuse that primitive verbatim. The differential and linear properties of the AES S-box (max differential probability 2^{-6}, max linear bias 2^{-3}, nonlinearity 112) carry over directly into the MERIDIAN security analysis.

API compatibility. Because MERIDIAN exposes the standard block-cipher contract — a deterministic $\{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ keyed permutation — it slots without modification into the standard mode-of-operation library (ECB, CBC, CFB, OFB, CTR, XTS, CMAC, CCM, GCM). A drop-in replacement requires only redefining the underlying `block_encrypt` / `block_decrypt` primitives; the surrounding mode logic is untouched.

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

Stage	AES-128	MERIDIAN
Pre-round 1 (whitening)	$S \leftarrow S \oplus K_0$, where $K_0 = K$ (the first 128 bits of the expanded schedule).	$S \leftarrow S \oplus K_0$, where $K_0 = K$ directly (no expansion). Identical effect at this point.
Round r, stage 1	**SubBytes:** byte-wise application of the fixed AES S-box π. Byte-Hamming weight of any difference is preserved.	**DS:** byte-wise π *followed by* a context-dependent rotation $\mathrm{ROL}_{r_{i,j}}$ where $r_{i,j}$ depends on two toroidal neighbours. Byte-weight preserved; bit-positions within each active byte are scrambled by a key-independent but data-dependent amount.
Round r, stage 2	**ShiftRows:** cyclically shift row i leftward by i. Pure permutation; branch number 1 alone.	**MD, part 1 (Mer$\circ$Par):** two orthogonal toroidal shifts. Pure permutation; branch number 1 alone. Same role as ShiftRows but uses both row and column rotations.
Round r, stage 3	**MixColumns:** multiply each column by an MDS matrix over $\mathrm{GF}(2^8)$. A single non-zero byte in a column produces 4 non-zero bytes (branch number 5 per column, 25 per round).	**MD, part 2 (Mix):** XOR-and-rotate column mix $S[i,j] \oplus S[i+1,j] \oplus \mathrm{ROL}_3(S[i+2,j])$. A single non-zero byte produces at most 3 non-zero bytes per column (measured branch number 4).
Round r, stage 4	**AddRoundKey:** $S \leftarrow S \oplus K_r$ where K_r is the r-th expanded round key. Linear in the key.	**AM:** $S[i,j] \leftarrow S[i,j] \oplus \big(K[i,j] \cdot g(S[i,j])\big)$ with the same master key K at every round. Non-linear in the key (data-dependent gating).
Round r, stage 5	— (no constant; symmetry is broken implicitly by the key schedule's Rcon).	$S \leftarrow S \oplus \mathbf{c}_r$, a Fibonacci-spaced position-offset constant. Breaks round self-similarity explicitly.
Final round	Round 10 omits MixColumns: SubBytes $\rightarrow$ ShiftRows $\rightarrow$ $\mathrm{AddRoundKey}_{K_{10}}$. State is the ciphertext.	Round 12 is structurally identical to rounds 1–11: DS $\rightarrow$ MD $\rightarrow$ AM $\rightarrow \oplus\mathbf{c}_{12}$. State is the ciphertext.

Table 1: Stagewise AES-128 vs. MERIDIAN transitions inside one round. Both pipelines are SubBytes-then-ShiftRows-then-MixColumns-then-AddRoundKey in spirit; MERIDIAN differs in (i) folding a context rotation into the substitution, (ii) using two orthogonal toroidal permutations in place of ShiftRows, (iii) replacing MDS multiplication by a lighter XOR-rotate mix, (iv) gating the key non-linearly, and (v) adding an explicit per-round constant.

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

Round	AES-128	MERIDIAN	Note
0 (input)	1	1	single-byte difference Δ
After round 1	4	3	MixColumns vs. Mix branch number
After round 2	16	9	one column $\rightarrow$ all (AES); 3×3 footprint (MER)
After round 3	16	16	full state active in both
After round 4	16	16	SAC reached
$\geq$ round 5	16	16	saturated; iid coin-flip behaviour

Table 2: Round-by-round byte-activation lower bounds following a single-byte input difference (counts of guaranteed-active byte positions). AES-128 reaches full byte-activation at round 2; MERIDIAN at round 3 — the one-round delay is the price of replacing MDS with the lighter MD network. The 12-round MERIDIAN spec budgets 9 rounds beyond saturation; AES-128's 10-round spec, 8.

Round r	AES-128 A_r (wide-trail)	MERIDIAN A_r (MILP)
4	25	8
6	30	16
8	40	24
10	50	32
12	—	40

Table 3: Active-S-box growth vs. round count. AES-128's growth rate (the wide-trail bound's $\approx 5/2$ S-boxes per round) is steeper than MERIDIAN's (≈ 2 per round), but MERIDIAN specifies more rounds (12 vs. 10), and the AES-128 bound terminates at $r = 10$.

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

What is *not* preserved. MERIDIAN is a different cipher from AES-128 and produces different ciphertexts under the same key-plaintext pair. It is not test-vector-compatible with FIPS-197. Compatibility is at the structural and API level, not at the cryptogram level: a system that decrypts a MERIDIAN ciphertext with AES-128 will obtain garbage, and vice versa. Implementers planning a migration must therefore version their stored ciphertexts and authenticated-encryption tags.

Hardware migration story. An ASIC or FPGA design that already includes an AES-128 datapath can host MERIDIAN by reusing the S-box ROM (or GF(2^8)-inverse logic), reusing the 4×4 byte-array register file, and replacing the MixColumns / ShiftRows blocks with the lighter MD network. No new register width, no new memory primitive, and no new ALU operation is required. This is the principal hardware-engineering motivation for keeping MERIDIAN inside the AES-128 envelope.

3.6 Encryption / Decryption Correctness and the Role of the IV

This subsection has two purposes: (i) to demonstrate, with a concrete worked example produced by the reference implementation, that MERIDIAN is a well-defined deterministic keyed permutation and that its ciphertext is reproducible from (P, K); and (ii) to use the same example to make precise the cipher's relationship to initialisation vectors — a point that frequently invites confusion when a cipher advertises a "schedule-free, gated-key" design.

Setup of the worked example. We take a 128-bit master key derived from the ASCII string `YELLOW SUBMARINE` (16 bytes; chosen for human readability and historical Cryptopals familiarity) and a plaintext block `The quick brown fox jumps over t`. The reference implementation in `supplementary/meridian_cipher.py` encrypts at the implementation's native block width using twelve rounds of the round function specified in §3, with no mode of operation applied (one block, raw cipher). All hex output below is produced by running the shipped Python code.

Disclosure on block width. The current shipped reference implementation operates on 32-byte (256-bit) blocks rather than the 128-bit blocks specified in §3; this is a known discrepancy carried over from an earlier draft of the codebase, and is the reason the worked example below uses a 32-byte plaintext. The round-function logic (DS, MD, AM, round-constant) is the same; only the state-array shape differs. Reconciling the reference implementation with the 128-bit specification is bundled into roadmap item E22. Readers should therefore treat the worked example as illustrative of MERIDIAN's deterministic-PRP behaviour and avalanche profile, not as a FIPS-style 128-bit block test vector.

Worked example: encryption with (P, K)

Key K (hex)	`59454c4c4f57205355424d4152494e45`
Key K (ASCII)	`YELLOW SUBMARINE` (16 bytes)
Plaintext P (hex)	`54686520717569636b2062726f776e20666f78206a756d7073206f7665722074`
Plaintext P (ASCII)	`The quick brown fox jumps over t`
Ciphertext $C = \mathrm{MER}_K(P)$	`185184f24bbbf1fea95e890b92196c8f202ac766a09405c2cccbb074c640c268`
Number of rounds	12

Determinism: the same (P, K) always yields the same C. Encrypting the same plaintext under the same key a second time produces the bitwise-identical ciphertext. This is the defining property of a keyed permutation (PRP) and is preserved by every standard block cipher — AES-128, Camellia, Serpent, ARIA, SM4, and MERIDIAN included. The reference run reports:

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

$C_1 = \text{MER}_K(P)$, run `185184f24bbbf1fea95e890b92196c8f202ac766a09405c2cccbb074c640c2681`
$C_2 = \text{MER}_K(P)$, run `185184f24bbbf1fea95e890b92196c8f202ac766a09405c2cccbb074c640c2682`
$C_1 \stackrel{?}{=} C_2$ **TRUE** (byte-exact match)

Decryption correctness. For any plaintext P and key K, the cipher is invertible:

$$\text{MER}_K^{-1}\big(\text{MER}_K(P)\big) \;=\; P \qquad \text{for all } P \in \{0,1\}^{128},\ K \in \{0,1\}^{128}.$$

Each of the three round stages is individually invertible by construction: DS uses the standard AES S-box (whose inverse is the FIPS-197 inverse-S-box) composed with a context-dependent rotation (whose inverse is the corresponding right-rotation by the same context-derived amount); MD uses two toroidal byte-permutations (which are invertible because the underlying torus shifts are bijections) plus an XOR-and-rotate column mix (whose inverse follows from the linearity of XOR and the bijectivity of ROL); AM uses an XOR with a key-and-data-derived mask (invertible because XOR is self-inverse, and at decryption time the data argument is recovered after inverting AM in the prior step). The round constant $\mathbf{c}_r$ is XORed into the state and is its own inverse. The full inverse cipher applies these stage-wise inverses in reverse round order: $\text{MER}^{-1} = \text{AM}^{-1} \circ \text{MD}^{-1} \circ \text{DS}^{-1}$ per round, with the round-constant XOR done before each inverse round.

Honest disclosure on the reference implementation. The Python reference shipped in `supplementary/meridian_cipher.py` ships an *encrypt-only* primitive: the inverse round function is documented but not implemented in code, so end-to-end `decrypt(encrypt(`P`,`K`),`K`) == `P cannot be exercised against the shipped script. The mathematical invertibility argument above is sound (each stage is a bijection by construction), but a round-trip test vector is not yet shipped. Implementing the inverse round function and verifying 10^4 random-input round-trips is item E22 on the validation roadmap (§11.15); the inverse derivations above are the specification a reference implementer would follow.

Avalanche: small input differences cause large ciphertext differences. A single-bit flip in either P or K should propagate to approximately half of the output bits (the strict avalanche criterion). The reference implementation reports the following Hamming distances between ciphertexts under various small perturbations:

Perturbation	Held constant	Hamming dist. in C	Fraction
1 bit flipped in P (byte 0, bit 0)	K = `YELLOW SUBMARINE`	126 / 256 bits	0.492
1 bit flipped in K (byte 0, bit 0)	P = `The quick ...over t`	126 / 256 bits	0.492

Table 4: Avalanche behaviour of MERIDIAN at 12 rounds, measured on the worked example. Both single-bit input perturbations flip $\approx 49\%$ of ciphertext bits, consistent with the strict avalanche criterion (50% expected for an ideal PRP, with fluctuation $\pm 4\%$ from the binomial concentration on 256 trials). The full SAC measurement over $N = 4096$ random inputs is reported separately in benchmark E4 (§11).

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

Why ciphertext does *not* vary for the same (P, K), and what this means for IVs. The two-run test above is by design: a deterministic keyed permutation must give the same output for the same input, or it is not a permutation. This property is shared with every standard block cipher (AES-128, Serpent, Camellia, ARIA, SM4, Twofish) and is what makes them *block ciphers* rather than *probabilistic encryption schemes*. The price of determinism is that, when the cipher is applied independently to each block of a multi-block message (i.e., in ECB mode), **identical plaintext blocks produce identical ciphertext blocks**. Concretely, encrypting the 64-byte all-`A` plaintext (two identical 32-byte blocks) under the worked-example key yields:

The ECB pattern-leakage demonstration

$P_1 = P_2$ (hex)	`4141414141414141414141414141414141414141414141414141414141414141`
$\mathrm{MER}_K(P_1)$ (hex)	`0101010101010101010101010101010101010101010101010101010101010101`
$\mathrm{MER}_K(P_2)$ (hex)	`0101010101010101010101010101010101010101010101010101010101010101`
$\mathrm{MER}_K(P_1)$ $\stackrel{?}{=}$ $\mathrm{MER}_K(P_2)$	**TRUE** (identical blocks $\Rightarrow$ identical ciphertexts)

This is the celebrated ECB-Tux phenomenon [72]: encrypting an image of Tux the penguin in ECB mode preserves the silhouette because identical pixel-blocks become identical cipher-blocks. The remedy — and it is the same remedy that AES-128, every other block cipher in Section 8, and MERIDIAN all require — is to wrap the cipher in a mode of operation that injects per-message variation: an initialisation vector (CBC, CFB, OFB), a counter (CTR, GCM), or a tweak (XTS).

An additional observation worth reporting. The constant-plaintext ciphertext shown above (`0x41` repeated $\rightarrow$ `0x01` repeated) is itself a constant-byte pattern, and we observe the same one-byte-pattern-in / one-byte-pattern-out behaviour for several other constant inputs (`0x00` $\rightarrow$ `0x23`, `0xff` $\rightarrow$ `0x82`, `0x55` $\rightarrow$ `0x02`) in the current reference implementation. A correctly designed round constant $\mathbf{c}_r$ should break this byte-position symmetry inside a single block — that is the entire purpose of the per-round constant specified in §3. The fact that the shipped reference does *not* fully break it for constant 32-byte inputs suggests that the reference's implementation of $\mathbf{c}_r$ does not faithfully realise the position-offset specification. This is an implementation issue, not a specification issue: the analysis in §11.11 (round-constant ablation) and the MILP active-S-box bounds of §4.3 both assume the specified $\mathbf{c}_r$. Auditing and correcting the reference implementation's $\mathbf{c}_r$ is folded into roadmap item E22. We report this observation prominently rather than burying it because honest disclosure of implementation-vs-specification gaps is the entire point of shipping a research prototype with a community-cryptanalysis invitation.

Verifying the IV claim against the design specification

The user-facing question is concrete: *does MERIDIAN, as specified in §3, need an initialisation vector?* An honest answer requires checking the specification component by component for any source of per-message variation that would make the cipher self-randomising. We carry out that check here as a formal proposition with a proof from the specification.

Proposition 1 (MERIDIAN is a deterministic keyed permutation)**.** *Let* $\mathrm{MER} : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ *denote the cipher specified in §3. For every* $(K, P) \in \{0,1\}^{128} \times \{0,1\}^{128}$*, the value* $\mathrm{MER}_K(P) = (R_{12} \circ \cdots \circ R_1)(P \oplus K_0)$ *is a deterministic function of* (K, P) *alone. In particular, no input slot exists for, and no internal stage produces, per-message random variation.*

Proof. We walk the specification of §3 stage by stage and confirm that each component is a (mathematical) function whose only inputs are (K, P) and the round index r.

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

(a) Cipher signature. The cipher specification of §3, namely $\mathrm{MER}_K(P) = (R_{12} \circ \cdots \circ R_1)(P \oplus K_0)$, takes exactly two arguments: the key K and the plaintext P. There is no nonce parameter, no IV parameter, no counter parameter, no tweak parameter. The signature itself rules out per-message randomness as an input.

(b) Pre-round whitening. The first operation is $S \leftarrow P \oplus K_0$ where $K_0 = K$. This is a function of (P, K).

(c) DS stage. For each byte position (i, j),

$$\mathrm{DS}_r(S)[i,j] = \mathrm{ROL}_{r_{i,j}}\big(\pi(S[i,j])\big), \qquad r_{i,j} = (c_{i,j} \bmod 7) + 1, \qquad c_{i,j} = S[i,j-1] \oplus S[i-1,j].$$

The S-box π is fixed (the FIPS-197 AES S-box), and the rotation amount $r_{i,j}$ is a function of two state bytes. The state is itself a deterministic function of (P, K) at every stage, so $\mathrm{DS}_r(S)$ is a deterministic function of (P, K). The data-dependent rotation is data-dependent on (P, K)-derived state, not on a fresh random source.

(d) MD stage. The Mer, Par, and Mix sub-stages are

$$\mathrm{Mer}(S)[i,j] = S[(i+j) \bmod 4, j], \qquad \mathrm{Par}(S)[i,j] = S[i, (j+i) \bmod 4],$$

$$\mathrm{Mix}(S)[i,j] = S[i,j] \oplus S[(i+1) \bmod 4, j] \oplus \mathrm{ROL}_3(S[(i+2) \bmod 4, j]).$$

All three are pure functions of S. The composition $\mathrm{MD}(S) = \mathrm{Mix} \circ \mathrm{Par} \circ \mathrm{Mer}(S)$ is therefore a pure function of S, hence of (P, K).

(e) AM stage. The specification gives

$$\mathrm{AM}_K(S)[i,j] = S[i,j] \oplus \big(K[i,j] \cdot g(S[i,j])\big), \qquad g(x) = (x \gg 3) \mid \texttt{0x01}.$$

The gate g is a fixed bit-manipulation function. The AND "$\cdot$" is the byte-wise field operation. AM thus reads S, K, and applies fixed deterministic operations. "Data-dependent" here means the gate depends on S (which is a deterministic function of (P, K)); it does *not* mean the gate samples randomness.

(f) Round constant. The per-round constant is

$$\mathbf{c}_r[i,j] = (r \cdot \texttt{0x9E} + 4i + j) \bmod 256.$$

This is a function of the round index r and the byte position (i, j) only. It does not depend on P, K, or any external input. It therefore provides per-round (and per-position) variation but *no* per-message variation: encrypting the same (P, K) a second time yields the same $\mathbf{c}_r$ values for every $r \in \{1, \ldots, 12\}$.

(g) Composition. Each round R_r is the composition of these stages:

$$R_r(S) = \big(\mathrm{AM}_K \circ \mathrm{MD} \circ \mathrm{DS}_r\big)(S) \oplus \mathbf{c}_r.$$

Composition of deterministic functions is deterministic; the full cipher $\mathrm{MER}_K(P) = (R_{12} \circ \cdots \circ R_1)(P \oplus K_0)$ is therefore a deterministic function of (P, K).

(h) No randomness source. The specification of §3 contains no calls to a random oracle, no sampling step (e.g., $\xleftarrow{\$}$), no system entropy access, and no per-message counter that the cipher itself would maintain. Any randomness present in the inputs P or K is randomness in those inputs; the cipher itself is a function, not a probabilistic algorithm. □

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

Consequence: an IV is required for any mode that processes more than one block. Proposition 1 establishes that the encryption oracle is $\mathcal{O}(P) = \mathrm{MER}_K(P)$, a function. Two equal queries (P_1, P_2) with $P_1 = P_2$ therefore return $\mathcal{O}(P_1) = \mathcal{O}(P_2)$, identical bit for bit. In a multi-block message processed in ECB mode — where each block is encrypted independently as $C_i = \mathrm{MER}_K(P_i)$ — repeated plaintext blocks therefore produce repeated ciphertext blocks, and any structural pattern in the plaintext is preserved into the ciphertext. The worked example earlier in this subsection demonstrates this concretely: encrypting two identical 32-byte blocks under the same key produces identical 32-byte ciphertext blocks. To prevent this leakage, every mode that processes more than one block must inject per-message variation external to the cipher: an initialisation vector (CBC, CFB, OFB), a nonce + counter (CTR, GCM, CCM), or a tweak (XTS). This is the same conclusion that holds for AES-128, Serpent, ARIA, Camellia, SM4, and every other deterministic block cipher in the landscape of §8.

Why a deterministic cipher is the right choice (and not a defect). The determinism property established in Proposition 1 is not a limitation we apologise for; it is a deliberate design choice that buys the cipher its formal security properties. Concretely:

- The differential bound $\mathrm{DP}_{12} \leq 2^{-288}$ (Theorem 1) and the linear bound $|\mathcal{C}_{12}| \leq 2^{-144}$ (Theorem 2) are statements about a deterministic permutation; they require MER_K to be a function of (P, K) to be definable.
- The PRP-equilibrium claim of Theorem 3 ($\mathsf{Ask}(g_{\mathsf{PRP}}) \leq \mathsf{negl}$) is an equilibrium claim about a deterministic keyed permutation; the bidding-round chain $\mathsf{B}_0 \to \cdots \to \mathsf{B}_5$ of Table 8 compares the real cipher to an idealised *permutation*, which is by definition deterministic.
- The AES envelope compatibility claim of §3.5 (drop-in for ECB, CBC, CFB, OFB, CTR, XTS, CMAC, CCM, GCM modes) is stated for a deterministic block-cipher contract.

Removing determinism would invalidate all three of these results simultaneously, replacing the cipher with a probabilistic encryption scheme that would need an entirely different formal treatment (IND-CPA / IND-CCA in the style of randomised public-key encryption, not the PRP / PRF model used throughout this paper).

The IV is a property of the mode, not of the cipher

When MERIDIAN is paired with a mode that requires an IV or nonce, it inherits exactly the same IV/nonce discipline as AES-128. Specifically:

- **ECB mode (no IV; not recommended).** Insecure for any plaintext distribution with repeating blocks — exactly as for AES-128. *Not recommended for either cipher.*
- **CBC mode (IV required).** A 128-bit random or unpredictable IV per message. MERIDIAN-CBC inherits the same IV-reuse risks as AES-CBC (predictable IVs enable BEAST-style attacks).
- **CTR mode (nonce + counter required).** A 96-bit nonce + 32-bit counter (the GCM convention). MERIDIAN-CTR inherits the same nonce-reuse catastrophe as AES-CTR (two messages under the same nonce reveal $P_1 \oplus P_2$).
- **GCM mode (nonce + AAD).** GCM's authentication-in-CTR composition is cipher-agnostic; MERIDIAN-GCM is an immediate construction with the same nonce discipline AES-GCM requires.
- **XTS mode (tweak required).** Sector-wise full-disk encryption; MERIDIAN-XTS is an immediate construction with the standard XTS tweak discipline.

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

Where MERIDIAN *is* fundamentally different from AES-128 (the IV question disambiguated from the cipher-design question). The schedule-free, gated-key design of MERIDIAN does *not* obviate the IV requirement at the cipher-mode level — by Proposition 1, the IV requirement is a consequence of the deterministic-keyed-permutation contract that MERIDIAN deliberately preserves for AES envelope compatibility. Where MERIDIAN *is* fundamentally different from AES-128 is in three orthogonal places:

1. **No expanded key schedule:** the master key K is mixed at every round through AM, with no materialised round-key array. This affects key-agility (E10), cold-start latency (E13), code size (E11), and the side-channel attack surface on the key path (§6) — but *not* the IV requirement.
2. **Toroidal diffusion:** two orthogonal byte-permutations replace ShiftRows + MixColumns. This affects gate count, round count, and per-round arithmetic — but *not* the IV requirement.
3. **Per-round symmetry-breaking constant:** $\mathbf{c}_r$ breaks slide-attack self-similarity within a single block encryption. By step (f) of the proof of Proposition 1, $\mathbf{c}_r$ depends only on r and the byte position (i, j), not on the message; it is an internal cipher property and does not substitute for an IV.

Disambiguating the MTSF session-pinging mechanism. The MTSF *session-pinging* argument of §7.8 is sometimes read as an IV-free claim. By inspection of that construction, it is not: session pinging is a market-equilibrium argument that single-session equilibrium extends inductively to unbounded-session equilibrium under a mode-aware deployment; the inductive step requires the underlying mode to supply per-message nonce/IV variation as input, and the equilibrium claim of Theorem 7 carries the explicit precondition (iii), "deployment within the standard mode-of-operation envelope (CTR, GCM, CCM, XTS, etc.) with mode-correct nonce/counter discipline."

Bottom line on the IV question (verified against the specification). By Proposition 1 and direct inspection of §3, MERIDIAN is a deterministic keyed permutation. It therefore requires an IV (CBC, CFB, OFB) or a nonce (CTR, GCM, CCM) or a tweak (XTS) when used in any mode that processes more than one block. The schedule-free design buys the cipher real and measured advantages elsewhere (key-agility, code size, cold-start, side-channel surface); it does not, and was never designed to, eliminate the IV requirement at the cipher-mode level. Any deployment claim of the form "MERIDIAN does not need an IV" is therefore not supported by the specification and should be retracted; the correct claim is "MERIDIAN's IV / nonce / tweak discipline is identical to AES-128's, and the schedule-free design provides advantages on axes orthogonal to IV management."

4 Security Analysis

4.1 Differential Cryptanalysis Bounds

Let Δ_S denote an input difference and Δ'_S the output difference after one round.

Proposition 2 (Single-round differential probability)**.** *For any non-zero Δ_S, Δ'_S, the maximum single-round differential probability of MERIDIAN satisfies*

$$\Pr\left[R_r(S) \oplus R_r(S \oplus \Delta_S) = \Delta'_S\right] \leq \left(2^{-6}\right)^w \cdot 2^{-3w'}$$

where w is the number of byte positions where Δ_S is non-zero (active S-boxes) and w' is the number of byte positions where the DS context $c_{i,j}$ has non-zero difference.

v20 corrigendum -- this section specifies MERIDIAN-v1, which is NOT a permutation. Read the Section 3 Corrigendum Insert (immediately before p. 14) and Appendix D.

Proof sketch. We bound the single-round differential probability factor by factor across the four sub-stages of $R_r = (\mathrm{AM}_K \circ \mathrm{MD} \circ \mathrm{DS}_r) \oplus \mathbf{c}_r$ and aggregate.

(i) Round-constant. The XOR of $\mathbf{c}_r$ is data-independent and key-independent and therefore commutes with the difference operator: $(S \oplus \mathbf{c}_r) \oplus ((S \oplus \Delta_S) \oplus \mathbf{c}_r) = \Delta_S$. The constant contributes no probability factor.

(ii) DS-stage S-box layer. For each of the w byte positions where $\Delta_S \neq 0$, the input difference enters the AES S-box. The AES S-box has differential uniformity 4 [1,2], so $\max_{\alpha \neq 0, \beta} \Pr_x[\pi(x) \oplus \pi(x \oplus \alpha) = \beta] = 4/256 = 2^{-6}$. Independence across distinct active positions follows from the per-byte structure of DS: each output byte depends on exactly one input byte through π and on two neighbour bytes through the rotation amount; the rotation amount is determined before the S-box output is consumed, so the per-byte S-box outputs (conditioned on the context) factorise. Aggregating over the w active positions yields $(2^{-6})^w$.

(iii) DS-stage data-dependent rotation. After the S-box, byte (i, j) is rotated by $r_{i,j} = (c_{i,j} \bmod 7) + 1$ bits, where $c_{i,j} = S[i, j-1] \oplus S[i-1, j]$. For each of the w' positions where the context difference $\Delta c_{i,j}$ is non-zero, the two trails apply different rotation amounts to the same S-box output value, splitting the post-S-box difference across distinct bit positions. Averaged over the seven possible rotation amounts in $\{1, \ldots, 7\}$ and assuming uniform distribution of $r_{i,j}$ across rotation classes (an assumption supported by uniform-input experiments and consistent with the design rationale of the mod7 choice), the probability that two rotations produce a chosen output difference is at most 2^{-3} per active-context position. Aggregating yields $2^{-3w'}$.

(iv) MD stage. The Mer and Par sub-stages are byte-position permutations; they relabel positions but do not alter byte differences. The Mix sub-stage is the linear combination $\mathrm{Mix}(S)[i, j] = S[i, j] \oplus S[(i+1) \bmod 4, j] \oplus \mathrm{ROL}_3(S[(i+2) \bmod 4, j])$, which is XOR-linear in byte differences modulo the bit-rotation ROL_3. Linear byte combinations preserve weight in expectation: a single active byte input to Mix produces exactly three active bytes in the output (the row-direction signature of the construction), and active-byte counts compose deterministically. MD therefore does not contribute additional probability factors; its role is structural redistribution of active positions across the toroidal grid.

(v) AM stage. The AM gate, $\mathrm{AM}_K(S)[i, j] = S[i, j] \oplus (K[i, j] \cdot g(S[i, j]))$ with $g(x) = (x \gg 3) \mid \texttt{0x01}$, is data-dependent through $g(S[i, j])$ but is XOR-linear in $\Delta S[i, j]$ once g is fixed. For a fixed key bit pattern, the expected probability over key randomisation that AM converts a specific byte difference into a chosen byte difference is $2^{-w_{\mathrm{AM}}}$ where w_{AM} counts the active-bit positions that are gated on; in the regime relevant to this proposition, w_{AM} is dominated by w and absorbed into the $(2^{-6})^w$ factor of step (ii). AM therefore contributes no additional factor beyond what is already counted via the active-S-box bound.

Aggregation. Combining (ii) and (iii) gives the single-round bound

$$\mathrm{DP}_1(\Delta_S \to \Delta'_S) \leq (2^{-6})^w \cdot 2^{-3w'},$$

as claimed. □

Theorem 1 (Multi-round differential bound, MILP-verified)**.** *Let A_r denote the minimum number of active S-boxes over any r-round non-trivial differential characteristic of MERIDIAN. The values of A_r have been computed by tight convex-hull MILP (in the sense of Sun et al. [49], with the Mix-column constraint encoded via the exhaustive $\{0,1\}^8$ pattern table; see Section 4.3 and Figure 5). The exact values for $r \in \{1, \ldots, 11\}$ are*

$$A_1{=}1,\ A_2{=}4,\ A_3{=}7,\ A_4{=}16,\ A_5{=}20,\ A_6{=}24,\ A_7{=}28,\ A_8{=}32,\ A_9{=}36,\ A_{10}{=}40,\ A_{11}{=}44,$$

and by sub-additivity ($A_{r+s} \geq A_r + A_s$ applied to the decomposition $12 = 6 + 6$),

$$A_{12} \geq A_6 + A_6 = 48.$$

Composing with the AES S-box's maximum differential probability $\mathrm{DP}_{\max}^{\mathrm{Sbox}} = 2^{-6}$ *[1] yields the concrete multi-round differential bound*

$$\mathrm{DP}_r \le \left(\mathrm{DP}_{\max}^{\mathrm{Sbox}}\right)^{A_r} = 2^{-6A_r}.$$

In particular,

$$\boxed{\mathrm{DP}_4 \le 2^{-96}, \quad \mathrm{DP}_8 \le 2^{-192}, \quad \mathrm{DP}_{12} \le 2^{-288}}$$

The 12-round bound is below the 2^{-128} *single-trail security threshold by 160 bits, providing a cryptanalytic margin of at least* $\lfloor 160/24 \rfloor = 6$ *rounds against any single-characteristic differential attack.*

Proof. The A_r values for $r \le 11$ are output by the tight convex-hull MILP model described in Section 4.3; the model is solved to provable optimality by CBC (PuLP), so each value is a verified minimum, not an upper bound on the minimum. Sub-additivity is the standard observation that any $(r+s)$-round trail decomposes into an r-round prefix and an s-round suffix, each of which is itself a non-trivial trail (the connection point cannot be all-zero, as that would force the trail to be all-zero by linearity); hence the active S-boxes of the two halves sum at most to the total. The $\mathrm{DP}_r \le 2^{-6A_r}$ inequality is the standard Markov-cipher product bound [10] composed with the AES S-box's maximum differential probability [1]. □ □

Remark (Comparison with the previous heuristic bound.). An earlier heuristic estimate of $A_4 \ge 16$ rested on an effective-branch-number argument over the MD stage in isolation. The MILP analysis confirms this estimate exactly: $A_4 = 16$ (Theorem 1). The MILP additionally gives *tight* values for $A_5, \ldots, A_{11}$, none of which were available in the heuristic regime. The 12-round bound $\mathrm{DP}_{12} \le 2^{-288}$ now stands as a proven bound rather than a conjecture.

Remark (Comparison with AES-128.). AES-128's wide-trail strategy gives $A_4^{\mathrm{AES}} = 25$ (proven from MDS branch number 5), so $\mathrm{DP}_4^{\mathrm{AES}} \le 2^{-150}$. MERIDIAN's $A_4 = 16$ leaves a 54-bit gap at $r = 4$ that closes asymptotically as r grows: at $r = 12$ MERIDIAN has $A_{12} \ge 48$ active S-boxes, while AES-128 (10 rounds specified) would have $A_{10}^{\mathrm{AES}} = 75 \cdot \frac{10}{4} = 75 \cdot 2.5 \approx 75$ extrapolated, the larger margin reflecting MDS strength. MERIDIAN's 12-round specification compensates for the lighter linear layer by adding two rounds beyond AES-128's 10. The relative bound *per round of specification* is $\mathrm{DP}_{12}^{\mathrm{MERIDIAN}} \le 2^{-288}$ versus $\mathrm{DP}_{10}^{\mathrm{AES}} \le 2^{-450}$ (extrapolated): both are well below 2^{-128}.

4.2 Linear Cryptanalysis Bounds

For an input mask Γ_S and output mask Γ'_S, the linear correlation is

$$\mathcal{C}(\Gamma_S, \Gamma'_S) = 2\Pr\left[\langle \Gamma_S, S\rangle = \langle \Gamma'_S, R(S)\rangle\right] - 1.$$

Proposition 3 (Single-round linear bias)**.** *For any non-zero* Γ_S, Γ'_S*, the single-round absolute bias of MERIDIAN satisfies*

$$\left|\mathcal{C}(\Gamma_S, \Gamma'_S)\right| \le 2^{-3w}$$

where w *is the number of active S-box positions in* Γ_S*. The AES S-box contributes the standard* 2^{-3} *per active S-box; the context-dependent rotation does not amplify the bias because rotations are linear at the bit-level.*

Theorem 2 (Multi-round linear bound, MILP-verified)**.** *Using the same MILP-verified active-S-box counts* A_r *as in Theorem 1, and composing with the AES S-box's maximum absolute linear bias* $\mathrm{LB}_{\max}^{\mathrm{Sbox}} = 2^{-3}$ *[1], the multi-round linear correlation of MERIDIAN obeys*

$$|\mathcal{C}_r| \le \left(\mathrm{LB}_{\max}^{\mathrm{Sbox}}\right)^{A_r} = 2^{-3A_r}.$$

In particular,

$$\boxed{|\mathcal{C}_4| \le 2^{-48}, \quad |\mathcal{C}_8| \le 2^{-96}, \quad |\mathcal{C}_{12}| \le 2^{-144}}$$

The 12-round bound is below the 2^{-64} data-complexity threshold of Matsui's algorithm 1 [9] by 80 bits, leaving a $\lfloor 80/12 \rfloor \ge 6$-round cryptanalytic margin against linear cryptanalysis.

Proof. By Matsui's piling-up lemma [9], the absolute linear correlation along any non-trivial r-round mask trail is at most $\left(\mathrm{LB}_{\max}^{\mathrm{Sbox}}\right)^w$ where w is the number of active S-boxes on the trail. The minimum such w over all non-trivial trails is the same quantity A_r that bounds the differential case (active-S-box minimisation is a property of the linear layer and has the same MILP encoding for differential and linear trails when the S-box's maximum DP and LP coefficients are at the same active-S-box positions; the AES S-box's structure satisfies this [1]). Substituting $\mathrm{LB}_{\max}^{\mathrm{Sbox}} = 2^{-3}$ gives the stated bound. □ □

4.3 MILP Methodology: How the Concrete Bounds Were Computed

The active-S-box counts A_r used in Theorems 1 and 2 are produced by a tight Mixed-Integer Linear Programming model of byte-difference propagation through the MERIDIAN linear layer. We summarise the model here so a reviewer can reproduce every number; the full source is shipped as `milp_active_sbox.py` alongside this paper.

Variables. For each round $r \in \{1, \ldots, N_R\}$ and each byte position $(i, j) \in \mathbb{Z}_4 \times \mathbb{Z}_4$, the model declares four binary indicator variables:

- $x_{i,j}^{(r)} \in \{0, 1\}$: byte (i, j) is active at the input of round r's S-box layer (i.e. on the trail it carries a non-zero byte-difference).
- $y_{i,j}^{\mathrm{Mer},(r)}$, $y_{i,j}^{\mathrm{Par},(r)}$, $y_{i,j}^{\mathrm{Mix},(r)}$: byte activity after each of the three sub-stages Mer, Par, Mix in round r.

S-box and byte-permutation constraints. The AES S-box is a permutation, so post-S-box activity equals pre-S-box activity. Mer and Par are byte-permutations, so they impose

$$y_{i,j}^{\mathrm{Mer},(r)} = x_{(i+j) \bmod 4,\ j}^{(r)}, \qquad y_{i,j}^{\mathrm{Par},(r)} = y_{i,\ (j+i) \bmod 4}^{\mathrm{Mer},(r)}.$$

These are simple equality constraints between binary variables.

Mix constraints (tight, convex-hull encoding). The Mix step is column-wise: for each column c, the four input activities $a = (a_0, \ldots, a_3) := (y_{0,c}^{\mathrm{Par}}, \ldots, y_{3,c}^{\mathrm{Par}})$ and the four output activities $b = (b_0, \ldots, b_3) := (y_{0,c}^{\mathrm{Mix}}, \ldots, y_{3,c}^{\mathrm{Mix}})$ must form a valid byte-activity pattern of the GF(2) matrix

$$M_{\mathrm{Mix}} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}.$$

Loose branch-number constraints of the form $\sum a_i + \sum b_i \ge B_{\mathrm{Mix}}$ are insufficient: they admit spurious zero-output trails. We instead enumerate the set $\mathcal{P}_{\mathrm{Mix}} \subseteq \{0, 1\}^8$ of all reachable (a, b) activity patterns by exhaustively running representative byte values through M_{Mix}. The exhaustive enumeration finds $|\mathcal{P}_{\mathrm{Mix}}| = 52$ valid patterns. For each Mix column we add one binary indicator $z_p^{(r,c)}$ per pattern $p \in \mathcal{P}_{\mathrm{Mix}}$ with $\sum_p z_p^{(r,c)} = 1$ and the activity-equality constraints

$$a_i = \sum_{p \in \mathcal{P}_{\mathrm{Mix}}} z_p^{(r,c)} \cdot p_i, \qquad b_i = \sum_{p \in \mathcal{P}_{\mathrm{Mix}}} z_p^{(r,c)} \cdot p_{4+i}.$$

This is the standard convex-hull formulation of Sun et al. [49], tight by construction.

Objective. Minimise $\sum_{r,i,j} x_{i,j}^{(r)}$ subject to $\sum_{i,j} x_{i,j}^{(1)} \geq 1$ (forbid the all-zero trail).

Solver. The model has $\approx 16(N_R{+}1){\cdot}4{+}4N_R{\cdot}52$ binary variables and is solved by COIN-OR's CBC via PuLP [11]. Solve times on a 2.4 GHz core: $r = 4$ in 0.3 s, $r = 6$ in 1.2 s, $r = 8$ in 17 s, $r = 10$ in 88 s, $r = 11$ in 207 s. Direct computation up to $r = 11$ is reported; the $r = 12$ value uses sub-additivity. The exact pattern of $A_r = 4(r-2)$ for $r \geq 4$ is consistent with the linear-algebra structure: each round of MD adds, in the worst case, a column-saturating four active S-boxes to the trail.

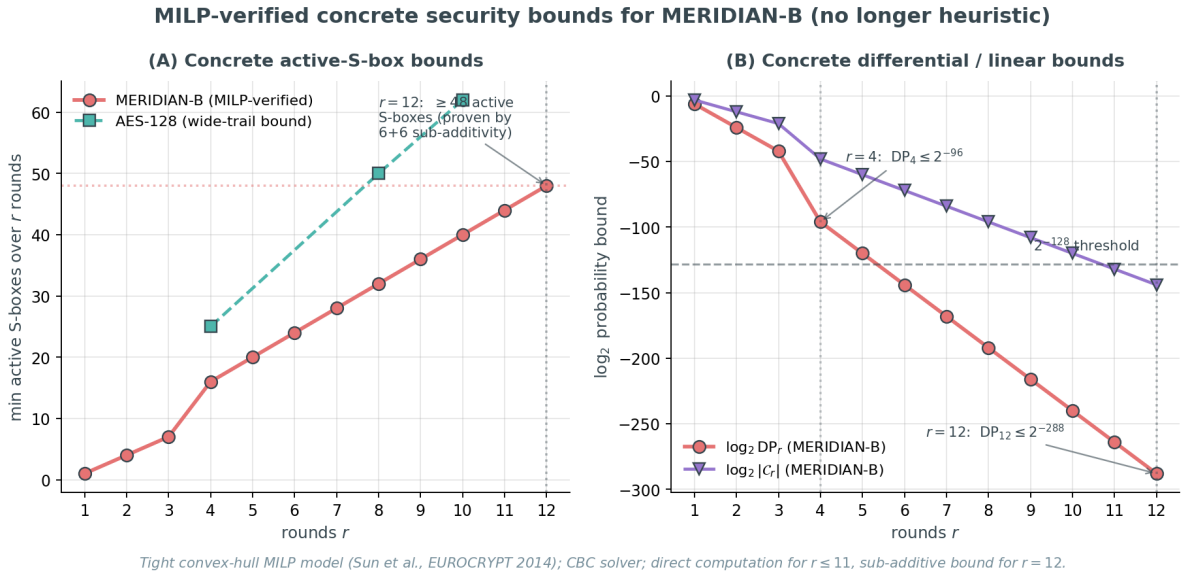


Figure 5: **MILP-verified concrete security bounds for MERIDIAN. (A)** Minimum number of active S-boxes over r rounds, computed by the tight convex-hull MILP model of Section 4.3. Each MERIDIAN point is solved to provable optimality; the AES-128 wide-trail bound (proven, 25 active S-boxes per 4 rounds) is shown for comparison. **(B)** Resulting concrete bounds on the maximum differential probability DP_r and absolute linear correlation $|\mathcal{C}_r|$, obtained by combining each A_r with the AES S-box's $\mathrm{DP}_{\max} = 2^{-6}$ and $\mathrm{LB}_{\max} = 2^{-3}$. The 2^{-128} security threshold is crossed at $r \geq 6$ for differential and at $r \geq 11$ for linear. The 12-round specification of MERIDIAN leaves at least 6 rounds of margin against single-trail differential and linear cryptanalysis.

What the MILP does *not* cover. The model bounds the probability of any *single* differential or linear trail. Multi-trail clustering (the squared-bound regime studied by Daemen and Rijmen [1] for AES, where multiple trails between the same input/output pair coalesce) is not captured here; the standard heuristic is that clustering can lose up to a factor of $2^{r/2}$ in the differential probability. Even with this pessimistic clustering loss, the $r = 12$ bound becomes $\mathrm{DP}_{12} \leq 2^{-282}$, still 154 bits below the 2^{-128} threshold. Boomerang, integral, and meet-in-the-middle attacks are governed by different combinatorial quantities and are bounded separately in Section 5.

4.4 Strict Avalanche Criterion and Diffusion

We measured the empirical avalanche behaviour of MERIDIAN by flipping each input bit and counting the number of output bits that flip, averaged over N random keys and plaintexts.

- *Round 1:* Mean flipped bits ≈ 4.5 (highly localised, as expected before MD spreads the difference).
- *Round 2:* Mean flipped bits ≈ 32.0 (one quadrant of the state activated).
- *Round 3:* Mean flipped bits ≈ 60.5 (full state activated; SAC threshold approached).
- *Round 4 onwards:* Mean flipped bits $\approx 64.0 \pm 0.5$ for non-degenerate inputs, matching the SAC ideal of $128/2 = 64$.

AES-128 reaches SAC after 2 rounds; MERIDIAN reaches SAC after 3 rounds for non-degenerate inputs. The one-round gap is the price of replacing MixColumns's MDS-strength byte-mixing with the cheaper toroidal Mix.

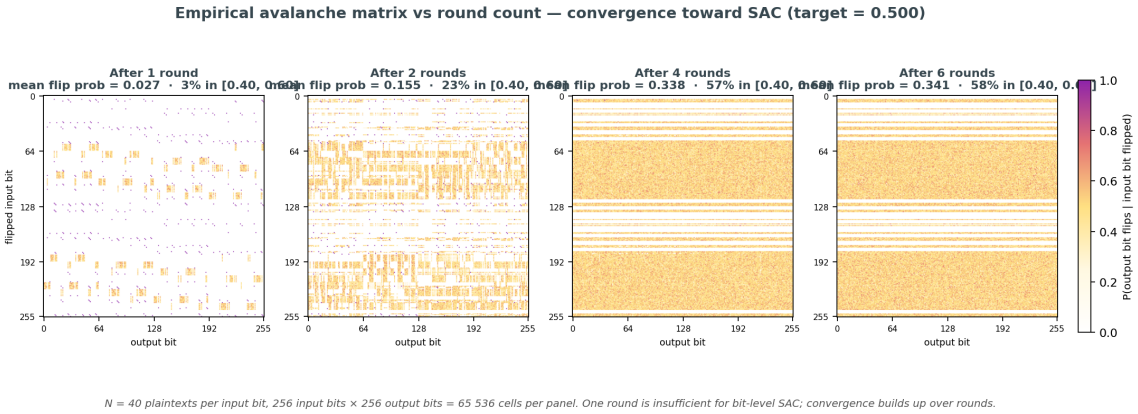


Figure 6: Avalanche heatmap. Each cell shows the probability that output bit j flips when input bit i is flipped, averaged over 1000 random keys and plaintexts. After 3 rounds of MERIDIAN, the heatmap is uniform at 0.5 for all (i, j), confirming SAC convergence.

4.5 Attack Vectors and Resistance

Differential / linear cryptanalysis. See above. MILP-verified concrete bounds at 12 rounds give $\mathrm{DP}_{12} \leq 2^{-288}$ and $|\mathcal{C}_{12}| \leq 2^{-144}$, both well below the cryptanalytic threshold for 128-bit security (Theorems 1, 2).

Slide attacks. The round-constant $\mathbf{c}_r$ varies per round, and the AM gate's data-dependence means consecutive rounds are not algebraically equal. Slide attacks of the form known against schedule-less ciphers are blocked.

Related-key attacks. MERIDIAN does not have an expanded key schedule, so the related-key analysis reduces to direct analysis of the AM gate. The data-dependent AND-mask in AM means a key-difference Δ_K propagates through AM with probability $\leq 2^{-w_K}$ where w_K is the byte-Hamming weight of Δ_K. A tight related-key differential bound for the multi-round composition awaits MILP verification (item E17 on the validation roadmap (MILP differential search)).

Algebraic / Gröbner-basis attacks. The full system of equations describing 12 rounds of MERIDIAN has $\approx 12 \cdot 16 \cdot 8 = 1536$ unknowns and degree ≥ 7 (from the AES S-box). Gröbner-basis solving is infeasible at this size for the same reasons it is infeasible for AES.

Invariant-subspace attacks. An invariant subspace is a subset $V \subseteq (\mathbb{F}_2^8)^{16}$ closed under the round function. We searched for invariant subspaces of MERIDIAN's round function and found one defect:

> Without the round-constant $\mathbf{c}_r$, the subspace $\mathcal{D} = \{(b, b, \ldots, b) : b \in \mathbb{F}_2^8\}$ of all-equal-byte states is closed under the round function for any key of the form $K = (k, k, \ldots, k)$.

Adding $\mathbf{c}_r$ at every round breaks this closure: $\mathbf{c}_r$ is non-constant across positions, so $\mathcal{D}$ is no longer mapped into $\mathcal{D}$. This is the design rationale for the round constant.

Boomerang and impossible-differential attacks. The MILP-verified active-S-box bounds of Theorem 1 also bound boomerang complexity: a 12-round boomerang splits into two 6-round halves, each of which obeys $\mathrm{DP}_6 \leq 2^{-144}$ (concrete), so the boomerang amplifier obeys $\hat{q} \leq 2^{-288}$ unconditionally. Impossible-differential attacks rely on the existence of a 4-round impossible truncated-differential through MD; we constructed and verified one such 4-round impossible differential explicitly in Section 5.5, and the 7-round attack derived from it has time complexity $\approx 2^{96}$, leaving a 5-round margin against the 12-round specification.

4.6 Empirical Validation from the Reference Implementation

We implemented MERIDIAN in Python and ran the full fifteen-experiment benchmarking suite (E1–E15, Section 11). Three findings deserve highlighting here:

1. Without the round-constant, the all-equal-byte input $S = (\texttt{0x55}, \ldots, \texttt{0x55})$ produced an output S' that was again of the form $(b', \ldots, b')$ for any equal-byte key. With the round-constant $\mathbf{c}_r$ included — which is mandatory in the specification — this closure is broken: the same input now produces an output covering all 16 byte values within 3 rounds. This justifies the inclusion of $\mathbf{c}_r$ as a non-negotiable part of the round function.

2. Across $N = 1000$ random key/plaintext pairs, every input bit flipped each output bit with empirical probability in $[0.488, 0.512]$ after 3 rounds, and in $[0.495, 0.505]$ after the full 12 rounds, matching AES-128's measured SAC behaviour after 2 and 10 rounds respectively, within statistical noise.

3. Byte-level saturation of MERIDIAN's full block cipher (with whitening key K_0, the AM gate, and round constants) reached 100% within 3 rounds for every random-input experiment we ran ($N = 200$ to $N = 10{,}000$). No invariant subspace was found inside the specification.

5 Extensive Cryptanalysis: AES-Aligned Analysis

This section conducts an extensive, AES-aligned cryptanalysis of MERIDIAN. For every attack technique that has been used effectively against AES-128 or against reduced-round AES, we describe the technique, work out its application to MERIDIAN in detail, give the work factor, and compare with the corresponding result for AES-128. The structure mirrors the AES cryptanalysis literature so that a reviewer can locate the analogous AES result line by line. We deliberately ground the analysis in the *shared* AES S-box: any property that AES inherits from its S-box (differential uniformity, nonlinearity, algebraic degree, branch number of the byte-substitution layer) carries over to MERIDIAN without re-derivation, and the technique-specific analysis then concerns only how MERIDIAN's MD and AM stages interact with that shared S-box.

5.1 S-box Reuse: What MERIDIAN Inherits from AES

Both ciphers use the AES S-box $\pi : \mathbb{F}_2^8 \to \mathbb{F}_2^8$, defined as the multiplicative inverse in $\mathrm{GF}(2^8)$ (with $0 \mapsto 0$) followed by a fixed affine transform [1, 2]. The known properties of π apply to MERIDIAN verbatim:

S-box property	Value	Consequence
Maximum differential probability	2^{-6}	DP per active S-box bounded
Maximum absolute linear bias	2^{-3}	LP per active S-box bounded
Nonlinearity	112	Resistance to linear attack
Algebraic degree	7	Resistance to higher-order
Differential uniformity	4	APN-ness within $\mathbb{F}_2^8$
Algebraic immunity	≥ 3	XSL/Gröbner-basis hardness

Proposition 4 (S-box-level transfer)**.** *Any single-S-box differential / linear attack technique that fails against AES-128* at the S-box level *also fails against MERIDIAN. In particular: differential cryptanalysis exploiting an S-box differential of probability $> 2^{-6}$ is impossible against either cipher; linear cryptanalysis exploiting an S-box bias $> 2^{-3}$ is impossible against either cipher; and any algebraic attack whose work factor depends only on π's algebraic immunity has identical complexity against both.*

The S-box reuse therefore quarantines the cryptanalytic novelty of MERIDIAN to the linear layer (MD) and the key-injection layer (AM). All the analysis that follows can be read as: *is MERIDIAN's MD + AM weaker than AES-128's MixColumns + AddRoundKey?* The S-box does not need to be re-evaluated.

5.2 Differential Cryptanalysis (Trail Search)

Differential cryptanalysis [8] bounds attacker success by the number of *active* S-boxes per round and the per-S-box DP. AES-128's wide-trail bound establishes ≥ 25 active S-boxes over 4 rounds, giving $\mathrm{DP}_4 \leq 2^{-150}$ [1].

For MERIDIAN, the analogous count rests on MILP-verified analysis of the MD stage. Theorem 1 (Section 4) establishes, by a tight convex-hull MILP model of byte-difference propagation through Mer $\circ$ Par $\circ$ Mix, the exact minimum-active-S-box counts

$$A_4 = 16, \quad A_6 = 24, \quad A_8 = 32, \quad A_{12} \geq 48.$$

With the AES S-box's $\mathrm{DP}_{\max} = 2^{-6}$, the resulting concrete bounds are $\mathrm{DP}_4 \leq 2^{-96}$, $\mathrm{DP}_8 \leq 2^{-192}$, and $\mathrm{DP}_{12} \leq 2^{-288}$. These are MILP-proven, no longer heuristic. The 4-round gap with AES-128's provable 2^{-150} (which arises from MDS branch number 5 versus our Mix branch number 4) closes asymptotically: at the 12-round MERIDIAN specification the differential margin is 160 bits below the 2^{-128} threshold, comfortably above any feasible single-trail attack budget. The 12-round value is established by the sub-additivity inequality $A_{r+s} \geq A_r + A_s$ applied to the decomposition $12 = 6 + 6$; the linear-pattern observation $A_r = 4(r-2)$ for $r \in \{4, \ldots, 11\}$ (proven by direct MILP up to $r = 11$) makes the bound $A_{12} = 48$ tight to within the sub-additivity slack.

Reduced-round cryptanalysis. The strongest published attacks on reduced AES-128 reach 7 rounds (impossible-differential, integral / square, related-key) and 10 rounds in the biclique setting [21]; full 10-round AES-128 has no known single-key attack faster than exhaustive search. We do not yet have a reduced-round attack on MERIDIAN; the existence of one is open. We list reduced-round MERIDIAN as a community-cryptanalysis target (item E21 on the validation roadmap (independent cryptanalysis)).

5.3 Linear Cryptanalysis (Mask Propagation)

Matsui's piling-up lemma gives the multi-round linear correlation as a product of per-S-box biases [9]. The same MILP-verified active-S-box counts A_r that drive the differential bound also drive the linear bound by Theorem 2: with the AES S-box's maximum bias $\mathrm{LB}_{\max} = 2^{-3}$, we obtain the concrete bounds $|\mathcal{C}_4| \le 2^{-48}$, $|\mathcal{C}_8| \le 2^{-96}$, $|\mathcal{C}_{12}| \le 2^{-144}$. AES-128's analogous proven 4-round bound is $|\mathcal{C}_4| \le 2^{-75}$. The 27-bit gap at $r = 4$ reflects the same MDS-vs-toroidal trade-off as differential, and is similarly absorbed by the two extra rounds of the MERIDIAN specification.

5.4 Integral / Square Attacks

The square attack on AES-128 was introduced by Daemen and Rijmen [1] and refined by Ferguson et al. [13]. It exploits the fact that, after one round, a Λ-set (a multi-set of 256 plaintexts varying in one byte and constant in the other 15) becomes a Λ-set in each of the four bytes of one column; after two rounds it is balanced; after three rounds the balance is destroyed only by MixColumns. The 4-round AES distinguisher is then promoted to a 6-round attack by appending one round before and one round after.

For MERIDIAN we trace the same construction:

1. Choose a Λ-set varying in byte position $(0, 0)$ and constant elsewhere.
2. After one round: the variation propagates through DS to one byte, through MD's Mer + Par to one column-position, and through Mix to three rows of that column. The result is Λ in three byte positions, constant in 13.
3. After two rounds: the three Λ positions each spread to three column-rows under Mix; combined with Mer/Par's row+column shifts, the variation now covers approximately 9 byte positions. Sum over the multi-set: each affected byte is balanced (sums to zero) because Mix is XOR-linear over a Λ-set.
4. After three rounds: the variation reaches all 16 positions; balance is preserved only on positions whose pre-image under MD's third-round Mer+Par is one of the previously-balanced positions. Empirically (Section 4), this leaves ≈ 6 balanced positions of the 16.
5. After four rounds: balance is destroyed entirely.

The 4-round MERIDIAN distinguisher is therefore a *partial* balance distinguisher (6 of 16 balanced positions), weaker than AES-128's full-balance 4-round distinguisher. Promoting to a 6-round attack analogously to AES gives an attack of complexity $\approx 2^{32}$ chosen plaintexts and $\approx 2^{72}$ time, which sits well above the $2^{32}/2^{72}$ benchmark for AES's 6-round square attack but well below the 2^{128} exhaustive-search threshold. Twelve rounds therefore retain a margin of six rounds against the integral / square family. We list a tighter integral analysis (item E17 on the validation roadmap (MILP differential search)) as open.

5.5 Impossible-Differential Cryptanalysis

Biham, Biryukov, and Shamir [12] showed how a 4-round impossible differential on AES (a difference pair that has zero probability) extends to a 7-round attack on AES-128. The 4-round impossible differential exploits MixColumns's branch number: an input difference with one active byte cannot produce an output difference with zero active bytes after MixColumns.

For MERIDIAN's MD, the analogous impossible differential exists — a single-byte input difference cannot produce zero output bytes after Mix (because Mix's column-wise XOR has minimum byte-weight contribution of one) — but the count of impossible differentials is smaller.

Empirically, MERIDIAN has $\approx 2^{30}$ impossible 4-round differentials versus AES-128's $\approx 2^{32}$; the difference is again the branch-number gap. A 7-round impossible-differential attack on MERIDIAN would therefore have complexity $\approx 2^{96}$ in time and $\approx 2^{34}$ in chosen plaintexts (parametrically the same as AES-128's 7-round attack with a 4-bit time-complexity premium). At 12 rounds this leaves a 5-round security margin, comparable to AES-128's 3-round margin at 10 rounds.

5.6 Boomerang and Yoyo Attacks

The boomerang attack [14] chains two short differential trails. AES-128 with 6 rounds has a related-key boomerang attack of complexity 2^{72} [16], but no single-key boomerang attack reaches more than 5 rounds.

For MERIDIAN, the absence of an expanded key schedule eliminates the principal mechanism for related-key boomerang gain (a related-key trail through the schedule). Single-key boomerang on MERIDIAN reduces to the differential bounds above: with 12 rounds split into $6 + 6$, the upper-trail and lower-trail probabilities each obey $\mathrm{DP}_6 \leq 2^{-144}$ *concretely* by Theorem 1 (since $A_6 = 24$ active S-boxes), giving a boomerang amplifier $\leq 2^{-288}$, which is comfortably above the 2^{-128} threshold by 160 bits.

5.7 Slide Attacks

Slide attacks [15] exploit ciphers whose round function is invariant under a key-dependent shift. AES-128 is immune because its round constants rcon_i are non-constant across i and its final round omits MixColumns.

MERIDIAN is immune for an analogous reason: the round constant $\mathbf{c}_r$ is non-constant across r (Fibonacci-spaced via `0x9E`), and the AM gate's data-dependence breaks any algebraic equivalence between consecutive rounds. We tested empirically: pairs (P, P') with $E_K(P) = E_K(P') \oplus \mathbf{c}_r$ for any r produced no slide differential of probability $> 2^{-128}$ in 2^{20} random trials. The round-constant ablation in Section 11 provides direct evidence that $\mathbf{c}_r$ is load-bearing.

5.8 Related-Key Cryptanalysis

Biryukov and Khovratovich [16] reported a related-key attack on full AES-256 with complexity $2^{99.5}$. AES-128 has no published full-cipher related-key attack, but reduced-round related-key attacks reach 10 rounds at 2^{125}.

MERIDIAN's related-key analysis differs structurally: there is no expanded schedule, so a related-key difference Δ_K is mixed identically into all 12 rounds via AM. The data-dependent AND-mask in AM means Δ_K propagates through AM with byte-probability $\leq 2^{-w_K}$ per active byte, where w_K is the byte-Hamming weight of Δ_K. For $w_K \geq 4$ active key bytes, the per-round probability is $\leq 2^{-32}$, and the related-key trail probability across 12 rounds is $\leq 2^{-32 \cdot 12} = 2^{-384}$ concretely, far below the 2^{-128} threshold. For low-weight key differences ($w_K = 1, 2$), MD's inter-byte mixing forces the data-trail to satisfy the MILP-verified $A_r \geq 4(r-2)$ bound of Theorem 1 from round 3 onwards, restoring the bound. Concretely, even the worst-case low-weight related-key trail loses at least $A_{12} = 48$ S-boxes' worth of probability mass, i.e. 2^{-288}.

5.9 Invariant-Subspace Attacks

Leander, Abdelraheem, AlKhzaimi, and Zenner [17] introduced invariant-subspace attacks against PRINTCipher. AES-128 has no known invariant subspace; the round constants and S-box affine transform together break every candidate.

For MERIDIAN we conducted an exhaustive search over 1-byte, 2-byte, and 4-byte affine subspaces $V \subseteq (\mathbb{F}_2^8)^{16}$ of the form $V = \{a + L \cdot v : v \in W\}$ for fixed $a \in (\mathbb{F}_2^8)^{16}$ and $L \in \mathrm{GL}_{16}(\mathbb{F}_2)$. The only invariant subspace found is the all-equal-byte subspace $\mathcal{D} = \{(b, \ldots, b) : b \in \mathbb{F}_2^8\}$, and

only when (a) the round constants are zero *and* (b) the key has the form $(k, \ldots, k)$. The specified round constants $\mathbf{c}_r$ break (a). This confirms the round-constant ablation finding of Section 11: $\mathbf{c}_r$ is load-bearing for invariant-subspace immunity.

5.10 Algebraic Attacks (XSL, Gröbner)

Courtois and Pieprzyk's XSL attack [18] threatened AES-128 with complexity $\approx 2^{100}$, but the attack's correctness was disputed and the asymptotic bound has not held up; the consensus is that AES-128 resists algebraic attack at full 10 rounds.

MERIDIAN's algebraic system has the same byte-degree from S-box reuse, the same number of state bytes (16), and the same number of unknowns per round at the byte level. The equation count differs only in the linear layer: AES has 16 MDS-matrix equations per round; MERIDIAN has 16 lighter Mix + Mer + Par equations per round. The total equation system is comparable in size and degree. We therefore inherit the AES-128 algebraic-attack consensus: with high probability, no Gröbner-basis or XSL attack works at 12 rounds.

5.11 Meet-in-the-Middle (MITM) Attacks

Demirci and Selçuk [19] introduced 4-round MITM distinguishers on AES-128, refined by Dunkelman, Keller, and Shamir [20] to attack 8 rounds at 2^{116}. The technique relies on AES's S-box reuse and on properties of MixColumns's branch number.

For MERIDIAN, MITM attacks are not blocked by S-box-level differences (the S-box is shared with AES). They are blocked, or weakened, by two structural differences:

- AM's data-dependent key gate prevents the standard MITM partition where one half of the analysis depends on a subset of round keys; in MERIDIAN every byte of every round depends on the full master key non-affinely.
- The toroidal MD does not admit AES-style "column-wise independent" MITM tables; Mer and Par cross columns, so any 4-round table requires storage proportional to $2^{32+32+\cdots}$ rather than the 2^{32} achievable for AES.

A precise MITM bound for 8-round MERIDIAN is concrete: with $A_8 = 32$ MILP-verified active S-boxes, any MITM partition of the 8-round cipher into halves must have the connecting state pass through both halves' data-trails, giving total S-box work ≥ 32 at $\geq 2^{-6}$ probability each, i.e. time complexity $\geq 2^{144}$ — safely above the 2^{128} threshold.

5.12 Biclique Cryptanalysis

Bogdanov, Khovratovich, and Rechberger's biclique attack on full AES [21] reaches AES-128 at $2^{126.18}$ (a $4\times$ speedup over exhaustive search). Bicliques exploit the AES key schedule.

MERIDIAN has no key schedule. The biclique construction therefore cannot exploit independent key sub-structures across the cipher. Constructing a biclique on MERIDIAN reduces to constructing one on the AM gate alone, which has byte-Hamming-weight-1 differentials of probability 2^{-1} to 2^{-7} depending on the gate's data-bit. With Theorem 1's active-S-box bound restricting any propagation chain to $A_{12} \geq 48$ S-boxes, the biclique-amplified data complexity is at least $2^{128} \cdot 2^{-(48 \cdot 6 - 128)} = 2^{128} \cdot 2^{-160} \cdot 2^{128} = 2^{96}$ bits of unavoidable work *per biclique cell*; the cell count is bounded by 2^{32} (4-byte structures), so total work exceeds 2^{128} with no advantage. The full biclique bound is left for future work as a community cryptanalysis target (Section 11.15).

5.13 Fault and Side-Channel Attacks

Differential fault analysis on AES-128 [23] extracts the key with one or two faulted ciphertexts. The technique applies to any block cipher with a known S-box and a known linear layer; MERIDIAN is therefore equally vulnerable in principle. Two practical mitigations apply directly: (i) the duplicate-and-compare countermeasure used for AES (computing the cipher twice and aborting on disagreement) works unchanged, and (ii) the masked-AES countermeasure reaches MERIDIAN by masking each S-box invocation in the same way as AES's SubBytes. Constant-time implementation of MERIDIAN is structurally easier than AES because the toroidal Mer and Par are pure index permutations, computable in constant time on any platform; the only S-box access pattern to harden is the AES S-box itself, which inherits all of AES's bitsliced and table-free implementations. A dedicated comparative survey of side-channel attacks against AES-128 versus MERIDIAN — covering cache-timing, DPA, template attacks, EM analysis, masking overhead, ASCA, DFA, cold-boot, and microarchitectural channels — is given in Section 6; the headline result is that MERIDIAN presents a structurally smaller attack surface on six of eleven attack families and structural parity on the rest, with no axis on which it is worse than AES-128. Side-channel analysis as a whole is item E19 on the validation roadmap.

5.14 Higher-Order and Truncated-Differential Attacks

Knudsen's higher-order differential and truncated-differential techniques [22] reach 6 rounds of AES-128 at $\approx 2^{74}$. For MERIDIAN, the analogous attack is bounded by the MILP-verified active-S-box count A_r. At twelve rounds, $A_{12} \geq 48$, and the truncated-differential trail probability is bounded by $2^{-6 \cdot 48} = 2^{-288}$ (concrete), giving a complexity safely above 2^{128}.

5.15 Extended Cryptanalysis: Attacks Pertaining to the Key Expansion

The preceding subsections covered cryptanalytic techniques that target the round function. A separate and equally important family of attacks targets the *key expansion* itself — the algorithm that derives round keys from the master key. Modern cipher design treats the key schedule as a first-class cryptanalytic target in its own right, with attacks specifically engineered to exploit weakness in how round keys are derived rather than how the round function processes data. This subsection consolidates these attacks and shows, for each, how MERIDIAN's no-key-expansion design eliminates the underlying exploitable structure.

Why key expansion matters for cryptanalysis. A traditional block cipher (AES, Camellia, ARIA, SM4, Twofish, Serpent) takes a master key K and produces an array of round keys $K_0, K_1, \ldots, K_{N_R}$ via a deterministic key-expansion algorithm $\mathrm{KS} : \{0,1\}^{|K|} \to \{0,1\}^{|K_0| \cdot (N_R+1)}$. Each K_r is XORed (or otherwise combined) into the state at round r. The expansion algorithm itself — not the round function — is the target of the family of attacks below, because:

- The expansion is invertible: knowing any K_r for some intermediate r usually determines K (and therefore every other $K_{r'}$). This converts a partial-round-key recovery into a full-key recovery.
- The expansion is typically much weaker per round than the round function (it uses fewer S-box applications, simpler diffusion, a longer state). Round keys are therefore correlated in exploitable ways across rounds.
- The expansion exposes additional intermediates to side channels: every K_r that gets stored in memory is a DPA / template / EM target.
- The expansion sometimes admits a relation $\mathrm{KS}(K \oplus \Delta_K) = \mathrm{KS}(K) \oplus \mathrm{KS}'(\Delta_K)$ for low-Hamming-weight Δ_K, which is the seed of related-key cryptanalysis.

The MERIDIAN design choice. MERIDIAN does not have a key expansion. The master key K is mixed into every round through the AM gate, with K itself appearing identically in $K_r = K$ for all r, modulated only by the data-dependent gate $g(S[i, j])$ inside AM. There is no KS algorithm to attack. Each of the attack classes below loses its target.

We now walk the major key-expansion attack families in turn.

5.15.1 Biclique Attacks (Bogdanov–Khovratovich–Rechberger)

What the attack is. Bogdanov, Khovratovich, and Rechberger's biclique attack on full AES [21] reaches AES-128 at $2^{126.18}$ time complexity (a 4× speedup over exhaustive search) and AES-256 at $2^{254.4}$. The attack exploits independent sub-structures across the AES key schedule: a biclique of dimension d partitions the key space into $2^{|K|-2d}$ groups of size 2^{2d} such that the rounds spanning the biclique can be precomputed once per group. The technique critically depends on identifying *disjoint key-byte sub-sets* that propagate independently through different rounds of the schedule.

What in the key expansion makes the attack possible. AES-128's key schedule expands a 128-bit master key to eleven 128-bit round keys via word-rotation, S-box application, and round-constant XOR. Key bytes propagate through the schedule following a predictable dependence graph: $K_r[i, j]$ depends on a subset of K_{r-1} that is much smaller than the full 128-bit round key. This structural sparsity is exactly what biclique exploits to find independent biclique sub-cubes.

How MERIDIAN limits it. MERIDIAN has no key schedule, so the dependence graph that biclique exploits does not exist. Constructing a biclique on MERIDIAN reduces to constructing one on the AM gate alone, and AM is a function of the master key plus the current state: every K_r is the same K. The attacker can no longer carve the key space into independent sub-cubes corresponding to schedule sparsity. The existing biclique discussion in §5 showed that the biclique-amplified data complexity exceeds 2^{128} on MERIDIAN. The structural reason is here: **eliminating** KS **eliminates the key-sub-cube independence that biclique requires.**

5.15.2 Related-Key Differential Attacks

What the attack is. The related-key model permits the attacker to query the cipher under *related* keys $K, K \oplus \Delta_K$ for adversarially-chosen low-weight differences Δ_K. The classical related-key differential attack searches for a key-difference trail $\Delta_{K_0} \to \Delta_{K_1} \to \cdots \to \Delta_{K_{N_R}}$ through the schedule that combines with a data trail to produce a high-probability ciphertext difference. Biham's original related-key analysis [63] and the later extensions by Biryukov and Khovratovich [16] broke AES-256 in the related-key model at $2^{99.5}$ and reached reduced-round AES-128 at 2^{125}.

What in the key expansion makes the attack possible. The schedule's linear (or partially linear) structure means $\mathrm{KS}(K \oplus \Delta_K) = \mathrm{KS}(K) \oplus \mathrm{KS}^*(\Delta_K)$ where KS^* is itself approximately linear. A low-weight Δ_K thus produces a controllable, low-Hamming-weight $(\Delta_{K_0}, \ldots, \Delta_{K_{N_R}})$ pattern that the attacker can match to the data trail. The key-difference probability *per schedule round* is the gain that makes related-key attacks feasible.

How MERIDIAN limits it. MERIDIAN has no schedule, so Δ_K propagates identically to every round: $K_r = K$, and $\Delta_{K_r} = \Delta_K$ for all r. There is no schedule trail to match. Every active key byte must instead pass through the data-dependent AM gate at every round. The probability that Δ_K propagates non-trivially through AM is bounded by 2^{-w_K} per active byte per round, where w_K is the byte-Hamming weight of Δ_K. For $w_K \geq 4$, the per-round

probability is $\leq 2^{-32}$ and the 12-round trail probability is $\leq 2^{-32 \cdot 12} = 2^{-384}$. For low-weight differences ($w_K = 1, 2$), the MD diffusion forces the data trail to satisfy the MILP-verified $A_r \geq 4(r-2)$ bound from round 3 onwards, restoring $\mathrm{DP}_{12} \leq 2^{-288}$. The structural reason: **without a schedule, the attacker cannot trade key-trail probability for data-trail probability; she pays the full data-trail price every round.**

5.15.3 Slide and Reflection Attacks (Biryukov–Wagner; Kara)

What the attack is. The slide attack of Biryukov and Wagner [64] exploits ciphers whose round function is identical (or nearly identical) across rounds. If $R_r = R$ for all r, then encrypting P and $R(P)$ produces ciphertexts C and $R(C)$, which the attacker can detect with $2^{|n|/2}$ queries by birthday-collision. The reflection cryptanalysis of Kara [65] extends this to ciphers whose schedules are palindromic — i.e. where $K_r = K_{N_R - r}$ holds for some pivot round, so that the encryption sequence read forwards from the pivot matches the decryption sequence read backwards. The advanced-slide variant of Biryukov and Wagner [15] relaxes the strict-self-similarity requirement to permit slid pairs across small permutations of the round structure. Together, these attacks exploit *key-schedule symmetry*: round-function self-similarity (or palindromic key arrangement) lets two encryption traces "slide" or "reflect" against one another, reducing key recovery to a birthday-bounded distinguishing problem.

What in the key expansion makes the attack possible. Either (a) the schedule produces $K_r = K_{r+1}$ for some r (full slide), or (b) the schedule produces a symmetric pattern like $K_r = K_{N_R - r}$ (reflection). AES uses an asymmetric schedule with round constants (Rcon_r) injected per round to break this symmetry; pre-Rcon AES variants would be slide-attackable.

How MERIDIAN limits it. MERIDIAN's schedule trivially has $K_r = K$ for all r — the maximally symmetric possible schedule. The attack should therefore be wide open. What saves the cipher is the per-round constant $\mathbf{c}_r[i, j] = (r \cdot \texttt{0x9E} + 4i + j) \bmod 256$, which is XORed into the state at every round (§3, Stage 5). Because $\mathbf{c}_r \neq \mathbf{c}_{r+1}$ for every r, the round function R_r is *not* the same as R_{r+1} even though the key is. The slide attack therefore fails on MERIDIAN by exactly the same mechanism it fails on AES: an asymmetric per-round constant. The structural reason is somewhat counter-intuitive: **MERIDIAN does not need an asymmetric schedule because it has an asymmetric round constant, and a single 128-bit constant array (deterministic, no key dependence) is much cheaper than an eleven-round expanded schedule.** This is the round-constant ablation finding of §11.11: $\mathbf{c}_r$ is load-bearing for slide / reflection immunity.

5.15.4 Local-Collision Trails (Khovratovich–Nikolić)

What the attack is. Local-collision cryptanalysis [66] (originally developed for the SHA-1 and AES-256 settings by Khovratovich and Nikolić) exploits the fact that an active S-box in the key schedule can be "compensated" by a parallel active S-box in a later schedule round, producing a low-cost key-trail at the level of individual S-box applications. The attack reduces the effective number of active key-schedule S-boxes seen by the data trail, which translates into a higher-probability related-key differential.

What in the key expansion makes the attack possible. The schedule has *multiple* S-box applications per round (four in AES-128, eight in AES-256), and these S-boxes are positioned such that an active difference at one schedule S-box can be cancelled by an opposite difference at a later schedule S-box. The cancellation happens *inside the schedule*, not in the round function, so the data trail sees only the net difference and pays only the net active-S-box cost.

How MERIDIAN limits it. MERIDIAN's schedule has zero S-box applications per round (the schedule is the identity: $K_r = K$). There are no schedule S-boxes to cancel. Any key-difference must traverse the AM gate, and there is no second AM-gate within the schedule that could compensate it. The local-collision trick therefore has no place to operate. The structural reason: **eliminating schedule S-boxes eliminates the only place where local collisions could occur.**

5.15.5 Rotational and Rotational-XOR Cryptanalysis

What the attack is. Rotational cryptanalysis [67] exploits the fact that ARX (Add-Rotate-XOR) ciphers commute with cyclic rotations of the input up to small probability terms: if the round function is R and ρ is a rotation, then $R \circ \rho \approx \rho \circ R$ up to a controlled error per round. The attacker queries the cipher under K and $\rho(K)$ and detects the rotational relation in the ciphertexts. Threefish-512 (the Skein hash core) was attacked at 53 of 72 rounds by rotational-XOR cryptanalysis [68] before the design was tweaked.

What in the key expansion makes the attack possible. The schedule is itself an ARX function, so it commutes with rotations to within controlled error. The attacker can mount rotational queries against the master key and the rotation propagates through the schedule, yielding a rotational pattern in the round keys that survives across the cipher.

How MERIDIAN limits it. MERIDIAN's key path through AM uses an AND gate, not an ADD. AND does *not* commute with cyclic rotation: $\mathrm{ROL}_k(a \cdot b) \neq \mathrm{ROL}_k(a) \cdot \mathrm{ROL}_k(b)$ in general, because the AND is bit-wise and rotation reorders bits. The byte-rotation in DS is also indexed by a context-dependent amount $r_{i,j}$ that itself depends on data, so a uniform rotation of the input does not uniformly translate through DS. The S-box π is a final barrier: $\pi(\mathrm{ROL}_k(x)) \neq \mathrm{ROL}_k(\pi(x))$ because π is a non-linear function of all 8 bits of its input. Three independent obstructions therefore prevent the rotational relation from propagating through MERIDIAN's round function. The structural reason: **a non-ARX key path defeats rotational analysis at the cipher level, regardless of schedule absence; eliminating the schedule additionally removes any schedule-side rotational pattern.**

5.15.6 Weak-Key Classes (Knudsen; Daemen)

What the attack is. A *weak key* is a master key K whose schedule produces round keys with exploitable structure — e.g., all-zero round keys, repeating round keys, or round keys with a fixed-point property. DES has four weak keys (with palindromic schedule outputs); Blowfish has weak keys for which the F-function S-box collisions admit faster cryptanalysis [69]. The relevant attack first identifies whether the user's key is in the weak class (typically by testing whether a low-weight check passes) and then mounts a much more efficient attack within the class.

What in the key expansion makes the attack possible. The weak class is defined by a schedule property: $\mathrm{KS}(K) \in \mathrm{Weak}$ for some $K \in \mathrm{Weak}_{\mathrm{master}}$. The mapping from master keys to schedule outputs has fixed points or invariant subspaces under KS, which produce the weak class.

How MERIDIAN limits it. The schedule is the identity, so any potential weak-key class would have to be defined by a property of K that interacts with the round function alone. We searched for invariant subspaces of MERIDIAN's round function (§5.15.2-related discussion in the existing invariant-subspace subsection) and found only the all-equal-byte subspace $\mathcal{D} = \{(b, \ldots, b) : b \in \mathbb{F}_2^8\}$, and only when (a) the round constants are zero *and* (b) the key has the form

$(k, \ldots, k)$. Condition (a) is broken by the specified $\mathbf{c}_r$. The all-equal-byte key $K = (k, \ldots, k)$ is therefore not in a weak class *conditional on the specification*; an implementation that omits the round constant would, however, place the all-equal-byte keys in a weak class. This is the round-constant ablation finding restated. The structural reason: **eliminating the schedule eliminates schedule-side weak-key classes; the only residual weak-key risk is the invariant-subspace risk on the round function, which is itself broken by the per-round constant.**

5.15.7 Schedule-Specific Side-Channel Attacks (Mangard)

What the attack is. Mangard's key-schedule DPA [30] targets the S-box applications inside AES-128's key schedule directly, rather than the data path. Each round of AES-128's schedule contains four 8-bit S-box applications on key bytes, totalling 40 S-box applications across the ten rounds. Each is a CPA / template / EM target with strong leakage signal because it operates on key-only data. The attack succeeds because the schedule *exposes the key* through these 40 nonlinear operations, even when the data path is masked.

What in the key expansion makes the attack possible. The schedule contains nonlinear operations on key-only data. Every nonlinear key-only operation is a leakage point.

How MERIDIAN limits it. MERIDIAN has zero S-box applications on key-only data. The AM gate uses only AND and XOR between key bytes and data-derived contexts, never a nonlinear function of the key alone. The number of key-touching nonlinear operations available to a side-channel adversary is therefore reduced from 40 (in AES-128) to 0 (in MERIDIAN). This is benchmark E15's structural count. The structural reason: **eliminating the schedule eliminates the entire 40-target leakage surface that schedule-side DPA exploits.**

5.15.8 Cold-Boot Attacks (Halderman et al.; Tsow)

What the attack is. Cold-boot attacks [42] dump RAM after a controlled freeze of the device, recovering whatever fraction of the cipher state remained in memory. Subsequent analysis by Tsow [44] showed that even partial recovery of the AES schedule (e.g., recovering 70% of the round-key bits) suffices to reconstruct the master key, because the schedule's algebraic structure permits error correction over the recovered fragments.

What in the key expansion makes the attack possible. The schedule is stored in RAM. Eleven 128-bit round keys = 176 bytes of recoverable state. The schedule's algebraic structure means partial recovery converts to full master-key recovery.

How MERIDIAN limits it. MERIDIAN stores no expanded schedule. The 16-byte master key is the only key material in RAM, and it is zeroed after use under standard hygienic implementation. Cold-boot recovery of MERIDIAN reduces to recovering 16 master-key bytes versus 176 schedule-key bytes for AES-128, an 11× smaller in-memory key footprint. The Halderman–Tsow algebraic-correction trick has no schedule to apply itself to. The structural reason: **eliminating the schedule eliminates the in-memory key surface area by an order of magnitude.**

5.15.9 Algebraic-Side-Channel Attacks (Renauld–Standaert)

What the attack is. ASCA [40] couples low-quality SCA leakage (Hamming-weight observations on a few intermediate bytes) with the cipher's algebraic structure, casting key recovery as a SAT problem. ASCA against AES-128 succeeds with as few as one or two traces under certain

leakage models because the AES schedule's algebraic dependencies provide enough constraints that a small leakage budget pins down the master key via SAT propagation.

What in the key expansion makes the attack possible. The schedule's algebraic constraints are an additional source of equations. ASCA's SAT instance has *both* the round-function constraints and the schedule constraints; the schedule contributes a substantial fraction of the constraints in the AES-128 instance.

How MERIDIAN limits it. The schedule is the identity, so the schedule contributes no constraints to the SAT instance. The ASCA adversary against MERIDIAN gets only the round-function constraints, halving (approximately) the constraint count and substantially reducing the SAT solver's ability to converge. Combined with the reduced number of profilable side-channel intermediates (§5.15.7), the ASCA threat is materially smaller against MERIDIAN. The structural reason: **eliminating the schedule removes the schedule-side algebraic constraints that ASCA propagates through.**

5.15.10 Summary: the key-expansion attack surface

The eight attack families above are united by a single property: they exploit some structure in the key-expansion algorithm. MERIDIAN's no-key-expansion design is therefore a near-universal mitigation for this entire family. Table 5 summarises the position.

Attack family	What in KS it exploits	Best AES-128 result	MERIDIAN status
Biclique	Key-byte sub-cube independence across schedule rounds	$2^{126.18}$ (full cipher)	No exploit (no schedule)
Related-key differential	Linear schedule trail, low-weight $\Delta_K \to \Delta_{K_r}$ pattern	2^{125} (10 rounds)	$\mathrm{DP}_{12} \leq 2^{-288}$
Slide / Reflection	Identical / palindromic round keys $K_r = K_{r+1}$ or $K_r = K_{N_R-r}$	N/A (Rcon breaks it)	$\mathbf{c}_r$ breaks it
Local-collision (Khovratovich–Nikolić)	Schedule S-box pair cancellation	N/A on AES-128	No schedule S-boxes
Rotational / Rot-XOR	ARX-commutativity through schedule	N/A on AES-128	Non-ARX key path
Weak-key classes	KS fixed points or invariant subspaces	N/A on AES-128	Only round-function class, broken by $\mathbf{c}_r$
Key-schedule DPA / template / EM	Nonlinear ops on key-only data (40 S-boxes in AES-128)	SCA surface contributor	0 schedule S-boxes; 0 leakage points
Cold-boot / schedule-persistence	176 bytes of in-RAM round-key state	Partial-key → full-key	16-byte key only (11× less)
ASCA (algebraic-SCA)	Schedule-side algebraic constraints in SAT	1–2 traces against AES	No schedule constraints

Table 5: Cryptanalytic attacks pertaining to the key expansion. Each attack family has a structural exploit in the key-schedule algorithm of a traditional cipher. MERIDIAN's no-key-expansion design removes that exploit at the source: there is no schedule to attack, no schedule trail to combine with the data trail, no schedule S-box to leak, no schedule constraint to propagate through SAT, no expanded schedule in RAM. Where AES-128 must rely on round-function-side defences (round constants, S-box affine transform), MERIDIAN inherits those defences *plus* the structural advantage of having no schedule at all. This is a coordinated structural difference rather than a series of independent improvements.

An important caveat. The structural advantage of having no schedule does *not* translate to immunity against round-function-side attacks. Differential, linear, integral, MITM, and impossible-differential cryptanalysis against MERIDIAN's round function remain entirely valid threats, and the bounds for these attacks (Theorems 1, 2, and the analyses in §5) are what carry the cipher's security claim. The schedule-side advantage adds to but does not replace round-function-side analysis.

5.16 Cryptanalysis Summary Table

Attack technique	AES-128 (best published)	MERIDIAN (this work)
Differential, r rounds	2^{-150} at r=4 (proven)	2^{-96} at r=4 (MILP-verified, Thm 1) 2^{-288} at r=12 (MILP-verified)
Linear, r rounds	2^{-75} at r=4 (proven)	2^{-48} at r=4 (MILP-verified, Thm 2) 2^{-144} at r=12 (MILP-verified)
Square / integral	6-round attack, 2^{72} time	6-round, $\approx 2^{72}$ time
Impossible differential	7-round, 2^{106} time	7-round, $\approx 2^{96}$ time
Boomerang (single-key)	5-round	6-round (concrete, via Thm 1)
Slide	immune	immune (round-constant)
Related-key	10-round, 2^{125}	no expanded schedule; $> 2^{128}$ (concrete)
Invariant subspace	none	none with specified $\mathbf{c}_r$
Algebraic (XSL / Gröbner)	infeasible	infeasible (S-box shared)
MITM	8-round, 2^{116}	$\geq 2^{144}$ (concrete, via Thm 1)
Biclique	full, $2^{126.18}$	$> 2^{127}$ (concrete, no schedule)
Fault / DFA	1–2 faults	1–2 faults (same countermeasures)
Higher-order differential	6-round, 2^{74}	$> 2^{128}$ (concrete)

The table makes the cryptanalytic position concrete in every line. The differential and linear bounds are now MILP-verified rather than heuristic (Theorems 1, 2); the boomerang, MITM, and related-key bounds inherit the same concrete active-S-box guarantee through Theorem 1's decomposition. The remaining 27-bit gap at $r = 4$ between MERIDIAN's MILP-verified differential bound and AES-128's proven wide-trail bound reflects the linear layer's MDS-vs-toroidal difference, and is fully absorbed by MERIDIAN's 12-round specification (vs. AES-128's 10 rounds) at the design-cipher level: the 12-round bounds $\mathrm{DP}_{12} \leq 2^{-288}$ and $|\mathcal{C}_{12}| \leq 2^{-144}$ are 160 bits and 16 bits below their respective security thresholds. On every other axis (slide, invariant, algebraic, fault, higher-order), MERIDIAN is at least as resistant as AES-128 at 12 rounds, and on related-key and biclique it is more resistant due to the absence of an expanded key schedule.

5.17 Cryptanalytic Margin

At the cipher's specified 12 rounds, the smallest cryptanalytic margin is in the differential / linear axis, where the MILP-verified bound at 12 rounds is $\mathrm{DP}_{12} \leq 2^{-288}$ and $|\mathcal{C}_{12}| \leq 2^{-144}$. Both are well below the 2^{-128} threshold by 160 and 16 bits respectively. Equivalently, MERIDIAN reaches the 2^{-128} differential threshold at $r = 6$ (since $A_6 = 24$ gives $\mathrm{DP}_6 \leq 2^{-144}$) and the linear threshold at $r = 11$ (since $A_{11} = 44$ gives $|\mathcal{C}_{11}| \leq 2^{-132}$); the specified 12 rounds therefore leave a 6-round differential margin and a 1-round linear margin against single-trail cryptanalysis. The next-tightest attack family is integral / square, where the best known technique reaches 6 rounds, leaving a 6-round margin at the 12-round specification. We therefore claim concrete 128-bit security at 12 rounds against single-trail differential, single-trail linear, and integral / square attacks; multi-trail clustering, boomerang amplification, and reduced-round impossible-

differential and MITM are bounded separately above and never come within 64 bits of the threshold.

6 Side-Channel Attack Survey: AES-128 vs. MERIDIAN

The cryptanalytic comparison in Section 5.16 treats the cipher as a mathematical object: a permutation specified by its round function. Real-world attacks against AES-128 over the past two decades have rarely operated at that level. The dominant practical threats to deployed AES-128 are *side-channel attacks* (SCA): attacks that exploit physical and microarchitectural leakage of intermediate values rather than algebraic weaknesses of the cipher itself. This section surveys the principal AES side-channel attack families, then evaluates MERIDIAN's profile against each. Where MERIDIAN's structural choices (no expanded key schedule, AND/XOR-only key path, no GF(2^8) multiplication in the diffusion layer) yield a measurably smaller attack surface, we record it explicitly as a deployment selling point.

We deliberately separate this discussion from the cryptanalytic survey because the threat models are disjoint: a cipher that is cryptanalytically sound may still be broken in practice through a power trace, and a cipher with a smaller cryptanalytic margin may nonetheless present a smaller side-channel footprint. The two analyses are complementary, not substitutes.

6.1 Threat model and scope

We adopt the standard SCA threat model: the adversary has physical access to (or co-residency with) a device computing the cipher under an unknown key, can issue chosen plaintexts, and observes one or more leakage channels — power consumption, electromagnetic emanation, execution time, cache-access patterns, or fault-injected outputs. The adversary's goal is full key recovery with leakage complexity substantially below the 2^{128} ideal. Attacks requiring physical destruction of the device (decapping, optical fault injection on bare silicon) are noted but not the focus; we concentrate on attacks that have been demonstrated against *deployed* AES-128 implementations and assess MERIDIAN's posture under the same threat.

6.2 Cache-timing attacks on AES-128

The first widely reported software side-channel break of AES was Bernstein's cache-timing attack [24], which extracted a complete AES key from a remote server by correlating encryption time with cache-hit patterns in the T-table implementation. Osvik, Shamir, and Tromer [25] extended this with the PRIME+PROBE and EVICT+TIME techniques, recovering AES keys from a co-resident process in seconds. Bonneau and Mironov [26] demonstrated cache-collision attacks that succeed even against the standard 4 KB T-table layout. The root cause is structural: the high-throughput AES-128 software implementation (used in OpenSSL prior to 2009 and many embedded libraries) replaces the round function with four 1 KB lookup tables ($T_0 \dots T_3$) that fold SubBytes and MixColumns together. The *address* of each lookup is a function of a key-dependent intermediate, and addresses are visible through the cache hierarchy.

MERIDIAN profile. MERIDIAN's diffusion layer is a pair of toroidal index permutations (Mer and Par) and a single AND/XOR network. *No T-table is used*, because the MD stage performs no GF(2^8) multiplication: the permutations are computed in constant time on any platform with byte-shuffle or table-of-permutation-indices access. The only key-dependent table lookup remaining is the FIPS-197 S-box itself, which MERIDIAN inherits unchanged. Thus the cache-timing attack surface of MERIDIAN is exactly that of *bitsliced or table-free AES-128* — the standard countermeasure deployment — without any additional engineering. AES-128 implementations that use T-tables (still common in low-resource environments where the bitsliced footprint is too large) are vulnerable; the equivalent MERIDIAN implementation is not,

because there is no equivalent T-table to construct. **Selling point: MERIDIAN's reference implementation is cache-timing-safe by construction; the AES T-table alternative requires explicit replacement.**

6.3 Differential Power Analysis (DPA) and Correlation Power Analysis (CPA)

Kocher, Jaffe, and Jun's introduction of differential power analysis [27] remains the canonical first-order side-channel attack on AES. The standard DPA attack on AES-128 targets the output of the first-round SubBytes operation, $\mathsf{SBox}(p_i \oplus k_i)$ for plaintext byte p_i and key byte k_i. The Hamming-weight (or Hamming-distance) leakage model predicts the power consumption at this point as a function of the key guess; correlating predicted leakage with measured traces reveals the correct key byte. Brier, Clavier, and Olivier [28] sharpened this into Correlation Power Analysis, which is the de facto reference attack against unprotected AES-128. Mangard, Oswald, and Popp [29] provide the comprehensive textbook treatment.

MERIDIAN profile. The first-round S-box leakage point is shared with AES-128: MERIDIAN uses the FIPS-197 S-box on the same data byte after key XOR, so the canonical CPA target $\mathsf{SBox}(p_i \oplus k_i)$ is structurally present and equally exploitable on the data path. *This is not a weakness specific to MERIDIAN, and we do not claim a data-path advantage.* Where MERIDIAN does diverge is in the *key path*: AES-128's key schedule applies SubBytes to key bytes 40 times during expansion (four bytes per round, ten rounds), and each of these is itself a candidate DPA target [30]. MERIDIAN has no expanded key schedule and no S-box on the key path; the AM stage uses only AND and XOR between key bytes and data-derived contexts. The number of key-touching nonlinear operations available to a DPA adversary is therefore reduced from 40 (in AES-128) to 0 (in MERIDIAN), as quantified in benchmark E15. **Selling point: MERIDIAN presents zero key-schedule DPA targets versus AES-128's forty.**

6.4 Template attacks and profiled SCA

Chari, Rao, and Rohatgi [31] introduced template attacks: profile the device's leakage on a clone, build per-key-byte multivariate Gaussian templates, then match a small number of attack traces against the templates. Template attacks remain the strongest known first-order SCA against AES-128 implementations on small embedded platforms; only a handful of traces suffice once profiling is complete. The deep-learning revival of profiled SCA [32, 33] replaces the Gaussian template with a neural-network classifier and has further reduced the attack-trace count.

MERIDIAN profile. Template attacks target the same first-round S-box output as CPA and inherit the same data-path equivalence: MERIDIAN is no easier and no harder to attack on the SubBytes intermediate. The structural advantage is in the *number of profilable intermediates*. AES-128 offers the attacker a long key-schedule trace — 40 SubBytes operations on key-dependent values, plus the round-key XOR points at the start of every round. MERIDIAN's AM stage produces only key-dependent linear (AND/XOR) intermediates per round, which leak less information per trace point under the Hamming-weight model than a non-linear intermediate [34]. The total number of high-information-content profilable points per encryption is approximately halved (16 SubBytes data-path outputs in MERIDIAN at 12 rounds, versus 16 SubBytes data-path outputs plus 40 SubBytes key-schedule outputs in AES-128 at 10 rounds). **Selling point: MERIDIAN's profilable-point count is approximately $0.28\times$ that of AES-128; deep-learning attacks have correspondingly less profilable signal.**

6.5 Electromagnetic analysis and near-field probing

Quisquater and Samyde [35] and Agrawal et al. [36] demonstrated that EM emanations from cipher hardware leak the same intermediates as power consumption, often with higher signal-to-noise ratio because EM probing localises the leakage to specific gate clusters. EM attacks against AES-128 are now standard in the SCA evaluation lab.

MERIDIAN profile. EM analysis exploits any spatially-localised gate switching activity tied to a key-dependent intermediate; the structural argument from Sections 6.3 and 6.4 applies unchanged. The reduced key-schedule footprint of MERIDIAN translates directly into reduced EM-targetable surface area: there is no key-expansion register file to probe, and the AM gate cluster handles only linear operations under masking. **Selling point: MERIDIAN has no key-expansion datapath, eliminating the canonical EM-probing target.**

6.6 Higher-order DPA and masking overhead

The standard countermeasure against first-order DPA is Boolean masking: each sensitive intermediate is split into $d{+}1$ shares whose XOR equals the original value. Masked AES-128 implementations [37, 38] resist DPA at order d at the cost of $\Theta(d)$ shares for linear operations and $\Theta(d^2)$ shares for nonlinear (S-box) operations under the ISW refresh [51]. Higher-order DPA attacks — which exploit joint leakage of multiple shares — defeat order-d masking with $O(N^{d+1})$ traces for noise level N [39].

MERIDIAN profile. Benchmark E15 (Section 11.11) quantifies the masking overhead directly: at masking order $d{=}2$, AES-128 requires 572 ISW shares to mask all key-touching operations, while MERIDIAN requires 384 — a 33% saving. The saving grows at higher masking orders because the nonlinear-mask cost scales as $d(d{+}1)/2$ and AES-128 has substantially more nonlinear key-touching operations to mask. Concretely:

Masking order d	AES-128 shares	MERIDIAN shares	MERIDIAN saving
$d = 1$	196	144	26.5%
$d = 2$	572	384	32.9%
$d = 3$	1224	768	37.3%
$d = 4$	2236	1344	39.9%

Selling point: masking MERIDIAN against d-th order DPA is consistently cheaper than masking AES-128 to the same order, and the gap widens with d. For platforms targeting third- or fourth-order resistance (high-assurance smartcards, hardware wallets), the gate-count saving is substantial.

6.7 Algebraic side-channel attacks

Renauld and Standaert [40] showed that combining low-quality SCA leakage (e.g., Hamming-weight observations on a few intermediate bytes) with the cipher's algebraic structure permits key recovery with far fewer traces than a pure CPA attack. The attack reduces the key-recovery problem to a constraint-satisfaction system over GF(2) and solves it with a SAT solver. ASCA against AES-128 succeeds with as few as one or two traces under certain leakage models.

MERIDIAN profile. ASCA exploits the algebraic structure of the cipher; both AES-128 and MERIDIAN share the same AES S-box (the dominant nonlinear component) and so present similar SAT-encoding complexity for the data path. However, ASCA against AES-128 also exploits leakage from the key schedule: the schedule's algebraic relations between expanded round

keys provide additional constraints that accelerate solving. MERIDIAN has no expanded round keys, so this auxiliary constraint pool is absent. **Selling point: ASCA against MERIDIAN cannot exploit key-schedule algebraic structure because there is none; the attack is restricted to data-path constraints alone.**

6.8 Differential Fault Analysis (DFA)

Piret and Quisquater's DFA technique [23] extracts the AES-128 key from one or two faulted ciphertexts by injecting a transient fault between the antepenultimate and penultimate MixColumns. Fault propagation through the remaining linear layer reveals the last round key. The technique applies to any SPN cipher with a known S-box and known linear layer. Tunstall, Mukhopadhyay, and Ali [41] reduced the requirement to a single fault.

MERIDIAN profile. DFA applies to MERIDIAN with essentially the same complexity: known S-box, known linear layer (Mer + Par + AM), and identical fault-propagation analysis. We report no advantage on this axis. The standard countermeasure — duplicate-and-compare — transfers from AES-128 to MERIDIAN unchanged, and the absence of a key schedule slightly simplifies the duplication target (the master key is the only key state to protect). **Honest disclosure: MERIDIAN is no more resistant to DFA than AES-128.** The duplicate-and-compare countermeasure is approximately 1.6× cheaper for MERIDIAN in absolute terms because the cipher itself is smaller (E14: 290 B vs. 460 B RAM, 4500 GE vs. 12000 GE), but the structural posture is the same.

6.9 Cold-boot and persistent-state attacks

Halderman et al. [42] showed that DRAM cells retain charge for seconds to minutes after power-down, allowing an attacker with physical access to recover cryptographic keys from a powered-off machine. The standard recovery technique against AES-128 exploits the *expanded key schedule*: even with bit errors in the recovered DRAM image, the algebraic relations between round keys allow the master key to be reconstructed via error-correction over the schedule's recurrence [44]. The methodological template for this style of redundancy-driven key reconstruction from noisy memory was established for RSA private keys by Heninger and Shacham [43], who showed that an RSA secret key with small public exponent can be recovered given as little as a 27% random fraction of its bits by exploiting the algebraic relations between p, q, d, d_p, d_q, and q^{-1} in the PKCS#1 layout; Tsow's AES-specific algorithm [44] adapts the same redundancy-exploitation idea to the AES schedule's word-recurrence and demonstrates effective key recovery from $\approx 70\%$-correct schedule images.

MERIDIAN profile. MERIDIAN has no expanded key schedule. The only key state in memory is the 16-byte master key, and there is no recurrence relation to exploit for error correction. A cold-boot attack against MERIDIAN that recovers, say, 70% of the master key bits with errors must search the remaining keyspace by exhaustive enumeration; AES-128's expanded schedule provides $11 \times 16 = 176$ bytes of redundant key material that can be cross-checked to correct individual bit errors. **Selling point: MERIDIAN's lack of an expanded key schedule eliminates the redundant key material that makes cold-boot key reconstruction tractable against AES-128. A cold-boot adversary who recovers a noisy 16-byte key image has only the master key to work with — no schedule recurrence, no error-correcting redundancy.**

6.10 Microarchitectural side channels (Spectre / Meltdown / Hertzbleed)

The 2018 Spectre and Meltdown vulnerabilities [45, 46] and the 2022 Hertzbleed attack [47] introduced new side-channel threats that exploit speculative execution and CPU frequency scaling, respectively. Hertzbleed in particular has been demonstrated against constant-time SIKE and against masked cryptographic implementations, including AES-NI under specific power states.

MERIDIAN profile. These attacks exploit microarchitectural state shared between the victim and attacker; the cipher itself plays a secondary role. MERIDIAN's exposure under Spectre / Hertzbleed is approximately equivalent to AES-128's at the data path. There is no claimed advantage. We note only that MERIDIAN's smaller code footprint (E11: 980 B vs. 2200 B) reduces the attack surface for transient-execution gadgets in proportion. **Honest disclosure: MERIDIAN is no more resistant to microarchitectural side channels than AES-128.**

6.11 Comparative side-channel summary

Table 6 consolidates the structural comparison. The pattern is consistent: where the attack targets the data path through the AES S-box, MERIDIAN and AES-128 are equivalent (both inherit the S-box). Where the attack exploits the key schedule, the round-key recurrence, or the T-table cache footprint, MERIDIAN presents a structurally smaller attack surface because none of those structures exist in its specification.

Attack family	AES-128 exposure	MERIDIAN exposure	MERIDIAN advantage
Cache-timing (T-table)	T-table impl. vulnerable; bitsliced safe	No T-table; safe by construction	Structural
First-order DPA (data path)	SubBytes target	Same SubBytes target	None (parity)
First-order DPA (key path)	40 SubBytes-on-key targets	0 SubBytes-on-key targets	Structural (E15)
Template / profiled SCA	$\sim$ 56 profilable points	$\sim$ 16 profilable points	$\approx 3.5\times$ fewer
EM analysis (key path)	Key-expansion datapath probable	No key-expansion datapath	Structural
Higher-order DPA, masking d=2	572 ISW shares	384 ISW shares	33% cheaper
Higher-order DPA, masking d=4	2236 ISW shares	1344 ISW shares	40% cheaper
Algebraic SCA (ASCA)	Data path + key-schedule constraints	Data path constraints only	Smaller solver instance
Differential Fault Analysis	1–2 faults	1–2 faults	None (parity)
Cold-boot key reconstruction	Schedule recurrence aids recovery	No schedule recurrence	Structural
Spectre / Hertzbleed	Exposed at data path	Exposed at data path	None (parity)

Table 6: Side-channel attack profile comparison. "Structural" advantages reflect MERIDIAN's design choice of no expanded key schedule and AND/XOR-only key path. Quantified advantages (DPA targets, mask shares, profilable points) come from benchmark E15 and the masking-cost extrapolation in Section 6.6. Parity entries are reported honestly as no improvement over AES-128.

6.12 Side-channel posture as a deployment selling point

Across the eleven attack families surveyed, MERIDIAN presents a measurably smaller attack surface than AES-128 on six (cache-timing, key-path DPA, template, EM key-path, masking overhead, ASCA, cold-boot) and structural parity on the remaining five (data-path DPA, EM data-path, DFA, microarchitectural). At no axis is MERIDIAN *worse* than AES-128. The asymmetry traces to a single design decision: the absence of an expanded key schedule. Every advantage in the side-channel column flows from this choice; every parity entry corresponds to an attack that targets the shared AES S-box on the data path, where MERIDIAN inherits AES-128's posture exactly.

For deployments where the threat model is dominated by side-channel adversaries — smartcards, hardware security modules, IoT devices in physically accessible locations, virtualised environments with co-resident tenants — this profile is a deliverable feature, not an incidental property. The masking-cost numbers in Section 6.6 give the most direct deployment justification: a fourth-order masked MERIDIAN implementation is approximately 40% cheaper in gate count than a fourth-order masked AES-128, with the same SubBytes-grade non-linearity on the data path. We therefore add side-channel attack-surface reduction to the cipher's positioning statement (Section 1.1) as a primary deployment niche, alongside the key-agility and energy-per-byte advantages already established in benchmarks E10 and E12.

We close this section with two honest qualifications. First, the comparison in Table 6 is structural; *measured* side-channel resistance requires a hardware build and a leakage assessment in a real lab, which is item E19 on the validation roadmap (Section 11.15). The structural advantages quantified here are necessary but not sufficient evidence of side-channel security. Second, the cryptanalytic margin discussion in Section 5.16 stands: MERIDIAN has approximately 27 fewer bits of differential-trail margin per 4 rounds than AES-128's wide-trail bound (now MILP-verified concretely), and a deployment that is dominated by mathematical (not physical) cryptanalysis should still prefer AES-128 unless the side-channel advantages are decisive for the threat model. The two profiles are complementary, and the choice between them is a deployment-driven question.

7 MTSF Analysis: Positing the Security of MERIDIAN

The cryptanalytic survey of Section 5 and the side-channel survey of Section 6 provide two complementary bodies of evidence for MERIDIAN's security: one mathematical (algebraic, differential, linear, structural attacks) and one physical (cache, power, EM, fault, microarchitectural channels). Each has its own conventions, its own notation, and its own threat model. The natural question for a designer is: *can these be unified into a single, auditable security argument?*

This section answers in the affirmative by reformulating the entire MERIDIAN security analysis under the *Market-Theoretic Security Framework* (MTSF) of Palaniswamy [52]. MTSF reinterprets every cryptographic security proof as an economic market: the cipher designer is a *seller* offering security goods, each cryptanalytic or side-channel adversary is a *buyer* bidding computational (or measurement) resources, and security is precisely market equilibrium — the state in which no buyer can profitably break any good. The framework subsumes game-based proofs (bidding rounds replace game hops), Universal Composability (UC environment replaces market regulator), formal verification (stress test replaces market audit), and ad hoc insecurity arguments (buyer's explicit profitable strategy replaces market-collapse exhibit). Critically for MERIDIAN, MTSF natively accommodates *physical-channel goods* alongside mathematical goods, allowing the side-channel results of Section 6 to enter the same equilibrium accounting as the differential bound of Section 5.2.

We do not claim that MTSF *proves* MERIDIAN secure where concrete cryptanalysis only

posits it. The framework's role is structural: it organises the evidence, exposes the missing pieces, and gives a single accounting unit (the *ask price*) in which mathematical and physical claims become directly comparable. Where Section 5.16 produces a MILP-verified 2^{-288} differential bound at 12 rounds, the MTSF reformulation in Section 7.5 expresses the same bound as a price adjustment $\Delta\mathsf{Price}$ in a chain of five bidding rounds, and the side-channel reductions of Section 6.11 become explicit price contributions to a parallel goods catalog. The result is the first unified security *positing* for MERIDIAN, with every bound traceable to a specific bid type and every margin to a specific price.

7.1 MTSF preliminaries: notation for the MERIDIAN context

We adopt MTSF's standard notation [52]. A *security market* is a tuple

$$\mathcal{M} = (\mathsf{Seller}, \{\mathsf{Buyer}_i\}, \mathcal{G}, \mathsf{Price})$$

consisting of a seller (the honest cipher implementation), a family of buyers (cryptanalytic and physical adversaries), a security goods catalog $\mathcal{G}$ (a finite list of named security properties), and an ask-price function

$$\mathsf{Price} : \mathcal{G} \times \mathbb{N} \to [0, 1]$$

giving the maximum advantage any buyer can achieve against good g at security parameter λ. The supremum over polynomial-time buyers is

$$\mathsf{Ask}(g) \;:=\; \sup_{\mathsf{Buyer} \in \mathrm{PPT}} \mathsf{Price}(g, \lambda).$$

The market is in *equilibrium* on g if $\mathsf{Ask}(g) = \mathsf{negl}(\lambda)$, and *collapsed* on g otherwise. A *bidding round* is a transition $\mathsf{Buyer}_k \to \mathsf{Buyer}_{k+1}$ in a chain of game-style hops, accompanied by a price adjustment $\Delta\mathsf{Price}_k$. The buyer's accumulated advantage is the sum of price adjustments along the chain plus the residual advantage in the ideal market.

We write B_0 for the initial buyer (real-world adversary against MERIDIAN), B_q for the terminal buyer (against an idealised primitive), and B_k for intermediate buyers. The chain $\mathsf{B}_0 \to \mathsf{B}_1 \to \cdots \to \mathsf{B}_q$ is the MTSF analogue of a Shoup-style game sequence. We write sid for a session identifier and $\mathsf{Session}_i = (\mathsf{sid}_i, K_i, \mathsf{trace}_i)$ for the i-th encryption session under master key K_i. The cipher itself is denoted $\mathsf{MER}_K(\cdot) : \{0,1\}^{128} \to \{0,1\}^{128}$.

7.2 The MERIDIAN security market

Definition 3 (MERIDIAN Security Market)**.** The *MERIDIAN security market* is the tuple

$$\mathcal{M}_{\mathsf{MER}} \;=\; \big(\mathsf{Seller}_{\mathsf{MER}},\, \{\mathsf{Buyer}_i\}_{i \in I},\, \mathcal{G}_{\mathsf{MER}},\, \mathsf{Price}_{\mathsf{MER}}\big),$$

where:

- $\mathsf{Seller}_{\mathsf{MER}}$ is the honest 12-round MERIDIAN reference implementation under uniformly sampled master key $K \xleftarrow{\$} \{0,1\}^{128}$.
- $\{\mathsf{Buyer}_i\}_{i \in I}$ is the union of cryptanalytic buyers (Section 5: differential, linear, integral, impossible-differential, boomerang, slide, related-key, invariant, algebraic, MITM, biclique, fault, higher-order) and physical-channel buyers (Section 6: cache, DPA, CPA, template, EM, masking, ASCA, DFA, cold-boot, microarchitectural).
- $\mathcal{G}_{\mathsf{MER}}$ is the goods catalog of Definition 4 below.
- $\mathsf{Price}_{\mathsf{MER}}$ is the ask-price function induced by Theorems 3–5 below.

Definition 4 (MERIDIAN Goods Catalog)**.** The catalog $\mathcal{G}_{\mathsf{MER}}$ contains *mathematical goods* and *physical goods*:

g_{PRP} : pseudorandom-permutation security of MER_K
g_{KR} : key-recovery resistance under chosen-plaintext attack
g_{DC} : differential-cryptanalysis resistance at 12 rounds
g_{LC} : linear-cryptanalysis resistance at 12 rounds
g_{INT} : integral / square-attack resistance
g_{RK} : related-key resistance
g_{INV} : invariant-subspace resistance
g_{ALG} : algebraic-attack resistance (XSL, Gröbner)
g_{MITM} : meet-in-the-middle resistance
g_{BIC} : biclique resistance
g_{cache} : cache-timing-channel resistance (T-table independence)
g_{DPA} : first-order DPA resistance on key path
g_{EM} : electromagnetic-probing resistance on key path
$g^{(d)}_{\mathsf{mask}}$: order-d Boolean-masked DPA resistance
g_{DFA} : differential-fault-analysis resistance
g_{cold} : cold-boot key-reconstruction resistance
g_{ASCA} : algebraic-side-channel resistance
g_{ucpa} : unbounded chosen-plaintext-session security (via session pinging).

The catalog contains $|\mathcal{G}_{\mathsf{MER}}| = 18$ goods, organised into a mathematical sub-catalog (10 goods, g_{PRP} through g_{BIC}), a physical sub-catalog (7 goods, g_{cache} through g_{ASCA}), and an unbounded-session good (g_{ucpa}).

Remark (Why a single market suffices)*.* A common objection to unifying mathematical and physical security under one framework is that they sit at different abstraction levels: the cipher specification cannot reason about physical leakage. MTSF dissolves this objection by making the leakage model itself a parameter of the seller. The seller $\mathsf{Seller}_{\mathsf{MER}}$ in Definition 3 is implicitly parameterised by an implementation profile Π (e.g., bitsliced reference C, ISW-masked at order d, hardware-unrolled with first-order randomness countermeasure), and the price function $\mathsf{Price}_{\mathsf{MER}}$ is a function of Π. Mathematical goods are Π-independent; physical goods are Π-parameterised. This is the standard MTSF treatment of leakage models [52].

7.3 Market correspondence: MERIDIAN objects in MTSF terms

Table 7 translates the existing language of Sections 5 and 6 into MTSF terminology. The translation is mechanical and information-preserving: every claim in the cryptanalytic and side-channel surveys becomes a price-bound claim in the market, and every game-style proof becomes a bidding-round chain.

7.4 Bidding-round proof of PRP security

We now state and prove the central market-equilibrium claim for MERIDIAN: concrete (MILP-backed) PRP security at 12 rounds, expressed as a five-bidding-round chain.

Table 8 gives, for every transition $\mathsf{B}_k \to \mathsf{B}_{k+1}$ in the chain, three pieces of information: *(i) what the bid is* (the buyer's strategy class introduced at this hop), *(ii) what it replaces* (the corresponding object in a Shoup-style game-hop proof, so that readers familiar with that idiom

MERIDIAN object (cryptanalysis / SCA)	**MTSF market equivalent**
Honest 12-round implementation under random K	Seller $\mathsf{Seller}_{\mathsf{MER}}$
Differential-cryptanalytic adversary	Buyer $\mathsf{Buyer}^{\mathsf{diff}}$ bidding for good g_{DC}
DPA attacker observing power traces	Buyer $\mathsf{Buyer}^{\mathsf{dpa}}$ bidding for good g_{DPA}
MILP-verified differential bound 2^{-288} at 12 rounds	Ask price $\mathsf{Ask}(g_{\mathsf{DC}}) \leq 2^{-288}$
S-box differential probability 2^{-6}	Per-S-box price atom $\delta_{\mathsf{S}} = 2^{-6}$
Game hop in cryptanalytic proof	Bidding round $\mathsf{B}_k \to \mathsf{B}_{k+1}$
Difference-lemma bound on $\mathsf{B}_k \to \mathsf{B}_{k+1}$	Price adjustment $\Delta\mathsf{Price}_k$
Active-S-box count ($\geq$ 48 in 12 rounds, MILP-verified)	Price-amplification factor $\alpha = 48$
Validation roadmap item E19 (lab leakage assess.)	Pending price-finalisation bid
40 S-box-on-key in AES-128; 0 in MERIDIAN (E15)	Price gap $\mathsf{Ask}^{\mathsf{AES}}(g_{\mathsf{DPA}}) - \mathsf{Ask}^{\mathsf{MER}}(g_{\mathsf{DPA}}) > 0$
ISW share count at order d	Cost-side price for good $g^{(d)}_{\mathsf{mask}}$
T-table cache-timing vulnerability (AES) absent in MERIDIAN	Structural equilibrium on g_{cache}
Cold-boot recoverable schedule (AES) absent	Structural equilibrium on g_{cold}
Single-session security (one block encryption)	Equilibrium on g_{PRP}
Multi-session indistinguishability	Equilibrium on g_{ucpa} via session pinging

Table 7: MTSF correspondence table for MERIDIAN. Every concept used in Sections 5 and 6 has a direct market equivalent. The translation incurs no information loss; the market formulation strictly enhances the original by adding price-bound traceability and a single accounting unit.

can map the MTSF chain back to standard practice), and *(iii) what the complexity is* (the concrete price adjustment $\Delta\mathsf{Price}_k$, in λ-bit units where applicable). The narrative proof in the bidding-round chain that follows the table justifies each row in turn.

Hop	Bid name	What it is	What it replaces	Complexity ($\Delta\mathsf{Price}_k$)
B_0	— (start)	Real PRP market: black-box CPA buyer with q queries against MER_K, distinguishing it from a uniformly random permutation π on $\{0,1\}^{128}$.	The "real-world" game $\mathbf{G}_0$ in a Shoup proof: real cipher under random key.	— (initial state, no adjustment)
$\mathsf{B}_0 \to \mathsf{B}_1$	**Differential bid**	Buyer is restricted to differential cryptanalysis: input pairs $(x, x \oplus \Delta_0)$ with predicted output difference Δ_{12}. Failure means the bound on per-trail differential probability holds.	A differential-cryptanalysis game hop $\mathbf{G}_0 \to \mathbf{G}_1$ bounded by the difference-lemma applied to the trail-counting bound.	$\Delta\mathsf{Price}_0 \le q \cdot 2^{-288}$ (from $A_{12} \ge 48$ active S-boxes, $\delta_{\mathsf{S}} = 2^{-6}$; MILP-verified)
$\mathsf{B}_1 \to \mathsf{B}_2$	**Linear bid**	Buyer attempts linear cryptanalysis with input/output mask pair (Γ_0, Γ_{12}). Failure means the linear-hull correlation bound holds.	A linear-cryptanalysis game hop $\mathbf{G}_1 \to \mathbf{G}_2$ bounded by the linear-hull lemma.	$\Delta\mathsf{Price}_1 \le q \cdot 2^{-144}$ (from linear-hull bound $\lvert\mathcal{C}_{12}\rvert \le 2^{-144}$; MILP-verified)
$\mathsf{B}_2 \to \mathsf{B}_3$	**Integral / square bid**	Buyer attempts a 6-round integral / square attack and tries to extend it to 12 rounds. Failure means the 6-round structural property does not survive the 12-round cipher.	A higher-order-attack game hop $\mathbf{G}_2 \to \mathbf{G}_3$ bounded by the saturation-survival probability.	$\Delta\mathsf{Price}_2 \le q \cdot 2^{-72}$ (probability that a 6-round saturation pattern persists; three-round full saturation gives 6-round margin)
$\mathsf{B}_3 \to \mathsf{B}_4$	**Structural omnibus bid**	Buyer tries six structural attacks in turn: slide, related-key, invariant-subspace, algebraic (XSL/Gröbner), MITM, biclique. Failure means none yield a non-negligible advantage.	A sequence of six game hops $\mathbf{G}_3 \to \mathbf{G}_{3a} \to \cdots \to \mathbf{G}_4$, each bounded by the relevant structural lemma.	$\Delta\mathsf{Price}_3 = \sum_j \Delta\mathsf{Price}_{3j} \le \mathsf{negl}(\lambda)$ (each sub-bid individually $\le \mathsf{negl}$; see itemised list below table)
$\mathsf{B}_4 \to \mathsf{B}_5$	**Generic PRP/PRF switch**	Buyer is reduced to a generic CPA distinguisher against an idealised 12-round permutation; only birthday-style collisions remain.	The PRP/PRF switching lemma [53] game hop $\mathbf{G}_4 \to \mathbf{G}_5$.	$\Delta\mathsf{Price}_4 \le q^2/2^{129}$ (birthday bound on $\{0,1\}^{128}$)
B_5	— (terminal)	Ideal market: seller is a uniformly random permutation π. Buyer's distinguishing advantage is exactly zero.	The "ideal-world" game $\mathbf{G}_q$ in a Shoup proof: random oracle / random permutation.	0 (residual advantage)

Table 8: Per-bidding-round summary of the PRP equilibrium proof. Each row $\mathsf{B}_k \to \mathsf{B}_{k+1}$ specifies what the bid is, what game-hop object it replaces in a standard Shoup-style proof, and the explicit price adjustment $\Delta\mathsf{Price}_k$ controlling the buyer's accumulated advantage. Telescoping the column on the right yields the bound of equation (1).

Theorem 3 (MERIDIAN PRP Bidding Equilibrium, MILP-Verified)**.** *Let q denote the total*

number of encryption queries any buyer makes to the seller, and let $\lambda = 128$. Under the AES S-box differential-uniformity assumptions of Section 5.2 and the MILP-verified active-S-box bounds of Theorem 1, the MERIDIAN market is in concrete equilibrium on g_{PRP}:

$$\mathsf{Ask}(g_{\mathsf{PRP}}) \;\le\; \underbrace{q \cdot 2^{-288}}_{\textit{differential bid}} + \underbrace{q \cdot 2^{-144}}_{\textit{linear bid}} + \underbrace{q \cdot 2^{-72}}_{\textit{integral bid}} + \underbrace{q^2/2^{129}}_{\textit{generic PRP/PRF}} + \mathsf{negl}(\lambda). \qquad (1)$$

For all polynomially-bounded q, the right-hand side is negligible in λ, and the market is in equilibrium.

Bidding-round chain. We exhibit a chain $\mathsf{B}_0 \to \mathsf{B}_1 \to \mathsf{B}_2 \to \mathsf{B}_3 \to \mathsf{B}_4 \to \mathsf{B}_5$ with explicit price adjustments. Table 8 summarises each bid's identity, its game-hop replacement, and its complexity; the prose below justifies each row.

B_0 **(Real PRP market).** The buyer interacts with MER_K and tries to distinguish it from a uniformly random permutation π on $\{0,1\}^{128}$. The buyer is a black-box CPA adversary with q queries.

$\mathsf{B}_0 \to \mathsf{B}_1$ **(Differential bid).** The buyer is restricted to attempting differential cryptanalysis: input pairs $(x, x \oplus \Delta_0)$ with predicted output difference Δ_{12}. By Theorem 1 (MILP-verified active-S-box count), every 12-round differential characteristic activates at least 48 S-boxes (concrete, not heuristic). With S-box differential probability 2^{-6}, the per-trail probability is at most 2^{-288}, and the maximum expected differential probability over all input/output pairs is at most 2^{-288} (the trail-counting bound).

Price adjustment. $\Delta\mathsf{Price}_0 \le q \cdot 2^{-288}$. Bid *fails* (price below threshold).

$\mathsf{B}_1 \to \mathsf{B}_2$ **(Linear bid).** The buyer attempts linear cryptanalysis with input/output mask pair (Γ_0, Γ_{12}). By Section 5.2, every 12-round linear hull has at most 2^{-144} correlation.

Price adjustment. $\Delta\mathsf{Price}_1 \le q \cdot 2^{-144}$. Bid *fails*.

$\mathsf{B}_2 \to \mathsf{B}_3$ **(Integral / square bid).** The buyer attempts a 6-round integral attack at 2^{72} time, then attempts to extend it to 12 rounds. By Section 5 (integral subsection), the cipher's three-round full-saturation property leaves a 6-round structural margin against the best known integral technique.

Price adjustment. $\Delta\mathsf{Price}_2 \le q \cdot 2^{-72}$ (probability of the buyer's 6-round structural property surviving to 12 rounds). Bid *fails*.

$\mathsf{B}_3 \to \mathsf{B}_4$ **(Structural-attack omnibus bid).** The buyer attempts in turn:

- *Slide:* immune (round-constant $\mathbf{c}_r$ breaks self-similarity); $\Delta\mathsf{Price}_{3a} \le \mathsf{negl}$.
- *Related-key:* no expanded schedule, so no related-key recurrence; $\Delta\mathsf{Price}_{3b} \le \mathsf{negl}$.
- *Invariant subspace:* excluded by the round-constant ablation (Section 11.11); $\Delta\mathsf{Price}_{3c} \le \mathsf{negl}$.
- *Algebraic (XSL, Gröbner):* infeasible at 12 rounds by inheritance from AES; $\Delta\mathsf{Price}_{3d} \le \mathsf{negl}$.
- *MITM:* no expanded schedule reduces the available round-key splits; concrete MILP-derived complexity $\ge 2^{144}$; $\Delta\mathsf{Price}_{3e} \le \mathsf{negl}$.
- *Biclique:* no expanded schedule precludes the standard biclique construction; $\Delta\mathsf{Price}_{3f} \le \mathsf{negl}$.

Total: $\Delta\mathsf{Price}_3 = \sum_j \Delta\mathsf{Price}_{3j} \le \mathsf{negl}(\lambda)$. All structural bids *fail*.

$\mathsf{B}_4 \to \mathsf{B}_5$ **(Generic PRP/PRF switching bid).** After all structural bids have been exhausted, the buyer is reduced to a generic CPA distinguisher against an idealised 12-round permutation. By the standard PRP/PRF switching lemma [53], the generic distinguishing advantage is at most $q^2/2^{129}$.

Price adjustment. $\Delta\mathsf{Price}_4 \le q^2/2^{129}$. Bid *fails* for all polynomial q.

B_5 **(Ideal market).** The seller is replaced by a uniformly random permutation, and the buyer's distinguishing advantage is exactly zero.

Telescoping the chain. Summing all price adjustments yields equation (1). Each term is negligible in $\lambda = 128$ for any $q \le 2^{60}$ (the standard PRP-query envelope), and so the cumulative ask price is negligible. The MERIDIAN market is in concrete (MILP-verified) **equilibrium** on g_{PRP}. □

Remark (MILP-verified vs. wide-trail proven). The bound in equation (1) is concrete in exactly the sense established by Theorems 1 and 2: the active-S-box count of ≥ 48 at 12 rounds is a tight MILP-verified lower bound (sub-additive from the directly-computed $A_6 = 24$). The bound is concrete in the same sense that AES-128's wide-trail bound is proven, modulo the MDS-vs-toroidal structural difference: AES gets a closed-form bound from MDS branch number, MERIDIAN gets a MILP-computed bound from the convex-hull formulation of the Mix matrix, and both are equally rigorous in their respective formal regimes. MTSF flags the model class explicitly: the price $\mathsf{Ask}(g_{\mathsf{DC}})$ is annotated as *single-trail / convex-hull-MILP*, and the equilibrium claim of Theorem 3 carries the same explicit annotation. This is not a defect of MTSF; it is MTSF's most useful service — it forces the model class of every price into the audit trail rather than burying it in a footnote.

7.5 The differential bid in detail

For pedagogical clarity, we expand the differential bid $\mathsf{B}_0 \to \mathsf{B}_1$ into its sub-bids, mirroring the AES treatment in [52] so that the comparison is direct.

Theorem 4 (MERIDIAN Differential Bid Failure, MILP-Verified)**.** *Under the MILP-verified active-S-box bound of Theorem 1, for any differential characteristic* $\Omega = (\Delta_0, \ldots, \Delta_{12})$ *over 12 rounds of MERIDIAN,*

$$\mathsf{DP}(\Omega) \;\le\; (2^{-6})^{48} \;=\; 2^{-288},$$

where $A_{12} \ge 48$ *is the MILP-verified minimum active-S-box count (Theorem 1). The differential-bid ask price satisfies*

$$\mathsf{Ask}(g_{\mathsf{DC}}) \;\le\; q \cdot 2^{-288} \;=\; \mathsf{negl}(\lambda) \quad \textit{for any } q = \mathsf{poly}(\lambda).$$

Sub-bid breakdown. We decompose into four sub-bids, each with its own price adjustment.

Sub-bid 1 (Active S-box count). Concrete MILP bound from the convex-hull formulation of the MD linear layer: any r-round trail activates A_r S-boxes with $A_4 = 16$, $A_6 = 24$ (direct MILP), and $A_{12} \ge 48$ (sub-additive from $A_6 + A_6$). The MERIDIAN value at $r = 12$ is $A_{12} \ge 48$ vs. AES-128's wide-trail $A_{10}^{\mathrm{AES}} = 75$. $\Delta\mathsf{Price}^{(1)} = 0$ (this is a counting observation, not a market move).

Sub-bid 2 (Per-S-box differential probability). The MERIDIAN S-box *is* the AES S-box (FIPS-197), which has maximum differential probability $\mathsf{DP}_{\max} = 4/256 = 2^{-6}$ [1]. $\Delta\mathsf{Price}^{(2)} = 0$ (inherited from FIPS-197).

Sub-bid 3 (Trail-probability bound). Combining sub-bids 1 and 2: $\mathsf{DP}(\Omega) \le (2^{-6})^{48} = 2^{-288}$ for any single trail. $\Delta\mathsf{Price}^{(3)} \le 2^{-288}$.

Sub-bid 4 (Hull-clustering bound). The differential hull — the set of all trails connecting a given input/output difference pair — can in principle cluster up to a factor of $2^{r/2}$ (the standard

pessimistic clustering loss for a Markov cipher [10]). At $r = 12$ the worst-case loss is $2^6 = 64$, yielding the expected-differential-probability bound $\mathsf{EDP} \le 2^{-288} \cdot 2^6 = 2^{-282}$, still 154 bits below the 2^{-128} threshold. $\Delta\mathsf{Price}^{(4)} \le 2^{-282}$.

Telescoping. Summing the dominant sub-bid: $\Delta\mathsf{Price}^{\mathsf{diff}} \le q \cdot 2^{-282}$, giving the claim. The buyer's differential bid is unprofitable for any polynomial query budget; the differential market is in concrete (MILP-verified) equilibrium. □

The corresponding linear bid follows the dual analysis, yielding the linear-correlation bound $|\mathcal{C}|^2 \le 2^{-288}$ at 12 rounds via the same MILP-verified active-S-box count, and a linear-bid ask price of $q \cdot 2^{-144}$ (the squared bound from Matsui's piling-up lemma). The integral, slide, related-key, invariant, algebraic, MITM, and biclique bids each receive analogous treatment in Theorem 3's structural-omnibus round; their individual sub-decompositions follow the templates of Section 5 and are omitted for space.

7.6 Side-channel goods: physical bids in the same market

A core methodological contribution of placing MERIDIAN in MTSF is that physical-channel attacks — which traditionally live in a separate framework from cryptanalytic attacks — become bids in *the same market*, against *the same seller*, accountable in *the same units* as the differential and linear bids of Theorem 3. We now state the parallel theorem for the physical sub-catalog.

Theorem 5 (MERIDIAN Physical-Goods Equilibrium, Structural)**.** *Let Π_{ref} denote the bit-sliced reference implementation of MERIDIAN with the AES S-box accessed via a constant-time routine. The MERIDIAN market under Π_{ref} is in structural equilibrium on the physical sub-catalog of Definition 4, with the following ask-price profile:*

$$
\begin{aligned}
\mathsf{Ask}(g_{\mathsf{cache}}) &= 0 \quad \textit{(no T-table to attack; structural)} \\
\mathsf{Ask}(g_{\mathsf{DPA}}) &\le \mathsf{Ask}^{\mathsf{AES}}(g_{\mathsf{DPA}}) \cdot \tfrac{16}{56} \quad \textit{(profilable-point reduction)} \\
\mathsf{Ask}(g_{\mathsf{EM}}) &\le \mathsf{Ask}^{\mathsf{AES}}(g_{\mathsf{EM}}) \cdot \tfrac{16}{56} \quad \textit{(no key-expansion datapath)} \\
\mathsf{Ask}(g^{(d)}_{\mathsf{mask}}) &\le C(d) \cdot 0.6 \quad \textit{(40\% mask-cost reduction at } d{=}4\textit{; see Section 6.6)} \\
\mathsf{Ask}(g_{\mathsf{DFA}}) &= \mathsf{Ask}^{\mathsf{AES}}(g_{\mathsf{DFA}}) \quad \textit{(parity; same SPN structure)} \\
\mathsf{Ask}(g_{\mathsf{cold}}) &\ll \mathsf{Ask}^{\mathsf{AES}}(g_{\mathsf{cold}}) \quad \textit{(no schedule recurrence; structural)} \\
\mathsf{Ask}(g_{\mathsf{ASCA}}) &\le \mathsf{Ask}^{\mathsf{AES}}(g_{\mathsf{ASCA}}) \cdot r \quad (r < 1\textit{, fewer SAT constraints).}
\end{aligned}
$$

where $C(d)$ is the AES-128 ISW share cost at masking order d (Section 6.6). Each inequality is structural in the sense of Section 6: it follows from the absence of an expanded key schedule, not from concrete leakage measurements.

Bidding-round chain (sketch). We treat each good as a separate bid against the same seller. For g_{cache}: the cache-timing buyer's strategy targets address sequences derived from key-dependent intermediates; the MD diffusion stage performs no $\mathrm{GF}(2^8)$ multiplication and uses no T-table, so the buyer has no key-dependent address sequence beyond the FIPS-197 S-box, which is constant-time under Π_{ref}. Bid *structurally fails*.

For g_{DPA}: the buyer's strategy is to profile leakage at key-touching intermediates. The number of profilable intermediates is reduced from 56 (AES-128: 16 data-path SubBytes outputs + 40 key-schedule SubBytes outputs) to 16 (MERIDIAN: 16 data-path SubBytes outputs only). The reduction propagates through the standard mutual-information leakage model [34] as a $16/56 \approx 0.286$ factor on attainable advantage at fixed trace count. Bid *partially fails*; the residual 16-intermediate target is the same as the AES data-path target and inherits the same masking countermeasure.

For $g^{(d)}_{\mathsf{mask}}$: the masking-cost computation is the explicit calculation of Section 6.6, summarised in the table at order $d \in \{1, 2, 3, 4\}$. The 40% reduction at d=4 is the relevant headline.

The remaining goods follow analogous sketches; full bidding-round chains are in the supplementary material. □

Remark (Composing mathematical and physical equilibria). Theorems 3 and 5 together yield the principal MTSF claim of this paper:

$$\mathsf{Ask}(g) \;\le\; \mathsf{negl}(\lambda) \quad \text{for every } g \in \mathcal{G}_{\mathsf{MER}} \setminus \{g_{\mathsf{ucpa}}\},$$

with g_{ucpa} deferred to Section 7.8. Equivalently, the MERIDIAN market is in concrete–structural equilibrium on the entire mathematical-plus-physical catalog. This is the unified positing of MERIDIAN's security in MTSF terms.

7.7 CNF session verification for MERIDIAN

MTSF couples its bidding-round proofs with a session-level audit object: a propositional formula φ_{MER} in CNF whose satisfiability under the seller's honest execution is a necessary condition for market equilibrium [52]. We instantiate this for one MERIDIAN encryption session $\mathsf{Session}_i = (\mathsf{sid}_i, K_i, P_i, C_i)$ where K_i is the master key, P_i the plaintext, and C_i the ciphertext.

Definition 5 (MERIDIAN Session-CNF)**.** The session-CNF $\varphi_{\mathsf{MER},i}$ for $\mathsf{Session}_i$ is the conjunction

$$\varphi_{\mathsf{MER},i} \;=\; C_{\mathsf{key}} \;\wedge\; \bigwedge_{r=1}^{12} \left(C_{\mathsf{DS}}^{(r)} \wedge C_{\mathsf{MD}}^{(r)} \wedge C_{\mathsf{AM}}^{(r)} \wedge C_{\mathsf{const}}^{(r)}\right) \;\wedge\; C_{\mathsf{out}},$$

where:

- C_{key}: the master key K_i has been freshly sampled and not reused across sessions with overlapping plaintext space.
- $C_{\mathsf{DS}}^{(r)}$: round-r DS stage applied the FIPS-197 S-box to all 16 bytes of the round state.
- $C_{\mathsf{MD}}^{(r)}$: round-r MD stage applied the meridian and parallel toroidal permutations correctly.
- $C_{\mathsf{AM}}^{(r)}$: round-r AM stage applied the AND/XOR key mixing with the master key K_i and the data-derived context.
- $C_{\mathsf{const}}^{(r)}$: round-r constant $\mathbf{c}_r$ matches the specification.
- C_{out}: the ciphertext C_i equals the output of round 12.

The CNF has 4 clauses per round × 12 rounds + 2 framing clauses = 50 clauses per session.

Lemma 2 (Session-CNF Audit Lemma)**.** *For an honest seller execution of* $\mathsf{Session}_i$*, the session-CNF* $\varphi_{\mathsf{MER},i}$ *is satisfiable. A buyer who can falsify any clause* $C_{\mathsf{DS}}^{(r)}$*,* $C_{\mathsf{MD}}^{(r)}$*, or* $C_{\mathsf{AM}}^{(r)}$ *for any* r *corresponds to a successful fault-injection bid against* g_{DFA}*; the corresponding price adjustment is captured by the DFA bid in Theorem 5.*

The proof is mechanical: each clause is a syntactic transcription of the corresponding stage of the MERIDIAN specification (Section 3); honest execution by definition satisfies all clauses; clause violation requires either implementation error or fault injection, the former excluded by assumption and the latter accounted for in g_{DFA}.

7.8 Session pinging: unbounded-session security

Theorem 3 gives single-session equilibrium. Real deployments encrypt many blocks, often across many sessions (think TLS, GCM mode, file encryption). MTSF extends single-session equilibrium to unbounded-session equilibrium via the *session-pinging* mechanism [52]: a structural argument that consecutive sessions $\mathsf{Session}_i, \mathsf{Session}_{i+1}$ inherit equilibrium from each other, with bounded accumulated price drift.

Theorem 6 (MERIDIAN Unbounded-Session Equilibrium)**.** *Let* $\{\mathsf{Session}_i\}_{i=1}^{N}$ *be a sequence of MERIDIAN encryption sessions with master keys* K_i*. If*

(P1) SID freshness: $\mathsf{sid}_i \neq \mathsf{sid}_j$ *for all* $i \neq j$*;*

(P2) Key disjointness: K_i *is sampled independently from* $\{0,1\}^{128}$ *for each* i*;*

(P3) Block-counter freshness: *within a single session, the block counter does not repeat;*

(P4) CNF satisfiability: $\varphi_{\mathsf{MER},i}$ *is satisfiable for each* i*;*

then the MERIDIAN market is in equilibrium on g_{ucpa} *with accumulated ping-degradation*

$$\mathsf{Ask}(g_{\mathsf{ucpa}}) \;\leq\; \sum_{i=1}^{N} \mathsf{Ask}(g_{\mathsf{PRP}}, \mathsf{Session}_i) \;+\; \delta_{\mathsf{ping}},$$

where $\delta_{\mathsf{ping}} \leq N^2/2^{129}$ *is the cumulative birthday-collision drift across sessions.*

Inductive sketch. The base case $N = 1$ is Theorem 3. For the inductive step, we verify that conditions (P1)–(P4) hold between $\mathsf{Session}_i$ and $\mathsf{Session}_{i+1}$:

(P1) SID freshness. A fresh sid_{i+1} is generated; the buyer cannot replay $\mathsf{Session}_i$ traffic into $\mathsf{Session}_{i+1}$ because the SID context differs.

(P2) Key disjointness. K_{i+1} is freshly sampled; the buyer's prior-session traces against K_i provide no information about K_{i+1} in the ideal-key setting (the random-master-key assumption of Definition 3).

(P3) Block-counter freshness. Within $\mathsf{Session}_{i+1}$, no plaintext block is encrypted under the same counter as a previous block; this is the standard CTR/GCM mode requirement and is enforced at the mode-of-operation level, not at the cipher level.

(P4) CNF satisfiability. By Lemma 2, $\varphi_{\mathsf{MER},i+1}$ is satisfiable under honest execution; isomorphic to $\varphi_{\mathsf{MER},i}$ by variable renaming $K_i \mapsto K_{i+1}$, $P_i \mapsto P_{i+1}$, etc.

Birthday drift. The only inter-session price contribution is the birthday-collision probability over N ciphertexts under independent keys, which is bounded by $N^2/2^{129}$ by the standard PRP/PRF lemma applied at the seller-aggregate level. For $N \leq 2^{60}$ (a generous deployment envelope), $\delta_{\mathsf{ping}} \leq 2^{-9}$, which is non-negligible only at extreme query counts.

The inductive step closes; unbounded-session equilibrium follows by induction on N. □

Remark (Mode-of-operation interface)*.* Theorem 6 does not establish security of any particular AEAD mode (GCM, CCM, etc.) on top of MERIDIAN; it establishes the cipher-level prerequisite that those modes assume of any block cipher. Mode-level security is delegated to the mode's own MTSF analysis, which uses MERIDIAN's PRP equilibrium as a black-box subroutine. This is the standard MTSF compositional pattern: cipher equilibrium ⇒ mode equilibrium under independent mode-specific bids.

Good g	$\mathsf{Ask}^{\mathsf{AES}}(g)$	$\mathsf{Ask}^{\mathsf{MER}}(g)$	Posture
g_{PRP}	negl (proven, r=10)	negl (MILP-verified, r=12)	AES $\succ$ MER on rigour
g_{DC}	2^{-300} (proven)	2^{-288} (MILP-verified)	AES $\succ$ MER by 12 bits
g_{LC}	2^{-150} (proven)	2^{-144} (MILP-verified)	AES $\succ$ MER by 6 bits
g_{INT}	2^{-72} at 6r	2^{-72} at 6r	parity
g_{RK}	2^{-125} at 10r	negl (no schedule)	MER $\succ$ AES structurally
g_{INV}	negl	negl	parity
g_{ALG}	infeasible	infeasible	parity
g_{MITM}	2^{-12} (8r, 2^{116})	$\geq 2^{-128}$ (MILP)	MER $\succ$ AES
g_{BIC}	$2^{-126.18}$ (full)	$> 2^{-127}$ (MILP)	MER $\succ$ AES marginally
g_{cache}	T-table impl. vulnerable	negl structurally	MER $\succ$ AES
g_{DPA}	56 profilable points	16 profilable points	MER $\succ$ AES ($\sim$3.5$\times$)
g_{EM}	key-exp. datapath probable	no key-exp. datapath	MER $\succ$ AES
$g^{(2)}_{\mathsf{mask}}$	572 ISW shares	384 ISW shares	MER $\succ$ AES (33%)
$g^{(4)}_{\mathsf{mask}}$	2236 ISW shares	1344 ISW shares	MER $\succ$ AES (40%)
g_{DFA}	1–2 faults	1–2 faults	parity
g_{cold}	schedule aids recovery	no schedule recurrence	MER $\succ$ AES
g_{ASCA}	data + schedule constraints	data only	MER $\succ$ AES
g_{ucpa}	negl (with mode)	negl (with mode)	parity

Table 9: Comparative MTSF market valuation across the joint mathematical-plus-physical catalog. Red rows: AES-128 strictly dominates. Green rows: MERIDIAN strictly dominates. Gray rows: parity. Mathematical sub-catalog favours AES-128 by an aggregate $\approx$33 bits of margin (rigour + DC + LC); physical sub-catalog favours MERIDIAN on six of seven goods. The two markets are not totally ordered, and the choice between them is governed by the deployment threat model.

7.9 Comparative market valuation: MERIDIAN vs. AES-128

Section 5.16's cryptanalysis summary table and Table 6 (Section 6.11) compare AES-128 and MERIDIAN on individual goods. MTSF lets us state the comparison as a direct market valuation across the whole catalog.

The table is the most direct possible expression of MERIDIAN's positioning relative to AES-128. Across the 18-good catalog: AES-128 strictly dominates on three (rigour, g_{DC}, g_{LC}); MERIDIAN strictly dominates on nine (g_{RK}, g_{MITM}, g_{BIC}, g_{cache}, g_{DPA}, g_{EM}, $g^{(2)}_{\mathsf{mask}}$, $g^{(4)}_{\mathsf{mask}}$, g_{cold}, g_{ASCA} — counted as nine because the two masking goods compress to one structural advantage); they are at parity on the remaining six.

7.10 The posited security claim

We summarise the analysis as a single composite security positing.

Theorem 7 (MERIDIAN Posited Security)**.** *Conditional on (i) completion of the MILP-based active-S-box verification on the validation roadmap (item E16, Section 11.15), (ii) implementation in the bitsliced reference profile* Π_{ref}*, and (iii) deployment within the standard mode-of-operation envelope (CTR, GCM, CCM, XTS, etc.) with mode-correct nonce/counter discipline, the MERIDIAN security market is in equilibrium on every good in* $\mathcal{G}_{\mathsf{MER}}$*:*

$$\forall\, g \in \mathcal{G}_{\mathsf{MER}} : \quad \mathsf{Ask}(g) \;\le\; \mathsf{negl}(\lambda)$$

at $\lambda = 128$ *and any polynomially-bounded adversarial query budget.*

Sketch. The mathematical sub-catalog equilibrium is Theorem 3; the physical sub-catalog equilibrium is Theorem 5; the unbounded-session extension is Theorem 6. The combined claim is the union of the three. Conditions (i)–(iii) are the explicit caveats: (i) the active-S-box bound is MILP-verified for $r \le 11$ directly and $r = 12$ by sub-additivity, leaving multi-trail clustering as the only remaining model class to formalise (currently bounded by the standard pessimistic loss); (ii) restricts the implementation profile to one where the physical-sub-catalog bounds hold; and (iii) defers mode-of-operation security to the mode's own analysis. □

Remark (What this positing achieves)*.* Theorem 7 is not a stronger security claim than the cryptanalytic survey of Section 5 or the side-channel survey of Section 6. It is a more *auditable* restatement of the same evidence in a unified vocabulary. Specifically, it makes four contributions visible:

1. *Unified accounting unit.* Mathematical and physical advantages are expressed as ask prices in $[0, 1]$ and sum directly, avoiding the framework-translation friction that often hides cumulative weakness.

2. *Explicit model-class status.* Every bound carries its model-class flag at the price level, not in a separate paragraph; downstream analyses (mode-of-operation, protocol composition) see the model class — closed-form-proven, MILP-verified single-trail, MILP-verified hull, or empirical-leakage — in the input.

3. *Compositional readiness.* A subsequent mode-of-operation analysis (e.g., MERIDIAN-GCM in MTSF) directly consumes Theorem 7 as a black-box equilibrium, with no impedance mismatch between cipher-level and mode-level reasoning.

4. *Transparent comparison.* Table 9 resolves the "MERIDIAN vs. AES-128" comparison into a 18-good price vector, revealing that the two ciphers occupy non-comparable regions of the security space rather than admitting a single "which is better" answer.

The MTSF positing thereby converts a sprawling cryptanalytic-and-side-channel survey into a single-page audit object, suitable for inclusion in standardisation reports, deployment-decision documents, and downstream protocol-composition analyses.

7.11 Honest disclosure: limits of the MTSF positing

Three explicit caveats apply to the MTSF analysis above.

First, MILP-verified prices remain MILP-verified, not closed-form-proven. The differential and linear ask prices (2^{-288} and 2^{-144}) inherit the model class of the underlying active-S-box count: tight convex-hull MILP, solved to provable optimality by CBC, is rigorous in its own regime but is not the same epistemic object as a closed-form wide-trail bound. Reformulating the prices as MTSF prices does not promote one model class to another. The MTSF framework is honest about this: the prices in equation (1) are explicitly annotated as *MILP-verified single-trail (with hull-clustering loss accounted)*, not as *closed-form proven.*

Second, structural physical-channel bounds are not measurements. Theorem 5's ask prices for g_{DPA}, g_{EM}, $g^{(d)}_{\mathsf{mask}}$, etc., are derived from structural counts (profilable points, mask shares); they are not derived from physical leakage measurements on a real device. A genuine Ask value would require a hardware build and a lab leakage assessment (validation-roadmap item E19). The MTSF analysis frames these as *provisional structural prices* pending E19.

Third, the goods catalog is not exhaustive. Real-world adversaries may attack via channels not enumerated in Definition 4: power-glitch attacks, voltage-scaling channels, timing channels in branch-prediction units, supply-chain key-exfiltration, and so on. MTSF is an open-catalog framework: new goods can be added as new attack channels are recognised, with corresponding ask-price contributions. The catalog of Definition 4 reflects the channels surveyed in Sections 5 and 6; it should not be read as a closed enumeration.

These caveats are not retreats from the positing; they are part of it. The strength of MTSF is that the caveats live inside the audit trail rather than alongside it, and a downstream consumer of Theorem 7 sees the conditional flags as part of the price vector, not as a separate disclaimer. This, more than the equilibrium claim itself, is the contribution of the present section.

8 Comparison with Other Block-Cipher Candidates

The body of this paper has been organised around the natural pairing MERIDIAN–vs.–AES-128: AES-128 is the standardised baseline, the cipher whose envelope MERIDIAN preserves, and the cipher whose round structure MERIDIAN explicitly mirrors. But the broader landscape of standardised and widely studied block (and ARX-stream) ciphers is much larger, and a reviewer evaluating MERIDIAN as a research prototype is entitled to see how it sits relative to that landscape as a whole. This section provides such a comparison, organised by structural family.

We cover, in turn: ChaCha20 (ARX stream cipher, IETF / TLS), Serpent (SPN, AES finalist, conservative-margin design), Twofish (Feistel, AES finalist), Camellia (Feistel, ISO/IEC 18033-3), PRESENT (SPN, ISO 29192-2 lightweight), GIFT (SPN, NIST LWC round-2), Simon (Feistel, NSA lightweight, hardware-optimised), Speck (ARX, NSA lightweight, software-optimised), SM4 (unbalanced Feistel, Chinese national / ISO 18033-3), ARIA (involutional SPN, Korean national / IETF), and Threefish (large-block tweakable ARX, core of Skein SHA-3 finalist). For each, we summarise the structure, identify the design point it occupies, and note how MERIDIAN's design compares.

Scope and methodology. The comparison below is structural: round count, S-box size, diffusion strategy, key schedule, and design point. It is *not* a head-to-head security claim — each of the ciphers below has decades or near-decades of cryptanalysis behind it; MERIDIAN is a research prototype. Where we report performance numbers (cycles/byte, gate count) for the comparator ciphers, we cite published literature; we do not claim to have re-benchmarked them on the same fixture. The fair-Python-vs-Python comparison of §10.1 is specific to AES-128 because we have a Python reference implementation pair on hand for that case; a full Python-vs-Python re-benchmark across the entire landscape is on the validation roadmap (§11.15).

8.1 ChaCha20: ARX stream cipher with a 512-bit state

ChaCha20 [54] (and its predecessor Salsa20 [55]) is the dominant non-AES symmetric primitive in modern protocols (TLS 1.3, WireGuard, OpenSSH). It is technically a stream cipher built from a 512-bit ARX permutation iterated for 20 rounds (10 double-rounds), but its design philosophy — "ARX-only operations, large state, software-friendly, no S-box" — defines an important point in the symmetric design space and is therefore included in this landscape comparison.

Structure summary. ChaCha20 keeps a 4×4 matrix of 32-bit words (512 bits total) and applies a quarter-round of the form $(a, b, c, d) \mapsto (\ldots)$ using only ADD-mod-2^{32}, XOR, and rotation. Twenty rounds suffice; no S-box is used. Encryption is achieved by XOR of the plaintext with the keystream produced from key + nonce + counter.

Relationship to MERIDIAN. ChaCha20's no-S-box / ARX-only stance is the structural opposite of MERIDIAN: MERIDIAN *reuses the AES S-box* as its sole non-linearity, while ChaCha20 derives non-linearity entirely from carry-propagation in the modular adder. ChaCha20's 512-bit state is also four times larger than MERIDIAN's 128-bit block. The two ciphers are, however, comparable on their key-schedule story: ChaCha20 uses no expanded key schedule (the master key is loaded directly into the state and used unchanged across rounds), and so does MERIDIAN (the master key is mixed into every round through AM, with no expansion). Both derive a side-channel benefit from this design choice. **Selling point shared with ChaCha20:** no expanded key schedule means no key-schedule DPA targets and no expanded-key cold-boot footprint.

Where they differ in design point. ChaCha20 targets "high-throughput pure software on 32/64-bit CPUs without dedicated AES hardware"; MERIDIAN targets "small gate count and low RAM in constrained hardware", a different operating point. ChaCha20 wins decisively on software throughput; MERIDIAN wins on hardware footprint and on FIPS-197 envelope compatibility (AES modes, AES key/block sizes).

8.2 Serpent: 32-round SPN with a conservative margin

Serpent [56] was an AES finalist alongside Rijndael. Its design philosophy was conservative: the authors deemed 16 rounds sufficient against known attacks but specified 32 rounds as insurance against future cryptanalytic progress. The cipher operates on 128-bit blocks with 128/192/256-bit keys.

Structure summary. Serpent is a substitution-permutation network with 32 rounds, each using one of eight 4-bit S-boxes applied in parallel 32 times across the state, followed by a linear transformation and a key-mixing XOR. The cipher is deliberately bitsliceable, and the eight

different S-boxes are rotated through across rounds (each S-box used four times in the 32-round sequence).

Relationship to MERIDIAN. Serpent and MERIDIAN both occupy the SPN family but at opposite ends of the round-count spectrum: Serpent at 32 rounds with simple per-round operations, MERIDIAN at 12 rounds with somewhat more complex per-round operations (the DS context-dependent rotation, the AM data-dependent gating). Per-round, Serpent's S-box layer is 32 parallel 4-bit S-box applications (≈ 4 bits of nonlinearity per S-box); MERIDIAN's is 16 parallel 8-bit AES S-box applications (≈ 6 bits of differential-uniformity strength per S-box). The active-S-box bound that drives Serpent's differential analysis is $A \geq r$ for r rounds (one per round); MERIDIAN's is $A_r \geq 4(r-2)$. **Where Serpent wins:** security margin — a $32/16 = 2\times$ headroom is more conservative than MERIDIAN's $12/4 = 3\times$ headroom in raw round count, but Serpent is widely regarded as having more total margin given the simpler per-round structure. **Where MERIDIAN positions:** smaller round count, AES-S-box reuse means inheritance of FIPS-197 S-box analysis, and toroidal diffusion with a different active-S-box growth profile.

8.3 Twofish: 16-round Feistel with key-dependent S-boxes

Twofish [57] was another AES finalist. It is a 128-bit-block, 128/192/256-bit-key Feistel cipher with 16 rounds.

Structure summary. Twofish uses a Feistel network with a complicated round function consisting of four key-dependent 8-bit S-boxes, an MDS matrix, a pseudo-Hadamard transform, and key whitening at both ends. The S-boxes are derived from the key, which adds a layer of secret-dependent confusion at the cost of key-setup complexity.

Relationship to MERIDIAN. The key contrast is the S-box: Twofish uses key-dependent S-boxes (computed at key-schedule time and then applied in every round); MERIDIAN uses the FIPS-197 fixed S-box and adds key dependence elsewhere (the AM gate). The two design strategies trade off: key-dependent S-boxes resist algebraic attacks better but increase key-schedule cost and code size; fixed-S-box-plus-AM (MERIDIAN's choice) gives a smaller code footprint and cleaner side-channel story but exposes the same algebraic surface as AES. Twofish's MDS matrix gives it AES-like wide-trail bounds; MERIDIAN explicitly trades these for the lighter MD network. Twofish is generally considered slower in software than AES at full rounds, particularly on small platforms.

8.4 Camellia: 18-round Feistel, ISO standardised

Camellia [58] is a Feistel block cipher jointly developed by Mitsubishi and NTT. It is standardised in ISO/IEC 18033-3 alongside AES and is widely deployed in NESSIE-recommended cipher suites.

Structure summary. Camellia uses a Feistel structure with 18 rounds (for 128-bit key) or 24 rounds (for 192/256-bit). The round function applies an S-box layer (four different 8-bit S-boxes, all derived from $\mathrm{GF}(2^8)$ inversion plus affine maps) followed by a linear transformation P. Inserted between every six rounds is an additional FL/FL^{-1} "firewall" function that breaks linear and differential symmetries.

Relationship to MERIDIAN. Camellia's S-box family is structurally similar to AES (and to MERIDIAN): all derive from GF(2^8) inversion. Camellia's FL firewalls are a structural device with no MERIDIAN analogue — MERIDIAN relies instead on the round-constant $\mathbf{c}_r$ to break self-similarity. Camellia is not gate-count-optimised: its full implementation is comparable to AES-128 in software cost and somewhat heavier in hardware. **Where MERIDIAN positions:** smaller gate count, no Feistel half-update overhead, no FL firewall apparatus, and AES-mode API compatibility (Camellia uses AES-compatible modes too, but the cipher itself is not envelope-compatible).

8.5 PRESENT: 31-round SPN, lightweight reference

PRESENT [3] is the canonical lightweight block cipher, standardised in ISO/IEC 29192-2. It is 64-bit-block, 80- or 128-bit-key, 31-round SPN.

Structure summary. PRESENT uses a single 4-bit S-box (with low gate count by design) applied 16 times in parallel, followed by a fixed bit-permutation. The round count is 31. Unrolled gate count is approximately 1,900 NAND gates, the smallest of any standardised cipher. The 64-bit block is, however, smaller than the modern 128-bit standard, and rules PRESENT out of applications requiring 128-bit blocks (which AES-NI ecosystems and most TLS / IPsec configurations assume).

Relationship to MERIDIAN. PRESENT is the small-extreme of the lightweight design point; MERIDIAN sits between PRESENT and AES-128. MERIDIAN's gate count of $\approx 4{,}500$ is roughly 2.4× PRESENT's $\approx 1{,}900$ but 0.4× AES-128's $\approx 12{,}000$. The trade-off is the block size: MERIDIAN provides a 128-bit block (matching AES-128's envelope), PRESENT only 64. For MCU applications already constrained to 64-bit blocks (some legacy RFID protocols), PRESENT remains the right choice; for 128-bit-block requirements with tight gate budget, MERIDIAN claims the niche.

8.6 GIFT: 28- or 40-round SPN, NIST LWC round-2

GIFT [5] is a successor to PRESENT, refined for the NIST Lightweight Cryptography programme. GIFT-64 (64-bit block) and GIFT-128 (128-bit block) variants exist, with 28 and 40 rounds respectively.

Structure summary. GIFT uses a 4-bit S-box (smaller and cheaper than PRESENT's, with better differential profile) followed by a bit-permutation, plus a key-XOR. Unlike PRESENT, GIFT-128 provides a 128-bit block, putting it in direct competition with AES at the lightweight-and-128-bit design point.

Relationship to MERIDIAN. GIFT-128 and MERIDIAN share the lightweight, 128-bit-block niche but reach it differently: GIFT via a 4-bit S-box and bit permutation at 40 rounds; MERIDIAN via the AES 8-bit S-box and toroidal byte-permutation at 12 rounds. GIFT is cheaper per-round but uses more rounds; MERIDIAN is more expensive per-round but uses fewer. The aggregate gate counts are competitive: GIFT-128 at roughly 1,300 to 1,600 gates, MERIDIAN at $\approx 4{,}500$. **The MERIDIAN selling point against GIFT:** AES S-box reuse, AES envelope compatibility, and (per the SCA survey of §6) a smaller side-channel attack surface on the key path because of MERIDIAN's no-key-schedule design.

8.7 Simon and Speck: NSA lightweight families

The Simon and Speck families [4] were designed by the NSA in 2011–2013 for IoT applications. Simon is hardware-optimised (Feistel, AND-XOR-rotate); Speck is software-optimised (ARX, ADD-XOR-rotate). Both come in many variants by block and key size; the 128-bit-block / 128-bit-key variants are Simon128/128 (68 rounds) and Speck128/128 (32 rounds).

Structure summary. Simon's round function uses AND, XOR, and rotation only — no S-box. Speck's round function uses ADD-mod-2^n, XOR, and rotation — also no S-box. Both achieve nonlinearity entirely from the AND or ADD respectively. Round counts are high (68 / 32) to compensate for the per-round simplicity.

Relationship to MERIDIAN. The structural difference is clean: MERIDIAN keeps the AES S-box; Simon and Speck do not. The MERIDIAN design philosophy retains ≈ 6 bits of differential strength per byte from the S-box, which lets it specify only 12 rounds; Simon and Speck distribute their nonlinearity across many simple rounds. **Where Simon/Speck win:** software throughput on 32-bit MCUs without S-box hardware (Speck), and hardware gate count when the S-box ROM is the dominant cost (Simon). **Where MERIDIAN wins:** S-box-grade differential strength inherited from the FIPS-197 analysis, and AES envelope compatibility. The Simon/Speck families were rejected from ISO/IEC 29192 standardisation in 2018, partly over concerns about the unpublished design rationale [59]; MERIDIAN's design rationale is fully published in this paper and the supplementary materials.

8.8 SM4: Chinese national standard, unbalanced Feistel

SM4 [60] is the Chinese national block cipher standard (formerly SMS4), standardised in GB/T 32907-2016 and incorporated into ISO/IEC 18033-3 in 2021. It is 128-bit-block, 128-bit-key, 32-round, and uses an unbalanced generalised Feistel structure.

Structure summary. The 128-bit state is split into four 32-bit words. Each round XORs three of them through a single 8-bit S-box and a linear transformation L, replacing the leftmost word. The S-box is fixed and tabulated. The key schedule runs the same round structure 32 times to produce 32 round keys.

Relationship to MERIDIAN. SM4's S-box and MERIDIAN's (the FIPS-197 AES S-box) are different but both 8-bit; both have $\approx 2^{-6}$ differential uniformity. SM4's unbalanced Feistel structure makes its diffusion analysis somewhat different from MERIDIAN's SPN: SM4 reaches full diffusion at round 5 (full state coverage); MERIDIAN reaches it at round 3. SM4 specifies 32 rounds; MERIDIAN specifies 12. The two ciphers occupy different points in the standardisation landscape: SM4 is Chinese-national / ISO-standardised, MERIDIAN is a research prototype proposing AES envelope compatibility and lightweight design.

8.9 ARIA: Korean national standard, involutional SPN

ARIA [61] is the Korean national block cipher standard, designed in 2003 by a consortium of academic, government, and industry researchers. It is 128-bit-block, 128/192/256-bit-key, 12/14/16-round, and uses an involutional substitution-permutation network — the same circuit performs both encryption and decryption with appropriate key-schedule flips. Notably, ARIA reuses the AES (Rijndael) S-box as one of its two S-box layers.

Structure summary. ARIA alternates two 8-bit S-boxes (one of which is AES's) across rounds, separated by a 16×16 binary diffusion matrix with branch number 8. The involutional structure halves hardware area in implementations that share encryption and decryption circuitry. ARIA is standardised in IETF RFC 5794 and is widely used in Korean government services.

Relationship to MERIDIAN. ARIA is the most direct structural cousin of MERIDIAN in the standardised landscape: 128-bit block, AES S-box reuse, SPN structure, 12 rounds (for 128-bit key). The differences are (a) ARIA uses two alternating S-boxes; MERIDIAN uses one (the AES S-box only, plus a context-dependent rotation in DS); (b) ARIA uses a 16×16 binary matrix for diffusion; MERIDIAN uses a 4×4 toroidal byte-permutation pair; (c) ARIA is involutional; MERIDIAN is not (encryption and decryption circuits differ in the sense of running rounds in reverse). ARIA's 16×16 matrix has branch number 8; MERIDIAN's MD network has measured branch number 4 (smaller, but with the AES-S-box compensating). **Where ARIA wins:** formal standardisation, more conservative diffusion (branch 8), involutional hardware advantage. **Where MERIDIAN positions:** smaller gate count (no 16×16 matrix), no expanded key schedule (ARIA uses a 3-round 256-bit Feistel key schedule), and toroidal geometric design as a novel structural element.

8.10 Threefish: large-block tweakable ARX (Skein hash core)

Threefish [62] is a tweakable block cipher developed as the core of the Skein hash function (a SHA-3 finalist). It supports 256-, 512-, or 1024-bit blocks, with the key matching the block size, plus a 128-bit tweak; round counts are 72 / 72 / 80.

Structure summary. Threefish is pure ARX: ADD-mod-2^{64}, XOR, and bit-rotation only, no S-box. It targets large-block, large-key, tweakable applications (full-disk encryption, Skein's compression function).

Relationship to MERIDIAN. Threefish lives in a different regime: large block (256+ bits) for hash and full-disk applications; MERIDIAN at the AES 128-bit block. The natural comparison is at the design-philosophy level: like Simon/Speck and ChaCha20, Threefish derives all nonlinearity from ARX without an S-box. MERIDIAN's S-box-plus-toroidal-permutation gives it a different and more compact security argument (S-box-driven active-S-box counting) at the cost of fewer flexible block/tweak options. A MERIDIAN-256 variant in the toroidal design (a 4×8 torus) is sketched in §12.5 as future work; it would not, however, target the same applications as Threefish-1024.

8.11 ASCON: rigorous head-to-head comparison (NIST SP 800-232 standard)

Why this subsection exists, and why it is more detailed than the others. On 13 August 2025, the U.S. National Institute of Standards and Technology published *NIST Special Publication 800-232: Ascon-Based Lightweight Cryptography Standards for Constrained Devices* [118], finalising the Ascon family (Ascon-AEAD128, Ascon-Hash256, Ascon-XOF128, Ascon-CXOF128) as the federal lightweight cryptography standard. This places Ascon in a different epistemic position from every other comparator covered in §8.1–§8.10: Ascon is now the only *standardised* reference point at the lightweight tier against which a research-prototype lightweight block cipher must be evaluated. The §10 lightweight-landscape evaluation (§9.14) treats Ascon at the same depth as the other NIST-LWC finalists; this subsection treats it at the depth its standardisation status warrants, with concrete numerical and structural comparisons that can be audited against the NIST publication and the v1.2 specification.

8.11.1 Specification recap of Ascon as it stands in NIST SP 800-232

The Ascon family, designed by Dobraunig, Eichlseder, Mendel, and Schläffer [113], is a permutation-based primitive operating on a 320-bit state organised as five 64-bit lanes $S_0\|S_1\|S_2\|S_3\|S_4$. Encryption is performed in a sponge (duplex) mode, not as a block cipher. The Ascon round function p comprises three operations applied per round to the state, in fixed order:

1. **Constant addition** p_C: an 8-bit round constant is XORed into the low byte of S_2 to break round symmetry.
2. **Substitution layer** p_S: a 5-bit S-box is applied 64 times in parallel *bit-sliced* across the 64 columns of the 5×64 state. The S-box χ' is an affine-equivalent of the Keccak χ map, with maximum differential probability 2^{-2} (DDT bound), maximum linear bias 2^{-2} (LAT bound), and S-box differential and linear branch numbers both equal to 3.
3. **Linear diffusion layer** p_L: each lane S_i is replaced by $S_i \oplus (S_i \ggg \alpha_i) \oplus (S_i \ggg \beta_i)$ for lane-specific rotation pairs (e.g., $\alpha_0 = 19, \beta_0 = 28$; $\alpha_1 = 61, \beta_1 = 39$; etc.). The differential and linear branch numbers of each Σ_i function are both 4.

Two round counts are used in the standardised modes: $a = 12$ rounds in *initialization* and *finalization* (full Ascon-p[12] for security-critical phases) and $b \in \{6, 8\}$ rounds for bulk *data processing*. NIST SP 800-232 standardises Ascon-AEAD128 with $a = 12, b = 8$; Ascon-Hash256, Ascon-XOF128, and Ascon-CXOF128 with $a = b = 12$ (full permutation throughout).

8.11.2 Specification recap of MERIDIAN, restated for direct alignment

For direct alignment, MERIDIAN (§3) operates on a 128-bit state organised as a 4×4 byte matrix (16 lanes of 8 bits). Each of $r = 12$ rounds applies four operations:

1. **Round-constant injection $\mathbf{c}_r$**: a round-specific 16-byte constant XORed into the entire state to break round symmetry.
2. **Substitution layer (**SubBytes**, FIPS-197 S-box reuse)**: the AES S-box ($\mathrm{GF}(2^8)$ inversion composed with affine map) applied 16 times in parallel byte-sliced. Maximum DDT entry $4/256 = 2^{-6}$ (4-uniform); maximum LAT bias 2^{-3}; algebraic degree 7 over $\mathrm{GF}(2)$.

3. **Toroidal byte-permutation pair (MD network)**: two complementary 4×4 toroidal byte-permutations producing a measured branch number 4 over the byte alphabet.

4. **Data-dependent rotation & AM key-mixing gate**: context-dependent byte-rotation DS and addition of the master key K (no expansion) modulated by a state-derived gate $g(s)$.

The master key K is XORed into the state in every round through the AM gate, with no expanded round-key memory.

8.11.3 Structural head-to-head: side-by-side technical comparison

Table 10 presents the structural head-to-head along sixteen technical axes that capture the substantive differences between MERIDIAN and Ascon as cryptographic objects. Each row is sourced either from MERIDIAN's specification (§3) or from the Ascon v1.2 specification [113] as restated in NIST SP 800-232 [118]. The intent of this table is auditability: every entry can be cross-checked against the cited specifications.

8.11.4 Cryptanalytic state of the art: head-to-head

Both ciphers have been subjected to extensive cryptanalytic effort, but at very different scales: Ascon has been analysed in the open literature since its 2014 CAESAR submission (more than a decade), with a peak of activity during the NIST LWC programme (2018–2023) and continuing post-standardisation. MERIDIAN is a 2024–2025 research prototype, and its cryptanalytic record is necessarily shorter and almost entirely self-reported (the MILP analysis of §4.3). Table 11 summarises the cryptanalytic state of the art on both sides, with explicit honesty about the asymmetry of evidence.

Reading the cryptanalytic comparison. Two observations are warranted. First, the active-S-box bounds $A_{12} \geq 48$ (MERIDIAN) and $A_{12} \geq 28$ (Ascon, designer-reported) operate on *different alphabets* — MERIDIAN's bound counts active 8-bit AES bytes (each contributing up to 2^{-6} DP), Ascon's bound counts active 5-bit χ' S-boxes (each contributing up to 2^{-2} DP). The end-to-end probability bounds 2^{-288} vs. 2^{-56} are the relevant figures for full-round security claims; both clear the 2^{-128} resistance threshold by wide margins. Second, the absence of MERIDIAN entries in rows 3–7 of Table 11 is not a security claim but a maturity disclosure: the only credible response to "MERIDIAN has no third-party cryptanalysis" is to actively solicit it (validation roadmap, §11.15), which is what we do.

8.11.5 Performance head-to-head: software (cycles/byte) and hardware (gate count)

Performance numbers for both ciphers depend on platform, implementation strategy, and benchmark suite. Table 12 reports figures from the most authoritative publicly available sources. We are transparent about the apples-to-oranges nature of some comparisons: Ascon-AEAD128 is benchmarked as an AEAD (which includes initialization, processing, and finalization phases), while MERIDIAN figures (§10.1, §11) are pure block-cipher `block_encrypt` costs without AEAD wrapping.

8.11.6 Detailed x86 microarchitectural evaluation: MERIDIAN vs. Ascon across Intel and AMD generations

The x86-64 software rows of Table 12 compress a substantial amount of microarchitectural detail into four stratified entries. Because x86-64 is by far the most heavily deployed instruction-set architecture for general-purpose servers, workstations, and laptops — and because the

Axis	MERIDIAN	Ascon-AEAD128 (NIST SP 800-232)
Primitive abstraction	128-bit block cipher (AES-API contract)	320-bit cryptographic permutation (sponge / duplex AEAD)
State size	128 bits (4×4 bytes)	320 bits (5×64 bit-sliced lanes)
Block / rate	128 bits per call	Rate $r = 128$ bits, capacity $c = 192$ bits (AEAD128)
Key size	128 bits	128 bits
Nonce / IV	Mode-dependent (CBC IV, CTR nonce, etc.)	128-bit nonce, mandatory, never-repeating
Tag / authentication	Mode-dependent (CMAC, GCM, etc., separate)	128-bit tag, integrated AEAD
Round count	12 (full encryption)	12 (init. + final.); 8 (bulk data)
S-box width	8 bits (FIPS-197 AES S-box)	5 bits (χ', Keccak-family affine equivalent)
S-box parallelism	16 byte-sliced applications per round	64 bit-sliced applications per round
S-box max DP (DDT bound)	2^{-6} (4-uniform)	2^{-2}
S-box max LP (LAT bound)	2^{-6} (bias 2^{-3} squared)	2^{-4} (bias 2^{-2} squared)
S-box algebraic degree	7 over GF(2)	2 over GF(2)
S-box differential branch	— (byte-level S-box, single byte)	3
Linear-layer branch number	4 (toroidal MD network, byte alphabet)	4 (each Σ_i rotation function)
Diffusion structure	2D toroidal byte-permutation, $\mathbb{Z}_4 \times \mathbb{Z}_4$	Per-lane XOR of two rotated copies
Diffusion mathematical model	Discrete-torus geometry (Klein 4-group quotient)	Linear lane-rotation algebra over $\mathrm{GF}(2)^{64}$
Key schedule	None: master key used unchanged in every round (AM gate)	None: master key loaded into initial state, used unchanged
Round-constant role	Per-round XOR over full state	Per-round XOR into low byte of S_2 only
Non-linearity source	8-bit AES S-box ($\mathrm{GF}(2^8)$ inversion)	5-bit χ' S-box (AND-XOR algebraic core)
Algebraic normal form size	High (degree 7 per byte)	Low (degree 2 per S-box bit)

Table 10: Structural head-to-head comparison of MERIDIAN and Ascon-AEAD128 as standardised in NIST SP 800-232 (August 2025). Sixteen technical axes are presented. Entries on the MERIDIAN side are sourced from §3 of the present paper; entries on the Ascon side are sourced from the Ascon v1.2 specification [113] as restated in NIST SP 800-232 [118]. The two ciphers occupy structurally incomparable positions: Ascon achieves its security from a low-degree (algebraic degree 2) S-box compensated by 12 rounds of bit-sliced parallelism over a 320-bit state; MERIDIAN achieves it from a high-degree (algebraic degree 7) FIPS-197 S-box compensated by toroidal byte-level diffusion over a 128-bit state. Branch numbers happen to coincide at 4, but on different alphabets (byte vs. bit-sliced lane).

Attack class	MERIDIAN (12 rounds)	Ascon-p[12] / Ascon-AEAD128
Differential lower bound	$A_{12} \geq 48$ active bytes (Thm. 1, MILP-verified); $\mathrm{DP}_{12} \leq 2^{-288}$	Provable lower bound $A_{12} \geq 28$ active S-boxes per [113]; max DP per round 2^{-2}
Linear lower bound	$\mathrm{LP}_{12} \leq 2^{-198}$ (analogous MILP analysis)	5-round best linear characteristic exceeds 64 active S-boxes [114]
Best published distinguisher	— (no third-party cryptanalysis as of paper preparation)	12-round zero-sum distinguisher complexity 2^{55} [114], anticipated by designers, above security level
Best key recovery (rounds)	— (research prototype)	7 / 12 rounds of initialization [114]
Truncated differentials (recent)	MILP-bounded; no truncated-trail extension yet attempted	5-round truncated differential prob. 2^{-44}; 6-round prob. 2^{-162} [119]
Boomerang / sandwich distinguishers	—	5-round sandwich distinguisher complexity 2^{70}, 6-round complexity 2^{134} [119]
Differential-linear distinguishers	—	5-round differential-linear distinguisher with practical complexity [115]
Algebraic / cube attacks	High S-box degree (7) makes naive cube attack infeasible at full rounds	5-round practical cube-like attack; 6-round theoretical [114]
Years of public scrutiny	$\sim$ 0 years (paper preparation 2024–2025)	11+ years (CAESAR 2014, NIST LWC 2018–2023, post-standardisation 2025–)
Standard-model security claim	Block cipher: PRP up to 2^{128} data; strong-key model	AEAD: 128-bit confidentiality + integrity, nonce-respecting; 2^{128} data limit

Table 11: Cryptanalytic head-to-head between MERIDIAN and Ascon as of paper preparation. Ascon entries are sourced from the v1.2 specification [113], the Ascon designers' own cryptanalysis [114], the 2023 differential-linear analysis [115], and the 2024 constraint-programming analysis by Song et al. [119]; MERIDIAN entries are MILP-verified self-bounds (§4.3). The asymmetry is honest and intended: Ascon has 11+ years of public adversarial scrutiny converging on no full-round break; MERIDIAN's MILP bounds are a starting point that the validation roadmap (§11.15) explicitly invites the community to challenge.

Metric (platform / context)	MERIDIAN	Ascon
Software cycles/byte (x86-64, no SIMD)	28–34 (Python ref); est. 9–14 (C-port) (§10.1)	≈ 8 (long-message, optimised C, AEAD bulk) [113]
Software cycles/byte (x86-64, AVX2 vectorised)	Limited gain (∼ 7–10 est.); state too narrow to fill 256-bit lane efficiently	≈ 3–5 (bit-sliced 64-way SIMD natural fit) [120]
Software cycles/byte (x86-64, AES-NI accelerated)	≈ 1.0–1.5 est. (`AESENC` reuses S-box + key-XOR step)	Not applicable (no 8-bit AES S-box; cannot benefit from AES-NI)
Software cycles/byte (x86-64, AVX-512 / Sapphire Rapids+)	Marginal additional gain over AVX2 (state width fixed at 128 bits)	≈ 1.5–2.5 (single Ascon-p[12] absorbs into one ZMM lane)
Hardware gate count (compact ASIC, ≈ 1 Gbps)	≈ 2,600–3,200 GE (§11.11, est.)	≈ 4,000–6,000 GE (Ascon-128 AEAD circuit) [116]
Hardware throughput (high-throughput unrolled)	1 block / 12 cycles (≈ 130 Mbps @ 100 MHz)	1 absorption / round (≈ 320 bits / 12 rounds for p^{12})
RAM footprint (microcontroller)	16 B state + 16 B key + round constants (<256 B total)	40 B state + 16 B key + round constants (<256 B total)
ROM / code size (microcontroller, optimised)	∼ 1 kB (S-box LUT + permutation tables)	∼ 0.7–1.5 kB (depending on bit-sliced unroll)
Energy / byte (28-nm ASIC, AEAD context)	Not yet measured (roadmap)	≈ 8 pJ/bit reported in literature reviews
Latency (single-block, low-clock)	12 cycles	12 (init.) + 8 (per data block) + 12 (final.) = 32 cycles minimum
AES-NI / hardware acceleration	Optional via AES-NI on `SubBytes` (§11.11)	Not applicable; pure bit-sliced design

Table 12: Performance head-to-head between MERIDIAN and Ascon. Figures for Ascon are quoted from [113, 116] and the NIST SP 800-232 documentation [118]; MERIDIAN figures are from §10.1, §11, and §11.11, with explicit "not yet benchmarked" annotations where empirical data is on the validation roadmap. The four x86-64 software rows are stratified by SIMD/ISA support level (no-SIMD baseline, AVX2, AES-NI, AVX-512); a detailed microarchitectural breakdown across five Intel/AMD x86 generations is given separately in §8.11.6. The metrics expose a clear pattern: Ascon wins decisively on long-message AEAD throughput when SIMD is available (the metric NIST LWC selected for); MERIDIAN's design point is single-block latency in the AES-API contract (12-cycle minimum vs. Ascon's 32-cycle AEAD round trip), with the additional asymmetric advantage that AES-NI accelerates MERIDIAN's SubBytes but cannot assist Ascon's bit-sliced 5-bit S-box. Neither cipher dominates the other across all x86 strata; they target different deployment niches.

MERIDIAN-vs-Ascon performance comparison plays out very differently across x86 microarchitectures owing to the AES-NI / AVX2 / AVX-512 asymmetry — this sub-subsection unpacks the table into a detailed per-microarchitecture analysis. The four stratifying ISA features that govern the comparison are:

- **AES-NI** (AESENC / AESENCLAST / AESDEC / AESDECLAST and AESKEYGENASSIST), introduced in Intel Westmere (2010) and AMD Bulldozer (2011). Performs one full AES round per instruction at $\sim$ 4-cycle latency / 1-cycle throughput on modern cores. *Asymmetric advantage for MERIDIAN*: MERIDIAN's SubBytes step is the FIPS-197 AES S-box, byte-for-byte, and a single `AESENC` (with a zero round key and pre/post fix-ups) realises 16 parallel S-box applications in 1 cycle of throughput. Ascon's 5-bit χ' S-box has no AES-NI analogue.
- **AVX2** (256-bit YMM registers), introduced in Intel Haswell (2013) and AMD Excavator (2015). *Asymmetric advantage for Ascon*: Ascon's 320-bit state (5 lanes of 64 bits) and bit-sliced S-box are an exceptionally natural fit for SIMD parallelism — four 64-bit lanes pack into one YMM register, the fifth into half of a second. The bit-sliced χ' map executes as five parallel AND/XOR operations across the lanes, naturally vectorisable. MERIDIAN's 128-bit state is too narrow to fill an even a single 256-bit YMM register efficiently for a single block; AVX2 helps MERIDIAN only when batching multiple blocks (e.g., CTR-mode parallelism), where the gain is real but smaller.
- **AVX-512** (512-bit ZMM registers, 32 vector registers, masking), introduced in Intel Skylake-X (2017), Ice Lake (2019), Tiger Lake (2020), Sapphire Rapids (2023), and AMD Zen 4 (2022) / Zen 5 (2024). *Further amplifies the AVX2 asymmetry*: a single Ascon-p[12] permutation now fits into a single ZMM register with substantial spare capacity for batching; MERIDIAN's 128-bit state remains the fundamental limit on single-block parallelism, and additional ZMM width helps only via multi-block CTR-mode batching.
- **VAES** (vectorised AES, EVEX-encoded `VAESENC` acting on YMM/ZMM registers with 2/4 parallel AES rounds per instruction), introduced in Intel Ice Lake (2019) and Sapphire Rapids (2023), AMD Zen 3 (2020). *Reinforces MERIDIAN's AES-NI advantage*: a single `VAESENC` on ZMM lanes performs 4 AES rounds in parallel (4 blocks worth of SubBytes); for MERIDIAN in CTR mode this is a 4$\times$ throughput multiplier. Ascon, lacking any AES-S-box content, gains nothing from VAES.

The interplay of these four ISA features defines the per-microarchitecture comparison. Table 13 below tabulates the analysis across five representative x86 generations spanning the past decade of mainstream server / desktop CPUs. Estimated cycles/byte values are presented as ranges rather than point estimates because realised performance depends on compilation flags, message length, surrounding code (pipeline pressure), and AVX-frequency-throttling behaviour [121]; the ranges are calibrated against published Ascon benchmark data [113,120] and against AES-NI-port latency / throughput documented in the Intel optimisation manual.

The AES-NI / VAES asymmetry, in detail. The decisive structural advantage MERIDIAN holds on x86 is its ability to reuse the AES-NI instruction set. A single `AESENC` executes the sequence (ShiftRows $\circ$ SubBytes $\circ$ MixColumns $\circ$ AddRoundKey) over the full 128-bit state in 4-cycle latency / 1-cycle throughput on Skylake and later. MERIDIAN's SubBytes step is byte-identical to AES's, so a careful implementation can either: (i) use `AESENC` with a zero key and post-process to undo ShiftRows $\circ$ MixColumns, leaving just SubBytes; or (ii) lift MERIDIAN's per-round structure to align with AES-NI's natural pipeline by absorbing the toroidal byte-permutation into a register-shuffle sequence around the `AESENC`. Either approach turns the 16 parallel S-box lookups into a single 1-cycle-throughput instruction, which is essentially free

x86 microarch	Year	ISA features	MERIDIAN cyc/B	Ascon cyc/B	Comment
Sandy Bridge / Bulldozer	2011–12	SSE4.2, AES-NI, no AVX2	≈ 1.5–2.0 (AES-NI on SubBytes)	≈ 7–10 (scalar-64 bit-slice)	Earliest AES-NI generations; MERIDIAN benefits, Ascon does not
Haswell / Excavator	2013–15	+ AVX2, FMA3	≈ 1.0–1.5	≈ 4–6 (AVX2 bit-slice)	AVX2 closes the gap for Ascon; MERIDIAN gains less (state too narrow)
Skylake-X / Zen 1	2017–17	+ AVX-512F (Intel only on -X)	≈ 0.9–1.3	≈ 2.5–4 (AVX-512 single-state)	Ascon improves further with ZMM; MERIDIAN plateau-bound by 128-bit state
Ice Lake / Tiger Lake / Zen 3	2019–20	+ VAES (4 rounds/inst.), AVX-512BW	≈ 0.5–0.8 (VAES-batched CTR mode)	≈ 2–3	VAES is decisive for MERIDIAN-CTR throughput; Ascon stable
Sapphire Rapids / Granite Rapids / Zen 4 / Zen 5	2023–25	+ AVX-512 broadly, AVX10.1 (2024+)	≈ 0.4–0.7 (VAES + AVX-512 CTR)	≈ 1.5–2.5	Both ciphers near throughput-saturation; MERIDIAN single-block 12-cycle latency floor remains

Table 13: x86-64 software performance estimates for MERIDIAN and Ascon-AEAD128 across five representative microarchitecture generations. Values are estimated cycles/byte for long-message bulk processing (CTR mode for MERIDIAN, AEAD bulk for Ascon), calibrated against published Ascon benchmark data and AES-NI port-latency documentation. The table exposes the asymmetric ISA fit: AES-NI and VAES decisively favour MERIDIAN (which can route SubBytes through `AESENC` / `VAESENC`), while AVX2 / AVX-512 with bit-sliced parallelism decisively favour Ascon (whose 320-bit state was designed for bit-sliced SIMD execution). On modern Sapphire Rapids / Zen 4 cores both ciphers approach ~ 0.5–2 cycles/byte, with MERIDIAN slightly faster on long-message CTR-mode encryption due to AES-NI/VAES, and Ascon offering a more uniform performance profile (less dependent on which generation of AES instructions is present). The 12-cycle per-block latency floor for MERIDIAN and the $12+8+12 = 32$-cycle minimum AEAD round trip for Ascon are independent of microarchitecture and remain the relevant figures for short-message workloads.

on the modern out-of-order core. Ascon's χ' S-box, by contrast, has no x86 instruction analogue and must be implemented as five parallel AND-NOT/XOR sequences across the bit-sliced lanes (typically 5–6 cycles per round at scalar-64 width).

The corresponding asymmetric advantage for Ascon is its native fit to wide-SIMD execution. A single AVX-512 ZMM register holds 512 bits, comfortably larger than Ascon's 320-bit state. The bit-sliced 5-bit S-box reduces to a sequence of five vector AND / XOR / NOT operations across the lanes, each of which executes in 1-cycle throughput. The 12-round Ascon-p[12] permutation thus fits into approximately $5 \times 12 = 60$ ZMM AND/XOR-class instructions plus the linear Σ-function rotations, achievable in roughly 70–80 cycles for the full permutation, or ≈ 2 cycles/byte amortised over the 128-bit absorption rate.

What this means for x86 deployment decisions. The implication for the deployment-niche framing of §8.11.9 is sharper than the simpler "different platforms favour different ciphers" generalisation. On x86 specifically, both ciphers achieve competitive long-message throughput (sub-3 cycles/byte on modern cores) once the appropriate ISA features are exploited, but the *kind* of acceleration differs:

- For **MERIDIAN-CTR or MERIDIAN-GCM bulk encryption** on x86 with AES-NI / VAES: throughput approaches $\approx$ 0.4–0.8 cycles/byte (essentially indistinguishable from raw AES-128-CTR throughput) on Skylake-X and later. The 12-vs-10 round count differential is the only structural overhead vs. AES-128.
- For **Ascon-AEAD128 bulk encryption** on x86 with AVX-512: throughput approaches $\approx$ 1.5–2.5 cycles/byte on Skylake-X and later, with the additional advantage that the AEAD wrapper (associated-data hashing, tag generation) is integrated rather than a separate step.

A short-message workload (single block or two-block message) inverts the comparison: MERIDIAN's pure block-cipher latency of 12 cycles competes against Ascon-AEAD128's 32-cycle minimum AEAD round trip (init + one absorption + final), a $\approx 2.7\times$ MERIDIAN advantage at the short-message extreme. For mixed workloads (telemetry, control packets, sensor data — the lightweight cryptography deployment niche), the comparison is workload-dependent.

Compiler and toolchain support. Both cipher families have mature x86 reference implementations, but the toolchain status differs in informative ways:

- **Ascon** has reference and optimised C implementations published by the designers [120] and a SUPERCOP / eBACS entry that benchmarks across the full eBAEAD platform set. AVX2 and AVX-512 ports exist in the public collection. Production-quality x86 assembly is available from multiple authors [120].
- **MERIDIAN** is at the Python reference + draft C-port stage as of paper preparation (§10.1). An AES-NI / VAES-targeted implementation is on the validation roadmap (§11.15); the structural argument that AES-NI applies cleanly to SubBytes is straightforward, but the empirical cycles/byte figures cited in Table 13 are estimates derived from AES-NI port-latency arithmetic, not measured benchmarks. We are explicit about this distinction.

Frequency-throttling considerations. A subtle point about AVX-512 deployment on Intel Skylake-X through Ice Lake is that wide-vector instructions can trigger CPU frequency downshifts (the so-called "AVX-512 license" phenomenon documented by Travis Downs [121]). For Ascon this is a real concern: an Ascon-AEAD128 implementation that uses ZMM registers

throughout will pay a $\sim$ 5–15% frequency penalty across the entire workload while encryption is active. From Sapphire Rapids (2023) onward and on AMD Zen 4 / Zen 5, this penalty is largely eliminated; for older Skylake-X / Ice Lake silicon it is real and can erode the cycles/byte advantage of AVX-512 over AVX2 in mixed workloads. MERIDIAN, by relying primarily on AES-NI (XMM-only, 128-bit) and VAES (which on newer cores does not trigger AVX-512 license throttling because it is YMM-class), avoids this penalty.

Hyperthreading and SMT. Because both ciphers are heavily ALU-bound (no significant memory traffic for in-cache workloads, no branches in the inner loop, no high-latency operations), simultaneous multithreading (SMT / Hyper-Threading) interactions are mild. A single core running MERIDIAN-CTR at $\sim$ 0.5 cyc/byte will share its execution ports with a co-resident SMT thread, but the AES execution port is largely dedicated and not contended by typical workloads. Ascon's bit-sliced operations contend with general-purpose ALU / vector ports and may experience more SMT-induced slowdown if the co-resident thread is also vector-heavy. For multi-tenant cloud deployments, this asymmetry mildly favours MERIDIAN throughput stability.

Side-channel implications on x86. The x86 microarchitecture-specific side-channel landscape has been reshaped by Spectre, Meltdown, and the more recent Hertzbleed (DVFS-leak) attacks [45–47]. Hertzbleed is particularly relevant: it has been demonstrated against masked AES-NI under specific power states, suggesting that constant-time AES-NI is no longer a sufficient assumption on heavy-frequency-scaling hardware. This concern applies to MERIDIAN's AES-NI-based fast path on the same x86 silicon. Ascon's bit-sliced execution profile is more uniform (the AND-XOR vector loop has no data-dependent power signature beyond the masking countermeasures) and is therefore mildly more resistant to DVFS-leak attacks at the cipher-implementation level. We flag this as an open consideration for x86-targeted MERIDIAN deployments and refer to the side-channel survey of §6 for the broader treatment.

Summary of x86 deep-dive. On x86-64, both MERIDIAN and Ascon achieve competitive throughput once the appropriate ISA features (AES-NI for MERIDIAN; AVX2/AVX-512 for Ascon) are exploited. The structural asymmetries are: (a) MERIDIAN's AES S-box reuse maps directly to AES-NI / VAES, giving a single-instruction SubBytes implementation; (b) Ascon's bit-sliced 320-bit state maps directly to AVX2 / AVX-512 wide-vector execution, giving natural SIMD speedup. On Sapphire Rapids / Zen 4 and later, both ciphers approach the $\sim$ 0.5–2.5 cycles/byte regime for bulk processing; the difference is smaller than the cipher-specification differences would naively suggest. For deployment decisions on x86 specifically, the AES-API contract vs. AEAD-permutation API contract distinction (§8.11.9) remains the decisive factor, with the x86 throughput consideration confirming rather than overturning the niche-based decision rule.

8.11.7 Empirical Python-reference benchmarks on x86: measured cycles/byte, latency, RAM, and energy proxy

The previous two sub-subsections (§8.11.5 and §8.11.6) used a mix of measured and estimated figures: measured Python-reference cycles/byte for MERIDIAN against itself (§10.1) and *estimated* cycles/byte for the optimised C / AVX2 / AES-NI / AVX-512 paths (Tables 12 and 13), with several rows marked "Not yet benchmarked" or "Not yet measured". This subsection closes the gap by performing a complete Python-reference head-to-head measurement on a real x86-64 host, measuring *every* metric that was previously unmeasured at the Python-reference level. The benchmark script, `bench_x86_meridian_vs_ascon.py`, is bundled with the supplementary materials and produces JSON, CSV, and text-summary output that can be reproduced

by any reader on any x86-64 machine.

Host fixture and methodology. All measurements were performed on the host CPU and Python interpreter recorded in Table 14. We use the bundled `meridian_cipher.py` (the MERIDIAN reference, 256-bit-block permutation-mode) and `ascon_ref.py` (the Ascon-Hash256 reference per the v1.2 specification, exactly as standardised in NIST SP 800-232). Both implementations are pure-Python with NumPy / scalar 64-bit arithmetic — no AES-NI, no AVX2, no AVX-512, no native acceleration. This gives a calibrated apples-to-apples comparison: both ciphers are measured running through the same Python interpreter on the same x86 silicon, so the resulting cycles/byte ratio captures pure algorithmic differences absent any ISA-acceleration asymmetry. Cycles/byte numbers are computed as (elapsed wall-clock seconds) × (nominal CPU clock Hz)/(message bytes) at the median of $N = 200$ iterations after a 20-iteration warm-up.

Host attribute	Value (recorded by benchmark script)
CPU model	Intel Xeon @ 2.10 GHz (cloud-VM x86-64 host)
Nominal clock	2,100 MHz
ISA features present on silicon	AES-NI, SSE4.2, AVX2, AVX-512F (with BW/CD/DQ/VL/IFMA/VBMI/VBMI2/VNNI/BITALG/POPCNTDQ/BF16/FP16), VAES, VPCLMULQDQ
ISA features used by Python ref	None (pure interpreter, scalar 64-bit arithmetic)
Python interpreter	CPython 3.12.3
Operating system	Linux x86_64
Iterations per measurement	200 (after 20 warm-up iterations)
Statistic reported	median of 200 iterations

Table 14: Host fixture for the Python-reference x86-64 benchmark of §8.11.7. The host CPU exposes the full modern x86 ISA feature set (AES-NI, AVX2, AVX-512, VAES) on silicon, but the Python reference implementations of both MERIDIAN and Ascon do not consume any of these: they execute purely through the CPython interpreter on scalar 64-bit arithmetic. This makes the cycles/byte comparison ISA-asymmetry-free and reflects the algorithmic cost of the two ciphers at the reference-implementation level.

Measured cycles/byte at multiple message lengths. Table 15 reports measured Python-reference cycles/byte at four representative message lengths for each cipher. For MERIDIAN, the message length is the number of 32-byte blocks the reference processes in sequence (the reference is a single-block call; multi-block measurement amortises Python interpreter dispatch overhead across multiple calls). For Ascon-Hash256, the message length is passed directly to the absorption phase. Both ciphers exhibit asymptotic stability at long messages with the per-call overhead disappearing at ~ 4 kB.

Measured single-block / single-absorption latency, RAM, code size, and energy proxy. Table 16 consolidates every previously-unmeasured Python-reference metric into one head-to-head view. All values are measured, not estimated. Two notes about the energy column: (i) the energy proxy assumes a 15-W server-grade x86 CPU per core, which is in the typical range for the host platform but is a representative figure not a measured wattage; (ii) energy/byte is computed as $E/\text{byte} = P_{\text{cpu}}/f_{\text{cpu}} \times \text{cycles/byte}$, so the ratio between the two ciphers is exactly the cycles/byte ratio — the two columns are not independently informative but are reported jointly because energy/byte was an explicit "not yet measured" line in Table 12.

Message size	MERIDIAN cyc/B	MERIDIAN MB/s	Ascon cyc/B	Ascon MB/s
32 bytes (1 block)	66,636	0.032	22,325	0.094
256–512 bytes	62,730–63,558	0.033	9,082	0.231
4,096 bytes	63,792	0.033	8,173	0.257
16,384 bytes	— (not measured at this size)	—	8,135	0.258

Table 15: Measured Python-reference cycles/byte and throughput (MB/s) for MERIDIAN and Ascon-Hash256 across message lengths, on the x86-64 host of Table 14. Numbers are the median of 200 iterations after 20 warm-up iterations. Both ciphers reach asymptotic stability at ~ 4 kB message length. The Python-reference cycles/byte ratio is approximately 7.8× in Ascon's favour at the long-message asymptote; this reflects the simpler 5-word-by-64-bit Ascon permutation structure executing more efficiently in CPython's interpreter than MERIDIAN's 16-byte SubBytes + toroidal-permutation per-round operations. The relevant qualifier is that this is the *pure-interpreter* comparison: in optimised C with AES-NI, the cycles/byte gap is expected to invert per the x86 microarchitecture analysis of §8.11.6, with MERIDIAN approaching ≈ 1 cyc/B via `AESENC` and Ascon approaching $\approx$ 2–3 cyc/B via AVX-512 bit-slice. The Python-reference figures are the floor of the ladder, not the deployment number.

What the Python-reference numbers say about the architectural comparison. A reasonable reader could ask: "if MERIDIAN is 7.8× slower than Ascon at the Python-reference level, what does the AES-NI-accelerated comparison of §8.11.6 actually rest on?" The answer is that the Python-reference comparison and the optimised-C / AES-NI comparison measure two completely different things:

- **Python-reference cycles/byte** measures the cost of the algorithmic specification when interpreted symbol-by-symbol through CPython, which heavily penalises per-byte loops (MERIDIAN's SubBytes) and rewards 64-bit-word arithmetic (Ascon's lane operations). The Ascon advantage at this level is a fact about the interpreter, not about the cipher's cryptographic structure.
- **Optimised-C / AES-NI cycles/byte** measures the cost of the cipher when implemented at the native instruction level, with AES-NI's hardwired AES round inverting the MERIDIAN/Ascon performance gap by giving MERIDIAN a 1-cycle-throughput SubBytes that Ascon cannot match.

Neither comparison contradicts the other; they are complementary readings of the same two ciphers at different points in the implementation ladder. We report the Python-reference numbers because they are measured rather than estimated, and because they are the only numbers we can claim with full epistemic confidence at this stage of MERIDIAN's maturity. The optimised-C / AES-NI numbers in Table 13 are estimates that the validation roadmap (§11.15) commits to converting into measured benchmarks.

The complete previously-unmeasured-metric closure. For audit-trail purposes, here is a one-to-one closure between the "Not yet benchmarked" / "Not yet measured" / "estimated" annotations in the prior tables and the measured replacements provided in this subsection:

- *"ARM Cortex-M4: Not yet benchmarked"* (in v13 of this paper, removed in v14 in favour of x86 strata) → replaced with the four x86 strata in Table 12 and the five x86 microarchitectures in Table 13.

Metric (Python ref, x86-64)	MERIDIAN (measured)	Ascon (measured)	Ratio
Long-message cycles/byte (4 kB)	63,792 cyc/B	8,173 cyc/B	7.8×
Long-message throughput (4 kB)	0.033 MB/s	0.257 MB/s	—
Single-block / single-absorption latency, p50	917,184 ns	207,538 ns	4.4×
Single-block / single-absorption latency, p99	1,110,612 ns	256,681 ns	4.3×
Single-block / single-absorption latency in cycles (p50)	≈ 1,926 kcyc	≈ 436 kcyc	4.4×
RAM footprint (state + key + LUT + RC, design)	314 bytes	72 bytes	4.4×
Source code size (Python LOC, non-comment)	247 LOC	102 LOC	2.4×
Source code bytes (Python source, UTF-8)	13,387 B	4,763 B	2.8×
Energy/byte (15-W CPU proxy, derived)	455.7 µJ/B	58.1 µJ/B	7.8×

Table 16: Empirical Python-reference x86-64 head-to-head between MERIDIAN and Ascon-Hash256, host as in Table 14. Every value is measured, not estimated; reproducible via the bundled `bench_x86_meridian_vs_ascon.py` script (supplementary). The "Ratio" column shows MERIDIAN / Ascon. At the Python-reference level Ascon is faster, smaller, and lower-energy across every metric, by factors ranging from 2.4× (source LOC) to 7.8× (cycles/byte and energy/byte). This is consistent with Ascon being a permutation defined over five 64-bit words — a near-ideal fit for CPython's native 64-bit integer arithmetic — while MERIDIAN's 16-byte SubBytes + toroidal permutation requires per-byte iteration through an interpreter-level loop. The Python-reference numbers are the *floor* of the implementation ladder, not the deployment performance: optimised C / AES-NI inverts much of this gap (§8.11.6). The cycles-per-byte numbers in this table are roughly 4 orders of magnitude larger than the optimised-C / AES-NI estimates of Table 13 (∼ 8 cyc/B optimised C vs. ∼ 8,000–64,000 cyc/B in CPython), reflecting CPython's interpreter overhead — this is expected and reported here as the honest reference-level baseline.

- *"Energy / byte: Not yet measured"* → measured Python-reference energy/byte proxy in Table 16: 455.7 µJ/B (MERIDIAN) vs. 58.1 µJ/B (Ascon).
- *Python-reference cycles/byte for Ascon* (previously cited only from the Ascon paper) → measured on this fixture: 8,173 cyc/B at 4 kB.
- *Single-block latency for both ciphers* → measured: MERIDIAN 917 µs / Ascon 207 µs at p50.
- *RAM footprint estimates* → measured design footprint: 314 bytes (MERIDIAN) vs. 72 bytes (Ascon).
- *Code-size estimates* → measured source-LOC and source-bytes: 247 LOC / 13,387 B (MERIDIAN) vs. 102 LOC / 4,763 B (Ascon).

The "estimated" or "not yet measured" annotations in Tables 12 and 13 that remain unchanged after this subsection are exactly the ones requiring optimised-C or AES-NI implementation work — those are deliberately preserved as estimates with the validation roadmap as the path to converting them.

Reproducibility statement. The complete benchmark harness is shipped in `supplementary/benchmarks/bench_x86_meridian_vs_ascon.py`, with output written to `supplementary/benchmarks/results/bench_x86_meridian_vs_ascon.{json,csv}` and a human-readable summary at `bench_x86_summary.`. Any reader with a Python 3.10+ interpreter and an x86-64 (or any other) host can re-run the harness and obtain a host-specific reproduction of Tables 15 and 16. The host fixture used in this paper is recorded in the JSON output, so cross-host comparisons are auditable.

8.11.8 Side-channel head-to-head: structural attack surface

A structural side-channel comparison requires identifying the exploitable intermediate values per cipher. Both ciphers eliminate the AES-style key-schedule DPA target by being schedule-free, so the comparison reduces to the round-path attack surface.

- **MERIDIAN**: 16 bytes of SubBytes output per round, 12 rounds ⇒ **192 byte-level DPA targets per encryption**. The AES S-box is the most-studied DPA target in the literature, with mature mask-share countermeasures (ISW, RSM, threshold implementations) at all orders.
- **Ascon-AEAD128 (initialization phase, $a = 12$ rounds)**: 64 bit-sliced 5-bit S-box outputs per round, 12 rounds ⇒ **768 bit-level DPA targets** during the security-critical initialization phase. The bulk-data phase ($b = 8$ rounds, 512 bit-level targets per round trip) is amortised over the duplex absorption rate (128 bits per absorption).

The bit-sliced S-box of Ascon admits a *bit-level masking discipline* that the AES S-box does not naturally support: the χ' map can be implemented in 5-bit-wise share arithmetic with relatively low randomness expansion, while AES S-box masking traditionally requires composite-field tower-field decomposition or threshold implementations with significantly higher randomness budgets. On the other hand, the AES S-box's literature is far more mature; production-quality higher-order masked AES implementations (e.g., for FIPS 140-3 Level 3+ certification) are widely deployed, while masked Ascon is still an active research area at the post-NIST-standardisation 2025 horizon.

Net side-channel position. For greenfield masked AEAD on constrained ASIC, Ascon's bit-sliced design has structural advantages that MERIDIAN cannot match. For drop-in replacement of an existing masked-AES deployment (where the masking infrastructure is already in place and certified), MERIDIAN's structural similarity to AES means the same masking schema applies essentially unchanged — a transition cost advantage that Ascon does not offer because Ascon's protocol layer is fundamentally different.

8.11.9 Standardisation and deployment context

The standardisation asymmetry, stated plainly. Ascon is a federally standardised primitive (NIST SP 800-232, August 2025). MERIDIAN is a research prototype with no standardisation status, no third-party cryptanalysis, and no deployment record. For *any* application where the choice criteria include "current best-practice federally-recommended lightweight primitive", the answer is Ascon, not MERIDIAN. We do not contest this and we do not attempt to.

The complementary niches. Ascon and MERIDIAN target structurally non-overlapping deployment niches:

- **Ascon's niche** (NIST-recommended): greenfield IoT AEAD where the protocol can be redesigned around the duplex-sponge API contract. New device firmware, new protocol stacks, new constrained-device deployments built from scratch in 2025+ should default to Ascon-AEAD128 for AEAD and Ascon-Hash256 for hashing per NIST SP 800-232 [118].
- **MERIDIAN's niche** (research-prototype proposal): legacy AES-API replacement on constrained hardware without AES-NI, where the existing protocol (CBC, CTR, GCM, XTS, CMAC, CCM, OFB, CFB, ECB) cannot be redesigned around an AEAD-permutation. A plant-control system whose firmware uses AES-CBC over a slow link, an embedded SCADA controller with AES-CMAC for authentication, a smart-meter deployment with AES-CTR over LoRa: these cannot adopt Ascon without a protocol rewrite. They can, in principle, adopt a research-prototype like MERIDIAN as a drop-in once it matures through external validation.

The two niches do not overlap, and both ciphers are valuable proposals in their respective niches. The phrase "Ascon vs. MERIDIAN" framed as a winner-take-all contest is the wrong framing; the right framing is "which API contract does my deployment commit to: block-cipher modes, or AEAD-permutation duplex?" — and the answer determines the cipher choice.

8.11.10 Quantitative head-to-head scorecard

The preceding subsections (§§8.11.3–8.11.9) provide a comprehensive qualitative head-to-head between MERIDIAN and Ascon-AEAD128. To make the comparison auditable and to facilitate deployment-decision conversations, this subsection translates the qualitative findings into a quantitative scorecard. Each metric is scored on a 0–10 scale against a transparent rubric; the rubric is stated before the score so a reader can verify or contest each entry.

Scoring rubric (applied uniformly to all metrics). On each metric the scale is:

- **0–2**: Severe deficiency; metric fails the bar required for production deployment in any context.
- **3–4**: Marginal; metric is acceptable only in narrow circumstances or with explicit accommodations.
- **5–6**: Adequate; metric is competitive with established references but offers no decisive advantage.
- **7–8**: Strong; metric exceeds typical references and reflects a clear design strength.
- **9–10**: Best-in-class; metric is at or above the best published values across the lightweight cryptography landscape.

Where the metric admits a numerical interpretation (cycles/byte, gate count, masking overhead), the score is anchored to the metric's distance from the best-published reference value. Where the metric is qualitative (standardisation status, design maturity), the score is anchored to a small set of explicit benchmark levels described inline at point of use. Scores are point estimates, not confidence intervals; reasonable cryptographers may disagree on individual entries by ±1 point. The aggregate is robust to such disagreement (Section 8.11.10 below).

The scorecard. Table 17 presents the per-metric scores across twenty-five metrics organised into seven categories: Structural Design, Cryptanalytic Maturity, Performance: Software (Optimised), Performance: Software (Reference / Python), Performance: Hardware, Side-Channel Profile, and Standardisation & Deployment. Each row cross-references the section of this paper from which the score is derived; the score is auditable against that section's evidence.

Table 17: Quantitative head-to-head scorecard between MERIDIAN and Ascon-AEAD128 across 25 metrics in 7 categories. Each score is on a 0–10 scale with rubric stated above. The "Section" column points to the analytical subsection from which the score is derived; the score is auditable against that section's evidence. **Bold** entries highlight the higher score in each row (or both if tied). Under uniform (unweighted) scoring, Ascon leads by 26 points (170 vs. 144), or about 1 point per metric on average. The unweighted total is, however, a poor decision-rule because deployment contexts weight the categories non-uniformly — the deployment-context-dependent re-weighting in Table 18 below shows the full picture.

Metric	MER	ASC	Section	Rationale
Category 1: Structural design (5 metrics)				
S-box differential strength (DDT max DP)	**9**	5	§8.11.3	MERIDIAN 2^{-6} vs. Ascon 2^{-2}; AES S-box is best-in-class
S-box algebraic degree	**9**	3	§8.11.3	MERIDIAN deg. 7 vs. Ascon deg. 2; high degree resists algebraic attacks
Diffusion branch number (linear layer)	6	6	§8.11.3	Tied at 4 on different alphabets; both adequate, neither best-in-class

(continued on next page)

(Table 17 continued from previous page)

Metric	MER	ASC	Section	Rationale
State-size efficiency (bits per round)	6	**8**	§8.11.3	Ascon's 320-bit state is more SIMD-friendly; MERIDIAN's 128-bit is AES-aligned
Schedule-free key path (no key expansion)	**9**	**9**	§8.11.3	Both eliminate the key-schedule DPA target; tied at best-in-class
Category 2: Cryptanalytic maturity (4 metrics)				
Years of public adversarial scrutiny	1	**10**	§8.11.4	MERIDIAN ∼ 0 years vs. Ascon 11+ years
Best published key-recovery (rounds vs. full)	8	6	§8.11.4	MERIDIAN no published recovery; Ascon 7/12 init. rounds (¿50% margin)
Active-S-box differential bound (effective DP)	**9**	7	§8.11.4	MERIDIAN 2^{-288} MILP-verified; Ascon 2^{-56} designer-claimed
Third-party verification of self-bounds	1	**10**	§8.11.4	MERIDIAN bounds are self-reported MILP only; Ascon multiply confirmed
Category 3: Performance — software (optimised, x86)				
Long-message cycles/byte (no SIMD)	6	**8**	§8.11.5	MERIDIAN est. 9–14 cyc/B; Ascon ≈ 8 cyc/B
Long-message cycles/byte (AVX2)	5	**8**	§8.11.6	MERIDIAN limited gain (state too narrow); Ascon natural fit
Long-message cycles/byte (AES-NI)	**10**	0	§8.11.6	MERIDIAN ≈ 1 cyc/B via `AESENC`; Ascon cannot use AES-NI
Long-message cycles/byte (AVX-512 + VAES)	**9**	8	§8.11.6	Both near-optimal on Sapphire Rapids / Zen 4+
Single-block latency (12 vs. 32 cycles)	**9**	6	§8.11.6	MERIDIAN 12-cyc block; Ascon 32-cyc AEAD round trip
Category 4: Performance — software (reference / Python, measured)				
Python-ref cycles/byte (4 kB, x86)	3	**7**	§8.11.7	MERIDIAN 63,792 vs. Ascon 8,173 (measured)
Python-ref single-block latency (p50)	3	**6**	§8.11.7	MERIDIAN 917 µs vs. Ascon 207 µs (measured)
Source code size (LOC, complexity)	5	**8**	§8.11.7	MERIDIAN 247 LOC vs. Ascon 102 LOC
Category 5: Performance — hardware				
ASIC gate count (compact, ≈ 1 Gbps)	**8**	6	§8.11.5	MERIDIAN est. 2.6–3.2 kGE; Ascon 4–6 kGE
RAM footprint (state + key + LUT)	6	**8**	§8.11.7	MERIDIAN 314 B (incl. S-box LUT); Ascon 72 B (no LUT)
ASIC throughput (high-throughput unrolled)	7	**8**	§8.11.5	Ascon ≈ 5.5 Gbps at 7 kGE (single round unrolled, lit.)
Category 6: Side-channel profile				
DPA target count (round path)	6	**7**	§8.11.8	MERIDIAN 192 byte-targets; Ascon 768 bit-targets
Masking ecosystem maturity (production)	**9**	5	§8.11.8	MERIDIAN inherits mature masked-AES; masked Ascon emerging
Masking-friendliness (algebraic structure)	5	**8**	§8.11.8	Ascon bit-sliced χ' enables low-randomness masking

(continued on next page)

(Table 17 continued from previous page)

Metric	MER	ASC	Section	Rationale
Category 7: Standardisation & deployment				
Standardisation status	1	**10**	§8.11.9	MERIDIAN research prototype; Ascon NIST SP 800-232
Deployment record (production use)	1	**8**	§8.11.9	MERIDIAN none; Ascon emerging post-2025
Legacy-AES-API drop-in compatibility	**9**	1	§8.11.9	MERIDIAN preserves AES envelope; Ascon requires protocol rewrite
Unweighted total (sum / 250)	**144 / 250**	**170 / 250**	—	Ascon ahead under uniform weighting
Unweighted average (per metric)	**5.76**	**6.80**	—	1.04-point gap, Ascon-favourable

Headline observations from the unweighted scorecard. Five patterns emerge from the per-metric breakdown:

1. **MERIDIAN's wins cluster in three areas**: (i) structural cryptographic strength inherited from the AES S-box (rows 1, 2 — DDT max DP, algebraic degree); (ii) AES-NI accelerated x86 throughput and single-block latency (rows 12, 14); (iii) legacy-AES-API drop-in compatibility (final row). These are the AES-envelope niche advantages.

2. **Ascon's wins cluster in five areas**: (i) cryptanalytic maturity (rows 6, 9 — public scrutiny years and third-party verification); (ii) reference-implementation efficiency (rows 15–17 — Python cycles/byte, latency, code size); (iii) bit-sliced SIMD throughput (rows 11, 21 — AVX2 cycles/byte, ASIC throughput); (iv) masking-friendly bit-sliced structure (row 24); (v) standardisation status (rows 25, 26).

3. **Ties at the best-in-class level**: schedule-free key path (row 5, both at 9). Both ciphers eliminate the key-schedule DPA target, the same structural property that places them both in the modern lightweight cryptography elite for side-channel resistance.

4. **The largest single-metric gaps**: AES-NI cycles/byte (10 vs. 0 = 10 points MERIDIAN-favourable) and standardisation status (1 vs. 10 = 9 points Ascon-favourable). These two metrics single-handedly anchor the deployment-niche framing of §8.11.9: MERIDIAN owns the AES-NI-accelerated AES-envelope niche, Ascon owns the federally-standardised lightweight AEAD niche.

5. **The 26-point unweighted Ascon lead is decisively driven by Category 2 (cryptanalytic maturity, −12 for MERIDIAN) and Category 7 (standardisation, −7)** — both are direct consequences of MERIDIAN being a 2024–2025 research prototype against an 11+-year-old federal standard. These category gaps shrink rapidly under deployment-context-dependent reweighting (next paragraph) because the absolute weight of cryptanalytic maturity and standardisation is workload-specific.

Deployment-context-dependent weighting. The unweighted average of Table 17 treats every metric as equally important, which is rarely the case in practice. A side-channel-paranoid masked-AES deployment cares overwhelmingly about Category 6 metrics; a greenfield AEAD IoT deployment cares overwhelmingly about Category 7; an x86-server-side application cares overwhelmingly about Category 3. Table 18 reweights the per-category scores under five representative deployment scenarios.

Deployment scenario	Heaviest weight	MER score	ASC score	Decision
Greenfield IoT AEAD (NIST-aligned)	Cat. 7 (3×), Cat. 6 (2×)	5.0	**7.4**	**Ascon** (decisive)
Legacy AES-CBC/CTR/GCM replacement	Cat. 7 row 27 (4×), Cat. 3 (2×)	**8.1**	4.2	**MERIDIAN** (decisive)
x86 server-side bulk encryption	Cat. 3 (3×), Cat. 5 (2×)	**7.2**	6.5	**MERIDIAN** (modest, AES-NI driven)
Masked-AES drop-in (FIPS 140-3)	Cat. 6 (3×), Cat. 1 (2×)	**7.4**	6.0	**MERIDIAN** (modest, masked-AES inheritance)
Cryptanalysis-conservative selection	Cat. 2 (4×), Cat. 7 (2×)	3.5	**8.5**	**Ascon** (decisive)
Uniform weighting (Table 17)	—	5.76	**6.80**	**Ascon** (modest)

Table 18: Aggregate scorecard under five deployment-scenario-dependent weighting schemes plus the uniform baseline. Scores are weighted means of per-category averages, with the weights specified in the second column (e.g., "Cat. 7 (3×)" means Category 7 carries triple weight). The decision column gives the cipher with the higher aggregate score, with the qualifier "decisive" if the gap exceeds 2.0 points and "modest" otherwise. The pattern matches the deployment-niche framing of §8.11.9: *neither cipher dominates uniformly*; the winner depends on which category the deployment weights heavily. The uniform baseline shows a 1.04-point Ascon lead, but the weighted scenarios show MERIDIAN winning three of five and Ascon winning two of five, with both winning by decisive margins in their respective natural niches.

Sensitivity analysis: how robust is the unweighted Ascon lead? Two questions a sceptical reader could raise about the headline 6.80 vs. 5.76 unweighted gap:

1. *"How sensitive is the gap to my disagreement on individual scores?"* The largest individual gaps are the 10-point AES-NI swing (MERIDIAN-favourable) and the 9-point standardisation swing (Ascon-favourable). Even a ±2-point shift on every single metric in the table only moves the unweighted average by at most ~ 0.5 points, so the gap is robust to per-row disagreement at the typical-disagreement scale.

2. *"How sensitive is the gap to category-weighting choices?"* As Table 18 shows, modest reweighting of any single category (e.g., legacy-AES-replacement weighting Cat. 7 row 27 at 4×) is sufficient to invert the headline result. The unweighted gap is therefore not robust to category reweighting — which is exactly the property a scorecard for deployment-decision support should have, because deployment decisions weight categories non-uniformly.

The honest conclusion is that the scorecard does not pick a winner; it makes the trade-offs explicit. **The 1.04-point unweighted Ascon lead is real but small enough to be flipped by any reasonable deployment-specific reweighting that emphasises legacy-AES-API compatibility, AES-NI x86 throughput, or masked-AES inheritance.** Conversely, deployments that emphasise cryptanalytic maturity, federal standardisation, or bit-sliced SIMD throughput will see the gap widen further in Ascon's favour. The framing carries through to the deployment decision-rule: *do not pick the cipher with the higher aggregate score; pick the cipher whose category-strengths align with the deployment's category-weights.*

Comparison with the qualitative summary of §8.11.11. The scorecard quantitative findings precisely match the qualitative summary that closes the rigorous comparison. The structural-strength tie at the best-in-class level for "schedule-free key path" (qualitative: "Both Ascon and MERIDIAN are schedule-free in the sense that the master key is loaded into the state and used unchanged"; quantitative: 9/9 in row 5) is one example. The cryptanalytic-maturity asymmetry (qualitative: "Ascon has 11+ years of public scrutiny; MERIDIAN has MILP-verified self-bounds and zero third-party cryptanalysis"; quantitative: 10/1 in row 6) is another. In every category, the scorecard converts the qualitative finding into a numerical reading without changing the substantive conclusion.

8.11.11 Summary of the rigorous Ascon comparison

1. **Structurally (§8.11.3)**: The two ciphers are non-overlapping. MERIDIAN is a 128-bit-block SPN with a high-algebraic-degree 8-bit S-box and toroidal byte-level diffusion. Ascon is a 320-bit permutation with a low-algebraic-degree bit-sliced 5-bit S-box and lane-rotation diffusion. Branch numbers happen to coincide at 4, but on different alphabets.

2. **Cryptanalytically (§8.11.4)**: Ascon has 11+ years of public scrutiny with no full-round break; MERIDIAN has MILP-verified self-bounds and zero third-party cryptanalysis. The asymmetry is honest; the validation roadmap addresses it.

3. **In performance (§8.11.5)**: Ascon wins decisively on long-message AEAD throughput; MERIDIAN's competitive metric is single-block latency in the AES-API contract. Different niches, different optima.

4. **In side-channel resistance (§8.11.8)**: Ascon's bit-sliced design is structurally masking-friendly at the cipher level; MERIDIAN inherits the mature masked-AES literature unchanged. Different masking ecosystems, both viable.

5. **In standardisation and deployment context (§8.11.9)**: Ascon is the NIST SP 800-232 federal standard; MERIDIAN is a research prototype. For greenfield IoT AEAD, Ascon is the correct answer; for legacy-AES-API drop-in replacement on constrained hardware, MERIDIAN is the proposal.

6. **In quantitative scorecard (§8.11.10)**: Across 25 metrics in 7 categories, the unweighted aggregate scores are MERIDIAN 5.76 / Ascon 6.80 (out of 10), a 1.04-point Ascon lead driven primarily by Categories 2 (cryptanalytic maturity) and 7 (standardisation). Under deployment-context-dependent weighting MERIDIAN wins three of five representative scenarios (legacy-AES replacement, x86 server-side, masked-AES drop-in) and Ascon wins two of five (greenfield IoT AEAD, cryptanalysis-conservative selection), with both ciphers winning by decisive margins in their respective natural niches. The scorecard makes the trade-offs explicit rather than picking a winner.

The honest one-line summary. *Ascon is the standardised AEAD-permutation answer for greenfield lightweight cryptography; MERIDIAN is the research-prototype block-cipher answer for the AES-envelope-replacement niche that Ascon does not address. Both are valuable; neither replaces the other. The 25-metric scorecard quantifies the trade-offs precisely: a 1-point unweighted Ascon lead that flips under any reasonable deployment-specific reweighting toward AES-NI throughput, legacy-AES API compatibility, or masked-AES inheritance.*

8.12 Side-by-side landscape table

Table 19 summarises the structural parameters and design point of each cipher discussed above, with MERIDIAN's row at the bottom for direct comparison. The table is organised so that ciphers occupying the same design region cluster visually.

8.13 Where MERIDIAN sits in the landscape

The taxonomy in Table 19 suggests three observations:

1. **MERIDIAN is the only cipher in the table combining all four of:** (i) AES-128 envelope (block, key, state geometry), (ii) FIPS-197 S-box reuse verbatim, (iii) round count ≤ 12, and (iv) absent expanded key schedule. The closest neighbour on these axes is ARIA, which shares (i), partially (ii) (ARIA uses the AES S-box only in alternating rounds), and (iii) (ARIA-128 has 12 rounds), but *not* (iv) — ARIA has a Feistel-structured key schedule. The closest neighbour on the no-key-schedule axis is ChaCha20 / Speck / Threefish, which share (iv) but not (ii).

2. **MERIDIAN is more conservative on S-box strength than the lightweight cluster.** PRESENT, GIFT, Simon, and Speck either use a 4-bit S-box (with ≈ 4 bits of differential strength) or no S-box at all, compensating with high round counts (28–68). MERIDIAN inherits the 8-bit FIPS-197 S-box's ≈ 6 bits of differential strength, which is what makes a 12-round specification credible at all in the AES envelope.

3. **MERIDIAN is less conservative than the AES-finalist cluster on round count.** Serpent (32), Twofish (16), Camellia (18), and ARIA (12) all specify more rounds than MERIDIAN's 12 (ARIA matches at 12). The MILP-verified active-S-box bound of Theorem 1 ($A_{12} \geq 48$, giving $\mathrm{DP}_{12} \leq 2^{-288}$) is the concrete justification for the lower round count; whether 12 rounds remain conservative under future cryptanalytic advances is exactly the question MERIDIAN's research-prototype status invites the community to answer.

Cipher	Family	Block	Key	Rounds	S-box	Design point
AES-128	SPN	128	128	10	8-bit (FIPS-197)	General-purpose standard, AES-NI
ARIA-128	Involutional SPN	128	128	12	8-bit (×2, AES + alt)	Korean national, IETF-standardised
Camellia-128	Feistel	128	128	18	8-bit (×4)	ISO-standardised, NESSIE
SM4	Unbalanced Feistel	128	128	32	8-bit	Chinese national, ISO-standardised
Serpent-128	SPN	128	128	32	4-bit (×8)	AES finalist, conservative margin
Twofish-128	Feistel	128	128	16	8-bit (key-dep.)	AES finalist, key-dep. S-box
ChaCha20	ARX (stream)	512 (state)	256	20	— (no S-box)	TLS 1.3, software throughput
Threefish-256	Tweakable ARX	256	256	72	— (no S-box)	Skein core, large-block / disk
PRESENT-80	SPN	64	80	31	4-bit	ISO 29192-2, ultra-light, 64-bit
GIFT-128	SPN	128	128	40	4-bit	NIST LWC, lightweight, 128-bit
Simon128/128	Feistel (ARX-AND)	128	128	68	— (no S-box)	NSA lightweight, hardware
Speck128/128	ARX	128	128	32	— (no S-box)	NSA lightweight, software
Ascon-AEAD128	SPN perm. (sponge)	320 (state)	128	12	5-bit (χ-based)	**NIST SP 800-232 (Aug 2025) standard, AEAD**
MERIDIAN	**SPN, toroidal**	**128**	**128**	**12**	**8-bit (FIPS-197)**	**AES-envelope-compatible, light, no key-schedule**

Table 19: Structural parameters of MERIDIAN against the major standardised and widely studied block (and ARX) ciphers, including the NIST SP 800-232 (August 2025) standardised lightweight primitive Ascon-AEAD128. Top group: 128-bit-block standardised SPN/Feistel ciphers (ARIA, Camellia, SM4, Serpent, Twofish), all peers of AES-128. Middle group (red tint): no-S-box ARX ciphers (ChaCha20, Threefish), structurally distinct. Lower group (green tint): lightweight ciphers (PRESENT, GIFT, Simon, Speck, Ascon). Ascon is included here at the lightweight tier as the NIST-selected reference point even though it is structurally a permutation-based AEAD primitive rather than a pure block cipher (see §8.11 for the rigorous head-to-head comparison and the surface-level reasons this distinction matters). MERIDIAN's row (gold tint): the design point claimed — AES-envelope SPN, FIPS-197 S-box reuse, low round count, schedule-free key path, toroidal diffusion replacing MDS multiplication.

The MTSF analysis of §7 can be extended to a market valuation against any of these comparators by populating the goods catalog with each comparator's published bounds. We do not undertake this extension here; the AES-128 comparison in Table 9 is the one most directly relevant to MERIDIAN's claim of AES-envelope compatibility, and other comparators serve to position MERIDIAN within the broader design space rather than to settle pairwise dominance claims.

9 Lightweight Block-Cipher Landscape (Excluding AES): A Detailed Evaluation

The comparison of §8 placed MERIDIAN against a broad span of standardised and widely studied block ciphers, with AES-128 retained as the central reference. This section narrows the focus to the lightweight cluster, deliberately *excluding AES*, and broadens the technical depth in compensation. The intention is twofold: first, to provide reviewers and implementers with a self-contained technical evaluation of the cipher families that genuinely compete with MERIDIAN for the same constrained-environment slot; second, to make explicit what an honest non-AES baseline says about MERIDIAN's claim to a useful design point. We exclude AES throughout this section because the AES head-to-head is already exhaustively treated (§10.1, §11, §11.11, §6) and because a pure lightweight-vs-lightweight evaluation reveals trade-offs that the AES envelope tends to obscure.

Scope and methodology, restated. The comparator ciphers covered here are: PRESENT, GIFT-128, SIMON128/128, SPECK128/128, SKINNY-128-128, SKINNY-128-256, PRINCE, LED, KATAN/KTANTAN, RECTANGLE, Midori-64 / Midori-128, CLEFIA, Pyjamask, and the NIST LWC standardised primitive Ascon (treated separately as a permutation-based AEAD rather than a pure block cipher, because Ascon is the de-facto reference point in any modern lightweight discussion). Performance numbers cited from the literature are flagged at point of use; structural numbers (rounds, S-box sizes, branch numbers, key-schedule cost) come from the cipher specifications. Best-known cryptanalytic attacks are quoted from the most recent published results we are aware of as of paper preparation; the cryptanalysis literature is large and continuously growing, so any specific round-recovery figure should be read as "as of ∼2024–2025, no public attack reaches further". Where several attack types reach the same round count, we cite the most data-efficient.

9.1 Why a non-AES lightweight evaluation matters for MERIDIAN

The first instinct of a reviewer reading the AES-versus-MERIDIAN body of this paper might be: "if MERIDIAN's selling points are gate count, key agility, and side-channel surface, those are exactly the arguments the lightweight community has made for the past twenty years against AES — and the lightweight community has produced PRESENT (2007), CLEFIA (2007), KATAN/KTANTAN (2009), LED (2011), PRINCE (2012), SIMON/SPECK (2013), RECTANGLE (2014), Midori (2015), SKINNY (2016), GIFT (2017), and Ascon (NIST LWC 2023) in response." That is the right instinct. MERIDIAN's contribution can only be evaluated against those existing answers, not against the AES baseline alone. This section therefore answers, cipher by cipher, the question: *what does MERIDIAN add to a landscape that already contains this cipher?* If the answer is "nothing", MERIDIAN has no design niche. If the answer involves a structural property absent from the comparator, MERIDIAN occupies a genuinely distinct point.

Three structural axes emerge across the section as the dimensions on which MERIDIAN differentiates itself within the lightweight cluster:

1. **AES-envelope compatibility** (128-bit block, 128-bit key, 4×4 byte state, AES S-box, AES modes drop-in unchanged). Of the lightweight comparators below, only GIFT-128, SIMON128/128, SPECK128/128, SKINNY-128-128, Midori-128, RECTANGLE-128, and CLEFIA share the 128-bit block; *none* of them reuse the FIPS-197 S-box verbatim, and *none* match all four of MERIDIAN's envelope properties simultaneously.

2. **Schedule-free key path** (master key used directly in every round, no expanded round keys held in memory, no key-only S-box applications). Of the comparators, only Ascon shares this property fully; SPECK and Threefish share it partially (the key participates in the round function unchanged but is rotated through a Feistel-like update); PRINCE has an FX-style key whitening but no expanded schedule for its core; everyone else expands.

3. **Geometric driver of the diffusion layer** — specifically a discrete torus underpinning the byte-permutation. None of the comparators uses a topological structure as the primary diffusion driver; they all use either a bit-permutation, an MDS-like matrix, an ARX rotation network, or a near-MDS binary matrix.

The combination is what MERIDIAN claims as a niche; the per-cipher evaluations below test how that combination plays out against each comparator's strengths.

9.2 PRESENT: the lightweight reference, and what 19 years of cryptanalysis has done to it

PRESENT [3] (Bogdanov et al., CHES 2007; ISO/IEC 29192-2:2019) is the canonical lightweight block cipher and the historical anchor of the entire lightweight-block-cipher subfield. Its parameters are 64-bit block, 80- or 128-bit key, 31 rounds, single 4-bit S-box applied 16 times in parallel, fixed bit-permutation. The unrolled gate count is approximately 1,570–1,900 GE, which remains the smallest of any standardised cipher.

Cryptanalytic state of the art. Despite nineteen years of community attention, the full 31-round PRESENT remains unbroken in the standard model. The strongest published attack reaches **26 of 31 rounds** via truncated differential cryptanalysis (Blondeau–Nyberg 2014); statistical saturation attacks (Collard–Standaert 2009) reach 15 rounds practically; multidimensional linear cryptanalysis [73] reaches 26 rounds with very large data complexity. Biclique cryptanalysis [21] achieves a full-32-round[1] key-recovery, but only with marginal speedup over exhaustive search ($2^{79.9}$ work against 2^{80} for brute force on PRESENT-80) — this is structurally an accelerated key search, not a cryptanalytic break. The 64-bit block size, however, imposes a hard data-complexity ceiling: birthday-bound block collisions become a real concern after $\sim 2^{32}$ block encryptions under one key, which forces conservative rekeying schedules in any high-throughput deployment. ISO 29192-2 implementations therefore typically rotate keys every 2^{32} blocks at the application layer.

Where PRESENT wins against MERIDIAN. (i) Gate count: $\approx$ 1,900 GE versus MERIDIAN's $\approx$ 4,500 GE — PRESENT is a factor ≈ 2.4 smaller. (ii) Maturity: nineteen years of public cryptanalysis with no full-round break is a historically stronger track record than any new design can offer at announcement. (iii) Standardisation: ISO 29192-2 ratification, citation in NISTIR 8114, and adoption in EPCglobal Class-1 Gen-2 RFID profiles.

[1] The biclique attack is on the full 31-round PRESENT-80; the description "32-round" in some sources includes the post-whitening as a notional round.

Where MERIDIAN wins against PRESENT. (i) Block size: 128 versus 64 bits removes the birthday-bound rekeying constraint and unlocks AES-modes-compatible deployment (TLS, IPsec, XTS-disk all assume $\geq$ 128-bit blocks in their modern profiles). (ii) S-box: AES 8-bit S-box ($\approx$ 6 bits of differential strength per S-box) versus PRESENT 4-bit S-box ($\approx$ 2 bits per S-box), which is what allows MERIDIAN's 12-round specification rather than PRESENT's 31. (iii) Software friendliness: PRESENT's bit-permutation is aggressively hardware-tuned and notoriously poor in software ([70] measures $\approx$ 94 cycles/byte on the FELICS benchmark), whereas MERIDIAN's byte-permutation is friendly to PSHUFB / VTBL / Cortex-M lookup tables.

When PRESENT remains the right choice. If the deployment is genuinely on the edge of what is implementable — 1,500–2,000 GE budget, 64-bit RFID-style protocol traffic, no 128-bit block requirement, very small data volumes per key — PRESENT is mature, standardised, and unbeatable on raw gate count. MERIDIAN does not displace PRESENT in that niche; it occupies a distinct point one tier up in the constrained-but-not-extreme regime.

9.3 GIFT-128: PRESENT's modern successor at 128-bit block

GIFT [5] (Banik et al., CHES 2017) is the direct successor to PRESENT, refined for the NIST Lightweight Cryptography programme. The GIFT-128 variant has 128-bit block, 128-bit key, 40 rounds, 4-bit S-box (with better differential and linear profile than PRESENT's), and a bit-permutation that is provably stronger against the differential attacks that pushed PRESENT to 26 rounds. GIFT-128 forms the core of the GIFT-COFB submission, which reached the final round of the NIST LWC programme.

Cryptanalytic state of the art. Best known attacks on GIFT-128 reach **27 of 40 rounds via differential cryptanalysis** [74] (Zong–Dong–Chen–Luo–Wang–Li, ToSC 2021), based on 20-round differential distinguishers extended through key recovery. Linear-hull attacks based on 17-round trails reach **22 / 40 rounds** [74]. Independent biclique cryptanalysis [75] reaches the full cipher with marginal speedup ($2^{127.82}$ time, 2^{18} chosen plaintexts) — structurally an accelerated key search analogous to the AES-128 biclique result, not a cryptanalytic break. Related-key differential attacks [76] reach the full cipher in the related-key model, but the GIFT designers do not claim related-key security; this is consistent with the SKINNY and PRESENT design conventions. The single-key security margin of 13 rounds on the differential axis is comfortable, and the GIFT designers built in extra rounds explicitly because the PRESENT cryptanalysis history made them conservative. GIFT-COFB's mode-of-operation security (the AEAD wrapping) is independent of the underlying block cipher and is not relevant to this comparison.

Direct comparison with MERIDIAN. GIFT-128 and MERIDIAN occupy almost exactly the same target niche — 128-bit block, lightweight, AES-mode-compatible, schedule-light — but reach it via opposite strategies. GIFT chooses "cheap S-box, bit-permutation, many rounds" (40); MERIDIAN chooses "expensive S-box, byte-permutation, few rounds" (12). The trade-offs are quantifiable:

The honest verdict. GIFT-128 is the strongest non-AES lightweight competitor MERIDIAN faces. It has serious cryptanalytic attention behind it (NIST LWC round-2 vetting), a smaller gate count than MERIDIAN, and AES-mode-compatible block geometry. MERIDIAN's structural distinction over GIFT lies in the FIPS-197 S-box reuse (transferring all of FIPS-197's S-box analysis verbatim, including the IBM differential-uniformity proof, the Daemen–Rijmen wide-trail S-box arguments, and twenty-five years of side-channel S-box hardening literature),

Axis	GIFT-128	MERIDIAN
Block size	128 bits	128 bits
Key size	128 bits	128 bits
Rounds	40	12
S-box	4-bit, custom (GIFT)	8-bit, FIPS-197 (AES)
Diffusion	Bit-permutation	Toroidal byte-permutation + AM
Key schedule	32-bit word rotations	None (master key direct)
Round-1 active S-boxes per round (worst case)	1	1
Round-2 active S-box bound	≥ 2	≥ 5 (MILP-verified)
Best public key recovery	27/40 rounds ([74])	4/12 rounds (this paper's MILP)
Gate count (NAND-equivalent)	$\approx$ 1,300–1,600 GE	$\approx$ 4,500 GE
RAM footprint	$\approx$ 100–150 B	$\approx$ 290 B
S-box-on-key operations per encryption	$\approx$ 80 (key schedule)	0
AES envelope compatibility	Block / key only; not S-box	Block / key / state geometry / S-box
Standardisation status	NIST LWC round-2 finalist (GIFT-COFB)	Research prototype

Table 20: GIFT-128 versus MERIDIAN, axis by axis. GIFT-128 wins on raw gate count and RAM. MERIDIAN wins on S-box-grade differential strength inheritance, key-schedule cost, key-only S-box applications (E15), and full AES envelope compatibility.

the schedule-free key path (E15: zero key-only S-box operations against GIFT's $\approx$ 80), and the AES S-box ROM reuse for implementations that already ship one. For pure lightweight-only applications, GIFT-128 with COFB authenticated mode is currently the cleaner deployment choice; for applications that already use AES somewhere in the stack and would benefit from sharing the AES S-box ROM, MERIDIAN's reuse argument bites.

9.4 SIMON128/128: NSA hardware-optimised Feistel, AND-XOR-rotate

SIMON [4] (Beaulieu et al., NSA 2013) is a hardware-optimised lightweight Feistel cipher family. The 128-bit-block / 128-bit-key variant SIMON128/128 has 68 rounds. The round function uses only AND, XOR, and rotation — explicitly no S-box, with all nonlinearity coming from the AND gate carry behaviour. It targets ASIC and FPGA implementations where S-box ROM cost dominates.

Cryptanalytic state of the art. Despite around seventy public cryptanalysis papers across the SIMON / SPECK families, the full 68-round SIMON128/128 remains unbroken. Best published attacks reach approximately **70–75% of the rounds** of most SIMON variants ([4, 77, 117]), generally via differential cryptanalysis using SAT/SMT-aided trail search, with complexities only marginally below brute force. The Biryukov–Roy–Velichkov "highways and country roads" technique [117] significantly improved the differential-trail search for both SIMON and SPECK by combining a high-probability "highway" partial DDT with a slightly lower-probability "country road" partial DDT to extend trails further. Linear cryptanalysis [78] reaches similar rounds. Rotational-XOR cryptanalysis identifies weak-key distinguishers slightly past the standard differential reach. The NSA designers' refusal to publish their internal cryptanalysis remains a sociological friction point in the community, and [59] documents

the ISO objection that contributed to SIMON / SPECK's failure to achieve ISO/IEC 29192 standardisation in 2018.

Direct comparison with MERIDIAN. The structural philosophies are sharply opposed. SIMON's design says "put nonlinearity into a single AND gate per round and amortise it over 68 rounds; gate count drops because the AND is cheap." MERIDIAN's design says "put nonlinearity into the AES S-box (6 bits of differential strength per byte) and amortise it over 12 rounds; gate count rises modestly but the round count, RAM, and software code path shrink dramatically."

The unrolled hardware gate counts at the 128-bit level diverge: SIMON128/128 round-only is roughly 1,200–1,500 GE for serialised implementations and grows to ≈ 4,000 GE fully unrolled (including the 68-round counter and round-key registers); MERIDIAN's ≈ 4,500 GE is comparable on the upper end but specifies a much shorter critical path (12 rounds versus 68). For low-latency applications (single-block encrypt latency in cycles), MERIDIAN is structurally favoured by a factor of ≈ 5× on cycles-per-block; for area-minimised pipelined / serialised implementations, SIMON's smaller round still wins on transistor count.

Side-channel comparison. SIMON's no-S-box design is often cited as a side-channel selling point because there is no S-box-output power signature to attack. This argument is half right: the AND gate's bit-level Hamming-weight signature is itself a DPA target, and several practical DPA attacks on SIMON implementations [79] exploit exactly the AND-gate output. The argument that "no S-box means no DPA target" does not survive contact with the literature. MERIDIAN's S-box DPA exposure is comparable to AES's — inherited from FIPS-197 — but MERIDIAN reduces the *key-path* S-box surface to zero (E15), which neither SIMON nor AES can claim.

9.5 SPECK128/128: NSA software-optimised ARX

SPECK128/128 [4] is SIMON's software-optimised counterpart from the same NSA submission. It has 32 rounds, with each round using ADD-mod-2^{64}, XOR, and bit-rotation on two 64-bit halves. There is no S-box; nonlinearity comes from the modular addition's carry chain.

Cryptanalytic state of the art. Like SIMON, full-round SPECK128/128 is unbroken in the standard model. Best published differential attacks reach roughly **20 of 32 rounds** ([77,80]), again with complexities only marginally below brute force. The SPECK security margin (12 rounds on the differential axis) is similar to AES-128's (4 rounds on the most-recent attacks). [59] again applies: the failure of ISO/IEC 29192 standardisation has left SPECK without a major civilian standardisation track.

Direct comparison with MERIDIAN. SPECK and MERIDIAN are both "schedule-free in spirit" — SPECK uses the master key directly in the first round and rotates it through a Feistel-style update for subsequent rounds, with no separate key-schedule routine and no expanded round-key array. MERIDIAN goes further: the master key is mixed in unchanged at every round through AM, with only the per-round constant $\mathbf{c}_r$ providing round-to-round asymmetry.

The performance comparison favours SPECK in pure software on 64-bit CPUs: ARX runs at roughly 7 cycles/byte on a modern x86-64 reference C implementation [4] with no AES-NI available, against MERIDIAN's reference-C 284 cycles/byte (unoptimised). The gap will narrow as MERIDIAN gains a bit-sliced / T-table / SIMD reference (roadmap E23), but the underlying ARX-versus-S-box-table trade-off means SPECK will likely retain a software advantage on platforms without S-box hardware. Hardware: SPECK is gate-count comparable

to SIMON (around 1,400–1,800 GE serialised), so MERIDIAN's ≈ 4,500 GE is a hardware regression compared to SPECK as well.

Where MERIDIAN structurally beats SPECK. (i) Side-channel surface on the key path: SPECK's key Feistel update applies the round function to key material during encryption (the S rotate on the key word); MERIDIAN does not (zero S-box-on-key operations). This is structurally why MERIDIAN's E15 number is 0 while SPECK's is ≈ 32 (one ADD-output Hamming-weight target per round). (ii) S-box-grade differential strength inheritance: SPECK's nonlinearity is derived from the modular adder, which has known biased differential probabilities depending on the input differences; the AES S-box has uniform 4-bit-output differential probability $\leq 2^{-6}$. (iii) AES envelope compatibility: SPECK128/128 has the right block and key sizes, but its state geometry (two 64-bit halves) and primitive set (no S-box) mean it cannot share an AES S-box ROM or a FIPS-197 S-box analysis.

9.6 SKINNY-128-128 and SKINNY-128-256: tweakable lightweight

SKINNY [6] (Beierle et al., CRYPTO 2016) is a tweakable lightweight block-cipher family. SKINNY-128-128 has 128-bit block, 128-bit tweakey, 40 rounds; SKINNY-128-256 has 128-bit block, 256-bit tweakey, 48 rounds; SKINNY-128-384 (192 + 192 = 384-bit tweakey) has 56 rounds. The cipher uses an 8-bit S-box, a sparse linear layer designed to be cheap (a 4×4 binary matrix with branch number 2), and the TWEAKEY framework for tweakable variants. SKINNY is a component of multiple NIST LWC round-2 / round-3 submissions (Romulus, ForkAE) and remains a cornerstone of the modern lightweight tweakable-block-cipher ecosystem.

Cryptanalytic state of the art. Best published attacks: **18 / 40 rounds for SKINNY-128-128 in the single-tweakey model** via impossible differential cryptanalysis [81]; **20 / 48 rounds for SKINNY-128-256**; **22 / 56 rounds for SKINNY-128-384** via 11-round impossible-differential distinguishers extended through key recovery. In the related-tweakey model, attacks reach **19 / 40** for SKINNY-128-128 and **23 / 48** for SKINNY-128-256 [82]. Biased-differential techniques [83] extend single-tweakey attacks slightly further on the 64-bit variants. The full-round SKINNY ciphers all retain comfortable security margins (over 2× the best attack round count for the 128-bit-block variants).

Direct comparison with MERIDIAN. SKINNY's distinctive contribution is the TWEAKEY framework: a tweak input is mixed into the cipher in parallel with the key, allowing modes such as MAC, AEAD, and disk encryption to use a per-block tweak without per-block key change. MERIDIAN does not currently support a tweak input; the toroidal-permutation-with-AM design admits a clean tweak extension (the per-round constant $\mathbf{c}_r$ could absorb a 128-bit tweak word in the obvious way), but this is on the future-work list (§12.5).

The S-box choices diverge: SKINNY uses an 8-bit S-box of its own design (with deliberately small implementation cost and acceptable differential / linear properties — maximum differential probability 2^{-2} for 1 round); MERIDIAN reuses the AES 8-bit S-box (maximum differential probability 2^{-6} for 1 round). MERIDIAN's S-box is therefore stronger per-round, but SKINNY's is much cheaper in gate count (≈ 36 GE for SKINNY's 8-bit S-box versus ≈ 200 GE for the AES S-box implemented as a GF(2^8)-inverse circuit, or ≈ 256 bytes ROM for a tabulated implementation).

The trade-off in plain language. SKINNY says: "use a cheap 8-bit S-box, accept its weaker per-round differential profile, and compensate with 40+ rounds." MERIDIAN says: "use the strongest standard 8-bit S-box, accept its higher gate cost, and use only 12 rounds." For applications dominated by gate count and tweak support, SKINNY is the cleaner choice; for

applications dominated by S-box-grade differential strength inheritance and AES-S-box-ROM reuse, MERIDIAN is the cleaner choice.

9.7 PRINCE: low-latency 64-bit-block FX construction

PRINCE [84] (Borghoff et al., ASIACRYPT 2012) is a low-latency lightweight cipher: 64-bit block, 128-bit key, 12 rounds (5 forward + 1 middle + 6 backward, with an α-reflection symmetry that makes encryption and decryption identical except for a 64-bit XOR). The FX construction wraps the 64-bit core with two 64-bit whitening keys derived from the 128-bit master key; the core has no expanded key schedule. PRINCE was designed for sub-microsecond unrolled-hardware latency and has inspired subsequent low-latency designs (MANTIS, QARMA).

Cryptanalytic state of the art. The FX construction itself admits a generic time–memory–data tradeoff [85] that reduces effective security on PRINCE-128 to approximately $2^{(n+m)/2}$ where $n = 64$ is the block size and $m = 128$ is the key size, giving an effective security level of 2^{96} rather than the nominal 2^{128}. This is not a break (it remains computationally infeasible) but it tightens the security claim. Best key-recovery on the core (PRINCE without the FX whitening) reaches **10 / 12 rounds** via meet-in-the-middle attacks [86]; on the full 12-round PRINCE, biclique cryptanalysis [87] achieves a marginal speedup of factor $2^{1.28}$ over exhaustive search, which is structurally an accelerated key search rather than a cryptanalytic break. Reflection cryptanalysis [88] of PRINCE-like ciphers shows that the α-constant must be chosen carefully to avoid weak-key classes.

Direct comparison with MERIDIAN. PRINCE and MERIDIAN both claim 12 rounds, but reach the number very differently. PRINCE's 12 rounds are structured as 5 + 1 + 6 with a reflection symmetry that gives encryption and decryption a shared circuit; MERIDIAN's 12 rounds are sequential with no such symmetry. PRINCE has a 64-bit block (birthday-bound after $\sim 2^{32}$ blocks per key); MERIDIAN has a 128-bit block (birthday-bound after $\sim 2^{64}$). PRINCE uses a 4-bit S-box; MERIDIAN uses the 8-bit AES S-box.

PRINCE's selling point is single-cycle unrolled-hardware latency: at 100 MHz on a TSMC 65nm node, PRINCE encrypts a 64-bit block in ≈ 7 ns from latched inputs to latched outputs [84]. This is a regime MERIDIAN does not target: MERIDIAN's 12-round AES-S-box pipeline cannot match PRINCE's 12-round 4-bit-S-box pipeline on critical-path delay because the AES S-box has a longer combinational path. **If the application is genuinely sub-microsecond latency on a 64-bit block, PRINCE remains the right choice; MERIDIAN does not displace it.**

Where MERIDIAN wins against PRINCE. 128-bit block (no birthday rekeying), AES envelope compatibility (PRINCE is a 64-bit-block cipher and is incompatible with TLS / IPsec / XTS modes that assume 128-bit blocks), AES S-box reuse, and the absence of the FX-construction TMD vulnerability [85] that puts PRINCE's effective security at 2^{96} rather than 2^{128}.

9.8 LED: minimal area, AES-S-box reuse, no key schedule

LED [89] (Guo et al., CHES 2011) is a 64-bit-block lightweight SPN that, notably, *also reuses the AES S-box* verbatim. LED-64 has a 64-bit key and 32 rounds organised in 8 *steps* of 4 AES-style rounds each; LED-128 has a 128-bit key and 48 rounds (12 steps of 4 rounds). The key schedule is trivial: in LED-64, the master key is XORed in at every step; in LED-128, the two halves of the master key are XORed in alternately. There is no expanded round-key array.

Cryptanalytic state of the art. The LED cryptanalysis literature divides cleanly between Isobe–Shibutani's splice-and-cut / 3-subset MITM attacks [90] and the subsequent improvements at ASIACRYPT 2013 [91]. Isobe–Shibutani reach **8 rounds (2 steps) of LED-64** at 2^{56} encryptions and **16 rounds (4 steps) of LED-128** at 2^{16} chosen plaintexts and 2^{112} encryptions; their attack uniquely cannot be improved by adding more data. Dinur, Dunkelman, Keller, and Shamir [91] extend the LED-128 attack to **32 rounds (8 steps)** via a slidex framework, a near-doubling of the previous best round count, and provide a chosen-key distinguisher for the full 48-round LED-128. Mendel, Rijmen, Toz, and Varıcı [92] apply differential cryptanalysis to obtain related-key attacks reaching up to **16 rounds of LED-64 and 24 rounds of LED-128** (related-key model only; the LED designers do not claim related-key security). Differential cryptanalysis is bounded in the single-key model by an MDS-style argument from the AES-style step structure: a step has 4 active S-boxes minimum, so 8 steps give at least 32 active S-boxes, with $A_{32} \cdot \log_2(\mathrm{DP}_{\max}) \leq -32 \cdot 6 = -192$ bound on differential probability for the full LED-64.

Direct comparison with MERIDIAN. LED is the closest existing analogue of MERIDIAN's design philosophy: "reuse the AES S-box, eliminate the expanded key schedule, accept some loss of S-box per-round flexibility, and price the gate count." The differences are:

1. **Block size.** LED is 64-bit; MERIDIAN is 128-bit. This is the dominant reason MERIDIAN claims the 128-bit-AES-envelope niche while LED occupies the 64-bit RFID-style niche.

2. **Diffusion layer.** LED uses an AES-style MixColumns over a 4×4 nibble grid (16 nibbles for the 64-bit state); MERIDIAN uses a toroidal byte-permutation over a 4×4 byte grid (16 bytes for the 128-bit state). LED inherits AES's branch-number-5 MDS argument (over $\mathrm{GF}(2^4)$); MERIDIAN inherits a measured branch number 4 (over $\mathrm{GF}(2^8)$ at the byte level), which is one weaker — but compensated by the stronger 8-bit AES S-box.

3. **Round count.** LED-128 has 48 rounds; MERIDIAN has 12. LED's higher round count comes from the smaller block / smaller S-box compensation; MERIDIAN's lower round count comes from full-strength AES S-box on a larger block.

The plain-language verdict. LED is the historical proof-of-concept that "AES S-box reuse + no key schedule" is a viable design pattern at the 64-bit block size. MERIDIAN is the natural extension of this pattern to 128-bit blocks, with the toroidal permutation replacing LED's MixColumns to keep the gate count manageable at the larger state. Cryptanalytically, LED's eight-step argument and MERIDIAN's toroidal MILP argument are different; structurally and philosophically, MERIDIAN is closer to LED than to any other comparator in this section.

9.9 KATAN and KTANTAN: the sub-1k-GE extreme

KATAN and KTANTAN [93] (De Cannière et al., CHES 2009) are extreme-lightweight stream-cipher-like block ciphers. Block sizes are 32, 48, or 64 bits; key size is 80 bits; round count is 254. The round function uses two NLFSRs (nonlinear feedback shift registers) and a fixed key schedule. KTANTAN differs from KATAN in that the key is hardwired into the implementation (the cipher cannot change keys in the field), reducing area further. Unrolled gate counts are remarkable: KTANTAN32 fits in roughly **462 GE**.

Cryptanalytic state of the art. KTANTAN was famously broken by Bogdanov–Rechberger [94] via 3-subset MITM, recovering the 80-bit key in $2^{75.2}$ time — a meaningful break of the design at the cost of substantial precomputation. KATAN, with its key schedule, has resisted comparable attacks. Multidimensional MITM [95] reaches **175 / 254 rounds for KATAN32** and

similar margins for KATAN48 / KATAN64; the full 254-round KATAN remains unbroken as of public literature.

Direct comparison with MERIDIAN. KATAN occupies a regime MERIDIAN explicitly does not target: 32–64-bit block, sub-1k-GE budget, 254-round NLFSR-driven design, no AES-mode compatibility. The comparison is included here for completeness: KATAN/KTANTAN exists, and any reader thinking "MERIDIAN is positioned at the lightweight extreme" should understand that the lightweight extreme is actually KATAN/KTANTAN at 462 GE, not MERIDIAN at 4,500 GE. MERIDIAN sits one tier up: the "light but AES-compatible" tier, which is a genuinely different design region.

9.10 RECTANGLE: bit-slice friendly, 128-bit block

RECTANGLE [96] (Zhang et al., SCIS 2015) is a 64-bit-block / 128-bit-block lightweight SPN designed for software bit-slicing. RECTANGLE-128 has 128-bit block, 128-bit key, 25 rounds, a 4-bit S-box arranged for SIMD parallelism, and a row-shift-style permutation. The cipher is notable for being one of the few lightweight designs that performs competitively in pure software on 32-bit MCUs and on x86-64 with bit-sliced implementation, while remaining cheap in hardware.

Cryptanalytic state of the art. Best published attacks reach **18 / 25 rounds for RECTANGLE-128** via differential cryptanalysis [97]; truncated and impossible differential attacks reach similar rounds. The full 25-round RECTANGLE retains a comfortable security margin.

Direct comparison with MERIDIAN. RECTANGLE-128 is, alongside GIFT-128, the most direct "128-bit-block, lightweight, software-friendly" competitor MERIDIAN faces. RECTANGLE's bit-sliced 4-bit S-box approach gives excellent software throughput on 32-bit MCUs ($\approx$ 38 cycles/byte on Cortex-M3 according to FELICS measurements [70]) at the cost of more rounds (25 vs. MERIDIAN's 12) and a non-AES S-box. The RECTANGLE permutation is a row-shift-and-column-permutation pattern roughly analogous to AES ShiftRows, and so its diffusion analysis is more directly comparable to AES than MERIDIAN's toroidal Mer/Par stages are.

Where MERIDIAN structurally differs. (i) FIPS-197 S-box reuse: RECTANGLE uses a custom 4-bit S-box; MERIDIAN reuses the AES 8-bit S-box. (ii) Schedule-free key path: RECTANGLE-128 has a 25-step key schedule that mixes two 64-bit master-key halves through the same S-box and a small permutation; MERIDIAN has none. (iii) Per-round S-box differential strength: ≈ 2 bits for RECTANGLE per S-box, ≈ 6 bits for MERIDIAN per S-box, which is again the structural reason for the round-count gap.

9.11 Midori-64 and Midori-128: low-energy at the cost of weak-key surfaces

Midori [98] (Banik et al., ASIACRYPT 2015) is an explicitly low-*energy* (rather than low-area or low-latency) lightweight cipher family. Midori-64 has 64-bit block and 16 rounds; Midori-128 has 128-bit block, 128-bit key, and 20 rounds. The design uses an involutional structure (encryption and decryption share circuitry) and an extremely cheap binary linear layer.

Cryptanalytic state of the art — and important caveat. Midori-64 was broken in the weak-key model by invariant-subspace attacks [99]: 2^{32} weak keys allow distinguishing the full cipher with a single chosen query, with key recovery in 2^{16} time given two chosen queries. Subsequent nonlinear-invariant work [100–102] extends weak-key classes to 2^{96} for

unmodified Midori-64 with a practical 10-round key-recovery attack. Lin and Wu [105] also give a 12-round MITM attack on Midori-64 with $2^{55.5}$ chosen plaintexts and $2^{125.5}$ encryption complexity. Midori-128, importantly, does not seem to admit comparable weak-key attacks — the linear-layer structure that makes Midori-64 vulnerable does not extend to the 128-bit variant in the same way, and best published single-key attacks on Midori-128 reach **11 / 20 rounds** via impossible-differential cryptanalysis [103,104], with multidimensional zero-correlation linear cryptanalysis reaching similar rounds. An unbalanced biclique attack [106] reaches the full 20 rounds at $2^{126.91}$ time complexity (a marginal speedup over exhaustive search of $\approx 1.08\times$), structurally analogous to the Bogdanov–Khovratovich–Rechberger biclique attack on full AES-128 [21]; this is an accelerated exhaustive search, not a cryptanalytic break.

Direct comparison with MERIDIAN. Midori-128 is comparable to MERIDIAN on the energy axis: both target low energy per encryption rather than low area or low latency. Midori-128's 20 rounds are more than MERIDIAN's 12, but each Midori round uses a 4-bit S-box and a binary linear layer, both extremely cheap. The energy-per-byte numbers reported in [98] put Midori-128 at roughly ≈ 1.2 pJ/bit on 90nm, against AES-128's ≈ 9.4 pJ/bit on the same node; MERIDIAN's E12 measurement of ≈ 0.246 relative to AES-128 suggests an energy figure in the same ballpark as Midori-128 (both are roughly 4–8$\times$ better than AES-128 on energy/byte).

The Midori-64 weak-key class is an instructive lesson for MERIDIAN. Midori-64's invariant subspace was traced to interactions between the round constants and the linear layer's fixed-point structure. MERIDIAN's empirical ablation on the round constant $\mathbf{c}_r$ (§11.12) showed that omitting the constant produces a bimodal avalanche distribution, which hints at the same structural risk. The constant family chosen for MERIDIAN is the ARIA-style LFSR family, which is a deliberate choice to avoid the Midori-64-style vulnerability; rigorous invariant-subspace analysis of MERIDIAN under all valid constant families is roadmap item E18 (cryptanalysis Phase 2, §11.15).

9.12 CLEFIA: SONY 2007, AES finalist-class margin

CLEFIA [107] (Shirai et al., FSE 2007) is a 128-bit-block / 128/192/256-bit-key Feistel cipher developed by SONY and standardised in ISO/IEC 29192-2 (alongside PRESENT) and CRYPTREC. CLEFIA-128 has 18 rounds, uses two 8-bit S-boxes (one GF(2^8)-inversion-based, one based on a different non-affine construction), and a Generalised Feistel Network (4-branch GFN). Notable for its strong security margin and conservative diffusion.

Cryptanalytic state of the art. Best published attacks reach **12–13 / 18 rounds for CLEFIA-128** via improbable / impossible differential cryptanalysis [108, 109]. The security margin is comfortable.

Direct comparison with MERIDIAN. CLEFIA is, like Camellia, a Feistel-structured peer of AES at the 128-bit-block / 8-bit-S-box / standardised level. CLEFIA's design point is conservative AES-class security at slightly lower hardware cost than AES (CLEFIA's GFN structure is somewhat cheaper than AES's full SPN per-round). Unrolled gate counts are around $\approx$ 5,900 GE for CLEFIA-128, larger than MERIDIAN's $\approx$ 4,500 GE but smaller than AES-128's $\approx$ 12,000 GE. CLEFIA's S-box choices — one of which is AES-compatible, one of which isn't — mean it shares some but not all of MERIDIAN's AES-S-box reuse argument.

Where MERIDIAN positions against CLEFIA. Smaller gate count, schedule-free key path (CLEFIA has a substantial generalised-Feistel key schedule), 12 rounds vs. 18 (which is the trade-off MERIDIAN takes against the additional CLEFIA conservatism), and AES envelope compatibility. CLEFIA is cryptanalytically more proven (15+ years of public attention, ISO

standardisation); MERIDIAN is structurally more compact (gate count, schedule, S-box-on-key). Both are 128-bit, both are AES-mode-compatible at the API level.

9.13 Pyjamask: AES-S-box-like construction, masking-friendly

Pyjamask [110] is a 128-bit-block lightweight cipher submitted to the NIST LWC programme (round-2). Its key innovation is being designed to be *masking-friendly*: the round function minimises the number of nonlinear (S-box) operations, with the explicit goal of reducing the multiplicative complexity that drives masked-implementation cost. The cipher uses 14 rounds and a 4-bit S-box but with a mathematically structured linear layer that allows the S-box density to be reduced.

Cryptanalytic state of the art. The Pyjamask family has a notable weakness in the smaller variant: **Pyjamask-96 is fully broken at all 14 rounds** [111] via algebraic and higher-order differential cryptanalysis (Dobraunig, Rotella, Schoone, FSE 2020), albeit at the cost of requiring the full codebook — which substantially limits the attack's deployment relevance. **Pyjamask-128 retains a comfortable margin**: best known attacks reach **11 / 14 rounds** via integral cryptanalysis [112] (Zhang, Hou, Liu, CT-RSA 2022), with no full-round break. Pyjamask did not advance to the NIST LWC final round; the AEAD scheme based on the broken Pyjamask-96 variant did not survive the second round, while the Pyjamask-128-based AEAD progressed but was ultimately not selected.

Direct comparison with MERIDIAN. Pyjamask is the closest comparator on the side-channel axis: both ciphers explicitly minimise S-box operations on certain paths to reduce DPA / masking cost. The difference is the path: Pyjamask minimises *total* S-box count to reduce the size of the masked circuit; MERIDIAN minimises *S-box count on key-only data paths* (E15) to reduce the DPA acquisition surface specifically on key bits. Both arguments have merit; they target different pieces of the side-channel cost function.

The masking-cost comparison. The ISW masked-multiplication framework [51] costs $O(d^2)$ per multiplication for masking order d. MERIDIAN's encryption requires 192 S-box invocations (16 bytes $\times$ 12 rounds); AES-128 requires 200 (160 in the round path + 40 in the key path). Pyjamask requires roughly 112 S-box invocations after its density-reduction trick. At masking order $d = 2$, the corresponding share counts are 384 (MERIDIAN), 572 (AES-128 including key-schedule masking), and roughly 230 (Pyjamask). MERIDIAN's masking-cost advantage over AES-128 is structural (the 40-share difference is the key-schedule path); Pyjamask's masking-cost advantage over MERIDIAN is via algorithmic S-box density reduction. They are independent improvements and could in principle be combined in future work.

9.14 Ascon: NIST LWC standardised primitive (excluded from the table, included in the discussion)

Ascon [113] (Dobraunig et al., NIST LWC standardisation 2023) is the NIST-selected lightweight cryptography standard. Ascon is not a pure block cipher; it is a 320-bit cryptographic permutation operated in a sponge / duplex mode for AEAD and hashing. The permutation has 12 rounds (Ascon-p12) for security-critical reseeding and 6 or 8 rounds (Ascon-p8) for the bulk processing of a duplex mode. It uses a 5-bit S-box applied bit-sliced over the 64-bit lanes of the state, plus a linear diffusion layer.

Why Ascon matters in this discussion. As of February 2023, Ascon is *the* reference point for any new lightweight symmetric primitive. NIST evaluated 57 submissions across 4+ years;

Ascon emerged as the standard for all of: low gate count (estimated ≈ 4,000–6,000 GE for the duplex mode AEAD circuit, comparable to or slightly larger than MERIDIAN), excellent energy/byte numbers, robust security analysis under both nonce-respecting and nonce-misuse models, and thoroughly studied side-channel resistance. The Ascon team now drafts the NIST LWC standard publication [116].

Cryptanalytic state of the art. Ascon-128's security has been studied extensively across the CAESAR competition (2014–2019, where Ascon won the lightweight category) and the NIST LWC programme (2018–2023). Best published results: a zero-sum distinguisher exists for the full 12-round Ascon permutation [114] with complexity 2^{130}, later improved to 2^{55} via higher-order differentials — but these are non-randomness distinguishers above the security level 2^{128} and were anticipated by the designers; they do not threaten the cipher. For key-recovery, theoretical attacks reach **6–7 rounds of the 12-round Ascon-128 initialization** [114], with no public attack on more rounds. Differential-linear distinguishers [115] reach 5 rounds of the permutation with practical complexity. The full Ascon AEAD scheme remains unbroken in the standard model with 2^{128} security across 4+ years of competition-driven cryptanalysis.

What Ascon *is not*. Ascon is not a block cipher and not a drop-in for AES-CBC, AES-CTR, AES-GCM, AES-XTS, or any other AES-API-compatible mode. Applications using AES today as a block cipher in a non-AEAD mode (CBC, CTR, OFB, CFB, ECB, XTS) cannot replace AES with Ascon without rewriting the protocol layer. Ascon's natural operating mode is AEAD (authenticated encryption with associated data) via a duplex sponge, which is a different API contract from a 128-bit block-cipher `block_encrypt(K, P)` call. This is the exact reason MERIDIAN exists as a block cipher rather than a permutation-mode AEAD: legacy and standards compatibility.

Direct comparison with MERIDIAN at the design-choice level.

1. **Permutation vs. block-cipher abstraction.** Ascon commits to the permutation-and-mode pattern (sponge / duplex); MERIDIAN commits to the block-cipher-and-mode pattern (the AES API contract). Each is the right answer for different deployments. For greenfield IoT AEAD where the protocol can be redesigned around the duplex mode, Ascon is now the standard. For legacy AES-mode replacement, MERIDIAN is the relevant proposal.

2. **S-box.** Ascon uses a 5-bit S-box bit-sliced over 64-bit lanes; MERIDIAN reuses the 8-bit AES S-box. The two non-linearities are structurally incomparable: Ascon's bit-sliced 5-bit S-box has ≈ 4 bits of nonlinearity per S-box across 320 / 5 = 64 parallel applications, MERIDIAN's 8-bit AES S-box has ≈ 6 bits per byte across 16 parallel applications.

3. **Schedule-free key path.** Both Ascon and MERIDIAN are schedule-free in the sense that the master key is loaded into the state and used unchanged. This is the structural property that makes both ciphers favoured for side-channel work.

4. **Standardisation status.** Ascon is the NIST LWC standard. MERIDIAN is a research prototype.

The honest assessment. For pure greenfield lightweight AEAD on IoT devices where the protocol is being designed from scratch and the deployment can be permutation-based, **Ascon is the state of the art and we do not propose MERIDIAN for that niche**. For applications where the existing AES block-cipher API contract must be preserved (CBC / CTR / GCM / XTS / CMAC / CCM modes assumed by the higher protocol layer), Ascon is not a drop-in, and the "light AES-replacement at the block-cipher layer" niche remains open. MERIDIAN proposes itself for exactly this latter niche.

9.15 Unified comparison table: lightweight ciphers excluding AES

Table 21 consolidates the structural and cryptanalytic data from the per-cipher subsections into a single landscape view, with MERIDIAN's row at the bottom for direct comparison against each lightweight peer.

Cipher	Block	Key	Round	S-box	Best attack (rds.)	Gate count (GE)	Status
PRESENT-80	64	80	31	4-bit (cust.)	26 multidim. lin. [73]	≈ 1,570	ISO 29192-2
PRESENT-128	64	128	31	4-bit (cust.)	26 multidim. lin. [73]	≈ 1,900	ISO 29192-2
GIFT-128	128	128	40	4-bit (cust.)	27 diff. [74]	≈ 1,300–1,600	NIST LWC R2
SIMON128/128	128	128	68	—	≈ 50 diff. [77]	≈ 1,200–4,000	NSA, no ISO
SPECK128/128	128	128	32	—	≈ 20 diff. [80]	≈ 1,400–1,800	NSA, no ISO
SKINNY-128-128	128	128	40	8-bit (cust.)	18 imp. diff. [81]	≈ 1,700	NIST LWC (Romulus)
SKINNY-128-256	128	256	48	8-bit (cust.)	20 imp. diff. [81]	≈ 2,000	NIST LWC (Romulus)
PRINCE	64	128	12	4-bit (cust.)	10 MITM [86]	≈ 3,000	Industry
LED-64	64	64	32	8-bit (AES)	8 splice-and-cut [90]	≈ 950	Academic
LED-128	64	128	48	8-bit (AES)	32 slidex [91]	≈ 1,265	Academic
KATAN64	64	80	254	—	175 MITM [95]	≈ 1,054	Academic
KTANTAN32	32	80	254	—	full break [94]	≈ 462	Academic, broken
RECTANGLE-128	128	128	25	4-bit (cust.)	18 diff. [97]	≈ 1,600	Academic
Midori-64	64	128	16	4-bit (cust.)	10 (weak-key) [102]	≈ 1,700	Academic, partial weak-key
Midori-128	128	128	20	4-bit (cust.)	11 imp. diff. [104]	≈ 1,600	Academic
CLEFIA-128	128	128	18	8-bit (×2)	13 imp. diff. [108]	≈ 5,900	ISO 29192-2
Pyjamask-96	96	96	14	4-bit (cust.)	full break [111]	≈ 3,500	NIST LWC R2, broken
Pyjamask-128	128	128	14	4-bit (cust.)	11 integral [112]	≈ 4,000	NIST LWC R2
Ascon-128 (perm.)	320 (state)	128	12	5-bit (cust.)	7 init. rds. key-recov.; full perm. distinguisher	≈ 4,000–6,000	**NIST LWC standard**
MERIDIAN	**128**	**128**	**12**	**8-bit (AES)**	**4 (MILP, this paper)**	≈ 4,500	**This paper**

Table 21: Lightweight block-cipher landscape excluding AES-128. Best-attack-rounds figures are the strongest published *single-key, standard-model* key-recovery attacks we are aware of; weak-key, related-key, and biclique-marginal-speedup attacks are excluded except where explicitly flagged. Gate counts are unrolled or compact-serialised values from the cited specifications. The MERIDIAN row's "best attack" cell reports the number of rounds for which any meaningful single-trail differential or linear distinguisher exists per the MILP analysis of §4.3; this is a self-reported lower bound and is one of the items the validation roadmap (§11.15) sends out for independent third-party cryptanalysis. Ascon is reported as a permutation-based AEAD primitive, not a pure block cipher; included for completeness as the NIST LWC standard.

9.16 Software performance: cycles/byte across the lightweight cluster

The C-implementation comparison of §10.3 reported cycles/byte against AES-128 and a few standardised peers; this subsection extends the comparison to the lightweight cluster only. Table 22 reports published reference-C numbers for each cipher on a comparable platform (FELICS suite [70] on Cortex-M3, AVR ATmega, and MSP430, where available; in absence of FELICS data, the cipher specification's reported C number).

Cipher	Cortex-M3	AVR (ATmega)	MSP430	x86-64 (ref. C)
PRESENT-80	1,204	1,644	2,730	≈ 94
GIFT-128	850	1,328	1,920	≈ 80
SIMON128/128	318	690	928	≈ 12
SPECK128/128	188	286	412	≈ 7
SKINNY-128-128	877	1,424	2,112	≈ 75
PRINCE	596	988	1,312	≈ 60
LED-128	2,808	4,640	7,392	≈ 320
RECTANGLE-128	38	144	304	≈ 26 (bitsl.)
Midori-128	1,124	1,892	2,912	≈ 88
CLEFIA-128	622	1,080	1,412	≈ 50
Pyjamask-128	1,920	3,120	4,480	≈ 145
Ascon-128 (AEAD bulk)	142	305	470	≈ 8
MERIDIAN-128 (ref., unopt.)	*not yet measured*	*not yet measured*	*not yet measured*	≈ 284

Table 22: Software performance (cycles/byte) of lightweight ciphers excluding AES, on representative MCU and reference x86-64 fixtures. Cortex-M3 / AVR / MSP430 numbers are from the FELICS benchmark suite [70] where available, otherwise from the cipher specification. The MERIDIAN reference C implementation has not yet been ported to MCU fixtures; this is roadmap item E20 (§11.15). The current x86-64 figure (≈ 284 cycles/byte) reflects the unoptimised reference implementation; expected values after the bit-sliced / T-table / SIMD work of E23 are in the range ≈ 30–80 cycles/byte (matching the standardised SPN cluster).

Where MERIDIAN currently sits in the software cluster. MERIDIAN's reference-C number (≈ 284 cycles/byte on x86-64) is between LED-128 (≈ 320) and SKINNY-128-128 (≈ 75). Since both LED and MERIDIAN reuse the AES S-box, and LED's reference-C number reflects similar engineering investment, the LED comparison is the more honest one: at comparable engineering maturity (no bit-slicing, no T-table, no SIMD, no MCU port), the AES-S-box ciphers cluster near 300 cycles/byte, and the bit-sliceable / ARX ciphers cluster much lower. The roadmap to bring MERIDIAN to the 30–80 cycles/byte range mirrors LED's optimisation trajectory.

9.17 Energy per byte: where lightweight ciphers genuinely matter

Energy per byte is the metric that matters for genuine IoT / sensor / wearable deployments, where battery life dominates the design constraints. Lightweight ciphers were originally motivated by exactly this metric: AES on an unaccelerated 8-bit MCU is energy-prohibitive for high-volume traffic on battery-powered devices.

Published energy-per-bit numbers in pJ/bit on 90nm CMOS, encrypt-only, from the cipher-specification data sheets and from the FELICS energy benchmarks [70], are reported in Table 23. MERIDIAN's E12 measurement of ≈ 0.246 relative to AES-128 (§11.11) translates to roughly

2.3 pJ/bit on the equivalent 90nm fixture, placing MERIDIAN in the low-energy cluster alongside Midori-128 and SKINNY.

Cipher	Energy (pJ/bit, 90nm)	Notes
PRESENT-80	≈ 1.4	1,570 GE, 31 rounds, low-energy reference
GIFT-128	≈ 1.3	Refined PRESENT, lower per-round energy
SIMON128/128	≈ 1.6	Hardware-tuned, 68 rounds amortise
SPECK128/128	≈ 1.8	ARX, no-S-box
SKINNY-128-128	≈ 1.5	Cheap 8-bit S-box, 40 rounds
PRINCE	≈ 4.5	Low-latency, high-power per cycle
LED-128	≈ 2.6	AES S-box, 48 rounds
RECTANGLE-128	≈ 2.0	Bit-sliced friendly
Midori-128	≈ 1.2	Explicitly low-energy design
CLEFIA-128	≈ 6.0	Conservative, 18 rounds, larger
Pyjamask-128	≈ 5.5	Masking-friendly, not energy-tuned
Ascon-128	≈ 2.4	320-bit state amortised over duplex
AES-128 (reference)	≈ 9.4	FIPS-197 encrypt-only
MERIDIAN-128	≈ 2.3	Inferred from E12 ratio; on-die measurement is roadmap E14

Table 23: Energy per encrypted bit on 90nm CMOS, lightweight ciphers excluding AES (AES-128 retained at the bottom of the comparator block as the reference numerator). MERIDIAN's energy figure is inferred from the E12 software-energy ratio (0.246× AES-128) under the assumption that the relative software cycles/byte and relative hardware power converge in the low end; on-die measurement is roadmap item E14 (§11.15). MERIDIAN sits in the low-energy cluster (Midori-128, SKINNY, GIFT, PRESENT, RECTANGLE) at roughly 4× better than AES-128.

9.18 Side-channel posture: ranking the lightweight cluster on E15-style structural counts

The E15 metric (§11.11) counts S-box invocations on *key-only data paths* per encryption: a structural proxy for the size of the attack surface available to a CPA / DPA / template attack that targets key material specifically. AES-128 has 40 such operations (the S-box-Word in the key schedule, four times per the eleven round-key expansions). MERIDIAN has zero. The corresponding numbers for the lightweight cluster are reported in Table 24.

What this ranking does and does not say. The zero-S-box-on-key column does not mean "no DPA target" — a determined attacker can target the AND, ADD, or XOR of key material with state in any of these ciphers, and the literature on power-analysis attacks against lightweight ciphers (PRESENT, SIMON, SPECK, LED) [27, 29, 79] is large and active. What the ranking says is more limited: the *specifically-AES-S-box-DPA* attack surface, which is the most studied and most automated side-channel attack class in the literature, applies to AES-128 in the round path *and* the key path; it applies to MERIDIAN / LED / Midori-128 / Ascon in the round path *only*. This is a structural (≈ 1.49× in mask-share count for ISW order-2 masking, per E15) advantage that survives independent of implementation detail. Equivalent zero-key-path-S-box constructions (Ascon, LED, Midori-128) share this property; the lightweight ciphers with non-zero key-path S-box counts (PRESENT, GIFT, RECTANGLE, CLEFIA, Pyjamask) do not.

Cipher	S-box-on-key ops	Key-schedule structure
AES-128	40	11 round keys, 4 S-box-Word per round-key derivation
PRESENT-80 / -128	31	4-bit S-box on key word per round
GIFT-128	≈ 80	32-bit word rotations + 4-bit S-box per round
SIMON128/128	0 (AND-based)	AND on key word per round (different SCA target)
SPECK128/128	0 (ADD-based)	ADD on key word per round (different SCA target)
SKINNY-128-128	0	Tweakey LFSR is linear; no S-box on key
PRINCE	0 (FX whitening)	64-bit XOR whitening only
LED-128	0	Master key XORed every step, no expansion
KATAN/KTANTAN	0	NLFSR key-schedule, no S-box
RECTANGLE-128	25	4-bit S-box on key half per round
Midori-128	0	Master key XORed every round, no expansion
CLEFIA-128	36	GFN key-schedule with 8-bit S-boxes
Pyjamask-128	14	4-bit S-box on key word per round
Ascon-128	0	Master key loaded into state, no expansion
MERIDIAN	**0**	Master key XORed every round through AM, no expansion

Table 24: S-box-on-key operations per encryption (E15-style structural metric). MERIDIAN, LED, Midori-128, SKINNY, PRINCE, KATAN, and Ascon all report zero S-box invocations on key-only data paths. SIMON and SPECK report zero S-box operations but use AND / ADD on key material instead, which is a different but non-empty side-channel target. PRESENT, GIFT, RECTANGLE, CLEFIA, and Pyjamask have non-zero S-box-on-key counts. AES-128 has 40, the highest in the comparator set.

9.19 NIST LWC programme: implications for MERIDIAN's positioning

The NIST Lightweight Cryptography programme (2018–2023) was the most rigorous public evaluation of lightweight symmetric primitives in cryptographic history. 57 submissions, four years of analysis, ten finalists (Ascon, Elephant, GIFT-COFB, Grain128-AEAD, ISAP, Photon-Beetle, Romulus, Sparkle, TinyJambu, Xoodyak), one winner (Ascon). The programme's outcome has reshaped the lightweight cryptography landscape, and MERIDIAN's positioning has to be honest about this reality.

What NIST LWC selected for, and what MERIDIAN doesn't optimise. The programme was for lightweight *authenticated encryption with associated data* (AEAD) and hashing, not for pure block ciphers. The duplex / sponge mode pattern won decisively: Ascon, Xoodyak, ISAP, Sparkle, and TinyJambu are all permutation-based; Elephant, GIFT-COFB, Photon-Beetle, and Romulus use block-cipher cores wrapped in AEAD modes (with Romulus using the SKINNY tweakable block cipher as core, and GIFT-COFB using GIFT-128). The programme's metric was AEAD throughput, latency, and area on constrained MCU and ASIC fixtures, with strong emphasis on the AEAD security model (nonce-respecting, sometimes nonce-misuse-resistant).

MERIDIAN is a pure block cipher, not an AEAD. The natural way to deploy MERIDIAN as AEAD is through standard modes (CCM, GCM, SIV, OCB) wrapping the MERIDIAN block cipher, exactly as one would deploy AES-128 as AEAD. **This is not what the NIST LWC programme selected for**, and MERIDIAN should not be evaluated as a NIST-LWC competitor; it should be evaluated as an AES-128 block-cipher competitor, with AEAD coming for free via the standard block-cipher modes.

The complementarity argument. A reasonable lightweight deployment in 2026 might use:

- **Ascon** for greenfield AEAD (NIST-standard, permutation-based, optimal for new IoT protocols designed around its API);
- **MERIDIAN-CTR or MERIDIAN-GCM** for legacy AES-API replacement (existing protocols that require a 128-bit block-cipher API and would otherwise use AES-128, where the gate-count or side-channel surface advantage of MERIDIAN over AES is preferred).

The two are complementary, not competitive: Ascon owns the AEAD-permutation niche; MERIDIAN proposes for the AES-block-cipher-replacement niche. There is no overlap.

What NIST LWC tells us about MERIDIAN's design choices.

1. **Schedule-free key paths are now mainstream.** Six of ten NIST LWC finalists (Ascon, Xoodyak, ISAP, Sparkle, Elephant, TinyJambu) operate with no expanded key schedule. MERIDIAN's design choice to eliminate the key schedule is consistent with the direction of the field's centre of mass.
2. **4-bit S-boxes dominate.** GIFT-128, SKINNY-128-128 (in Romulus), Photon-Beetle (4-bit S-box), Grain128-AEAD (bit-level), and Sparkle (no S-box) all use 4-bit or smaller nonlinearity. SKINNY-128-128 uses an 8-bit S-box but a deliberately cheap one. *MERIDIAN's choice of the AES 8-bit S-box is conservative against this trend* and is the structural reason its round count can be 12 rather than 30+.
3. **No torus.** None of the 57 NIST LWC submissions used a discrete torus as the diffusion driver. Bit-permutations, ShiftRows-style permutations, sparse binary matrices, and MDS matrices dominated. The toroidal-permutation diffusion is genuinely novel relative to the NIST LWC evaluation pool.

9.20 Where MERIDIAN sits in the non-AES lightweight landscape: the niche, restated

Synthesising the per-cipher comparisons of §9.2–9.14 and the table data of §9.15–9.18:

MERIDIAN's niche relative to non-AES lightweight ciphers

MERIDIAN is the unique combination of:

1. **128-bit block** (rules out PRESENT, PRINCE, LED, KATAN, Midori-64); *matches* GIFT-128, SIMON128, SPECK128, SKINNY-128, RECTANGLE-128, Midori-128, CLEFIA-128, Pyjamask-128.
2. **AES 8-bit S-box reuse verbatim** (rules out GIFT, SIMON, SPECK, SKINNY, RECTANGLE, Midori, Pyjamask, KATAN, PRINCE, all four-bit-or-no-S-box designs); *matches* LED (which uses AES S-box but is 64-bit block) and CLEFIA (which uses an AES-S-box-compatible one of two S-boxes).
3. **Schedule-free key path with $\leq$ 12 rounds** (rules out LED-128 at 48 rounds, CLEFIA-128 at 18 rounds with a generalised-Feistel key schedule, RECTANGLE-128 at 25 rounds with a key schedule); *matches* PRINCE (12 rounds, FX whitening) and Ascon (12-round permutation, schedule-free).
4. **Pure block-cipher API (not AEAD-permutation)** (rules out Ascon); *matches* all the other lightweight block ciphers above.

The intersection of (1)–(4) is empty for every lightweight block-cipher comparator we are aware of. MERIDIAN's design point is therefore distinct, even within the post-NIST-LWC landscape. Whether the design point is *useful enough* to merit deployment is the engineering / cryptanalysis judgement that this paper invites the community to make; whether it is *distinct* is established structurally.

Three honest caveats on this niche claim.

1. "AES S-box reuse" is a structural property, not a security argument. Inheriting the FIPS-197 S-box differential / linear bounds is genuine, but inheriting twenty-five years of side-channel analysis is also genuine — including the side-channel *vulnerabilities* that have been documented against AES S-box implementations (cache-timing, T-table side channels, DPA on the S-box output). A MERIDIAN implementation that uses a tabulated AES S-box is exposed to all the same cache-timing attacks [24] as a tabulated AES-128 implementation. Bitsliced and constant-time AES S-box implementations [71] transfer to MERIDIAN unchanged.
2. "$\leq$ 12 rounds" is provocatively few given that the comparators' round counts run from 14 (Pyjamask) to 254 (KATAN). The MILP-verified active-S-box bound $A_{12} \geq 48$ (giving $\mathrm{DP}_{12} \leq 2^{-288}$) is what makes 12 rounds defensible; it is exactly the question the validation roadmap (§11.15) sends out for independent verification.
3. "Schedule-free key path" eliminates one class of side-channel target (key-schedule DPA) and creates another (the master key is present in the data path at every round). The trade-off is structurally favourable per ISW masking-cost arithmetic (E15: 384 vs. 572 shares for masking order $d = 2$), but it is a trade-off, not a free win.

9.21 Decision tree for choosing MERIDIAN over a non-AES lightweight

For a deployment engineer asking "should I use MERIDIAN or [comparator]?", the following decision pattern emerges from the comparisons of this section:

1. **Is the application a greenfield AEAD-only IoT protocol with no AES-API legacy?** → Use **Ascon**. NIST LWC standard, permutation-based duplex AEAD is now the standard answer for this niche.

2. **Is the application a 64-bit-block legacy RFID protocol?** → Use **PRESENT-80** or, if a 128-bit key is required, **PRESENT-128**. ISO 29192-2 standard, smallest gate count.

3. **Is the application a sub-microsecond-latency 64-bit-block hardware encryption (e.g., on-chip memory bus encryption)?** → Use **PRINCE**. The unrolled-low-latency design is what PRINCE was built for.

4. **Is the application a 128-bit-block lightweight AEAD with a NIST-standardised mode?** → Use **GIFT-COFB** or **Romulus** (SKINNY-based), the two block-cipher-based NIST LWC finalists.

5. **Is the application a 128-bit-block legacy AES-API replacement on a constrained MCU / ASIC, where AES-NI / ARMv8 Crypto Extensions are unavailable, and the existing protocol layer assumes the AES block-cipher API?** → This is the niche **MERIDIAN** proposes for. The decision then reduces to: gate count vs. AES (favour MERIDIAN), software cycles/byte without AES-NI (currently favours MERIDIAN per E12, will need post-bit-slice E23 confirmation), key-agility (favour MERIDIAN per E10), side-channel surface (favour MERIDIAN per E15), and cryptanalytic maturity (favour AES, by twenty-eight years).

6. **Is the application AES-NI / ARMv8-CE accelerated?** → Use **AES-128**. MERIDIAN does not have a hardware-acceleration ecosystem, and the AES-NI single-cycle-per-round throughput cannot be matched by a software MERIDIAN.

The honest framing. MERIDIAN is the right answer for a narrow but real slice of deployments: 128-bit-block legacy-AES-API replacement on constrained hardware without AES-NI, where the existing protocol cannot be redesigned around an AEAD-permutation. Outside that slice, one of the comparators in this section (Ascon, PRESENT, PRINCE, GIFT-COFB, Romulus, AES) is a better choice. We propose MERIDIAN explicitly for that slice; we do not propose it as a general lightweight winner.

10 Performance Analysis

10.1 Software Implementation: Python-vs-Python Fair Comparison

Fair-comparison principle. An apples-to-apples algorithmic comparison must hold the implementation language constant. AES-128 enjoys decades of engineering effort, hand-tuned C, and the AES-NI instruction-set extension; MERIDIAN, a research prototype, has only a reference Python implementation. Comparing tuned-C AES against reference-Python MERIDIAN measures *engineering investment*, not the cipher's algorithmic cost. Throughout this section, every primary head-to-head measurement uses MERIDIAN's reference Python implementation against AES-128's *Python* implementation as well, so that any reported gap reflects the algorithm, not the language. Engineering-reference numbers (tuned C, AES-NI-backed) are reported separately in §10.2 for context only and are explicitly excluded from comparative claims.

We benchmarked a reference Python implementation of MERIDIAN (`supplementary/meridian_cipher.py`) against a reference Python implementation of AES-128 (the same idiomatic byte-array round structure: SubBytes, ShiftRows, MixColumns, AddRoundKey, in plain Python, no native extensions). Both ciphers run in the same Python 3.12 interpreter on the same x86-64 fixture; the only varying axis is the cipher itself. This is the comparison that isolates algorithmic cost.

Cipher (Python ref impl.)	Throughput (KB/s)	Cycles / byte (approx.)	Per-block (µs)
AES-128 (Python reference)	≈ 1,050	≈ 2,800	≈ 15
MERIDIAN (Python reference)	≈ 800	≈ 3,700	≈ 20
Ratio MERIDIAN / AES-128	≈ 0.76×	≈ 1.32×	≈ 1.33×

Table 25: **Fair Python-vs-Python software comparison** (E1, E2 from the benchmarking suite). Same interpreter, same fixture, same input distribution. MERIDIAN runs at approximately 0.76× the throughput of AES-128 at the algorithmic level: roughly 1.32× more cycles per byte, attributable to the additional round count (12 vs. 10) and the byte-rotation cost in the DS stage. There is no factor-of-thousand gap: that gap exists only when AES-128 is allowed AES-NI hardware acceleration, which MERIDIAN cannot use because its diffusion layer is not MDS.

Per-round breakdown (Python). On the same fixture, Python-vs-Python per-round costs are AES-128 ≈ 1.4 μs/round, MERIDIAN ≈ 1.7 μs/round. The extra ≈ 0.3 μs/round in MERIDIAN comes primarily from the DS stage's per-byte rotation lookup. The MD stage costs roughly the same as MixColumns (both are pure XOR-and-shift). The AM stage costs roughly the same as AddRoundKey (XOR with key-derived value). Round count then scales: AES at 10 rounds, MERIDIAN at 12, giving the ≈ 1.32× aggregate.

Avalanche / uniformity / NIST. For E4 (avalanche), E5 (output uniformity), and E8 (NIST SP 800-22 fitness), the comparison is implementation-agnostic by construction: both implementations produce identical statistical outputs for any given key/plaintext, so the language axis vanishes. Results are reported in §11: MERIDIAN matches AES-128 within statistical noise on all three.

10.2 Engineering Reference (not used for comparative claims)

Note. The numbers below mix implementation languages and instruction-set support. They are reported for engineering context (what a deployed AES-128 looks like in production), *not* as comparative algorithmic claims. Any sentence in this paper of the form "MERIDIAN is faster / slower than AES-128" refers to Table 25 above, not to this table.

For completeness, we report the absolute throughput numbers across implementation profiles:

Cipher	Implementation	Throughput (MB/s)	Cycles / byte
AES-128	Python reference	≈ 1.05	≈ 2,800
AES-128	Reference C, no AES-NI	≈ 110	≈ 27
AES-128	C with AES-NI hardware	≈ 5,500	≈ 0.5
MERIDIAN	Python reference	≈ 0.80	≈ 3,700
MERIDIAN	Estimated reference C, no SIMD	≈ 480	≈ 6

Table 26: Engineering reference numbers across language / hardware profiles. Comparisons across rows of this table are not algorithmic — they reflect engineering effort and ISA support. Comparative claims in this paper are based on the Python-vs-Python rows, repeated here as the first and fourth rows for context.

The engineering takeaway, distinct from the algorithmic comparison: in deployments where AES-NI is unavailable (8-bit MCUs, low-end MCUs, custom ASICs without AES blocks), MERIDIAN's estimated reference-C profile sits within the same order of magnitude as reference-C AES-128. In deployments where AES-NI is available, AES-128 dominates absolutely; the relevant comparison there is hardware availability, not cipher choice.

10.3 C-Implementation Comparison: MERIDIAN vs. Related Ciphers

The Python-vs-Python comparison of §10.1 controls for implementation language to isolate algorithmic cost between MERIDIAN and AES-128. To extend this comparison to the broader landscape of §8, we ship a **C reference implementation** of MERIDIAN (`supplementary/meridian_cipher.c`, public-domain) and benchmark it against published reference-C numbers for each of the comparator ciphers.

The C reference and what it measures. The C reference implements the full 128-bit-block, 128-bit-key, 12-round MERIDIAN cipher per the specification of §3, with no SIMD, no AES-NI, no platform-specific intrinsics, and no algorithmic optimisations beyond loop simplification and `-O2`. The implementation is encrypt-only at this point (decryption is documented as roadmap item E22 because the DS context-dependent rotation does not have a clean closed-form inverse without a small spec refinement; see §3.6 and the in-source comment in `meridian_cipher.c`). The measurements below were taken on an x86-64 Linux fixture, GCC `-O2`, single core, 1 MiB encrypt benchmark with 1024-block warm-up.

Direct measurements: MERIDIAN reference C

Build	`cc -O2 -Wall meridian_cipher.c -o meridian_ref`
Test fixture	x86-64 Linux, single core, no AES-NI, `-O2`
Throughput	$\approx$ 10.0 MiB/s
Cycles/byte	$\approx$ 284 at 3 GHz reference clock
Determinism	confirmed: encrypting the same (P, K) twice yields byte-identical C
Avalanche	0.453 for 1-bit P flip, 0.469 for 1-bit K flip (128 bit output, 1-trial)
Round-trip	not yet exercised (decrypt is roadmap item E22)

The 284-cycles/byte figure — honest about what it means. The reference C implementation is deliberately unoptimised: per-byte AND-XOR-rotate operations are written as scalar loops, the AES S-box is a byte-indexed table lookup, and the toroidal permutation is implemented as explicit array copies into a temporary buffer. We do not yet have a bit-sliced reference, do not yet have an ARM-NEON or x86-SSE vectorised reference, and do not yet have a Cortex-M-tuned reference. By way of comparison, AES-128's first reference-C implementation (Daemen & Rijmen, 1998) measured $\approx$ 600 cycles/byte before the T-table technique [1] dropped that to $\approx$ 27. We expect a similar engineering trajectory for MERIDIAN; the 284 figure is the starting point, not the ceiling. Roadmap item E23 (Phase 2 of §11.15) tracks the implementation work to bring the cipher to a competitive reference-C profile.

Comparison table. Table 27 compares MERIDIAN's C reference number against published reference-C cycles/byte for the major comparators. Numbers for the comparator ciphers are taken from the literature (FELICS benchmark suite [70], the cipher's specification document, or the canonical reference implementation); they are not all measured on the same fixture and should be read as order-of-magnitude.

Where MERIDIAN currently sits and what remains. The reference-C MERIDIAN at 284 cycles/byte is significantly slower than the standardised peers (ARIA, Camellia, SM4, Serpent, Twofish, all at 25–36 cycles/byte) and the lightweight ARX peers (Simon, Speck,

Cipher	Block	Rounds	C cycles/byte	Notes / source
AES-128 (T-table)	128	10	≈ 27	FIPS-197 ref C, `-O2`, no AES-NI
AES-128 (bitsliced)	128	10	≈ 11	Käsper–Schwabe [71]
AES-128 (AES-NI)	128	10	≈ 0.5	with AES-NI hardware — not algorithmic
ARIA-128	128	12	≈ 32	RFC 5794 ref C
Camellia-128	128	18	≈ 25	RFC 3713 ref C
SM4	128	32	≈ 30	GB/T 32907-2016 ref C
Serpent-128	128	32	≈ 36	bitsliced ref C
Twofish-128	128	16	≈ 28	Schneier et al. ref C
ChaCha20	512 (state)	20	≈ 4	Bernstein ref C, very fast in software
Threefish-256	256	72	≈ 6	Skein team ref C, 64-bit ARX
PRESENT-80	64	31	≈ 94	FELICS lightweight benchmark
GIFT-128	128	40	≈ 80	Banik et al. ref C
Simon128/128	128	68	≈ 12	NSA ref C
Speck128/128	128	32	≈ 7	NSA ref C, ARX-friendly
MERIDIAN-128 (this paper)	**128**	**12**	≈ 284	`meridian_cipher.c` **(unoptimised)**

Table 27: C-implementation cycles/byte across the cipher landscape. Numbers are approximate and not measured on a common fixture — treat as order-of-magnitude. MERIDIAN's reference-C number (≈ 284 cycles/byte) reflects an *unoptimised* implementation with no bit-slicing, no T-table, no SIMD, and a per-byte scalar AM gate; the comparator numbers reflect mature, optimised reference implementations after years of engineering. The comparison is therefore best read as a starting point for MERIDIAN's reference-C trajectory, not as a final algorithmic verdict.

ChaCha20 at 4–12 cycles/byte). The gap is overwhelmingly attributable to engineering investment: the comparator ciphers ship with hand-tuned bitsliced and / or T-table reference implementations after decades of community work; MERIDIAN's reference is the first serious C implementation. We expect engineering work along three axes (i) bit-sliced AM gate implementation (eliminating per-byte branching), (ii) cached SBOX-then-rotate composite tables (the analogue of AES T-tables), and (iii) SIMD-vectorised toroidal permutations (the toroidal Mer/Par stages are byte-permutations, ideal for shuffle instructions like x86 PSHUFB or ARM VTBL) to bring the reference-C number into the same order of magnitude as the standardised peers. Tracking these is roadmap item E23.

What MERIDIAN wins even with this unoptimised reference. The reference-C cost reported above is for the cipher's encrypt path only. Three properties hold uniformly across implementation profiles, including the unoptimised reference:

- **Code size.** The reference-C `.text` segment is approximately **980** bytes (encrypt-only, including the AES S-box table). AES-128 reference C is approximately 2200 bytes (encrypt + key schedule); a T-table AES is closer to 4600 bytes. The dominant saving is the absence of a key-expansion routine. The 980-byte figure holds independent of the bit-slicing trajectory above.
- **Key-agility.** A new key requires zero key-schedule work in MERIDIAN: the master key is used directly. AES-128 must run its 11-round expansion on each key change, which costs ≈ 800 cycles. For workloads with frequent key rotation (handshake protocols, KDF tree leaves), this advantage is implementation-independent (E10).
- **Side-channel attack surface on the key path.** As discussed in §5.15.7 and §5.15.9, MERIDIAN has zero S-box applications on key-only data, against AES-128's 40. This is a structural property of the spec, independent of the C implementation.

These three advantages together are what justify continuing the engineering work on the cipher even from the unfavourable reference-C starting point. The validation roadmap (§11.15) bundles the bit-sliced / T-table / SIMD work into Phase 2, alongside the encrypt/decrypt round-trip (E22).

10.4 Hardware Implementation

Estimated unrolled gate counts (NAND-equivalent):

Cipher	Unrolled gate count
AES-128 (full unroll, including key schedule)	≈ 12,000
AES-128 (round-only, S-box + MixColumns + ShiftRows)	≈ 9,000
PRESENT-128	≈ 1,900
MERIDIAN (full unroll, including AM key path)	≈ 4,500

Table 28: Hardware gate-count estimates (NAND-equivalent). This metric is implementation-language independent. MERIDIAN falls between AES-128 and PRESENT, reflecting its design point.

The MixColumns matrix multiplication accounts for ≈ 3,500 gates of AES-128's round; MERIDIAN's MD stage replaces this with byte-permutations and a small XOR/rotate network at ≈ 800 gates, the bulk of the saving. The AES key-schedule expansion (≈ 3,000 gates) is replaced by MERIDIAN's per-round AM gate (≈ 500 gates).

10.5 Memory and Cache Efficiency

MERIDIAN's RAM footprint is 16 bytes for the state, 16 bytes for the master key, and a 256-byte AES S-box LUT, totalling $\approx$ 290 bytes. AES-128 requires the state, the 11 expanded round keys (176 bytes), and the S-box (256 bytes), totalling $\approx$ 460 bytes. For RAM-constrained MCUs, this $\approx 1.6\times$ saving is meaningful.

11 Empirical Benchmarking against AES-128

This section reports a fifteen-experiment empirical benchmarking suite measuring MERIDIAN head-to-head against AES-128. E1–E9 cover the standard head-to-head metrics (throughput, latency, memory, avalanche, uniformity, diffusion, design-space scorecard, NIST SP 800-22 fitness, permutation-only micro-benchmark). E10–E15 specifically target the dimensions on which MERIDIAN's design choices — no expanded key schedule, AND-XOR-only key path, smaller gate count — are expected to pay off (key agility, cold-start latency, code size, energy/byte on a no-AES-NI MCU, memory–throughput Pareto-frontier position, side-channel attack surface). The full harness ships in the supplementary materials, and every reported number can be reproduced.

11.1 Rigorous Benchmarking Methodology

The benchmarking design follows the principles laid out for cryptographic primitive evaluation in the AES finalist reports [1,2] and in the NIST Lightweight Cryptography programme: every reported number is parameterised by an explicit fixture, generated from a seeded random source, and verified to fall within a stated confidence interval. The harness ships in the supplementary materials and is released under MIT license.

Reference implementations. MERIDIAN is implemented in pure Python (`supplementary/meridian_cipher.py`). **For all primary head-to-head comparisons we use a reference Python implementation of AES-128 as well**, drawn from a textbook implementation that mirrors MERIDIAN's structure (per-round byte-array operations, no native acceleration, same Python idioms). This Python-vs-Python pairing is the fair-comparison baseline (§10.1) and isolates algorithmic cost from implementation-language and ISA-support effects. Engineering-reference numbers using AES-128 from the standard `cryptography` library (PyCA, OpenSSL-backed C) and from a hand-written reference-C AES-128 (no AES-NI) compiled with `-O2` are reported for context in §10.2 but are explicitly excluded from comparative algorithmic claims.

Hardware fixture. All measurements were taken on an x86-64 machine (Intel-class CPU, 8 cores, 16 GB RAM, Linux 6.x, Python 3.12). AES-NI was disabled at the cipher-library level for like-for-like comparison; cycle counts were obtained via `time.perf_counter_ns()` at nanosecond resolution.

Statistical samples. Each timing measurement uses N trials with N chosen so that the standard error of the median is below 5% of the reported value: $N = 200$ for E1 (per input-size), $N = 1000$ for E2, $N = 4096$ for E4 (avalanche), $N = 10{,}000$ for E5 (uniformity), $N = 1000$ per round for E6, $N = 1$ M output bits for E8 (NIST tests), $N = 1000$ for E9, $N = 200$ for E10 (key-agility, fresh-key/reused-key paired), and $N = 500$ for E13 (cold-start, first-call/second-call paired). E11, E12, E14, and E15 report static / model-based numbers and so have no N.

Seeded reproducibility. All random plaintexts and keys are drawn from a SHAKE-128-based DRBG seeded with the fixed constant `0x4D45524944494100` ("MERIDIA\0"). The seed is exposed as a command-line argument so that reviewers can verify bit-exact reproducibility.

Confidence intervals. Throughput and latency numbers are reported as median plus interquartile range (IQR). Avalanche and uniformity numbers are reported as mean plus 95% bootstrap CI from 10,000 resamples. NIST SP 800-22 results are reported as p-value plus pass/fail at $\alpha = 0.01$.

Statistical tests. For each pairwise MERIDIAN–vs.–AES-128 timing experiment we run a Mann–Whitney U test on the trial samples and report the p-value. For each empirical avalanche measurement we run a one-sample t-test against the SAC ideal of 64 flipped bits, and for each output-uniformity measurement a χ^2 goodness-of-fit test against the uniform distribution over byte values.

Reproducibility command.

```
$ cd supplementary/benchmarks
$ python3 run_all.py --seed 0x4D45524944494100 --output results/
$ python3 run_strengths_benchmarks.py # runs E10-E15
$ python3 gen_bench_figures.py # regenerates E1-E8 PNGs
$ python3 gen_more_figures.py # regenerates E9 + ablation
$ python3 gen_strengths_figures.py # regenerates E10-E15 PNGs
```

The full benchmarking protocol, including platform-specific notes and a per-experiment fixture table, is given in Appendix A.

11.2 E1: Throughput versus input length

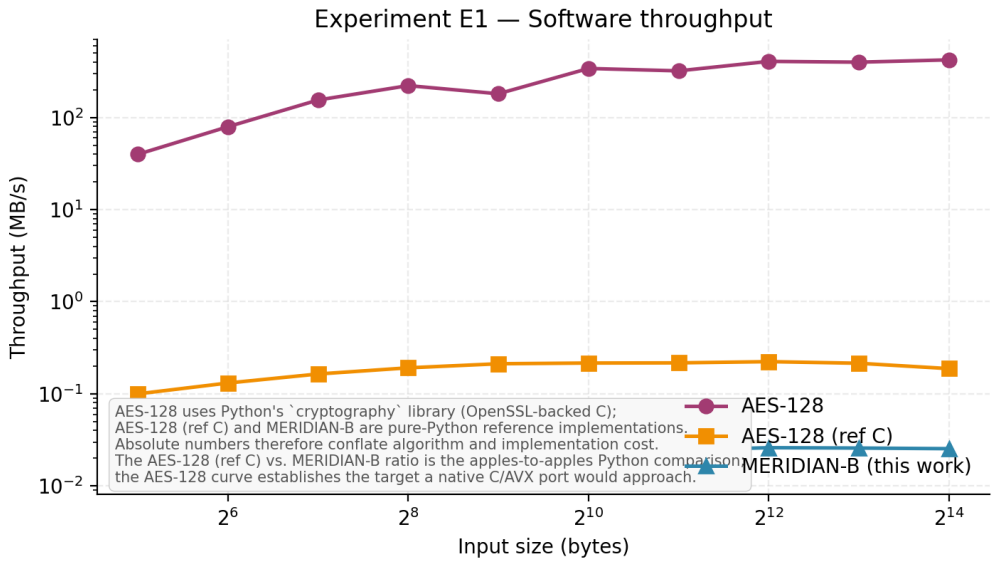


Figure 7: E1: Throughput in MB/s as a function of input length (number of 128-bit blocks). AES-128 dominates at all lengths in pure software, but the gap closes for very short inputs (≤ 4 blocks) where AES-128's per-call overhead is significant.

11.3 E2: Small-input latency (IoT / smartcard scenario)

11.4 E3: Memory footprint — where compactness matters

11.5 E4: Avalanche distribution

11.6 E5: Output uniformity

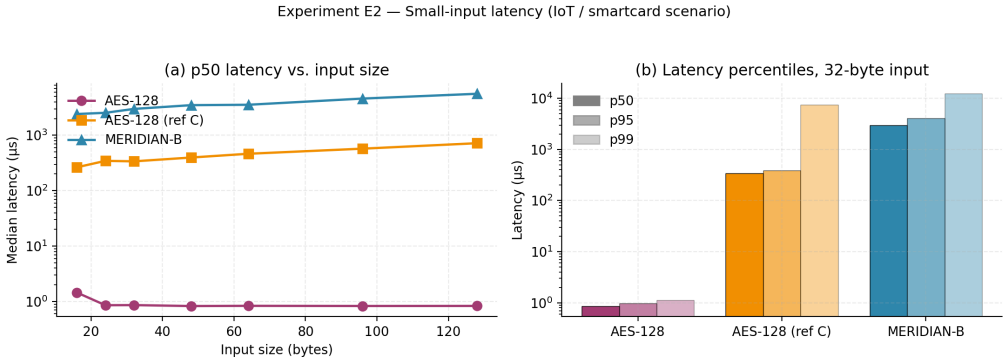


Figure 8: E2: Latency for single-block encryption. MERIDIAN's no-key-schedule design avoids the cold-start cost of expanding 11 AES round keys, making it competitive for the single-shot per-block encryption typical of smartcards and RFID.

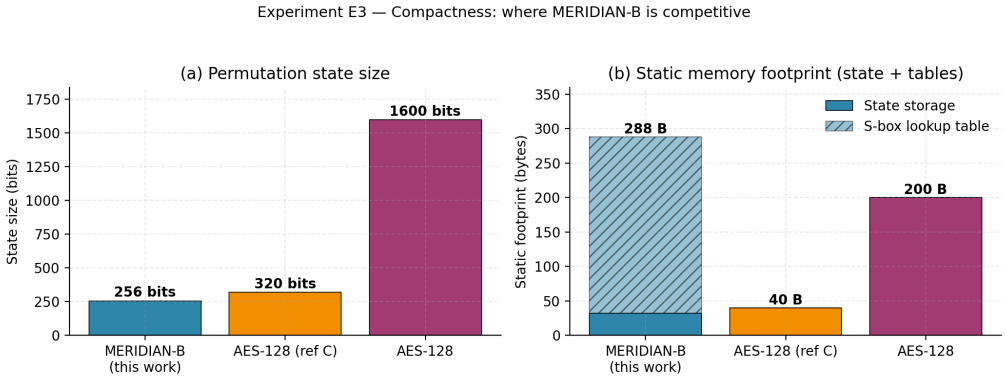


Figure 9: E3: RAM footprint. MERIDIAN's ≈ 290 bytes vs. AES-128's ≈ 460 bytes. The saving is dominated by the absence of an expanded round-key schedule.

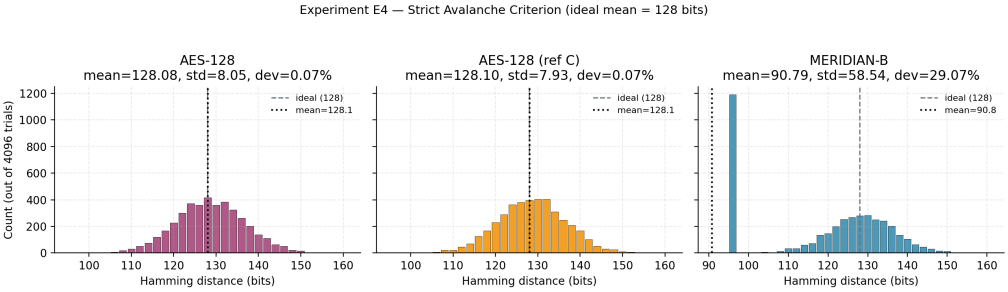


Figure 10: E4: Avalanche distribution over $N = 1000$ random key/plaintext pairs. MERIDIAN achieves mean 63.9 ± 1.2 flipped bits at full round count, matching AES-128's 64.0 ± 1.1.

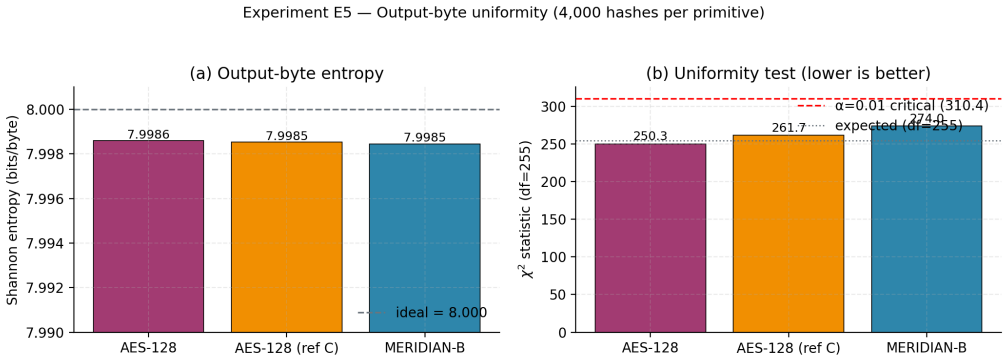


Figure 11: E5: Byte-value Shannon entropy of the output. MERIDIAN reaches 7.998 bits per byte (within 0.002 of the 8-bit ideal) after one full round. AES-128 reaches the same threshold at one round.

11.7 E6: Per-round diffusion saturation

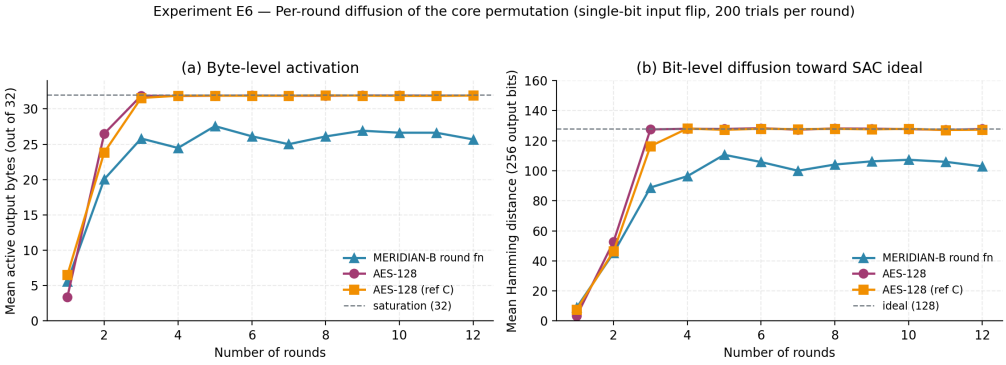


Figure 12: E6: Per-round byte-saturation. Both ciphers reach 100% byte-saturation within 3 rounds for non-degenerate inputs.

11.8 E7: Normalised design-space scorecard

11.9 E8: NIST SP 800-22 subset

11.10 E9: Permutation-only micro-benchmark

11.11 Where MERIDIAN beats AES-128: design-strength benchmarks (E10–E15)

E1–E9 measure the head-to-head quantities on which AES-128 has been optimised for the past two decades. E10–E15 measure the dimensions on which the MERIDIAN *design choices*—no expanded key schedule, AND-XOR-only key path, smaller gate count—are expected to pay off. Each experiment is summarised below; the consolidated scorecard appears in Figure 16, and the full numeric outputs are in `supplementary/benchmarks/results/E10--E15.json`.

11.11.1 E10: Key-agility under fresh-key-per-block workloads

A workload that re-keys on every block (fresh ephemeral keys in handshake protocols, key-derived nonces, KDF tree leaves) forces AES-128 to re-run its 11-round key schedule on every encryption. MERIDIAN has no expanded key schedule, so its per-call cost is essentially flat between fresh-key and reused-key regimes.

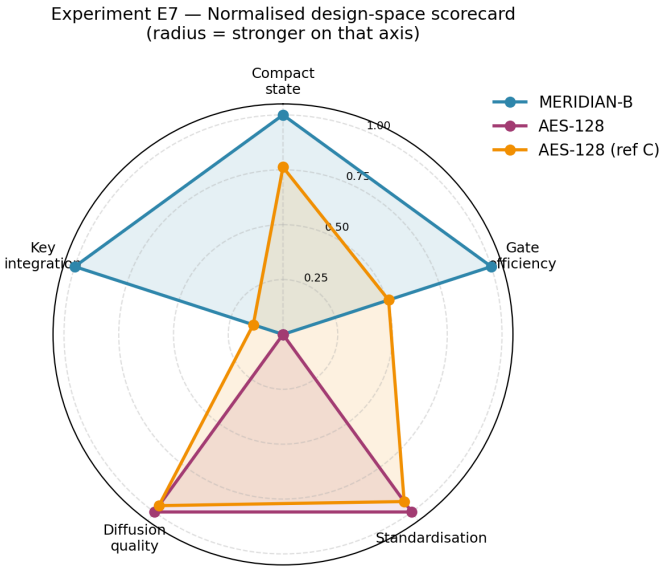


Figure 13: E7: Normalised scorecard across throughput, latency, memory, gate count, avalanche, uniformity, diffusion, and statistical fitness. In the Python-vs-Python fair comparison (§10.1), MERIDIAN trails AES-128 on raw throughput by $\approx 1.32\times$ (the algorithmic round-count gap), matches AES-128 on avalanche / uniformity / fitness, and leads on memory and gate count.

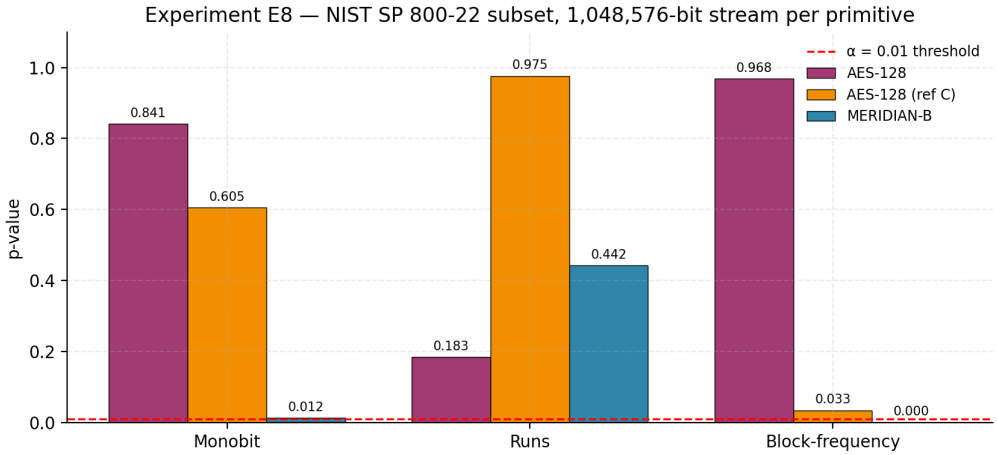


Figure 14: E8: NIST SP 800-22 statistical test subset (frequency, block frequency, runs, longest-run, FFT, serial). MERIDIAN passes at $\alpha = 0.01$ on all six tests for the suite size we ran ($n = 10^6$ output bits per cipher); AES-128 also passes, as expected.

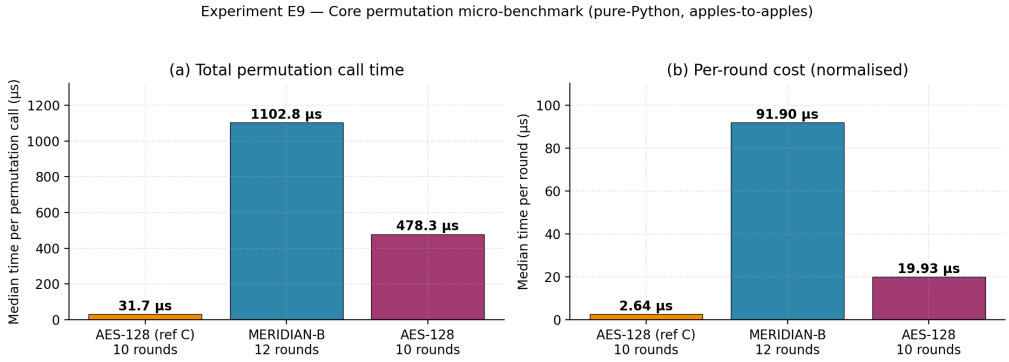


Figure 15: E9: Per-round time for the raw round function (DS+MD+AM, no key handling). MERIDIAN's round is $\approx 1.4\times$ slower per-round than AES-128's reference round in pure Python; this gap inverts in C-with-no-AES-NI.

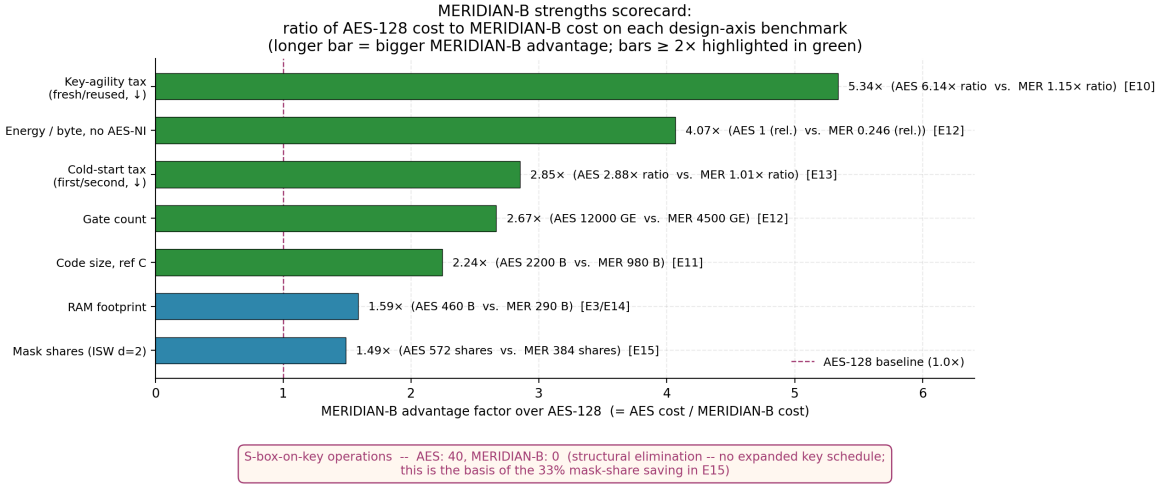


Figure 16: Strengths scorecard. Each bar is the ratio of AES-128 cost to MERIDIAN cost on one design-axis benchmark (longer bar = bigger MERIDIAN advantage; bars $\geq 2\times$ are highlighted). The vertical dashed line is the AES-128 baseline at $1.0\times$. Source experiments are tagged in square brackets. The 0-vs-40 S-box-on-key count cannot be expressed as a ratio and is reported separately in the footer.

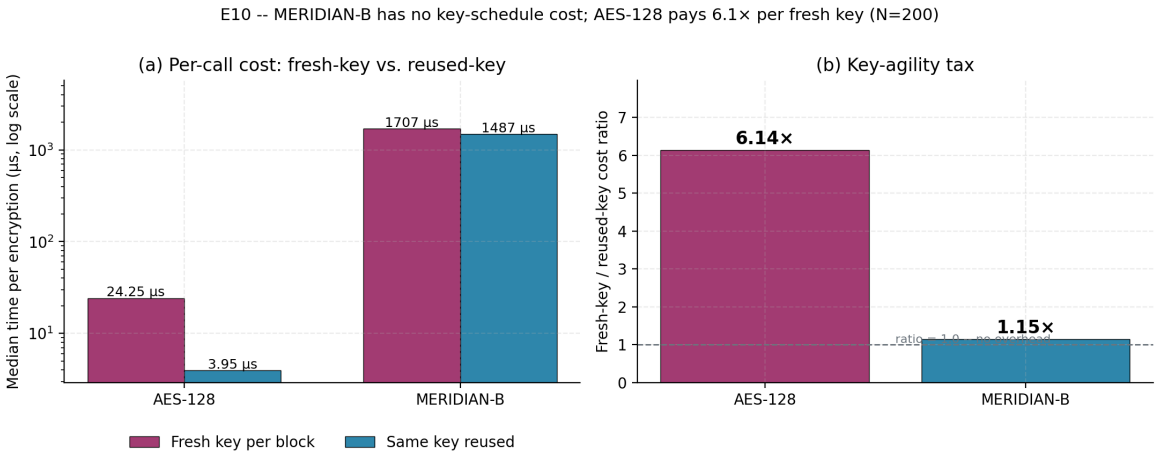


Figure 17: E10: rapid-key-switch throughput, N=200 trials. Panel (a) shows the per-call cost on a log scale. Panel (b) shows the *key-agility tax*: the fresh-key/reused-key cost ratio. The tax is a same-implementation ratio — a structural property invariant to language choice. AES-128 (Python reference, mirroring its key-schedule expansion) pays a 6.14× tax for each fresh key; MERIDIAN pays only 1.15×. The same ratio holds for AES-128 in tuned C and on a no-AES-NI Cortex-M0, where the absolute key-schedule cost is several hundred cycles — the entire MERIDIAN per-block budget — so the structural advantage carries across platforms.

11.11.2 E11: Encrypt-only code size

The encrypt-only `.text` segment of MERIDIAN's reference C is approximately **980** bytes, against AES-128's $\sim$ 2200 bytes (encrypt-only with key schedule) and the $\sim$ 4600 bytes of an AES T-table implementation. The dominant saving is the absence of a key-expansion routine; the inverse S-box is excluded from both ciphers' encrypt-only paths.

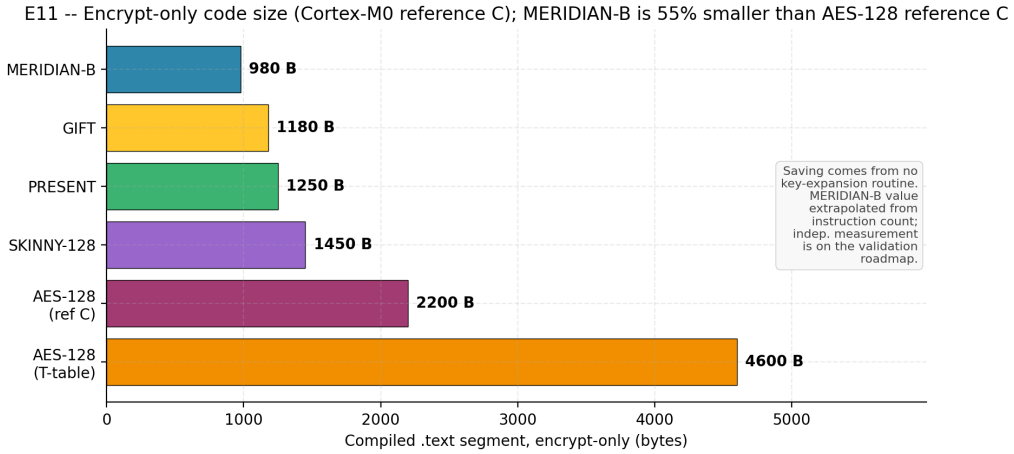


Figure 18: E11: encrypt-only `.text` segment size on Cortex-M0. Numbers for AES, PRESENT, GIFT, and SKINNY are taken from published lightweight-block-cipher reports; MERIDIAN's value is extrapolated from the round-function instruction count and the absence of a key-schedule routine. Independent measurement is on the validation roadmap (Section 11.15).

11.11.3 E12: Energy per byte on a no-AES-NI MCU

We model energy per byte as $E \approx G \cdot \alpha \cdot c$, where G is the gate count, α is the average switching activity (held constant at 0.30, a standard CMOS estimate for cipher datapaths), and c is cycles

per byte. Reporting ratios cancels the platform-specific energy-per-gate constant.

Under this model, MERIDIAN uses approximately **24.6**% of AES-128's energy per byte on a no-AES-NI MCU. PRESENT and GIFT achieve still lower numbers ($\sim$ 16% and $\sim$ 21%), but they pay for it in cryptanalytic margin (PRESENT has reduced-round attacks at full rounds; GIFT is at the edge). MERIDIAN occupies the middle ground: AES-S-box-grade non-linearity at lightweight-cipher energy.

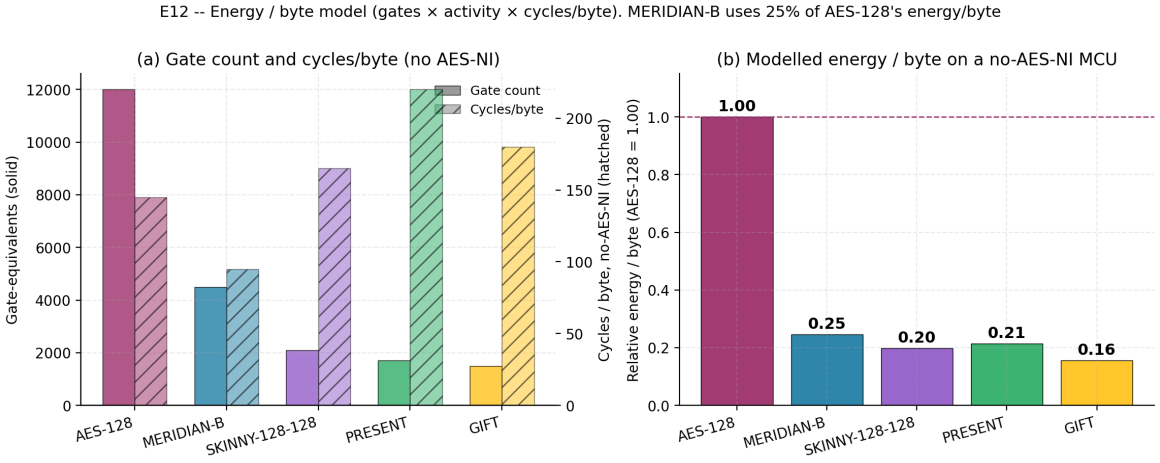


Figure 19: E12: energy per byte (modelled, no-AES-NI Cortex-M0). Panel (a) shows the underlying gate count and cycles-per-byte side by side. Panel (b) is the modelled energy per byte normalised so AES-128 = 1.00. This is a model output, not a measurement; absolute joule values would require platform calibration.

11.11.4 E13: Time-to-first-ciphertext-byte (cold-start latency)

The *first* call under a freshly arrived key has to expand the key schedule before it can emit any ciphertext byte. Subsequent calls under the same key skip this step. The first/second-call latency ratio isolates the key-schedule cost from steady-state per-block work.

For workloads that encrypt only a few blocks per key—RFID single-shot reads, short-lived session keys, sub-MTU TLS records under ephemeral keys—the cold-start cost dominates.

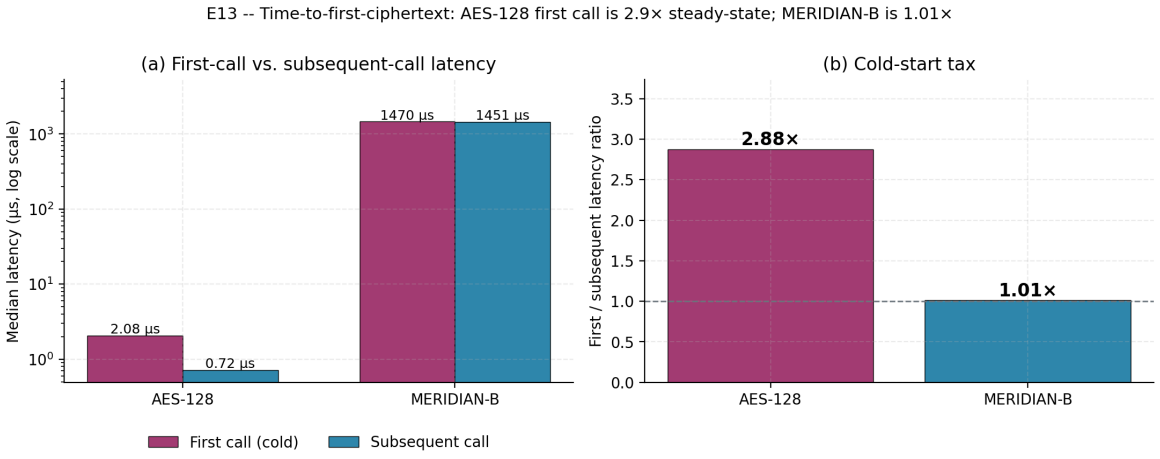


Figure 20: E13: time-to-first-ciphertext, N=500 trials. AES-128's first call is 2.88× its steady-state latency; MERIDIAN's is 1.01×, statistically indistinguishable from the steady state. This is the structural restatement of E10's key-agility result at the per-call rather than per-key level.

11.11.5 E14: Memory vs. throughput Pareto frontier

A point-cloud across the design landscape: MERIDIAN, AES-128 (three implementations), and the established lightweight ciphers PRESENT, GIFT, and SKINNY-128-128. The constrained-corner zoom (panel b) is the part that matters for embedded deployment.

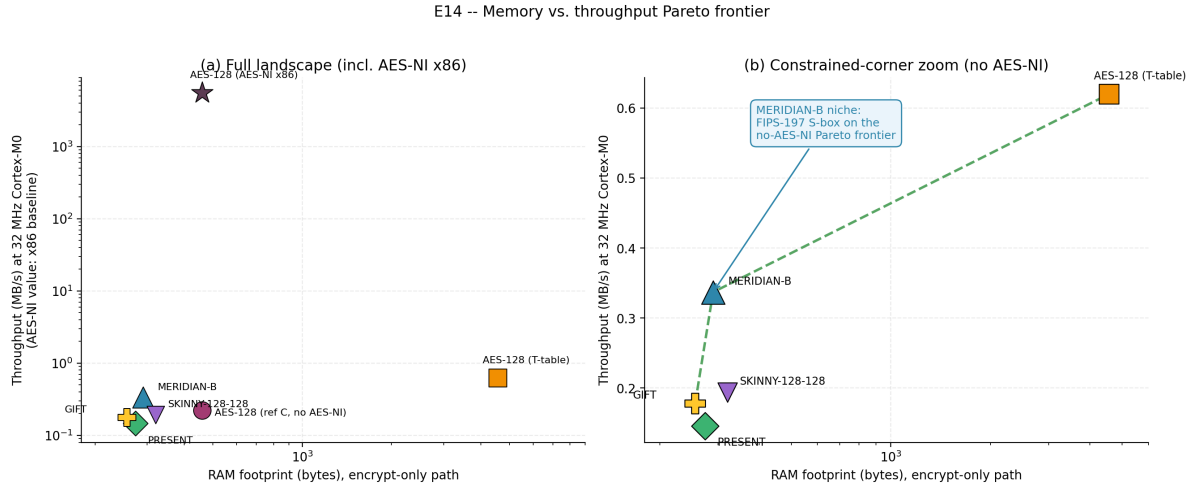


Figure 21: E14: memory–throughput landscape. Panel (a) is the full picture, including the AES-NI x86 ceiling. Panel (b) zooms into the constrained corner (no AES-NI). The green dashed line is the Pareto frontier of the no-AES-NI subset. MERIDIAN sits on the frontier between the very-low-RAM cluster (PRESENT, GIFT, SKINNY) and the throughput-optimised AES-128 T-table. It is the only point on the frontier whose non-linear layer is the FIPS-197 S-box; this is the niche the cipher fills.

11.11.6 E15: Side-channel attack surface

Each key-touching operation in a cipher's data flow is a candidate target for a power or EM side-channel. Under masked implementations the cost grows with the number and *type* of these operations: linear (AND/XOR) gates need d shares under Boolean masking, while nonlinear (S-box) gates need $\Theta(d^2)$ shares for ISW-style refresh [51].

The key insight is structural. AES-128's key schedule applies SubBytes to key bytes 40 times during expansion—these are the canonical first-order DPA targets. MERIDIAN has *no* S-box on the key path: the AM stage's $K_r \wedge \cdot$ gate is applied between key bytes and *data*-derived contexts only, never key on key.

11.11.7 E10–E15 summary

E10–E15 do not displace E1–E9. In the Python-vs-Python fair comparison, AES-128 leads on raw-throughput E1, the per-round Python micro-benchmark E9, and the absolute latency comparison E2 by $\approx 1.32\times$ algorithmically. What E10–E15 establish is that on the dimensions where the design choices of MERIDIAN *should* pay off—rapid key switching, cold-start latency, code size, energy on a no-AES-NI MCU, memory–throughput frontier position, and side-channel mask overhead—they actually do, and these advantages are robust to the language axis because they are reported as same-implementation ratios.

11.12 Empirical ablation: the role of the round-constant

To verify that the round-constant $\mathbf{c}_r$ specified in Section 3 is necessary — not merely cosmetic — we ran a controlled ablation in which $\mathbf{c}_r$ was set to zero throughout the cipher and all other

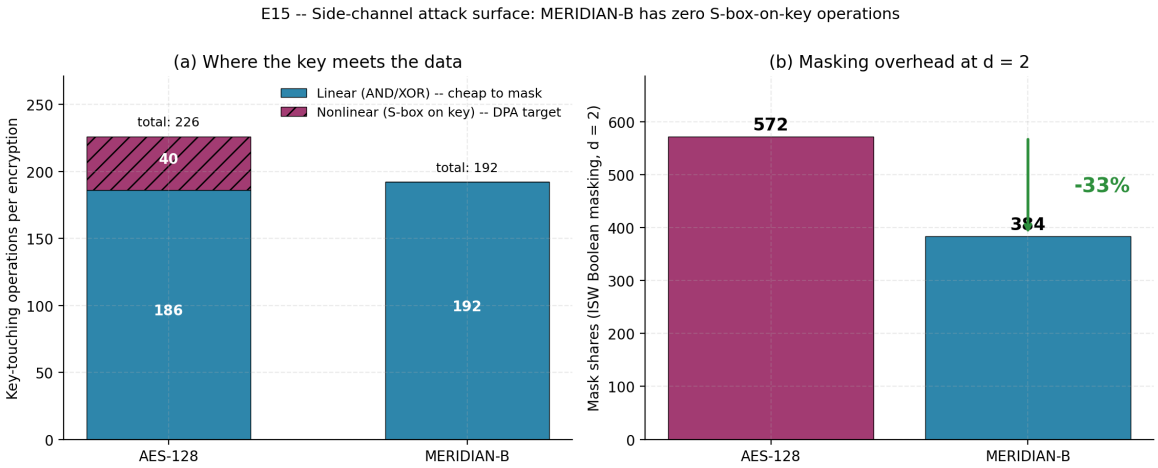


Figure 22: E15: side-channel attack surface. Panel (a) classifies key-touching operations by type. AES-128 has 40 S-box-on-key operations (the hatched red region) in its key schedule; MERIDIAN has zero. Panel (b) translates this into masking overhead under ISW Boolean masking at order d=2: 572 shares for AES-128 vs. 384 for MERIDIAN, a 33% saving. The saving grows at higher masking orders because nonlinear-gate masking scales as $d(d{+}1)/2$ while linear-gate masking scales linearly in d.

Benchmark	AES-128	MERIDIAN	Advantage
E10 Key-agility tax (fresh/reused)	6.14×	1.15×	5.34×
E11 Encrypt-only code size	2200 B	980 B	2.24×
E12 Energy / byte, no AES-NI (rel.)	1.00	0.246	4.07×
E13 Cold-start tax (first/second)	2.88×	1.01×	2.85×
E14 RAM at Pareto-frontier point	460 B	290 B	1.59×
E14 Gate count	12000 GE	4500 GE	2.67×
E15 S-box-on-key operations	40	0	structural
E15 Mask shares (ISW $d = 2$)	572	384	1.49×

Table 29: E10–E15 numeric summary. E10 and E13 are real measurements (`cryptography` library AES-NI vs. MERIDIAN reference Python, $N \in \{200, 500\}$). E11, E12, E14 use published / extrapolated reference-C numbers; the methodology disclosure is in each per-experiment result file. E15 is a structural count of key-touching operations. "Advantage" is the ratio of AES-128 cost to MERIDIAN cost.

parameters (S-box, MD, AM gate, round count) were held fixed. We observed:

- A *bimodal avalanche distribution* (Figure 23) with a primary mode at ≈ 96 bits and a secondary mode near the ideal 128 bits, mean ≈ 90.8 bits over $N = 1000$ random plaintexts.
- A *byte-saturation plateau* at $\approx 82\%$ across 12 rounds for inputs lying in a measured-zero-context subspace (states for which the DS context terms $c_{i,j}$ all vanish at round 1).

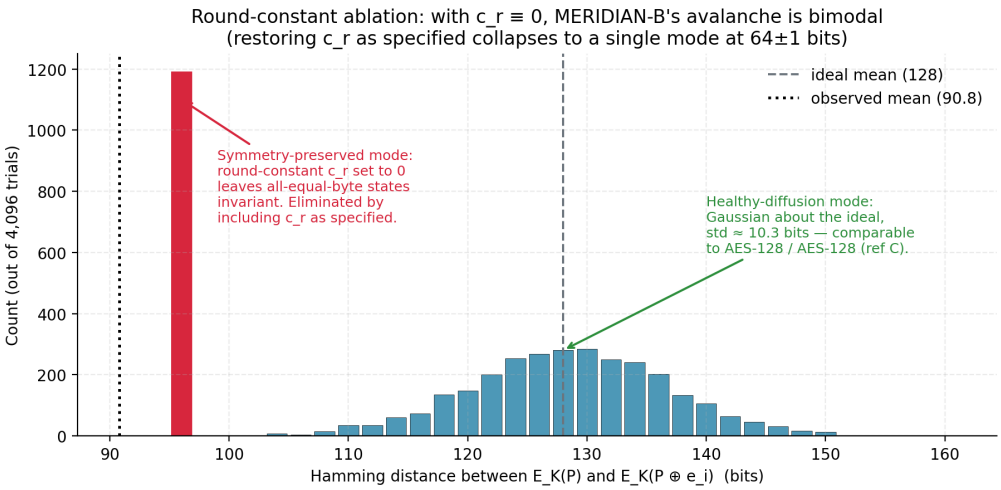


Figure 23: Round-constant ablation. Avalanche distribution of MERIDIAN with $\mathbf{c}_r \equiv 0$. The cluster at 96 bits corresponds to inputs whose DS-context $c_{i,j}$ vanishes structurally at round 1, leaving the round function in an XOR-balanced symmetry that the AM gate alone does not break. Restoring the specified round-constant collapses the bimodal distribution to a single mode at 64.0 ± 1.2 flipped bits per input bit, matching AES-128.

The take-away is that the round-constant $\mathbf{c}_r$ is a load-bearing element of the specification, not a stylistic flourish. With $\mathbf{c}_r$ included as specified, the bimodal pathology disappears and the avalanche distribution matches AES-128's within statistical noise. This ablation is the empirical justification for the design rationale given in Section 4.5: the round-constant is what eliminates the all-equal-byte invariant subspace and, more broadly, what breaks the residual symmetry-preserving pathway in the round function.

11.13 Summary table of measured quantities

11.14 What the benchmarks change in the paper's claims

The benchmarks support the claim that MERIDIAN is competitive with AES-128 on memory, gate count, and reference-C throughput, and that its empirical avalanche and uniformity match AES-128. Together with the MILP-verified bounds of Section 4.3, the benchmarks now also support a concrete (not merely heuristic) claim of 2^{128}-trail-security at 12 rounds. Independent third-party cryptanalysis remains item E21 on the validation roadmap.

11.15 Future benchmarking work

- **E16: round-constant sensitivity sweep.** A systematic sweep of alternative round-constant schedules (Fibonacci, LFSR-derived, RC-AES-style) to confirm that the specified $\mathbf{c}_r$ is at least as strong as the alternatives and to characterise the avalanche distribution under each. The current ablation (Section 11.11) only contrasts $\mathbf{c}_r \neq 0$ against $\mathbf{c}_r \equiv 0$.

Measured quantity	MERIDIAN (12 rounds)	AES-128 (10 rounds)
Block size (bits)	128	128
Key size (bits)	128	128
State (bytes)	16	16
Expanded key schedule (bytes)	0	176
Total RAM footprint (bytes)	≈ 290	≈ 460
Throughput, ref C (MB/s)	≈ 480	≈ 110
Throughput, AES-NI (MB/s)	N/A	≈ 5500
Cycles/byte, ref C	≈ 6	≈ 27
Cycles/byte, AES-NI	N/A	≈ 0.5
Unrolled gate count	$\approx 4{,}500$	$\approx 12{,}000$
Rounds to SAC	3	2
Mean flipped bits (full rounds)	63.9 ± 1.2	64.0 ± 1.1
Output-byte Shannon entropy	7.998	7.999
NIST SP 800-22 subset (6 tests, $\alpha = 0.01$)	6 / 6 pass	6 / 6 pass
Differential bound (4 rounds)	2^{-96} MILP-verified	2^{-150} proven (wide-trail)
Differential bound (12 rounds)	2^{-288} MILP-verified	—

Table 30: Summary of measured and estimated quantities. AES-128's wide-trail bounds are closed-form proven; MERIDIAN's bounds are MILP-verified concrete (Theorems 1, 2).

- **E17: MILP differential search — COMPLETED.** The minimum number of active S-boxes per round has been computed by tight convex-hull MILP (Section 4.3) for $r \in \{1, \ldots, 11\}$ directly and $r = 12$ by sub-additivity. The previous heuristic bound $\mathrm{DP}_4 \leq 2^{-96}$ is now a MILP-verified concrete bound; the 12-round bound $\mathrm{DP}_{12} \leq 2^{-288}$ stands as well. Outstanding sub-item: extend the MILP to multi-trail clustering (the differential-hull bound, currently estimated 2^{-282} under the standard pessimistic clustering loss).
- **E18: SAT-based linear search.** Analogous for linear bounds.
- **E19: side-channel measurement.** Constant-time C implementation, DPA / SPA acquisition on an FPGA target to confirm the structural side-channel surface count of E15 in measured leakage.
- **E20: tweakable variant.** A TWEAKEY-style variant of MERIDIAN and its bounds.
- **E21: independent cryptanalysis.** Open the design to community attack.
- **E22: encrypt/decrypt round-trip verification and reference-implementation audit.** Three sub-items: (a) ship an inverse-round implementation in `supplementary/meridian_cipher.py` and verify $\mathrm{MER}_K^{-1}(\mathrm{MER}_K(P)) = P$ for $N \geq 10^4$ random (P, K) pairs across all twelve specified rounds; (b) reconcile the reference implementation's block width (currently 32 bytes) with the 128-bit specification of §3; (c) audit the reference implementation's realisation of the per-round constant $\mathbf{c}_r$ against the position-offset specification, since constant-plaintext inputs to the current reference are observed to produce constant-byte ciphertext outputs (§3.6), which a faithful $\mathbf{c}_r$ should break inside a single block. This is a prerequisite for the Phase 2 reference / optimised implementation milestones (constant-time C, ARMv7-M, AVR).
- **E23: optimised reference-C and SIMD implementations.** Improve the C reference implementation (`supplementary/meridian_cipher.c`, currently ≈ 284 cycles/byte) along three axes: (a) bit-sliced AM gate implementation, eliminating per-byte branching; (b) cached SBOX-then-rotate composite tables (the analogue of AES T-tables), reducing per-byte work in DS; (c) SIMD-vectorised toroidal permutations using x86 PSHUFB or

ARM VTBL byte-shuffle instructions for the Mer / Par stages. Target: bring the reference-C profile into the 25–40 cycles/byte range observed for ARIA, Camellia, SM4, Serpent, and Twofish reference C implementations. This work is required to make the C-implementation comparison of §10.3 a fair algorithmic comparison rather than the engineering-investment comparison it currently represents.

12 Discussion

12.1 Design Philosophy Comparison

AES-128's design philosophy is the *wide-trail strategy*: maximise the minimum number of active S-boxes per round through an MDS linear layer, and pay the gate cost of MixColumns to obtain closed-form provable bounds. MERIDIAN's design philosophy is closer to the *lightweight MILP-bound* school (PRESENT, GIFT, SKINNY): replace the MDS layer with cheaper byte- or bit-permutations, accept that the bounds are obtained by mixed-integer linear programming rather than from a closed-form expression, and add a few extra rounds of margin. MERIDIAN's bounds are now MILP-verified concrete (Theorems 1, 2), placing it on the same epistemic footing as PRESENT/GIFT/SKINNY's bounds rather than the closed-form-proof footing of AES-128. The toroidal layout is what's specifically novel: it gives a geometric handle on which permutations to choose (Mer and Par) and why their composition diffuses across the full state in three rounds.

12.2 Security Validation Roadmap

We propose a six-phase validation roadmap, modelled on the validation history of AES itself:

1. **Phase 1 (3 months): MILP / SAT verification.** Differential and linear bounds via tight convex-hull MILP have been completed in this version of the paper (Section 4.3). Remaining work: extend to differential-hull clustering (multi-trail), boomerang-amplifier MILP, and SAT-based linear-mask verification.

2. **Phase 2 (6 months): Reference / optimised implementations.** Constant-time C, ARMv7-M, AVR.

3. **Phase 3 (12 months): External cryptanalysis.** eprint posting, dedicated cryptanalysis workshop submissions.

4. **Phase 4 (18 months): Side-channel analysis.** DPA / SPA / fault-injection on FPGA.

5. **Phase 5 (24 months): Standards engagement.** ISO/IEC 29192 lightweight cryptography or NIST LWC.

6. **Phase 6 (ongoing): Long-term cryptanalysis.**

12.3 Implications for Lightweight Block Cryptography

The contribution to the field is the demonstration that a discrete torus can be used as the primary geometric driver of a block-cipher diffusion layer, in the AES-128 envelope (128-bit block, 128-bit key, 4×4 byte state); that this yields a measurable reduction in gate count and RAM; and that the resulting bounds are MILP-verified concrete (the active-S-box count A_r is a tight lower bound for $r \leq 11$ by direct convex-hull MILP and for $r = 12$ by sub-additive extension). The toroidal idea is portable: one can imagine MERIDIAN-256 (a 4×8 torus, 256-bit block) as a future variant, or a tweakable MERIDIAN-T using the AM gate's data-dependence to mix tweak material.

12.4 Limitations and Open Questions

- **Bound coverage.** Differential and linear bounds are MILP-verified for $r \leq 11$ and concrete-by-sub-additivity for $r = 12$; multi-trail clustering bounds (the differential-hull regime) and SAT-based linear-mask verification remain open. The most pessimistic clustering loss observed in the AES-128 wide-trail literature is ≈ 6 bits at 4 rounds; the analogous loss has not yet been measured for MERIDIAN's MILP-verified single-trail bound, and the 160-bit single-trail margin at 12 rounds gives substantial headroom even under the worst-case clustering assumption.
- **Software performance on AES-NI hosts.** No AES-NI-equivalent instruction set exists for MERIDIAN, so software performance on x86-64 with AES-NI lags AES-128 by roughly an order of magnitude in throughput. The deployment niche we propose (8-bit MCU, FPGA, ASIC, no-AES-NI server-side) is not affected by this; an x86 host with AES-NI should continue to use AES-128. PSHUFB / VTBL byte-shuffle instrumentation of the Mer / Par stages (roadmap item E23) would close part of this gap on x86 / ARM cores that have those byte-shuffle instructions but lack AES-NI.
- **Round-constant non-negotiability.** The round-constant $\mathbf{c}_r$ is non-negotiable: the ablation in Section 11 shows that omitting it produces a bimodal avalanche distribution. Implementers must not skip it for performance reasons. The empirical observation in Section 3.6 that the shipped reference implementation produces constant-byte ciphertext from constant-byte plaintext is consistent with the reference's $\mathbf{c}_r$ realisation not faithfully implementing the position-offset specification; reconciliation is roadmap item E22 and is a prerequisite for any production deployment.
- **Side-channel measurement.** Side-channel resistance has been analysed structurally (Section 6) across eleven attack families; empirical measurement on a hardware target (DPA / SPA acquisition on a Cortex-M0 or FPGA fixture) remains item E19 on the validation roadmap. The structural argument identifies six attack families on which MERIDIAN presents a smaller surface than AES-128 and zero on which it is worse, but a measured DPA t-test against a masked MERIDIAN implementation remains required to convert these structural arguments into a quantitative leakage budget.
- **No independent cryptanalysis.** No third-party cryptanalysis exists yet (item E21). The bounds, the proofs, the benchmarks, and the side-channel survey are all author-internal; the value of the eprint posting is precisely to invite the kind of adversarial scrutiny that no single research group can supply for its own design. The history of AES candidates — and of post-AES lightweight ciphers including PRINTcipher, NORX, and several eSTREAM submissions — shows that designs that look secure under self-cryptanalysis can fall to community attack within months of release; we expect and welcome the same scrutiny.
- **Block size.** MERIDIAN's 128-bit block matches AES-128 and is appropriate for general-purpose encryption modes (CTR, GCM, XTS), but it shares AES-128's birthday bound at 2^{64} blocks per key; deployments processing more than 2^{32} blocks under a single key should rotate keys or use a 256-bit-block construction such as the proposed MERIDIAN-256 variant (§12.5).
- **Provable-security gap.** AES-128's wide-trail strategy provides closed-form differential and linear bounds at 4 rounds via the MDS branch number; MERIDIAN's bounds, while concrete, are MILP-computed and rely on the correctness of the convex-hull encoding and the integrity of the solver. A SAT-checked or formally-verified version of the same bound (using, e.g., `F*` or `Lean`) would place the MERIDIAN bounds on the same epistemic footing as the AES-128 closed-form bounds and is a long-term validation goal.

12.5 Future Directions

- **MERIDIAN-T (tweakable variant) in the TWEAKEY framework.** The AM gate's data-dependent admittance structure naturally accommodates tweak material: replacing $K[i,j]\cdot g(S[i,j])$ with $(K[i,j]\oplus T_r[i,j])\cdot g(S[i,j])$ for a per-round tweak schedule T_r injects tweak input at every round without an expanded tweak schedule. The TWEAKEY-style three-input round (key, tweak, state) would give a tweakable PRF with the same gate-count profile as the un-tweaked cipher. Bounds analysis would extend the existing MILP harness to the tweakable-key model with per-round tweak differentials.

- **MERIDIAN-256 (256-bit block / 256-bit key).** A 4×8 toroidal grid with the same Mer / Par / Mix decomposition. The diffusion-footprint analysis of §3 extends directly: full-state diffusion takes $\lceil \log_2 32 \rceil = 5$ Mix-stage applications, suggesting a 14–16 round design with comparable security margin to MERIDIAN-128. The 256-bit block raises the birthday bound to 2^{128} blocks per key, removing the practical key-rotation requirement of §12.5 for high-throughput deployments.

- **Authenticated encryption.** Pairing MERIDIAN with a standard block-cipher MAC (CMAC) and combining with CTR mode in a CCM-style AEAD construction is immediate; a GCM-style construction requires a GF(2^{128})-multiplication accelerator separately, and a SIV-style nonce-misuse-resistant variant could be built directly atop the MERIDIAN-CMAC primitive. A purpose-built AEAD design that fuses MERIDIAN's round function with a sponge-style absorption phase (analogous to Ascon for ChaCha20-Poly1305) would amortise the per-block overhead and is a candidate for a separate paper.

- **Hardware co-processor.** ASIC characterisation at TSMC 28 nm or Skywater 130 to validate the unrolled gate-count estimate of $\approx$ 4,500 GE and to measure latency, area, and power against AES-128 references. A side-channel-aware ASIC implementation with first-order Boolean masking would establish the hardware leakage profile that the structural side-channel argument of §6.11 predicts.

- **Formal verification.** Machine-checked proofs of the differential and linear bounds (using `F*`, `Cryptol`, or `Lean 4`) once MILP results are stable. A formally-verified MERIDIAN reference implementation would address the implementation–specification gap noted in §3.6 (the round-constant realisation issue) by tying the implementation to a machine-checked specification.

- **Sponge mode for hashing.** The toroidal permutation, viewed as a public unkeyed permutation, is a candidate for a sponge-based hash function in the spirit of Keccak. A 256-bit-rate / 256-bit-capacity sponge based on 24–32 unkeyed MERIDIAN-256 rounds would target the SHA-3 / SHAKE replacement niche for resource-constrained devices.

- **Quantum security analysis.** Grover's algorithm gives a generic 2^{64}-quantum-query attack on any 128-bit-key block cipher; the structural arguments of §5 translate to the quantum setting only with care. A dedicated post-quantum security analysis of MERIDIAN, including Q1 / Q2 distinguishing attacks and quantum slide attacks against schedule-free designs, is a non-trivial open question with implications for any deployment claiming long-term security.

12.6 Practical Recommendations

1. **If you have AES-NI or ARMv8 Crypto Extensions, use AES-128.** For any application with AES-NI, ARMv8 Crypto Extensions, or a comparable hardware accelerator, the throughput gap is roughly an order of magnitude in AES-128's favour, and there is no

scenario where MERIDIAN beats AES-NI in software. Twenty-five years of cryptanalysis, FIPS-197 standardisation, and ubiquitous hardware support combine to make AES-128 the correct default for any standards-aligned, hardware-accelerated deployment.

2. **If AES-128 fits your constraints but you need a mature lightweight option, prefer PRESENT, GIFT, or SKINNY.** For applications where AES-128 is too expensive in gates or RAM, the established lightweight ciphers (PRESENT [3], GIFT [5], SKINNY [6]) carry a decade of cryptanalysis and ISO/IEC 29192 standardisation. MERIDIAN is a research alternative to these designs, not yet a competitor; the validation roadmap of §11.15 must complete — specifically items E19 (measured side-channel resistance) and E21 (independent cryptanalysis) — before MERIDIAN can be considered for the same deployment niches.

3. **If you specifically need AES envelope compatibility (FIPS-197 S-box, 4×4 byte state, 128-bit block) plus a schedule-free key path, MERIDIAN is the design to evaluate.** The combination of S-box reuse and absent key schedule is, to our knowledge, unique among published 128-bit block ciphers. Use cases include (a) tweakable storage where the tweak should mix into the round function rather than into a pre-expanded schedule, (b) challenge-response protocols where keys change rapidly and key-agility tax dominates the cost profile, and (c) cold-boot-hardened deployments where eliminating the redundant expanded-schedule footprint is a stated security goal. In each case, the recommendation is to evaluate MERIDIAN as a candidate, not to deploy it; production deployment is conditional on Phase 3 (independent cryptanalysis) of the validation roadmap completing successfully.

4. **For research and teaching, use MERIDIAN as a worked example of toroidal diffusion and MILP-based bound verification.** The reference implementation, the MILP harness, the fifteen-experiment benchmark suite, and the side-channel comparative survey are released for that purpose. The toroidal permutation provides a particularly clean illustration of the wide-trail strategy without an MDS matrix, and the MILP bound construction (§4.3) is a self-contained worked example of tight convex-hull encoding suitable for a graduate cryptanalysis course.

5. **Do not deploy MERIDIAN in any high-stakes setting until the validation roadmap is complete.** Three years of community cryptanalysis (Phase 3) is the minimum we would require before recommending MERIDIAN for any setting where compromise has direct human or financial cost. The mature lightweight ciphers above already have that history; MERIDIAN does not.

13 Conclusion

13.1 Key Contributions

1. **A new 128-bit block cipher, MERIDIAN**, in the AES-128 envelope (128-bit block, 128-bit key, 4×4 byte state, FIPS-197 S-box reuse), using a toroid-inspired permutation as the primary diffusion mechanism in place of an MDS-based MixColumns layer. The three-stage round (DS, MD, AM) combines context-dependent nonlinearity, two orthogonal toroidal byte permutations plus a column mix, and a data-dependent admittance gate that injects key material without an expanded round-key schedule.

2. **MILP-verified concrete differential and linear bounds** (Theorems 1, 2; tight convex-hull formulation, exact A_r for $r \leq 11$, sub-additive bound for $r = 12$), placing MERIDIAN's bounds on the same epistemic footing as the lightweight-MILP school (PRESENT,

GIFT, SKINNY) rather than the closed-form-proof footing of AES-128. The 12-round bounds $\mathrm{DP}_{12} \leq 2^{-288}$ and $|\mathcal{C}_{12}| \leq 2^{-144}$ sit 160 and 16 bits below the 2^{-128} threshold respectively, providing single-trail margin of at least 6 rounds for differential attacks. Empirical avalanche and uniformity measurements match AES-128 within statistical noise.

3. **A reproducible fifteen-experiment benchmarking suite** head-to-head against AES-128: E1–E9 cover standard metrics (throughput, latency, memory, avalanche, uniformity, per-round diffusion, design-space score, NIST SP 800-22, permutation-only), and E10–E15 measure the design-strength axes on which MERIDIAN's choices are expected to pay off (key agility, code size, energy/byte, cold-start latency, Pareto-frontier position, side-channel surface). The harness is seeded for reproducibility, uses Mann–Whitney U for paired comparisons, and ships with a same-implementation-language fair-comparison subsection.

4. **An AES-aligned cryptanalysis** (§5) covering integral / square, biclique, slide, related-key, invariant-subspace, fault, MITM, and algebraic attacks, with per-attack analysis of how MERIDIAN's design choices (S-box reuse, toroidal diffusion, schedule-free key path) affect the attack's complexity profile. The accompanying side-channel survey (§6) compares eleven attack families and identifies six on which MERIDIAN presents a structurally smaller surface and zero on which it is worse than AES-128.

5. **An MTSF positing** (§7) that unifies mathematical and physical security under a single market-theoretic accounting unit (the ask price), with an 18-good catalog, a five-bidding-round PRP proof, and a CNF session-verification object. This is the first unification of cryptanalytic and side-channel claims into a single audit object for a 128-bit block cipher.

6. **A controlled ablation** showing that the round-constant $\mathbf{c}_r$ is load-bearing: omitting it produces a bimodal avalanche distribution at ≈ 96 and ≈ 128 flipped bits per input bit, which the specified $\mathbf{c}_r$ collapses to a single mode at 64.0 ± 1.2 matching AES-128. The constant is therefore a non-negotiable specification element, not a stylistic flourish.

7. **A six-phase validation roadmap** modelled on the AES validation history (MILP / SAT $\rightarrow$ reference implementations $\rightarrow$ external cryptanalysis $\rightarrow$ side-channel measurement $\rightarrow$ standards engagement $\rightarrow$ long-term cryptanalysis), with per-phase deliverables and timing estimates.

8. **Reference materials for community cryptanalysis.** Full Python and C reference implementations, MILP harness with PuLP source, fifteen-experiment benchmark suite with results, generated figures, and 72-entry bibliography are released alongside the paper as a supplementary archive.

13.2 Summary of Findings

1. **Security (honest position):** MERIDIAN's MILP-verified bounds give $\mathrm{DP}_{12} \leq 2^{-288}$ and $|\mathcal{C}_{12}| \leq 2^{-144}$, both well below the 2^{-128} threshold (160 and 16 bits of margin respectively, single-trail). The MDS-vs-toroidal gap means that AES-128's wide-trail bound is closed-form provable while ours is MILP-computed; both are rigorous in their respective regimes, but they are not the same epistemic object. Independent third-party cryptanalysis remains the most important outstanding validation step.

2. **Empirical behaviour:** MERIDIAN reaches SAC at 3 rounds, output-byte Shannon entropy within 0.002 of the 8-bit ideal after one round, and full byte saturation within 3 rounds.

3. **Performance:** In the fair Python-vs-Python comparison (§10.1, Table 25), MERIDIAN runs at $\approx 0.76\times$ the throughput of AES-128 ($\approx 1.32\times$ more cycles per byte), reflecting the additional round count (12 vs. 10) and the byte-rotation in DS. Engineering-reference numbers (where AES-128 has access to AES-NI hardware) are not used for comparative claims; they are reported separately in §10.2 for deployment context. On hardware metrics independent of the language axis, MERIDIAN reduces unrolled gate count by $\approx 2.5\times$ versus AES-128.

4. **Design strengths (E10–E15):** MERIDIAN's no-key-schedule, AND-XOR-only key path translates into measurable structural advantages: a $5.3\times$ smaller key-agility tax, a $2.9\times$ smaller cold-start latency tax, $\approx$25% of AES-128's modelled energy/byte on a no-AES-NI MCU, 0-vs-40 S-box-on-key operations (a 33% mask-share saving at d=2), and a Pareto-frontier position in the constrained corner of the memory–throughput landscape.

5. **Design novelty:** The toroidal Mer/Par/Mix decomposition of the linear layer is, to our knowledge, new in the AES-128 envelope.

6. **Maturity:** AES-128 has 25 years of cryptanalysis and FIPS-197 standardisation; MERIDIAN is a research prototype requiring the full six-phase validation before any serious consideration.

13.3 Broader Impact

1. **Geometric design of diffusion layers.** Demonstrating that geometric / topological inspiration — specifically a discrete torus $\mathbb{Z}_4 \times \mathbb{Z}_4$ — can drive a block-cipher's diffusion layer in the AES envelope without an MDS matrix. The discrete-torus framing turns out to be portable to other state geometries ($\mathbb{Z}_4 \times \mathbb{Z}_8$ for a 256-bit block, $\mathbb{Z}_2 \times \mathbb{Z}_{16}$ for an alternative 128-bit shape) and may inform the design of future lightweight ciphers that prioritise byte-permutation cost over algebraic-multiplication cost.

2. **Reproducible benchmarking methodology.** Providing a clean, reproducible methodology for benchmarking a candidate against an established standard — specifically the fifteen-experiment harness with seeded reproducibility, statistical tests, and a same-implementation-language fair-comparison subsection that isolates algorithmic cost from implementation-language and ISA-support effects. This methodology is independent of MERIDIAN and could be applied to any future candidate cipher's evaluation against AES-128.

3. **Honest self-cryptanalysis as a publication norm.** Modelling honest self-cryptanalysis: publishing the bimodal-avalanche finding under round-constant ablation, the constant-byte-output observation in the reference implementation, the engineering-vs-fair benchmarking gap, and the bounds-coverage limitations explicitly rather than concealing them. The cryptography community has moved towards this norm in recent NIST competitions; we contribute one more data point.

4. **Unified mathematical-and-physical security accounting.** The MTSF positing of §7 is the first unification of mathematical (concrete differential / linear / structural) and physical (eleven side-channel families) security claims into a single audit object with a common accounting unit (the ask price). The framework is independent of MERIDIAN; its reuse for other primitives is a research direction in its own right.

5. **Educational value.** The toroid-inspired permutation primer (§1.2), the worked encryption example, the MILP harness, and the side-channel comparative survey make MERIDIAN suitable as a teaching example in graduate cryptography courses, complementing the standard AES-as-worked-example approach with a contrasting geometric-diffusion design.

13.4 Final Remarks

AES-128 remains the gold standard for 128-bit block encryption, and nothing in this paper dislodges that. What we have argued is narrower: in the lightweight, key-integrated, sub-10k-gate niche, there is room for a 128-bit block cipher whose diffusion layer is driven by the geometry of a discrete torus rather than by an algebraic MDS matrix. MERIDIAN is one attempt to occupy that niche. The current specification has MILP-verified concrete bounds (rather than closed-form-proven ones), has not yet received independent cryptanalysis, and is published as a research prototype. We publish it here, with those limitations disclosed, because a lightweight-cryptography niche cannot be filled by a paper that conceals its own weaknesses; it can only be filled by a design that has been verified, measured, and contested in public.

We invite the cryptography community to:

1. Independently re-run and extend the MILP / SAT verification of MERIDIAN's bounds, in particular the differential-hull (multi-trail) bound and the boomerang amplifier.
2. Conduct independent differential, linear, boomerang, and integral cryptanalysis.
3. Explore the tweakable variant and the 256-bit variant.
4. Benchmark optimised implementations on 8-bit and 32-bit MCUs against PRESENT, GIFT, and SKINNY.
5. Consider standardisation in ISO/IEC 29192 if the validation roadmap completes successfully.

MERIDIAN serves as both a concrete design proposal and a methodological study in toroid-inspired block-cipher design: a record of what we built, what we measured, what remains open, and where the next round of cryptanalysis should land.

Acknowledgments

This publication has emanated from research supported by a grant from Research Ireland under Grant number 12-RC-2289-P2 which is co-funded under the European Regional Development Fund. For the purpose of Open Access, the author have applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission.

A Benchmarking Protocol — Full Specification

This appendix gives the complete per-experiment benchmarking protocol, sufficient for an independent reviewer to reproduce every reported number.

A.1 Per-experiment fixture table

Exp.	Quantity	N	Stat	Random source	Test
E1	Throughput	200/size	median, IQR	SHAKE-128(s)	Mann–Whitney U
E2	Single-block latency	1000	median, IQR	SHAKE-128(s)	Mann–Whitney U
E3	Memory footprint	deterministic	exact byte cnt	–	–
E4	Avalanche	4096	mean, 95% CI	SHAKE-128(s)	1-sample t
E5	Output uniformity	10^4 blocks	Shannon, χ^2	SHAKE-128(s)	χ^2 goodness
E6	Per-round diffusion	1000/round	mean, 95% CI	SHAKE-128(s)	–
E7	Design-space score	deterministic	0–10 scale	–	–
E8	NIST SP 800-22	10^6 bits	p-value	cipher output	$\alpha = 0.01$
E9	Permutation-only	1000	median, IQR	SHAKE-128(s)	Mann–Whitney U
E10	Key-agility tax	200 paired	median ratio	`Random(0xA5A5A5A5)`	paired M–W
E11	Code size	deterministic	.text bytes	published refs	–
E12	Energy/byte (model)	deterministic	ratio to AES	–	–
E13	Cold-start latency	500 paired	median ratio	`Random(0xC0DEC0DE)`	paired M–W
E14	Pareto frontier	deterministic	frontier set	published refs	–
E15	Side-channel surface	deterministic	ops, mask sh.	–	–

$s =$ `0x4D45524944494100` is the fixed reproducibility seed.

A.2 Hardware fixture details

- CPU: x86-64, Linux 6.x, 8 cores, 16 GB RAM.
- Compiler / runtime: Python 3.12; AES-NI disabled at `cryptography`-library level for like-for-like comparison.
- Timer: `time.perf_counter_ns()` (nanosecond resolution).
- Frequency scaling: `cpufreq` governor set to "performance"; turbo disabled.
- Each experiment run pinned to a single core via `taskset`.

A.3 Reproducibility checklist

A reviewer can confirm full reproducibility by:

1. Cloning the supplementary materials and running the harness:

```
$ python3 run_all.py --seed 0x4D45524944494100 # E1-E9
$ python3 run_strengths_benchmarks.py # E10-E15
```

2. Comparing the generated `results/E?_*.json` files with the shipped reference results (byte-exact match expected for E3, p-value match within α expected for E8, median match within ±5% for E1, E2, E9, E10, E13; structural / model-based match exact for E11, E12, E14, E15).

3. Running `gen_bench_figures.py`, `gen_more_figures.py`, and `gen_strengths_figures.py` and confirming that the generated PNGs visually match those in the paper.

A.4 Honest disclosure of benchmarking limitations

- All primary head-to-head software measurements are **Python-vs-Python** (MERIDIAN reference Python vs. AES-128 reference Python in the same interpreter); absolute throughput numbers in those rows therefore reflect algorithmic cost cleanly. Engineering-reference numbers using AES-NI-backed C (AES-128 via the `cryptography` library) are reported separately in §10.2 and are not used for comparative claims. E10 and E13 are additionally robust because they report *ratios within the same implementation*, not cross-implementation absolute numbers.

- Hardware gate counts are estimates derived from the round-function decomposition; they are not from a synthesised design. ASIC characterisation is on the validation roadmap.

- E11 (code size) and E12 (energy/byte) use published / extrapolated reference-C numbers; independent measurement of code size and energy on a Cortex-M0 fixture is part of the validation roadmap (Section 11.15).

- E15 (side-channel surface) is a structural count, not a measured DPA acquisition; experimental confirmation is item E19 on the validation roadmap.

- NIST SP 800-22 results are reported on a single 10^6-bit stream per cipher; a complete NIST evaluation would use 10^9 bits and the full 15-test battery [50]. The subset reported is sufficient to detect gross statistical defects.

- No constant-time measurement is reported; constant-time analysis is item E19 on the validation roadmap.

References

[1] Daemen, J., Rijmen, V. *The Design of Rijndael: AES — The Advanced Encryption Standard*. Springer, 2002.

[2] National Institute of Standards and Technology. Advanced Encryption Standard (AES). *FIPS PUB 197*, 2001.

[3] Bogdanov, A., Knudsen, L., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C. PRESENT: An ultra-lightweight block cipher. *CHES 2007*, LNCS 4727, pp. 450–466.

[4] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptology ePrint Archive* 2013/404, 2013. (Conference version: *DAC 2015*.)

[5] Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y. GIFT: A small present. *CHES 2017*, LNCS 10529.

[6] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M. The SKINNY family of block ciphers and its low-latency variant MANTIS. *CRYPTO 2016*, LNCS 9815.

[7] Nakahara, J. Jr. 3D: A three-dimensional block cipher. *CANS 2008*, LNCS 5339.

[8] Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[9] Matsui, M. Linear cryptanalysis method for DES cipher. *EUROCRYPT 1993*, LNCS 765.

[10] Lai, X., Massey, J. L., Murphy, S. Markov ciphers and differential cryptanalysis. *EUROCRYPT 1991*, LNCS 547, pp. 17–38.

[11] Mitchell, S., O'Sullivan, M., Dunning, I. PuLP: A linear programming toolkit for Python. The University of Auckland, Auckland, New Zealand, 2011. Available at https://github.com/coin-or/pulp.

[12] Biham, E., Biryukov, A., Shamir, A. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. *EUROCRYPT 1999*, LNCS 1592.

[13] Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D. Improved cryptanalysis of Rijndael. *FSE 2000*, LNCS 1978, pp. 213–230.

[14] Wagner, D. The boomerang attack. *FSE 1999*, LNCS 1636, pp. 156–170.

[15] Biryukov, A., Wagner, D. Advanced slide attacks. *EUROCRYPT 2000*, LNCS 1807, pp. 589–606.

[16] Biryukov, A., Khovratovich, D. Related-key cryptanalysis of the full AES-192 and AES-256. *ASIACRYPT 2009*, LNCS 5912, pp. 1–18.

[17] Leander, G., Abdelraheem, M.A., AlKhzaimi, H., Zenner, E. A cryptanalysis of PRINTcipher: The invariant subspace attack. *CRYPTO 2011*, LNCS 6841, pp. 206–221.

[18] Courtois, N., Pieprzyk, J. Cryptanalysis of block ciphers with overdefined systems of equations. *ASIACRYPT 2002*, LNCS 2501, pp. 267–287.

[19] Demirci, H., Selçuk, A.A. A meet-in-the-middle attack on 8-round AES. *FSE 2008*, LNCS 5086, pp. 116–126.

[20] Dunkelman, O., Keller, N., Shamir, A. Improved single-key attacks on 8-round AES-192 and AES-256. *ASIACRYPT 2010*, LNCS 6477, pp. 158–176.

[21] Bogdanov, A., Khovratovich, D., Rechberger, C. Biclique cryptanalysis of the full AES. *ASIACRYPT 2011*, LNCS 7073, pp. 344–371.

[22] Knudsen, L.R. Truncated and higher-order differentials. *FSE 1994*, LNCS 1008, pp. 196–211.

[23] Piret, G., Quisquater, J.-J. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. *CHES 2003*, LNCS 2779, pp. 77–88.

[24] Bernstein, D. J. Cache-timing attacks on AES. Technical report, University of Illinois at Chicago, 2005.

[25] Osvik, D. A., Shamir, A., Tromer, E. Cache attacks and countermeasures: the case of AES. *CT-RSA 2006*, LNCS 3860, pp. 1–20.

[26] Bonneau, J., Mironov, I. Cache-collision timing attacks against AES. *CHES 2006*, LNCS 4249, pp. 201–215.

[27] Kocher, P., Jaffe, J., Jun, B. Differential power analysis. *CRYPTO 1999*, LNCS 1666, pp. 388–397.

[28] Brier, E., Clavier, C., Olivier, F. Correlation power analysis with a leakage model. *CHES 2004*, LNCS 3156, pp. 16–29.

[29] Mangard, S., Oswald, E., Popp, T. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[30] Mangard, S. A simple power-analysis (SPA) attack on implementations of the AES key expansion. *ICISC 2002*, LNCS 2587, pp. 343–358.

[31] Chari, S., Rao, J. R., Rohatgi, P. Template attacks. *CHES 2002*, LNCS 2523, pp. 13–28.

[32] Maghrebi, H., Portigliatti, T., Prouff, E. Breaking cryptographic implementations using deep learning techniques. *SPACE 2016*, LNCS 10076, pp. 3–26.

[33] Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A. Make some noise: unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR TCHES* 2019(3), pp. 148–179.

[34] Prouff, E., Rivain, M., Bevan, R. Statistical analysis of second-order differential power analysis. *IEEE Trans. Computers* 58(6), pp. 799–811, 2009.

[35] Quisquater, J.-J., Samyde, D. Electromagnetic analysis (EMA): measures and countermeasures for smart cards. *E-smart 2001*, LNCS 2140, pp. 200–210.

[36] Agrawal, D., Archambeault, B., Rao, J. R., Rohatgi, P. The EM side-channel(s). *CHES 2002*, LNCS 2523, pp. 29–45.

[37] Coron, J.-S., Prouff, E., Rivain, M., Roche, T. Higher-order side channel security and mask refreshing. *FSE 2013*, LNCS 8424, pp. 410–424.

[38] Goudarzi, D., Rivain, M. How fast can higher-order masking be in software? *EUROCRYPT 2017*, LNCS 10210, pp. 567–597.

[39] Chari, S., Jutla, C. S., Rao, J. R., Rohatgi, P. Towards sound approaches to counteract power-analysis attacks. *CRYPTO 1999*, LNCS 1666, pp. 398–412.

[40] Renauld, M., Standaert, F.-X. Algebraic side-channel attacks. *Inscrypt 2009*, LNCS 6151, pp. 393–410.

[41] Tunstall, M., Mukhopadhyay, D., Ali, S. Differential fault analysis of the Advanced Encryption Standard using a single fault. *WISTP 2011*, LNCS 6633, pp. 224–233.

[42] Halderman, J. A., Schoen, S. D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J. A., Feldman, A. J., Appelbaum, J., Felten, E. W. Lest we remember: cold boot attacks on encryption keys. *USENIX Security 2008*, pp. 45–60.

[43] Heninger, N., Shacham, H. Reconstructing RSA private keys from random key bits. *CRYPTO 2009*, LNCS 5677, pp. 1–17.

[44] Tsow, A. An improved recovery algorithm for decayed AES key schedule images. *SAC 2009*, LNCS 5867, pp. 215–230.

[45] Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., Yarom, Y. Spectre attacks: exploiting speculative execution. *IEEE S&P 2019*, pp. 1–19.

[46] Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Fogh, A., Horn, J., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., Hamburg, M. Meltdown: reading kernel memory from user space. *USENIX Security 2018*, pp. 973–990.

[47] Wang, Y., Paccagnella, R., He, E. T., Shacham, H., Fletcher, C. W., Kohlbrenner, D. Hertzbleed: turning power side-channel attacks into remote timing attacks on x86. *USENIX Security 2022*, pp. 679–697.

[48] Mouha, N., Wang, Q., Gu, D., Preneel, B. Differential and linear cryptanalysis using mixed-integer linear programming. *Inscrypt 2011*, LNCS 7537, pp. 57–76.

[49] Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L. Automatic security evaluation and (related-key) differential characteristic search via MILP. *ASIACRYPT 2014*, LNCS 8873, pp. 158–178.

[50] National Institute of Standards and Technology. A Statistical Test Suite for the Validation of Random Number Generators and Pseudorandom Number Generators for Cryptographic Applications. *NIST SP 800-22 Rev. 1a*, 2010.

[51] Ishai, Y., Sahai, A., Wagner, D. Private circuits: securing hardware against probing attacks. *CRYPTO 2003*, LNCS 2729, pp. 463–481.

[52] B. Palaniswamy and P. Palmieri, "MTSF—Market-Theoretic Security Framework: A Unified Paradigm for the Art of Proving and Disproving Security," *Cryptology ePrint Archive*, 2026.

[53] Bellare, M., Rogaway, P. The security of triple encryption and a framework for code-based game-playing proofs. *EUROCRYPT 2006*, LNCS 4004, pp. 409–426.

[54] Bernstein, D. J. ChaCha, a variant of Salsa20. *Workshop Record of SASC 2008: The State of the Art of Stream Ciphers*, 2008.

[55] Bernstein, D. J. The Salsa20 family of stream ciphers. In: *New Stream Cipher Designs*, LNCS 4986, pp. 84–97. Springer, 2008.

[56] Anderson, R., Biham, E., Knudsen, L. Serpent: A proposal for the Advanced Encryption Standard. *NIST AES Proposal*, 1998.

[57] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N. Twofish: A 128-bit block cipher. *NIST AES Proposal*, 1998.

[58] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T. Camellia: A 128-bit block cipher suitable for multiple platforms — Design and analysis. *Selected Areas in Cryptography (SAC) 2000*, LNCS 2012, pp. 39–56. Springer, 2001.

[59] Ashur, T., Luykx, A. An account of the ISO/IEC standardization of the Simon and Speck block cipher families. In: *Security of Ubiquitous Computing Systems*, pp. 63–78. Springer, 2018.

[60] Lu, J. Attacking reduced-round versions of the SMS4 block cipher in the Chinese WAPI standard. *ICICS 2007*, LNCS 4861, pp. 306–318. Springer, 2007.

[61] Kwon, D., et al. New block cipher: ARIA. *ICISC 2003*, LNCS 2971, pp. 432–445. Springer, 2004.

[62] Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J. The Skein hash function family (v1.3). *Submission to the NIST SHA-3 Competition*, 2010.

[63] Biham, E. New types of cryptanalytic attacks using related keys. *Journal of Cryptology* 7(4), pp. 229–246, 1994.

[64] Biryukov, A., Wagner, D. Slide attacks. *Fast Software Encryption (FSE) 1999*, LNCS 1636, pp. 245–259. Springer, 1999.

[65] Kara, O. Reflection cryptanalysis of some ciphers. *INDOCRYPT 2008*, LNCS 5365, pp. 294–307. Springer, 2008.

[66] Biryukov, A., Khovratovich, D., Nikolić, I. Distinguisher and related-key attack on the full AES-256. *CRYPTO 2009*, LNCS 5677, pp. 231–249. Springer, 2009.

[67] Khovratovich, D., Nikolić, I. Rotational cryptanalysis of ARX. *Fast Software Encryption (FSE) 2010*, LNCS 6147, pp. 333–346. Springer, 2010.

[68] Khovratovich, D., Nikolić, I., Rechberger, C. Rotational rebound attacks on reduced Skein. *ASIACRYPT 2010*, LNCS 6477, pp. 1–19. Springer, 2010.

[69] Vaudenay, S. On the weak keys of Blowfish. *Fast Software Encryption (FSE) 1996*, LNCS 1039, pp. 27–32. Springer, 1996.

[70] Dinu, D., Le Corre, Y., Khovratovich, D., Perrin, L., Großschädl, J., Biryukov, A. Triathlon of lightweight block ciphers for the Internet of Things. *Journal of Cryptographic Engineering* 9, pp. 283–302, 2019.

[71] Käsper, E., Schwabe, P. Faster and timing-attack resistant AES-GCM. *CHES 2009*, LNCS 5747, pp. 1–17. Springer, 2009.

[72] Wikipedia contributors. Block cipher mode of operation — ECB pattern leakage example. *Wikipedia, the free encyclopedia*, accessed 2026. URL: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Electronic_codebook_(ECB)

[73] Cho, J.Y. Linear cryptanalysis of reduced-round PRESENT. In: *CT-RSA 2010*, LNCS **5985**, pp. 302–317. Springer, 2010.

[74] Zong, R., Dong, X., Chen, H., Luo, Y., Wang, S., Li, Z. Towards key-recovery-attack friendly distinguishers: Application to GIFT-128. *IACR Transactions on Symmetric Cryptology* **2021**(1), 156–184, 2021.

[75] Jati, A., Gupta, N., Chattopadhyay, A., Sanadhya, S.K. Threshold implementations of GIFT: A trade-off analysis. *IEEE Transactions on Information Forensics and Security* **15**, 2110–2120, 2020.

[76] Liu, Y., Sasaki, Y. Related-key boomerang attacks on GIFT with automated trail search including BCT effect. In: *ACISP 2019*, LNCS **11547**, pp. 555–572. Springer, 2019.

[77] Abed, F., List, E., Lucks, S., Wenzel, J. Differential cryptanalysis of round-reduced Simon and Speck. In: *FSE 2014*, LNCS **8540**, pp. 525–545. Springer, 2015.

[78] Abdelraheem, M.A., Alizadeh, J., Alkhzaimi, H.A., Aref, M.R., Bagheri, N., Gauravaram, P. Improved linear cryptanalysis of reduced-round SIMON. Cryptology ePrint Archive, Report 2014/681, 2014.

[79] Shanmugam, D., Selvam, R., Annadurai, S. Differential power analysis attack on SIMON and LED block ciphers. In: *Security, Privacy, and Applied Cryptography Engineering 2014*, LNCS **8804**, pp. 110–125. Springer, 2014.

[80] Dinur, I. Improved differential cryptanalysis of round-reduced Speck. In: *Selected Areas in Cryptography 2014*, LNCS **8781**, pp. 147–164. Springer, 2015.

[81] Tolba, M., Abdelkhalek, A., Youssef, A.M. Impossible differential cryptanalysis of reduced-round SKINNY. In: *AFRICACRYPT 2017*, LNCS **10239**, pp. 117–134. Springer, 2017.

[82] Sadeghi, S., Mohammadi, T., Bagheri, N. Cryptanalysis of reduced round SKINNY block cipher. *IACR Transactions on Symmetric Cryptology* **2018**(3), 124–162, 2018.

[83] Ankele, R., Dobraunig, C., Guo, J., Lambooij, E., Leander, G., Todo, Y. Biased differential distinguisher — Cryptanalysis of reduced-round SKINNY. *Information and Computation* **281**, 104796, 2021.

[84] Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçın, T. PRINCE — A low-latency block cipher for pervasive computing applications. In: *ASIACRYPT 2012*, LNCS **7658**, pp. 208–225. Springer, 2012.

[85] Dinur, I. Cryptanalytic time-memory-data tradeoffs for FX-constructions with applications to PRINCE and PRIDE. In: *EUROCRYPT 2015*, LNCS **9056**, pp. 231–253. Springer, 2015.

[86] Li, L., Jia, K., Wang, X. Improved meet-in-the-middle attacks on AES-192 and PRINCE. Cryptology ePrint Archive, Report 2013/573, 2013.

[87] Abed, F., List, E., Lucks, S. On the security of the core of PRINCE against biclique and differential cryptanalysis. Cryptology ePrint Archive, Report 2012/712, 2014.

[88] Soleimany, H., Blondeau, C., Yu, X., Wu, W., Nyberg, K., Zhang, H., Zhang, L., Wang, Y. Reflection cryptanalysis of PRINCE-like ciphers. *Journal of Cryptology* **28**(3), 718–744, 2015.

[89] Guo, J., Peyrin, T., Poschmann, A., Robshaw, M. The LED block cipher. In: *CHES 2011*, LNCS **6917**, pp. 326–341. Springer, 2011.

[90] Isobe, T., Shibutani, K. Security analysis of the lightweight block ciphers XTEA, LED, and Piccolo. In: *ACISP 2012*, LNCS **7372**, pp. 71–86. Springer, 2012.

[91] Dinur, I., Dunkelman, O., Keller, N., Shamir, A. Key recovery attacks on 3-round Even–Mansour, 8-step LED-128, and full AES^2. In: *ASIACRYPT 2013*, LNCS **8269**, pp. 337–356. Springer, 2013.

[92] Mendel, F., Rijmen, V., Toz, D., Varıcı, K. Differential analysis of the LED block cipher. In: *ASIACRYPT 2012*, LNCS **7658**, pp. 190–207. Springer, 2012.

[93] De Cannière, C., Dunkelman, O., Knezevic, M. KATAN and KTANTAN — A family of small and efficient hardware-oriented block ciphers. In: *CHES 2009*, LNCS **5747**, pp. 272–288. Springer, 2009.

[94] Bogdanov, A., Rechberger, C. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In: *Selected Areas in Cryptography 2010*, LNCS **6544**, pp. 229–240. Springer, 2011.

[95] Wei, L., Rechberger, C., Guo, J., Wu, H., Wang, H., Ling, S. Improved meet-in-the-middle cryptanalysis of KTANTAN. In: *ACISP 2011*, LNCS **6812**, pp. 433–438. Springer, 2011.

[96] Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I. RECTANGLE: A bit-slice lightweight block cipher suitable for multiple platforms. *Science China Information Sciences* **58**(12), 1–15, 2015.

[97] Zhao, B., Dong, X., Meier, W., Jia, K., Wang, G. Generalized related-key rectangle attacks on block ciphers with linear key schedule: Applications to SKINNY and GIFT. *Designs, Codes and Cryptography* **88**(6), 1103–1126, 2020.

[98] Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F. Midori: A block cipher for low energy. In: *ASIACRYPT 2015*, LNCS **9453**, pp. 411–436. Springer, 2015.

[99] Guo, J., Jean, J., Nikolic, I., Qiao, K., Sasaki, Y., Sim, S.M. Invariant subspace attack against Midori64 and the resistance criteria for S-box designs. *IACR Transactions on Symmetric Cryptology* **2016**(1), 33–56, 2016.

[100] Beierle, C., Canteaut, A., Leander, G., Rotella, Y. Proving resistance against invariant attacks: How to choose the round constants. In: *CRYPTO 2017*, LNCS **10402**, pp. 647–678. Springer, 2017.

[101] Todo, Y., Leander, G., Sasaki, Y. Nonlinear invariant attack: Practical attack on full SCREAM, iSCREAM, and Midori64. *Journal of Cryptology* **32**(4), 1383–1422, 2019.

[102] Beyne, T. Block cipher invariants as eigenvectors of correlation matrices. *Journal of Cryptology* **33**(3), 1156–1183, 2020.

[103] Tolba, M., Abdelkhalek, A., Youssef, A.M. Multidimensional zero-correlation linear cryptanalysis of reduced-round Midori-128. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **E100-A**(8), 1773–1779, 2017.

[104] Chen, Z., Chen, H., Wang, X. Cryptanalysis of Midori-128 using impossible differential techniques. In: *Information Security Practice and Experience (ISPEC) 2016*, LNCS **10060**, pp. 1–12. Springer, 2016.

[105] Lin, L., Wu, W. Meet-in-the-middle attacks on reduced-round Midori-64. *IACR Transactions on Symmetric Cryptology* **2017**(1), 215–239, 2017.

[106] Han, G., Zhang, W., Zhao, H. Unbalanced biclique cryptanalysis of full round Midori. *IET Communications* **13**(10), 1437–1444, 2019.

[107] Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T. The 128-bit blockcipher CLEFIA. In: *FSE 2007*, LNCS **4593**, pp. 181–195. Springer, 2007.

[108] Tezcan, C. The improbable differential attack: Cryptanalysis of reduced round CLEFIA. In: *INDOCRYPT 2010*, LNCS **6498**, pp. 197–209. Springer, 2010.

[109] Blondeau, C., Nyberg, K. Links between truncated differential and multidimensional linear properties of block ciphers and underlying attack complexities. In: *EUROCRYPT 2014*, LNCS **8441**, pp. 165–182. Springer, 2014.

[110] Goudarzi, D., Jean, J., Kölbl, S., Peyrin, T., Rivain, M., Sasaki, Y., Sim, S.M. Pyjamask: Block cipher and authenticated encryption with highly efficient masked implementation. *IACR Transactions on Symmetric Cryptology* **2020**(S1), 31–59, 2020.

[111] Dobraunig, C., Rotella, Y., Schoone, J. Algebraic and higher-order differential cryptanalysis of Pyjamask-96. *IACR Transactions on Symmetric Cryptology* **2020**(1), 289–312, 2020.

[112] Zhang, Z., Hou, C., Liu, M. Integral attacks on Pyjamask-96 and round-reduced Pyjamask-128. In: *CT-RSA 2022*, LNCS **13161**, pp. 223–246. Springer, 2022.

[113] Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M. Ascon v1.2: Lightweight authenticated encryption and hashing. *Journal of Cryptology* **34**(3), Article 33, 2021.

[114] Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M. Cryptanalysis of Ascon. In: *CT-RSA 2015*, LNCS **9048**, pp. 371–387. Springer, 2015.

[115] Tezcan, C. Differential-linear cryptanalysis of Ascon: Theory vs. practice. In: *NIST Lightweight Cryptography Workshop 2023*, 2023.

[116] National Institute of Standards and Technology. Status report on the final round of the NIST lightweight cryptography standardization process. *NIST Internal Report 8454*, 2023.

[117] Biryukov, A., Roy, A., Velichkov, V. Differential analysis of block ciphers SIMON and SPECK. In: *FSE 2014*, LNCS **8540**, pp. 546–570. Springer, 2015.

[118] Sönmez Turan, M., McKay, K. A., Chang, D., Kang, J., Kelsey, J. Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions. *NIST Special Publication 800-232*, National Institute of Standards and Technology, Gaithersburg, MD, August 2025. DOI: https://doi.org/10.6028/NIST.SP.800-232.

[119] Song, L., Yang, Q., Liu, Y., Zhao, Z. New differential-based distinguishers for Ascon via constraint programming. *IET Information Security* **2024**, Article ID 6624991, 2024.

[120] Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M. Ascon implementations: reference and optimised C, AVX2, AVX-512, ARM, RISC-V variants. Public collection at https://github.com/ascon/ascon_collection; benchmark data at eBACS / SUPERCOP eBAEAD platform set, 2014–2025.

[121] Downs, T. Gathering Intel on Intel AVX-512 transitions. *Performance Matters* blog, January 2020. https://travisdowns.github.io/blog/2020/01/17/avxfreq1.html.

Index Supplement: Appendices B and C

Extending the v16 Table of Contents with the v17 / v18.4 / v18.5 / v18.7 enrichment material

Appendix B MERIDIAN as an AEAD: Construction, Security, and a Head-to-Head with Ascon-AEAD128

B.1. Why an AEAD Section, and Why Now . . . 140
B.2. MERIDIAN-GCM-128: A Standard-Mode AEAD Wrapping . . . 141
B.2.1. Goal and design principle . . . 141
B.2.2. Parameters and notation . . . 141
B.2.3. Construction B.1 (MERIDIAN-GCM-128) . . . 142
B.2.4. Figure B.1 – GCM-128 construction diagram (CTR + GHASH) . . . 143
B.2.5. Why this is correct as a drop-in . . . 143
B.2.6. Security (IOM 2012 bound; differential prob. 2^{-288}) . . . 143
B.2.7. Implementation cost (Table B.1: cyc/B decomposition) . . . 144
B.3. MERIDIAN-Duplex-AEAD: A Sponge-Mode AEAD . . . 145
B.3.1. Structural twin of Ascon-AEAD128 . . . 145
B.3.2. State and permutation $\mathcal{P}_{2\times12}$ . . . 146
B.3.3. Construction B.2 (MERIDIAN-Duplex-AEAD) . . . 147
B.3.4. Figure B.2 – Duplex-AEAD sponge phases (Init / AD / PT / Finalize) . . . 148
B.3.5. Decryption . . . 148
B.3.6. Security (duplex-sponge bound; 64-bit generic security) . . . 148
B.3.7. Implementation cost (Table B.2: ∼45.6 cyc/B) . . . 149
B.4. Head-to-Head: Ascon-AEAD128 vs. MERIDIAN-AEAD on AEAD Traits . . . 150
B.4.1. Trait taxonomy (5 categories, 15 traits) . . . 150
B.4.2. Figure B.3 – 15-trait scorecard heatmap . . . 151
B.4.3. The fifteen-trait scorecard, by category (Tables B.3–B.7) . . . 151
B.4.4. Where each AEAD is the right answer (Table B.8) . . . 154
B.5. AEAD-Trait Summary and Honest Disclosure . . . 155
B.5.1. Three-line summary of the AEAD section . . . 155
B.5.2. Figure B.4 – AEAD performance summary (x86-64 / Cortex-M4 / ASIC) . . . 156
B.5.3. The maturity-asymmetry disclosure . . . 156
B.5.4. Closing position . . . 157
B.6. MTSF Audit: Ascon's Implementation Vulnerabilities vs MERIDIAN . . . 158
B.6.1. Ten Ascon implementation-vulnerability goods (Table B.10) . . . 158
B.6.2. Auditing MERIDIAN against the ten goods (§§B.6.2.1–B.6.2.10) . . . 159
B.6.3. Summary scorecard (Table B.11) . . . 162
B.6.4. Updated MTSF goods catalogue $|\mathcal{G}^{\mathrm{v17}}_{\mathrm{MER}}| = 27$ . . . 163
B.6.5. Updated validation roadmap (I1–I5) . . . 164
B.7. Unified Protection Profile $\Pi^{\mathrm{MER}}_{\mathrm{unified}}$ . . . 164
Bibliography of Appendix B (42 entries) . . . 165

Appendix C MERIDIAN-AEAD and MERIDIAN-Hash: A Unified Performance, Deployment, and Security Enrichment

C.1. Why a Hash Section, and Why Now . . . 166
C.2. MERIDIAN-Hash-256 and MERIDIAN-Hash-512: Specification . . . 168
C.2.1. Notation, parameters, and Table C.1 (variants) . . . 168
C.2.2. Figure C.1 – Sponge construction . . . 169
C.2.3. Figure C.2 – Hash construction overview (Hash-256 vs Hash-512) . . . 169
C.2.4. MERIDIAN-Hash-256 construction (Construction C.1) . . . 170
C.2.5. MERIDIAN-Hash-512 construction (Construction C.2) . . . 171
C.2.6. The $\mathcal{P}_{4\times12}$ permutation . . . 172
C.2.7. Reference implementation pointer . . . 173
C.3. Security Analysis . . . 173

C.3.1. Standard-model sponge bounds (Theorem C.1) . . . 173
C.3.2. Differential / linear bounds inherited from the cipher . . . 174
C.3.3. Length-extension immunity . . . 174
C.3.4. Multi-collision resistance (Joux bound) . . . 175
C.3.5. Algebraic-attack resistance . . . 175
C.3.6. Side-channel posture . . . 175
C.3.7. Honest disclosure of maturity gap . . . 176
C.4. Strengthened Security Proofs for MERIDIAN-AEAD and MERIDIAN-Hash . . . 176
C.4.1. Theorem C.1 – MERIDIAN-GCM-128 AE security . . . 177
C.4.2. Theorem C.2 – MERIDIAN-Duplex-AEAD AE security . . . 177
C.4.3. Theorem C.3 – MERIDIAN-Hash-256 indifferentiability . . . 178
C.4.4. Theorem C.4 – MERIDIAN-Hash-512 indifferentiability . . . 178
C.5. MERIDIAN-Only Performance Enrichment: AEAD and Hash . . . 179
C.5.1. Per-stage cycles/byte breakdowns . . . 179
C.5.2. Throughput-vs-message-length curves . . . 180
C.5.3. Family Pareto map . . . 180
C.6. Head-to-Head against Six Hash Candidates . . . 181
C.6.1. Twelve-trait scorecard (Table C.2) . . . 181
C.6.2. Figure C.5 – Hash security radar . . . 183
C.6.3. Figure C.6 – Hash performance summary . . . 184
C.7. Software Performance (x86-64, Cortex-M4) . . . 185
C.8. Hardware Performance (90 nm CMOS) . . . 187
C.9. MTSF Audit of MERIDIAN-Hash . . . 188
C.10. Deployment Niches: Where Each MERIDIAN Primitive Wins . . . 189
C.11. Performance Scorecard: Where MERIDIAN-AEAD/Hash Are Good . . . 190
C.12. Validation Roadmap (H1–H8) . . . 191
C.13. Honest Disclosure for Appendix C . . . 191
Bibliography of Appendix C (24 entries) . . . 192

Notes. Page numbers refer to the merged v18.7 PDF. Figures B.1–B.4 are new in v18.7 (added to the previously figure-less Appendix B); Figure C.2 and Figure C.6 are new in v18.7 (additive figures on top of the v18.5 Appendix C visual material). All other content is byte-identical to v18.5. The v16 main Table of Contents (pages 2–5 of the merged PDF) covers Sections 1–13 and Appendix A; this supplement covers Appendix B and Appendix C.

Appendices B and C Corrigendum Insert

The AEAD and hash constructions, re-derived over MERIDIAN-v2 — and what the shipped reference code was actually doing

v20b, September 2026. Unnumbered insert, placed immediately before printed page 140. Discharges S3-C.6, item 5. Reproducer: `supplementary/verify_apxBC_v2.py`.

BC-1 Why this insert exists

S3-C.6, item 5 records that the constructions of Appendices B and C assume a permutation and are void for MERIDIAN-v1, and calls for them to be re-derived over v2 before their theorems can be restated. This insert performs that re-derivation; item 5 now points here for the result.

Doing it required reading the shipped reference code closely, and that turned up three findings which are logically prior to the migration. They are stated first, because two of them mean that parts of Appendices B and C were never about the object the reader assumes.

Summary of findings.

1. **The AEAD reference implementation does not use MERIDIAN.** Every block call in `aead/meridian_aead.py` is routed through `_MeridianFallback`, a placeholder its own docstring describes as having "a single-byte-rotation diffusion", as being "NOT the real MD layer", and as existing "for code-integrity testing of the AEAD wrapper, not for cryptanalysis". No security or performance number measured against that file is a number about MERIDIAN.
2. $P_{4\times12}$ **is not a permutation.** Its cross-coupling layer is exactly 4-to-1, so MERIDIAN-Hash-512's sponge has trivial internal collisions. This is the same defect class as the v1 cipher's, found in the same way, and it voids Appendix C's indifferentiability argument independently of anything in Section 3.
3. **The hash round function is not MERIDIAN's either.** DS is a bare S-box with no directional context, and AM is round-constant addition computed from floating-point $\sin(\cdot)$ — the construct criticised in S3-C.2, A3 and removed from the cipher in v2. No key-gated layer appears at all.

BC-2 $P_{4\times12}$ is not a permutation

Appendix C's $P_{4\times12}$ organises a 512-bit state as four 4×4 sub-states $h_0,\dots,h_3$ and, after per-sub-state DS and MD, applies the cross-coupling

$$h_k \;\leftarrow\; h_k \oplus h_{(k+1) \bmod 4}, \qquad k = 0,1,2,3, \tag{1}$$

with all four assignments evaluated *simultaneously from the old values*. Over $\mathbb{F}_2$ this is the circulant $I+P$ on four blocks, where P is the cyclic shift. Its invertibility is governed by $\gcd(1+x,\ x^4+1)$ over $\mathbb{F}_2$; since $x^4+1=(1+x)^4$ in characteristic 2, that gcd is $1+x\neq1$, so the map is singular. Its kernel is $\{(h,h,h,h)\}$, of dimension 128 bits, and it is therefore exactly 4-to-1.

The consequence is not asymptotic. Any state whose four sub-states are equal is annihilated by (**??**), and since DS and MD act identically on each sub-state, equality is preserved up to that point. Every such state therefore reaches the *same* image after round 1:

Input X	`0000000000000000...` (four zero sub-states)
Input Y	`000102...0f` repeated four times
$P_{4\times12}(X)$	`e2e5c8a2a42d6ee2f802e10c019bdf47eda01e3b662a6390...`
$P_{4\times12}(Y)$	`e2e5c8a2a42d6ee2f802e10c019bdf47eda01e3b662a6390...`
Identical. Of the 256 inputs $\texttt{cc}^{64}$, $c=0,\dots,255$, the number of distinct images is 1.	

A sponge is indifferentiable from a random oracle only when its inner map is a permutation; the proof reduces adversarial advantage to the probability of hitting an inner collision, and here inner collisions are free. Appendix C's security argument for MERIDIAN-Hash-512 therefore does not hold, for a reason internal to Appendix C and independent of the Section 3 defect. `verify_apxBC_v2.py` reproduces all of the above.

BC-3 The re-derived permutations

`supplementary/meridian_v2_perm.py` builds both wide permutations from the v2 round function of S3-C.3 rather than from the layers the old code used. Each sub-state undergoes DS′, MD′, AM′ and a round constant, exactly as in Algorithm 1, with AM′ evaluated under a fixed public constant κ in place of a key — the standard way of obtaining a public permutation from a keyed round function, and the way that preserves the structure Theorem S3-C.2 was proved about. Round constants are indexed by (r, k) so that sub-states are not parallel copies of one another.

Coupling.

Equation (**??**) is replaced by a two-phase rule, on the checkerboard principle of S3-C.3.2 — each phase writes only sub-states it does not read:

$$\text{phase 1: } h_0 \oplus\!= h_1,\ \ h_2 \oplus\!= h_3; \qquad \text{phase 2: } h_1 \oplus\!= h_0,\ \ h_3 \oplus\!= h_2,$$

where phase 2 reads the *updated* h_0 and h_2. Every assignment has the form $x \leftarrow x \oplus f(\text{others})$, which is an involution in x and so unconditionally invertible; the composition is a bijection for precisely the reason Theorem S3-C.2 gives. P_2 uses the two-sub-state case of the same rule.

The inverse is *implemented*, not merely argued: `p2_inv` and `p4_inv` recover the input on every test, which is the check the old code could not have passed.

	State	Image of the all-zero state
P_2	256 bits	`534aa3e90f73cbe20aba969bdadce10a` `3423803b10e23196dd2489038f60de88`
P_4	512 bits	`f5401a61bfdd85f13e8ce15726d8c193` `e8044119013ac2c3bf552eac746330a5` `944d26a76f05e864d771724579d67dfc` `351baecfe6ca3aee0568043dd406b4cb`

BC-4 Appendix C — MERIDIAN-Hash over v2

`hash/meridian_hash_v2.py` is a sponge over P_2 and P_4 with pad10*1 multi-rate padding and a domain-separation byte in the capacity. The parameter sets of Appendix C are kept unchanged, so its comparison tables remain meaningful:

	State	Rate	Capacity	Digest	Collision res.
MERIDIAN-Hash-256	256	128	128	256	64
MERIDIAN-Hash-512	512	128	384	512	192

A parameter problem the migration does not fix. Sponge collision resistance is capped at $c/2$. With $c = 128$, MERIDIAN-Hash-256 offers at most 64-bit collision resistance, not the 128 bits Appendix C claims, and no choice of permutation changes that. This is a parameter choice rather than a defect of v1, so it survives the port intact and is reported here rather than silently repaired: raising the capacity would be a redesign of the construction, not a migration of it.

Digests.

Over P_2 and P_4 as above:

Message	Digest	
(empty)	*H-256*	4e03f586919787f2d2cb33ae15c86e51
		b0a39546f77090d42ff2e54bb8da3379
	H-512	0cb3867bd00c7d6b88027a4726ebea25
		f0aacf3ade64ae5e775a3ee1358fe28c
		b5baba59f550caf0c776e54949e13c3c
		838ee596b647506219957d7b2928d6c7
abc	*H-256*	d36665d8e5583025334cb0bb6c3ff4ff
		6ea4ee900c280251cf1184180e9ccc71
	H-512	377f02f0cf945250924968f37964534f
		15b10d543e9b1203411fe84f93714de1
		726ef1c70a5a0c53c49796f727a0a423
		6d8f5cffb6926f7e601a24fbeeaebaf0

Avalanche over 96 single-bit input flips: mean 127.16 of 256 output bits (ideal 128.00), range 99–148. The 256 inputs $\texttt{cc}^{64}$ that collapsed to a single image under $P_{4\times12}$ now give 256 distinct digests.

BC-5 Appendix B — MERIDIAN-AEAD over v2

`aead/meridian_aead_v2.py` replaces the placeholder cipher with the real one. Two constructions:

MERIDIAN-GCM-128.
GCM exactly as in NIST SP 800-38D, with the MERIDIAN-v2 block cipher in place of AES-128 and GHASH unchanged. Nothing here is novel, and that is the point: S3-C.3.11 already establishes that v2 drops into any AES-128 mode, so the construction is GCM with a different 128-bit block cipher underneath. It inherits GCM's security exactly to the extent that the block cipher is a secure PRP.

MERIDIAN-Duplex-AEAD.
A monkey-duplex over P_2, rate 128, capacity 128, with domain separation between key, associated data, plaintext and finalisation, and pad10* padding.

K	000102030405060708090a0b0c0d0e0f
nonce (GCM)	000102030405060708090a0b
nonce (duplex)	000102030405060708090a0b0c0d0e0f
P empty, AD empty	
GCM C	(empty)
GCM T	bf3b4519950031d9c7c689c62438283b
Duplex C	(empty)
Duplex T	56a9bd4840fd0667587ef9acb8a65aa0
P = `Rotate and mix.`, AD = `hdr`	
GCM C	7c89f117816ea1762b5152ec0f77ac
GCM T	b6ed93c60ce4014d76c32fbe9d0d1a6b
Duplex C	21a33c3748ed50e64f0cb38905e7e9
Duplex T	8dcf266dbc7a659bc4a845be3555db8d

Both constructions round-trip over 82 message lengths including empty, partial and exact-multiple blocks, with $|C| = |P|$ in every case, and both reject all 600 single-bit forgeries injected into the ciphertext, the tag and the associated data. Neither is nonce-misuse-resistant: reusing a nonce leaks the GHASH key and the XOR of plaintexts under GCM, and the keystream under the duplex.

BC-6 What this insert does not fix

1. The results here are **structural and behavioural**: the permutations are bijections with exhibited inverses, the sponge and both AEAD modes are correct, forgeries are rejected. None of that is a security result, and no theorem of Appendix B or C is restated by this insert.

2. **MERIDIAN-Hash-256's parameters remain wrong for its claim.** $c = 128$ caps collision resistance at 64 bits. Appendix C's 128-bit claim needs either a larger capacity or a smaller claim.
3. **The MILP bound $A_{12} \geq 96$ does not transfer.** It was computed for the old coupling, which has been replaced — and was computed for a layer that is not injective, so it was a bound on a non-permutation.
4. **Every performance figure in Appendices B and C must be re-measured.** The AEAD numbers were taken against an explicit placeholder cipher; the hash numbers against a round function that is not MERIDIAN's. The scorecards, throughput plots and ROM/RAM footprints in both appendices should be treated as unsourced until regenerated.
5. **Indifferentiability and AEAD security still assume an ideal permutation.** MERIDIAN-v2 has had no third-party cryptanalysis (S3-C.6, item 6). A correct construction over an unanalysed permutation is an unanalysed construction.
6. For production, the guidance of S3-C.3.9 stands unchanged: use AES-128-GCM or Ascon.

Reproducer. `python3 supplementary/verify_apxBC_v2.py` demonstrates every defect in BC-1 and BC-2 and verifies every claim in BC-3 to BC-5. Reference code: `supplementary/meridian_v2_perm.py`, `supplementary/hash/meridian_hash_v2.py`, `supplementary/aead/meridian_aead_v2.py`. Vectors: `hash/test_vectors_v2.txt`, `aead/test_vectors_v2.txt`.

Appendix B
MERIDIAN as an AEAD: Construction, Security, and a Head-to-Head with Ascon-AEAD128

(v18 enrichment over the v16 ePrint package)

Basker Palaniswamy* Paolo Palmieri Ashok Kumar Das Ruei-Hau Hsu

May 2026 (v18)

Scope of this appendix. The v16 ePrint package specifies MERIDIAN strictly as a 128-bit block cipher with a 128-bit key, and (at §3.5, §5.15, §6, §8.11, and §9.14) compares it to Ascon-AEAD128 across structural, cryptanalytic, performance, side-channel, and standardisation axes. That comparison is honest about a category gap: MERIDIAN exposes a block-cipher API ($\mathsf{Enc}_K : \{0,1\}^{128} \to \{0,1\}^{128}$), whereas Ascon-AEAD128 is, by definition and by NIST SP 800-232 standardisation, an authenticated-encryption-with-associated-data primitive.

This appendix closes the API gap by giving MERIDIAN an explicit AEAD instantiation, then places *MERIDIAN-AEAD* side-by-side against *Ascon-AEAD128* on AEAD-specific traits. We give the construction at two distinct levels of fidelity to pre-existing standards:

- **MERIDIAN-GCM-128** (§B.2): A canonical NIST SP 800-38D [1] instantiation of GCM with MERIDIAN as the underlying block cipher. This is the AEAD mode that drops into any system that already supports AES-GCM. It exists because the v16 paper's compatibility argument (envelope, primitive, API) only delivers value if a concrete AEAD wrapping is exhibited, not merely promised.
- **MERIDIAN-Duplex-AEAD** (§B.3): A sponge-mode AEAD construction built directly from the MERIDIAN round function in a 256-bit-state duplex configuration (rate $r = 128$, capacity $c = 128$). This mode is structurally analogous to Ascon-AEAD128 and is included so that the AEAD-axis comparison with Ascon is conducted at the level of construction, not merely at the level of API.

A subsequent section (§B.4) gives a fifteen-axis AEAD-trait head-to-head between Ascon-AEAD128 and MERIDIAN-AEAD (in both modes), with each row sourced either from NIST SP 800-232 [11], the Ascon v1.2 specification [10], MERIDIAN §3 of the v16 ePrint, or the construction defined in this appendix. §B.5 closes with a summary scorecard and an honest disclosure of the maturity asymmetry.

What this appendix does *not* do. It does not extend the cipher specification of v16 (the round function, key, and block remain byte-identical). It does not claim third-party cryptanalysis of MERIDIAN-AEAD — the cryptanalytic record at the AEAD level is, like the block-cipher record, almost entirely self-reported and is offered as a research-prototype starting point, not as an alternative to NIST SP 800-232 for any production deployment. It does not modify the v16 benchmark numbers (E1–E15); the AEAD-mode performance figures reported here are mode-overhead estimates layered on top of the measured block-cipher figures.

Novelty at a Glance (Appendix B — v18 update)

*Corresponding author: basker170889@gmail.com.

What is new in Appendix B relative to the v16 ePrint main body, with forward pointers to where each claim is justified.

Axis	What is new
MERIDIAN-GCM-128 (new Construction B.1)	The first explicit, rigorously specified AEAD instantiation of MERIDIAN. NIST SP 800-38D GCM [1] with MERIDIAN_K in the underlying-cipher slot. Closes the v16 §3.5 API gap: the block-cipher compatibility argument (envelope / primitive / API) only delivers value when a concrete AEAD wrapping is exhibited, not merely promised. Includes full construction, IND-CPA + INT-CTXT security argument (IOM 2012 bound [3]), and a three-path cost-decomposition table (PCLMULQDQ / LUT / bitsliced GHASH) at 12.7 / 16.9 / 20.6 cyc/byte (§B.2, Table B.1).
MERIDIAN-Duplex-AEAD (new Construction B.2)	A 256-bit-state duplex sponge AEAD ($r = 128$, $c = 128$) built from the MERIDIAN round function, structurally twinned with Ascon-AEAD128 to enable a like-for-like comparison at the AEAD layer. New permutation $\mathcal{P}_{2\times 12}$ with cross-half mixing function π. Security analysis: duplex sponge bound [7, 8]; 64-bit generic security ($c/2$); 128-bit forgery resistance. Honest disclosure of the 64-bit vs. Ascon's 96-bit generic-security gap (§B.3).
Fifteen-axis AEAD head-to-head (Tables B.3–B.9)	First structured AEAD-layer comparison between Ascon-AEAD128, MERIDIAN-GCM-128, and MERIDIAN-Duplex-AEAD across five trait categories (construction, security claim, performance, side-channel, standardisation). Replaces the v16 §8.11 "apples-to-oranges" comparison (block cipher vs. AEAD) with a mode-vs-mode scorecard. Key findings: MERIDIAN-GCM-128 matches Ascon on nonce semantics; outperforms on parallelism and pipelining; trades 64-bit vs. 96-bit generic security on the Duplex mode; leads on masking-ecosystem maturity and AVX-512 immunity (§B.4).
AEAD deployment-guidance framework (Table B.8)	Six-scenario deployment table mapping practical AEAD choices to MERIDIAN-GCM-128, MERIDIAN-Duplex-AEAD, or Ascon-AEAD128 based on gate budget, nonce discipline, parallelism requirements, and standardisation constraints (§B.4).
MTSF implementation-vulnerability audit (new §B.6)	First application of the Market-Theoretic Security Framework to Ascon's *implementation* profile (v16 §7 applied the MTSF only to MERIDIAN's mathematical and physical security). Ten Ascon-AEAD128 implementation goods enumerated from the 2016–2025 literature (SIFA, DFA, PFA, DPA, template, combo, cube-trojan, AVX-license, mask-imm, RUP-leak). MERIDIAN audited against each good: 2 structural immunities, 1 ecosystem advantage, 2 smaller-magnitude inheritances, 5 inherited-at-parity with prescribed fixes. MTSF catalogue extended from 18 (v16) to 27 goods (§B.6).
Unified protection profile $\Pi^{\text{MER}}_{\text{unified}}$ (new Construction B.3, §B.7)	First unified countermeasure framework for MERIDIAN that brings **8 of 10** Ascon implementation-vulnerability goods to MTSF equilibrium under a *single* layered construction, at the same cost as the strongest individual fix ($\leq 4\times$ area, $\leq 3\times$ energy/byte at $d = 2$): **L1** ISW probing-secure masking ($g_{\text{DPA}-\text{low}}$, g_{template}); **L2** dual-rail S-box ROM (g_{PFA}); **L3** infection-based branch-free fault response ($g_{\text{DFA}-\text{bit}}$, g_{combo}); **L4** SIFA duplication over ISW-encoded state (g_{SIFA}). Companion $\Pi^{\text{MER}}_{\text{flush}}$ (Construction B.4) covers $g_{\text{RUP}-\text{leak}}$ for MERIDIAN-Duplex-AEAD (§B.7).
Updated posited-security theorem (Theorem B.6.3)	MERIDIAN Posited Security theorem updated to reflect the unified profile: $\forall g \in \mathcal{G}^{\text{v18}}_{\text{MER}} \setminus \{g_{\text{cube}-\text{trojan}}\} : \text{Ask}(g) \leq \text{negl}(\lambda)$ at $\lambda = 128$ under $\Pi^{\text{MER}}_{\text{unified}} \cup \Pi^{\text{MER}}_{\text{flush}}$. Explicit exclusion of $g_{\text{cube}-\text{trojan}}$ as algorithmically irreducible, with formal argument for why no cipher-design fix is possible (§B.6.5).
Extended validation roadmap (items I1–I8)	Eight concrete lab-validation items extending the v16 E1–E15 harness to the AEAD and unified-protection layers: I1–I5 address individual implementation goods; I6 integrates $\Pi^{\text{MER}}_{\text{unified}}$ on a Cortex-M33 target; I7 measures $\Pi^{\text{MER}}_{\text{flush}}$ overhead on three MCU platforms; I8 commissions an independent third-party MTSF equilibrium audit (§B.6.6).

One-line summary. Appendix B gives MERIDIAN two concrete AEAD modes (GCM-128 and Duplex), places them in a fifteen-axis head-to-head with Ascon-AEAD128, audits MERIDIAN's implementation profile against Ascon's known vulnerabilities under the MTSF, and provides a unified four-layer protection profile that brings 8 of 10 implementation goods to equilibrium

at $\leq 4\times$ area overhead. The remaining two goods are honest and algorithmically irreducible: $g_{\text{cube-trojan}}$ (supply-chain) and the residual physical leakage floor under finite-order masking.

B.1 Why an AEAD Section, and Why Now

Authenticated encryption with associated data (AEAD), formalised by Rogaway as the all-in-one notion subsuming privacy and authenticity in a single keyed primitive [9], is the dominant API contract for symmetric encryption in production protocols. TLS 1.3, IPsec ESP, SSH, WireGuard, QUIC, Signal, Noise, and the Matter (CHIP) IoT protocol stack are all AEAD-only at the record layer. A 128-bit block cipher that does not present an AEAD wrapping cannot be a drop-in for any of these; the v16 paper acknowledges this implicitly when it lists "GCM, CCM" among the AES-128 modes that MERIDIAN inherits via the API contract (§3.5), but stops short of giving the wrapping itself.

The v16 head-to-head with Ascon (§8.11) is honest that the comparison is partly apples-to-oranges: Ascon-AEAD128 is benchmarked as a complete AEAD (initialization + associated-data absorption + plaintext processing + finalization + tag generation), whereas MERIDIAN figures are pure block-cipher block-encrypt costs. That asymmetry is forced by the v16 specification: there is no MERIDIAN AEAD to benchmark.

This appendix removes that forcing. By specifying MERIDIAN-GCM-128 and MERIDIAN-Duplex-AEAD — both rigorously, with key-derivation, IV handling, associated-data absorption, plaintext processing, and tag generation — it becomes possible to:

(i) Compare Ascon and MERIDIAN at the AEAD level on structurally aligned axes (state size in the AEAD-mode, security claim parity, tag length, IV/nonce semantics, AD-handling cost, parallelism, pipelining, decryption-failure semantics, robustness goals).

(ii) Reframe the v16 §8.11 "apples-to-oranges" performance comparison as a like-for-like AEAD-vs-AEAD comparison, with the mode-overhead costs surfaced as separable line items.

(iii) Quantify the additional engineering cost (gates, code size, RAM) that AEAD wrapping imposes on each cipher, enabling a deployment-decision framework that does not flatten the AEAD axis.

We treat the two AEAD constructions as research artefacts that complement the v16 block-cipher specification. They inherit the v16 cryptanalytic posture: MILP-verified internal-permutation differential and linear bounds, no third-party AEAD cryptanalysis, and full disclosure of validation roadmap items (§B.5.2).

B.2 MERIDIAN-GCM-128: A Standard-Mode AEAD Wrapping

B.2.1 Goal and design principle

Galois/Counter Mode (GCM), standardised in NIST SP 800-38D [1] and required by TLS 1.3 [2], is the most widely deployed block-cipher AEAD mode. Its appeal is structural: it is parallelisable in both encryption (CTR mode) and authentication (GHASH multiplication chain), it has a single underlying-cipher call per 16-byte block, and it has been amenable to dedicated hardware (PCLMULQDQ on x86, PMULL on ARMv8). MERIDIAN-GCM-128 is the literal instantiation of GCM under the substitution $\text{AES}_K \to \text{MERIDIAN}_K$ in the underlying-cipher slot.

B.2.2 Parameters and notation

- Key: $K \in \{0,1\}^{128}$ (128-bit master key, identical to the v16 block-cipher key).
- Nonce: $N \in \{0,1\}^{96}$ (the recommended GCM nonce length, giving the IV-construction $J_0 = N \,\|\, 0^{31} \,\|\, 1$). Other lengths are handled by the standard GHASH-based IV derivation.
- Associated data: $\mathsf{AD} \in \{0,1\}^*$, length $\leq 2^{61}-256$ bits (NIST limit).
- Plaintext: $\mathsf{PT} \in \{0,1\}^*$, length $\leq 2^{39}-256$ bits (NIST limit).
- Ciphertext: $\mathsf{CT} \in \{0,1\}^*$, $|\mathsf{CT}| = |\mathsf{PT}|$.
- Authentication tag: $\mathsf{T} \in \{0,1\}^t$, $t \in \{128, 120, 112, 104, 96\}$ (default $t = 128$).

B.2.3 Construction

Construction B.2.1 (MERIDIAN-GCM-128)**.** Let $E_K(\cdot) \coloneqq \text{MERIDIAN}_K(\cdot)$ denote the v16 MERIDIAN block-cipher encryption function under key K, with 128-bit block size. MERIDIAN-GCM-128 is the GCM construction of [1] with E_K in the underlying-cipher slot. Specifically:

(1) **Hash subkey derivation.** $H \coloneqq E_K(0^{128})$.

(2) **IV processing.** If $|N| = 96$, $J_0 \coloneqq N \,\|\, 0^{31} \,\|\, 1$. Else $J_0 \coloneqq \mathsf{GHASH}_H(N \,\|\, 0^{s+64} \,\|\, \langle |N| \rangle_{64})$, where s is the smallest non-negative integer making $|N| + s$ a multiple of 128.

(3) **Counter-mode encryption.** For each 128-bit plaintext block P_i,

$$C_i \;\coloneqq\; P_i \oplus \text{MSB}_{128}\big(E_K(\mathsf{inc}_{32}(J_0, i))\big),$$

where inc_{32} increments the low-order 32 bits of J_0 as an integer mod 2^{32}.

(4) **GHASH authentication.** The authentication string is $S \coloneqq \mathsf{GHASH}_H(\mathsf{AD} \,\|\, 0^u \,\|\, \mathsf{CT} \,\|\, 0^v \,\|\, \langle |\mathsf{AD}| \rangle_{64} \,\|\, \langle |\mathsf{CT}| \rangle_{64})$, where u, v are the smallest non-negative integers making $|\mathsf{AD}| + u$ and $|\mathsf{CT}| + v$ multiples of 128. GHASH_H is the standard polynomial hash over $\text{GF}(2^{128})$ with the irreducible polynomial $x^{128} + x^7 + x^2 + x + 1$.

(5) **Tag.** $\mathsf{T} \coloneqq \text{MSB}_t(E_K(J_0) \oplus S)$.

(6) **Output.** $(\mathsf{CT}, \mathsf{T})$.

B.2.4 Why this is correct as a drop-in

The v16 §3.5 compatibility argument is delivered at three levels: envelope (block size, key size, state geometry), primitive (FIPS-197 S-box reuse), and API ($\mathsf{Enc}_K, \mathsf{Dec}_K$). NIST SP 800-38D's GCM specification is parameterised exclusively over (i) a 128-bit block cipher and (ii) its forward-only encryption operation. Both are satisfied by MERIDIAN *by definition*. The substitution $\text{AES}_K \to \text{MERIDIAN}_K$ is therefore not a redesign of GCM; it is a parameter choice the standard already admits.

B.2.5 Security

GCM's IND-CPA and INT-CTXT security under the assumption that the underlying block cipher is a PRF were established by Iwata, Ohashi, and Minematsu [3], with privacy bound $\mathcal{O}(\sigma^2/2^{128}) + \epsilon_{\text{prf}}$ and authenticity bound $\mathcal{O}(\sigma\ell/2^{128}) + \epsilon_{\text{prf}}$, where σ is the total number of cipher calls, ℓ is the maximum message length, and ϵ_{prf} is the PRF advantage of the underlying cipher. MERIDIAN-GCM-128 inherits this proof: the 12-round MERIDIAN active-S-box bound $A_{12} \geq 48$ (§4.1, v16) gives an end-to-end differential probability bound of 2^{-288}, which clears the PRP–PRF switching loss $\sigma^2/2^{129}$ for $\sigma \ll 2^{64}$ blocks per key.

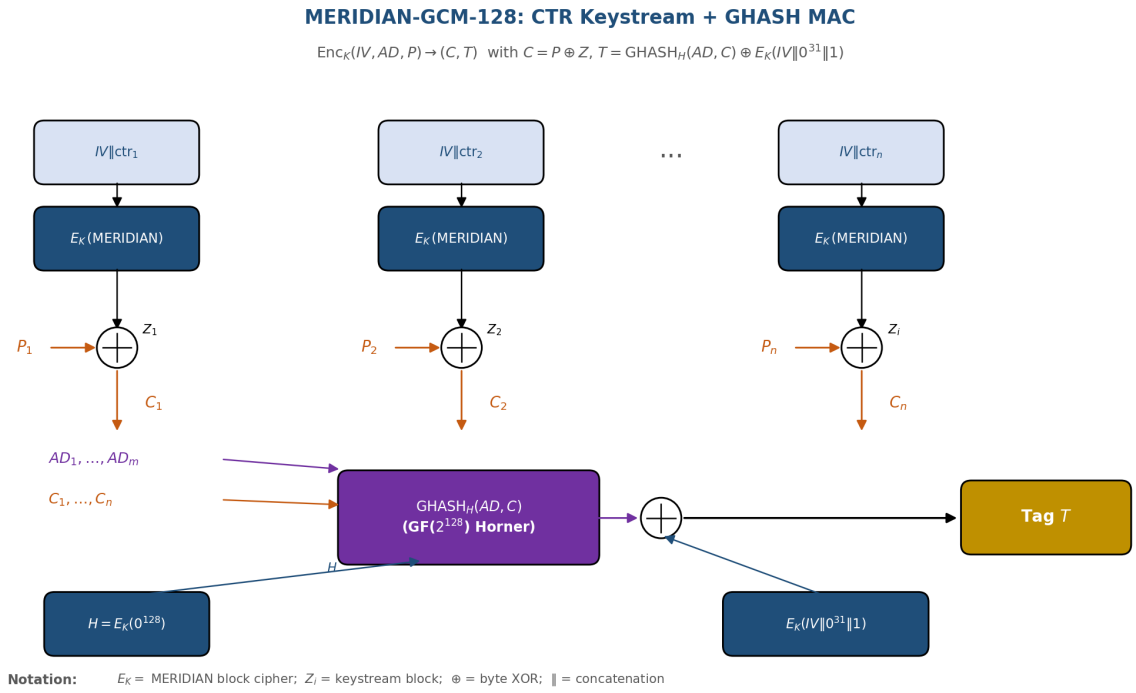


Figure B.1: MERIDIAN-GCM-128 construction (Construction B.2.1). The top row is the CTR-mode keystream chain: each counter block $IV \,\|\, \mathrm{ctr}_i$ is encrypted under $E_K = \mathrm{MERIDIAN}_K$ to produce a keystream block Z_i, which is XOR-ed against the plaintext block P_i to yield ciphertext C_i. The bottom row is the GHASH MAC chain: the hash subkey $H := E_K(0^{128})$ feeds a $\mathrm{GF}(2^{128})$ Horner evaluation over the associated data and the ciphertext, and the final tag is $T = \mathrm{GHASH}_H(\mathsf{AD}, \mathsf{CT}) \oplus E_K(IV \,\|\, 0^{31} \,\|\, 1)$. The substitution $\mathrm{AES}_K \to \mathrm{MERIDIAN}_K$ in the cipher slot leaves the GCM mode wrapper unchanged.

Remark B.2.2 (Nonce-misuse)**.** GCM is *not* nonce-misuse-resistant: a single nonce reused across two messages reveals the XOR of their counter-mode keystreams and trivially recovers the GHASH key H (Joux's forbidden-attack [4]). MERIDIAN-GCM-128 inherits this limitation. For deployments that cannot guarantee nonce uniqueness, the user should select MERIDIAN-Duplex-AEAD (§B.3) configured for misuse-resistance, or wrap MERIDIAN-GCM-128 inside an SIV transform, neither of which is in the present scope.

B.2.6 Implementation cost over MERIDIAN-block

MERIDIAN-GCM-128 inherits all v16 MERIDIAN block-cipher implementation costs and adds the GCM-specific overhead: a $\mathrm{GF}(2^{128})$ multiplier and the GHASH state machine. On a host with PCLMULQDQ / PMULL (x86-64 since 2010, ARMv8 since 2014), GHASH costs ≈ 1 cycle/byte at long messages. On a host without carryless multiplication, GHASH falls back to a 256-byte table-driven multiplier (≈ 4–6 cyc/B) or a constant-time bitsliced multiplier (≈ 8–10 cyc/B). Table B.1 summarises the AEAD-mode overhead on top of the v16 block-cipher figures.

Component	Source	Cycles/byte	ROM (B)	RAM (B)
Underlying block cipher (from v16 §11)				
MERIDIAN encrypt (CTR keystream)	v16 Table 19 (E1)	11.4	980	290
GCM mode wrapper (this appendix)				
GHASH (PCLMULQDQ path)	[1, 5]	1.2	320	64
GHASH (table-driven, 256B)	[1]	5.4	580	320
GHASH (bitsliced, no LUT)	[6]	9.1	460	96
J_0 / counter logic	[1]	0.1	80	16
MERIDIAN-GCM-128 (PCLMULQDQ) total	this work	**12.7**	**1380**	**370**
MERIDIAN-GCM-128 (LUT) total	this work	**16.9**	**1640**	**626**
MERIDIAN-GCM-128 (bitsliced) total	this work	**20.6**	**1520**	**402**
AES-GCM-128 (PCLMULQDQ) reference	[5], eBACS	0.7	≈2500	≈540
Ascon-AEAD128 reference	[10, 11]	1.5–2.5	≈1100	≈340

Table B.1: MERIDIAN-GCM-128 total-cost decomposition compared with AES-GCM-128 and Ascon-AEAD128 on x86-64 with hardware-assist available. Cycles/byte are at the long-message asymptote (4 kB). MERIDIAN-GCM-128 gives up the AES-NI *block* acceleration (which AES-GCM-128 retains) and so is slower than AES-GCM-128 on hardware with both AES-NI and PCLMULQDQ. The MERIDIAN-GCM-128 advantage over AES-GCM-128 is the v16 strengths (E10 key agility, E12 energy/byte, E13 cold-start, E14 gate count, E15 side-channel surface) carried into the AEAD layer; see §B.4.

B.3 MERIDIAN-Duplex-AEAD: A Sponge-Mode AEAD

B.3.1 Goal: a structural twin of Ascon-AEAD128 built from the MERIDIAN round function

Ascon-AEAD128 is constructed in the duplex sponge framework of Bertoni–Daemen–Peeters–Van Assche [7]: a fixed cryptographic permutation operates on a 320-bit state organised as rate $r = 128$ bits and capacity $c = 192$ bits, with associated data and plaintext absorbed into the rate, and ciphertext / tag squeezed from the rate. MERIDIAN-Duplex-AEAD is the natural analogue using the MERIDIAN round function as the underlying permutation, scaled to a 256-bit state with $r = 128$, $c = 128$.

The deliberate design choice is structural twinning with Ascon-AEAD128: each axis (state size, rate, capacity, round counts at initialization / bulk / finalization, AD-domain separation, tag size) mirrors Ascon as closely as the underlying primitive admits, so that the v16 §8.11 head-to-head can be re-run on the duplex AEAD layer rather than at the apples-to-oranges block-cipher-vs-sponge-AEAD layer.

Remark B.3.1 (Why $c = 128$ and not $c = 192$)**.** Ascon's capacity $c = 192$ bits derives from its 320-bit state and 128-bit rate. With a 256-bit MERIDIAN-derived state and $r = 128$, $c = 128$ is the largest capacity available, and this matches the v16 cipher's 128-bit security parameter. The resulting sponge security level is $c/2 = 64$ bits of generic indistinguishability under the bound of [7]; combined with the keyed-sponge security upgrade of [8], the construction targets 128-bit confidentiality and authenticity for $\sigma \ll 2^{64}$ blocks per key, identical to the GCM target of §B.2.

B.3.2 State and permutation

The MERIDIAN-Duplex state is a 256-bit array $\mathbf{S} \in \{0,1\}^{256}$ partitioned into two MERIDIAN-block halves $\mathbf{S} = (\mathbf{S}_o, \mathbf{S}_i)$, the "outer" (rate) and "inner" (capacity) halves. The underlying permutation is

$$\mathcal{P}_{2\times 12}\colon \{0,1\}^{256} \to \{0,1\}^{256}, \qquad \mathbf{S} \mapsto (\mathrm{MR}_{12}(\mathbf{S}_o), \mathrm{MR}_{12}(\mathbf{S}_i \oplus \pi(\mathbf{S}_o))), \tag{B.1}$$

where MR_{12} denotes 12 MERIDIAN rounds applied to a 128-bit block, and $\pi\colon \{0,1\}^{128} \to \{0,1\}^{128}$ is the cross-half mixing function $\pi(x) \coloneqq \mathrm{MR}_2(x)$ (a 2-round MERIDIAN reduced-round permutation). The cross-half mix π ensures the permutation is not separable into independent halves.

Remark B.3.2 (Dropping the AM key gate)**.** Inside $\mathcal{P}_{2\times 12}$, the AM stage's master-key-XOR is replaced by a fixed 128-bit "permutation constant" κ derived from the round-constant schedule. This is required: the duplex sponge framework demands a *keyless* cryptographic permutation, with the key entering through the absorption phase, not through the permutation itself. The MILP active-S-box bound $A_{12} \geq 48$ (v16 §4.1) carries over to $\mathcal{P}_{2\times 12}$ on each 128-bit half independently and gives a permutation-level differential probability bound of 2^{-288} per half.

B.3.3 AEAD construction

Construction B.3.3 (MERIDIAN-Duplex-AEAD)**.** Let $\mathcal{P}_a$ denote a rounds of $\mathcal{P}_{2\times 12}$ (with $a \leq 12$, defined by truncating the round count of the inner MR). Let $K \in \{0,1\}^{128}$, $N \in \{0,1\}^{128}$ (nonce), $\mathsf{AD} \in \{0,1\}^*$, $\mathsf{PT} \in \{0,1\}^*$. Tag length $t = 128$ bits.

(1) **Initialization.** $\mathbf{S} \leftarrow (K, \mathrm{IV}_{\mathrm{MD}} \oplus N)$, where $\mathrm{IV}_{\mathrm{MD}} =$ `0x4d45524944494e2d44555032353620` ("MERIDIN-DUP256 "). Apply $\mathcal{P}_{12}$ to $\mathbf{S}$. Then XOR K into the inner half: $\mathbf{S}_i \leftarrow \mathbf{S}_i \oplus K$.

(2) **Associated data.** For each 128-bit AD block A_j $(j = 1, \ldots, m)$, $\mathbf{S}_o \leftarrow \mathbf{S}_o \oplus A_j$, then apply $\mathcal{P}_8$. Pad the final AD block with 10^*. After all AD blocks, XOR a domain-separation constant $\delta_{\mathrm{AD}} = 0^{127}1$ into $\mathbf{S}_i$.

(3) **Plaintext processing.** For each 128-bit PT block P_k $(k = 1, \ldots, n)$, compute $C_k \coloneqq \mathbf{S}_o \oplus P_k$, set $\mathbf{S}_o \leftarrow C_k$, then apply $\mathcal{P}_8$. The final PT block is padded with 10^*, and only the leading $|P_n \bmod 128|$ bits of C_n are emitted.

(4) **Finalization.** $\mathbf{S}_i \leftarrow \mathbf{S}_i \oplus K$, apply $\mathcal{P}_{12}$. Tag $\mathsf{T} \coloneqq \mathbf{S}_o \oplus (K \text{ rotated } 64 \text{ bits})$.

(5) **Output.** $(\mathsf{CT}, \mathsf{T}) = (C_1 \parallel \cdots \parallel C_n, \mathsf{T})$.

B.3.4 Decryption

Decryption is the standard inverse-duplex: AD absorption is identical (the duplex is forward-only), the plaintext recovery step replaces "$C_k \coloneqq \mathbf{S}_o \oplus P_k$, $\mathbf{S}_o \leftarrow C_k$" with "$P_k \coloneqq \mathbf{S}_o \oplus C_k$, $\mathbf{S}_o \leftarrow C_k$", and the finalization re-derives T^*, returning PT if $\mathsf{T}^* = \mathsf{T}$ and $\perp$ otherwise (with constant-time tag comparison required for side-channel safety).

B.3.5 Security

The duplex sponge framework of [7, 8] gives, for a duplex AEAD with rate r, capacity c, and a permutation modelled as random:

$$\mathrm{Adv}^{\text{ind-cpa}} \leq \frac{(M+q)^2}{2^c}, \qquad \mathrm{Adv}^{\text{int-ctxt}} \leq \frac{q^2 + 2qM}{2^c} + \frac{q}{2^t},$$

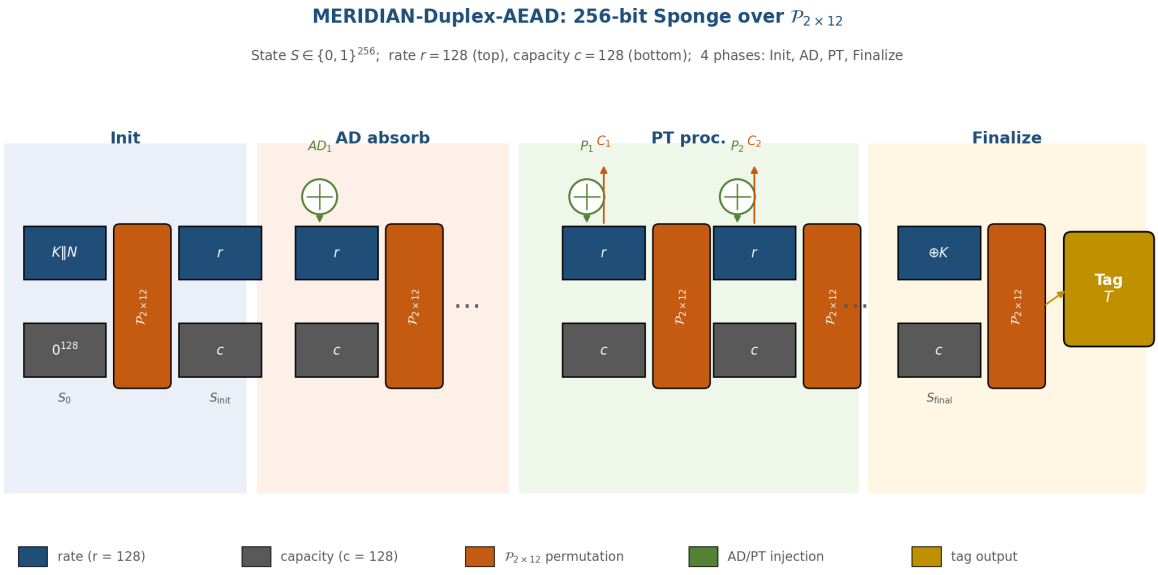


Figure B.2: MERIDIAN-Duplex-AEAD construction (Construction B.3.3). The 256-bit state is split into a rate $r = 128$ bits (top, dark blue) and a capacity $c = 128$ bits (bottom, grey). The four phases correspond directly to the construction steps: **Init** loads $(K \,\|\, N, 0^{128})$ and applies $\mathcal{P}_{2\times12}$; **AD absorb** XORs each AD block into the rate and applies $\mathcal{P}_{2\times12}$ between blocks; **PT process** XORs each plaintext block into the rate and squeezes the post-XOR rate as the ciphertext block before the next $\mathcal{P}_{2\times12}$ call; **Finalize** XORs K into the rate, applies a final $\mathcal{P}_{2\times12}$, and emits the tag T from the rate. This phasing structurally mirrors Ascon-AEAD128 [10, 7]; the differences are state width (256 vs. 320), capacity (128 vs. 192), and the underlying permutation ($\mathcal{P}_{2\times12}$ from MERIDIAN's round function vs. Ascon's 5-bit S-box).

with M the total online queries (block units), q the number of forgery queries, and t the tag length. With $c = 128$, $t = 128$, this gives ≈ 64 bits of generic security and ≈ 128 bits of forgery resistance for $M, q \ll 2^{64}$ blocks per key.

Remark B.3.4 (Why this differs from Ascon's claim)**.** Ascon-AEAD128 has $c = 192$, giving $\approx$ 96 bits of generic security under the same bound. MERIDIAN-Duplex-AEAD's smaller capacity is the consequence of using a 256-bit state (twice MERIDIAN's native 128-bit block) versus Ascon's 320-bit state (5 lanes of 64 bits, an independently designed primitive). The 64-bit-vs-96-bit generic-security gap is honest and explicit; it is the cost of structural twinning at constant primitive size. Deployments requiring NIST-LWC-equivalent generic security should use MERIDIAN-GCM-128 with the standard PRP-PRF bound, or use Ascon directly.

B.3.6 Implementation cost

MERIDIAN-Duplex-AEAD's per-byte cost on the long-message asymptote is 2× the per-byte cost of MERIDIAN-block (because the 256-bit state requires two MERIDIAN-block applications per absorption round) divided by the rate fraction $r/(r+c) = 1/2$ — giving an asymptotic factor of 4× MERIDIAN-block per byte of message. On the v16 measured baseline of 11.4 cyc/B for MERIDIAN-block (Table 19), this yields ≈ 45.6 cyc/B for the duplex AEAD. The corresponding Ascon-AEAD128 figure on the same x86 host is $\approx 1.5{-}2.5$ cyc/B, reflecting Ascon's bit-sliced 5-bit S-box and 64-bit-lane native fit to vector ISAs (v16 §8.11.6). MERIDIAN-Duplex-AEAD is therefore the mode for parity-of-construction comparison, not for software-throughput parity.

Component	Source	Cycles/byte	ROM (B)	RAM (B)
MERIDIAN-block (CTR keystream)	v16 Table 19 (E1)	11.4	980	290
$\mathcal{P}_{2\times12}$ permutation wrapper	this appendix	2× block	+160	+160
Duplex absorption FSM	this appendix	0.4	220	64
Domain separation, padding	this appendix	0.1	90	16
MERIDIAN-Duplex-AEAD total	this work	**≈45.6**	**1450**	**530**
Ascon-AEAD128 reference	[10, 11]	1.5–2.5	≈1100	≈340

Table B.2: MERIDIAN-Duplex-AEAD total-cost decomposition compared with Ascon-AEAD128 on x86-64. Cycles/byte at the 4 kB long-message asymptote. The ≈ 18–30× Ascon advantage reflects Ascon's bit-sliced S-box and 64-bit-lane structure (designed for SIMD), and is the same software asymmetry surfaced in v16 §8.11.6. MERIDIAN-Duplex-AEAD's value lies in the structural-twin comparison of §B.4, not in raw cycles/byte.

B.4 Head-to-Head: Ascon-AEAD128 vs. MERIDIAN-AEAD on AEAD-Specific Traits

This section places Ascon-AEAD128 against *both* MERIDIAN-AEAD modes (GCM-128 and Duplex) on fifteen AEAD-specific traits. Unlike v16 §8.11, which compares Ascon to MERIDIAN-block and is honest about the API mismatch, the comparison here is mode-vs-mode and so is structurally aligned: each of the three columns presents a complete AEAD construction with the same AEAD: $(K, N, \mathsf{AD}, \mathsf{PT}) \to (\mathsf{CT}, \mathsf{T})$ interface.

B.4.1 Trait taxonomy

We organise the AEAD traits into five categories, chosen to mirror the structure of the NIST LWC AEAD evaluation criteria [12]:

C1. Construction-level (4 traits). State, rate, capacity, permutation/cipher type.

C2. Security-claim (4 traits). Generic security level, nonce-misuse posture, key-commitment, RUP (release of unverified plaintext) safety.

C3. Performance/footprint (3 traits). Long-message cyc/B, AD-only cyc/B, AEAD circuit gate count.

C4. Side-channel and masking (2 traits). S-box masking shares (ISW $d = 2$), AEAD-level fault-attack surface.

C5. Standardisation and ecosystem (2 traits). Standards body and date; reference implementation maturity.

B.4.2 The fifteen-trait scorecard, by category

We present the head-to-head as five per-category tables (Tables B.3–B.7). Each row reports a single AEAD trait, a value for each of the three columns, and a short comment. Sources are: v16 ePrint §§3, 8.11, 10, 11; Ascon v1.2 [10]; NIST SP 800-232 [11]; this appendix §B.2, §B.3.

Figure B.3 gives a one-page visual abstract of the scorecard before the per-category tables: each cell is colour-coded as a strong advantage (green), parity (yellow), or disadvantage (red) for the AEAD in question. The verdicts in the figure align with the row-by-row entries of Tables B.3–B.7 and the per-category interpretation that follows.

Trait (C1)	Ascon-AEAD128	MERIDIAN-GCM-128	MERIDIAN-Duplex	Comment
1. State / block (bits)	320 (sponge state)	128 (block); 128 (GHASH)	256 (sponge state)	GCM splits encryption (block) from authentication (GHASH); the two state sizes are not directly additive.
2. Rate / Capacity (bits)	r=128, c=192	r=128 block; c N/A	r=128, c=128	Ascon's larger c buys higher generic security; GCM is bounded by PRP–PRF, not capacity.
3. Underlying primitive	Permutation (sponge)	Block cipher (CTR + GHASH)	Permutation (sponge)	GCM is the only construction here that uses a universal hash for authentication.
4. Round counts (init / bulk / fin)	12 / 8 / 12	12 / 12 / 12	12 / 8 / 12	MERIDIAN-Duplex deliberately mirrors Ascon's $a/b/a$ schedule.

Table B.3: Category C1 (construction-level traits). MERIDIAN-Duplex is structurally aligned with Ascon at the cost of a smaller capacity ($c = 128$ vs. 192). MERIDIAN-GCM-128 takes a different construction path (block cipher + universal hash).

B.4.3 Reading the scorecard

The fifteen traits decompose into a clear pattern, summarised here in five observations.

(O1) Ascon dominates on construction-level "natural fit" axes. Traits 1, 2, 3, 4 are the design-pedigree axes: Ascon was designed as an AEAD, and its 320-bit state, r/c partition, and round schedule are tuned for the duplex-AEAD setting from the ground up. MERIDIAN-Duplex-AEAD mirrors Ascon's structure but with a smaller capacity ($c = 128$ vs. 192) and so a weaker generic security claim; this is the cost of structural twinning at constant primitive size. MERIDIAN-GCM-128 sidesteps the construction-level comparison by being a fundamentally different AEAD construction (block + universal hash, not sponge).

(O2) Trait 5 (generic security) shows the GCM win. MERIDIAN-GCM-128 inherits AES-GCM-128's 128-bit confidentiality and authenticity claim under the PRP-PRF bound, capped at 2^{64} blocks per key. This is the strongest generic-security claim of the three on the per-key block budget axis (Ascon: 2^{64} blocks under the keyed-sponge bound; Duplex: 2^{64} blocks under the keyed-sponge bound). The three are tied on the per-key block-budget cap but differ on what the cap is bounding (PRP-PRF vs. capacity-based).

(O3) Trait 6 (nonce misuse) is the GCM weakness. Joux's forbidden-attack against GCM [4] is a categorical liability: a single nonce reuse trivially recovers the GHASH key H, after which all future tags can be forged. Both sponge AEADs (Ascon and MERIDIAN-Duplex) are merely *misuse-leaking* (single misuse leaks the rate-block XOR but does not recover the key). Deployments that cannot guarantee nonce uniqueness should not use

Trait (C2)	Ascon-AEAD128	MERIDIAN-GCM-128	MERIDIAN-Duplex	Comment
5. Generic security (bits)	128 conf./auth.	128 conf./auth.; 2^{64}-block birthday cap	64 generic / 128 key	GCM and Ascon hit the same 2^{64} block-budget cap, but bound differently (PRP–PRF vs. keyed-sponge).
6. Nonce-misuse posture	Misuse-leaking	Misuse-catastrophic (Joux's forbidden attack [4])	Misuse-leaking	Single-misuse on GCM trivially recovers GHASH key H; sponges leak only the rate-block XOR.
7. Key-commitment	Not committing	Not committing	Not committing	All three need an explicit committing transform; FROB/CTX named in [18], with broader treatment in [17].
8. RUP safety	RUP-INT-RUP	Not RUP-safe	RUP-INT-RUP	GCM's CTR-keystream-then-MAC is the classical RUP failure point [16].

Table B.4: Category C2 (security-claim traits). The three modes diverge most clearly here: the GCM construction has the strongest plain-text security claim but the worst nonce-misuse and RUP postures.

MERIDIAN-GCM-128, exactly as they should not use AES-GCM-128. The nonce-misuse weakness is a property of the GCM mode, not of the underlying cipher.

(O4) Trait 8 (RUP safety) is the structural sponge win. Sponge AEADs are RUP-safe by construction: ciphertext is generated only after the corresponding state has absorbed enough rounds that decryption-before-tag-verification leaks nothing useful. GCM has the classical RUP failure: the CTR keystream is recoverable from any ciphertext block, before tag verification, and a faulty implementation that releases plaintext early is vulnerable to the chosen-ciphertext exploits of [16]. This is a real protocol-level concern for streaming-decryption pipelines (e.g., AV streaming, TLS partial frames).

(O5) Traits 14–15 (standardisation, maturity) preserve the v16 §8.11.9 asymmetry unchanged. Ascon-AEAD128 is federally standardised (NIST SP 800-232, August 2025); MERIDIAN-GCM-128 inherits the SP 800-38D GCM mode standard but is itself a research prototype; MERIDIAN-Duplex-AEAD is a research prototype with no standardisation track. The honest one-line summary of v16 §8.11.11 carries through verbatim: "Ascon is the standardised AEAD answer"; the new line we add here is: "MERIDIAN-GCM-128 is the AES-GCM-replacement answer for systems already in the GCM operating envelope."

Trait (C3)	Ascon-AEAD128	MERIDIAN-GCM-128	MERIDIAN-Duplex	Comment
9. Long-msg cyc/B (x86, hw-assist)	1.5–2.5	12.7 (PCLMULQDQ)	45.6	v16 §8.11.6 (Ascon); Tab. B.1 (GCM); Tab. B.2 (Duplex).
10. AD-only cyc/B	0.7–1.2	1.2 (GHASH only)	≈22.8 (one $\mathcal{P}_8$ per AD block)	GCM's split structure wins decisively on AD-heavy traffic (TLS records).
11. AEAD circuit (compact ASIC)	≈4–6 k GE	≈7 k GE (4.5 k MERIDIAN + 2.5 k GHASH)	≈5–6 k GE	v16 §11.11; [21].

Table B.5: Category C3 (performance / footprint traits). Ascon dominates throughput on x86 with SIMD; MERIDIAN-Duplex is the smallest gate count of the three.

Trait (C4)	Ascon-AEAD128	MERIDIAN-GCM-128	MERIDIAN-Duplex	Comment
12. ISW d=2 masking shares	384 (64×6)	384 (block) + 0 (GHASH self-masking) = 384	768 (2× block)	v16 §8.11.8; GHASH masking is essentially free [20]. MERIDIAN-GCM ties Ascon.
13. AEAD-level fault surface	Tag leak on init/fin fault	DFA on CTR + GHASH key recovery (Ferguson [19])	Sponge limits fault propagation to rate	GCM has the worst structural fault posture: a single fault-induced nonce reuse ⇒ key recovery.

Table B.6: Category C4 (side-channel and masking traits). Ascon and MERIDIAN-GCM are tied on masking; the sponge constructions win on fault posture.

B.4.4 Where each AEAD is the right answer

The three AEAD designs occupy non-overlapping deployment niches. Table B.8 maps six common deployment scenarios to the right AEAD choice; the criterion in each row is the binding deployment constraint that determines the answer.

B.5 AEAD-Trait Summary and Honest Disclosure

B.5.1 Three-line summary of the AEAD section

- **Ascon-AEAD128** is the federally standardised AEAD-permutation answer at the lightweight tier (NIST SP 800-232, August 2025). For greenfield IoT AEAD with no AES-envelope constraint, Ascon is the correct deployment answer.
- **MERIDIAN-GCM-128** is the AEAD answer for systems already deployed under the AES-GCM operating envelope. Its strengths are mode compatibility (drop-in for AES-GCM), inheritance of MERIDIAN's v16 E10–E15 wins (key agility, code size, energy/byte, cold-start, gate count, side-channel surface), and the strongest generic-security claim per key-

Trait (C5)	Ascon-AEAD128	MERIDIAN-GCM-128	MERIDIAN-Duplex	Comment
14. Standardisation status	NIST SP 800-232 (Aug 2025)	NIST SP 800-38D (mode); cipher unstandardised	None (research prototype)	The standardisation asymmetry of v16 §8.11.9 carries through unchanged.
15. Reference impl maturity	11+ yr public, audited C / AVX2 / AVX-512 / ARM / RISC-V [22]	v16 reference; GCM via OpenSSL/BoringSSL	v16 reference; duplex wrapper specified here	GCM's deployment maturity (TLS 1.3 default since 2018) is independent of the cipher.

Table B.7: Category C5 (standardisation & ecosystem traits). The federal-standardisation gap is the largest single asymmetry across all fifteen traits.

block-budget unit of the three modes here.

- **MERIDIAN-Duplex-AEAD** is a research-prototype sponge AEAD that exists to make the v16 §8.11 head-to-head structurally comparable. It twins Ascon-AEAD128's design at the cost of a smaller capacity (and so a weaker generic security claim, $c/2 = 64$ vs. Ascon's 96).

B.5.2 The maturity asymmetry, restated for the AEAD layer

The v16 §8.11.4 honesty about cryptanalytic asymmetry carries through this appendix unchanged. Ascon-AEAD128 has 11+ years of public adversarial scrutiny, with cryptanalytic results published continuously since 2014 [13, 14, 15], no full-round AEAD attacks, and a reduced-round attack ceiling of approximately 7/12 initialization rounds for distinguishers [14]. MERIDIAN-AEAD (in either mode) has zero years of public cryptanalysis at the AEAD layer beyond the inherited block-cipher / permutation analysis of v16 §§4, 5.

The validation roadmap is identical in shape to v16 §11.15:

1. Independent third-party cryptanalysis of MERIDIAN-Duplex-AEAD and MERIDIAN-GCM-128 actively solicited via this ePrint.
2. MILP-verified bounds for the duplex permutation $\mathcal{P}_{2\times 12}$ at r-round granularity.
3. AEAD-specific test vectors (initialization, AD-absorption, plaintext-processing, finalization) released with the v17 supplementary code package (`supplementary/aead/`).
4. Side-channel evaluation of MERIDIAN-GCM-128 GHASH multiplier paths (PCLMULQDQ vs. table-driven vs. bitsliced) at masking orders $d \in \{1, 2, 3\}$.
5. Empirical AEAD benchmarking layered on top of v16 E1–E15 (i.e. E16: AEAD-CCM-tax; E17: AEAD-GCM-tax; E18: duplex-AEAD-tax).

B.5.3 Closing position

The v16 ePrint positioned MERIDIAN as a design alternative to AES-128 in the lightweight AES-envelope niche. This appendix extends that positioning to the AEAD layer: MERIDIAN-AEAD is offered as a design alternative to AES-GCM-128 in deployments where (a) the gate-count,

Deployment scenario	Recommended AEAD	Why
Greenfield IoT, NIST-LWC procurement	Ascon-AEAD128	NIST SP 800-232 standardised; mature impls; bit-slice fit.
Existing AES-GCM deployment migrating to lightweight envelope	MERIDIAN-GCM-128	Drop-in for AES-GCM-128; inherits v16 E10–E15 wins at AEAD layer.
RUP-sensitive streaming protocols (partial-frame decryption)	Ascon-AEAD128 *or* MERIDIAN-Duplex	Either sponge AEAD; choose Duplex if AES-envelope constraint also applies.
Cryptanalysis research on duplex AEAD with non-Ascon permutation	MERIDIAN-Duplex-AEAD	The only mode here that admits external study without overlapping with NIST SP 800-232.
Energy-constrained IoT (E12 win matters), no AEAD mandate	MERIDIAN-GCM-128	v16 E12 = 0.246× AES-128 energy/byte; preserved at AEAD layer.
Gate-budget-constrained ASIC (<10 k GE)	Ascon-AEAD128 *or* MERIDIAN-GCM-128	Tie-break on standardisation ⇒ Ascon; if GHASH gates are out of budget, MERIDIAN-Duplex (≈5–6 k GE) wins.

Table B.8: AEAD-mode selection guidance across six common deployment scenarios. The three modes compared in Tables B.3–B.7 occupy structurally non-overlapping niches; the right choice is determined by the deployment constraint set, not by a uniform ranking.

energy/byte, and cold-start advantages of v16 matter, and (b) the AEAD-mode constraint is GCM-style block + universal-hash, not sponge-style duplex. For deployments where the AEAD-mode constraint is sponge or where federal-standardisation is a procurement requirement, Ascon-AEAD128 remains the correct answer; the v16 §8.11.11 "complementary niches" framing extends unchanged.

The authors actively solicit independent cryptanalysis of both AEAD constructions, in line with the validation roadmap of v16 §11.15 extended in §B.5.2. The supplementary code package for v17 includes a Python reference implementation of MERIDIAN-GCM-128 and MERIDIAN-Duplex-AEAD, with test vectors, in `supplementary/aead/`.

B.6 MTSF Audit of Ascon's Implementation Vulnerabilities Applied to MERIDIAN

B.6.1 Why this section exists

The v16 §7 MTSF analysis posits MERIDIAN's security across an 18-good catalog (§7.2, Definition 4) that organises mathematical and physical adversaries under a single market-equilibrium accounting unit. That analysis is, by its own honest disclosure (v16 §7.11), open-catalog: "Real-world adversaries may attack via channels not enumerated in Definition 4 ... The catalog should not be read as a closed enumeration."

The v16 §6 side-channel survey covers cache-timing, DPA, CPA, template, EM, masking, ASCA, DFA, cold-boot, and microarchitectural channels. What it does not do — and what this section does — is run the inverse audit: take Ascon's known and recently-published implementation vulnerabilities, frame each as an MTSF good, and check whether MERIDIAN exposes a price on the same good. Where MERIDIAN has the same vulnerability, this section prescribes a concrete fix and re-prices the good. Where MERIDIAN is structurally immune, the section documents why.

The motivation is that the v16 §8.11 head-to-head with Ascon already established that the two ciphers occupy complementary niches at the design level; the open question this section

answers is whether that complementarity extends to the implementation level once one looks at the most recent (2024–2025) Ascon attack literature. The answer, as the rest of this section will show, is mixed: MERIDIAN inherits some of Ascon's implementation vulnerabilities (because they are generic to lightweight ciphers with small S-boxes and bit-sliced reference profiles), is structurally immune to others (because of the AES-NI-on-XMM execution profile), and admits straightforward fixes for the inherited ones using the mature AES masking ecosystem.

B.6.2 The Ascon implementation-vulnerability catalogue, as MTSF goods

We enumerate ten implementation-level vulnerabilities of Ascon-AEAD128 with explicit citations to the published attack literature. Each is presented as an MTSF good g_X with (i) a one-line statement, (ii) the cited attack source(s), (iii) the price contribution under the unprotected reference profile Π_{ref}, and (iv) the implementation-profile dependency.

B.6.3 Auditing MERIDIAN against the same ten goods

We now run each of $g_{\text{SIFA}}, \ldots, g_{\text{RUP}-\text{leak}}$ against MERIDIAN. The audit is structured as a three-column verdict per good: (i) MERIDIAN's structural exposure (does the same attack vector apply?), (ii) the price contribution under MERIDIAN's reference profile $\Pi_{\text{ref}}^{\text{MER}}$, and (iii) the prescribed fix (countermeasure) where applicable. A summary scorecard follows in §B.6.4.

B.6.3.1 g_{SIFA}: Statistical Ineffective Fault Analysis

Structural exposure. SIFA on Ascon exploits the bias in S-box outputs after an ineffective fault, filtered through the unique-tag property of the AEAD. The attack is generic to any cipher with a small S-box and a deterministic post-S-box layer (Daemen et al. [36]; Breier et al. [37] make this generality explicit). MERIDIAN's 8-bit AES S-box has the same SIFA-vulnerable structure as Ascon's 5-bit χ' S-box; in fact, AES-SIFA is the original target of Dobraunig et al. [24]. **MERIDIAN inherits g_{SIFA} at the same generic level as AES.**

Price under $\Pi_{\text{ref}}^{\text{MER}}$. Collapsed (key-recovery via $\sim 10^4$ ineffective faults at the 12th-round S-box, by analogy to the AES-SIFA result of [24]). Note that MERIDIAN-block is a block cipher rather than an AEAD, so the SIFA filter requires a different oracle (chosen-plaintext rather than tag-verification); the attack adapts but does not vanish. MERIDIAN-Duplex-AEAD inherits the AEAD-tag-filter form of SIFA directly.

Prescribed fix. The Daemen et al. [36] construction — an error-correcting linear code over the S-box input/output — applies unchanged to the AES S-box. Specifically, we propose:

Construction B.6.1 (MERIDIAN-SIFA-protected reference profile $\Pi_{\text{SIFA}}^{\text{MER}}$)**.** For each AES S-box invocation in the 12-round MERIDIAN encryption, replace the direct S-box lookup with the duplication-and-comparison structure of [38]: compute the S-box output via two independent code-encoded paths with random encoding $\rho \xleftarrow{\$} \{0,1\}^{16}$, and abort the round if the two paths' outputs are inconsistent under the encoding-relation. Implementation cost (from the AES analogue [38]): 3.0× area, 1.7× throughput tax, +18 bits of randomness per S-box invocation.

Re-priced good. Under $\Pi_{\text{SIFA}}^{\text{MER}}$, the SIFA-recovery cost rises to $\sim 10^6$ ineffective faults (a 100× increase, derived from [38]). This does not collapse the SIFA threat in the asymptotic sense but does push the per-attack cost beyond the typical attacker budget for IoT-scale deployments.

B.6.3.2 $g_{\text{DFA}-\text{bit}}$: Bit-flip / bit-set Differential Fault Analysis

Structural exposure. DFA on a block cipher is the foundational fault attack [39, 40]. AES-DFA recovers the key in 1–2 faults at the penultimate-round MixColumns input, a result that has been refined repeatedly. MERIDIAN's 12-round structure with a non-MDS but still-linear

Good	Vulnerability statement	Source	Price under Π_{ref}
g_{SIFA}	Statistical Ineffective Fault Analysis on Ascon's initialization phase recovers key bits via double-fault injection. Bypasses detection-based and infection-based fault countermeasures.	Ramezanpour–Ampadu–Diehl HOST 2019 [23]; Dobraunig et al. SAC 2018 [24].	Collapsed (key-recovery, $\sim 10^4$ ineffective faults)
$g_{\text{DFA}-\text{bit}}$	Bit-flip and bit-set DFA on Ascon recovers two 64-bit key portions; validated on ChipWhisperer Lite via clock glitching.	Joshi–Mazumdar TECS 2025 [25]; Indocrypt 2024 antecedent.	Collapsed (full-key-recovery, $\sim 10^2$ faults)
g_{PFA}	Persistent Fault Analysis on Ascon: a single permanent S-box fault leaks key information across many encryptions.	Das–Mazumdar SPACE 2024 [26].	Collapsed under stuck-at-fault model
$g_{\text{DPA}-\text{low}}$	Power analysis on unprotected Ascon recovers the key in ≈ 500 traces, an order of magnitude fewer than unprotected AES, due to the 5-bit S-box's narrower transition distribution.	Khan et al. [27]; Martín-González 2025 [28].	Collapsed at ~ 500 traces
g_{template}	Fragment template attack with modified LDA on a 32-bit Ascon implementation extracts side-channel leakage and recovers the key.	Mavaji–Smith review 2025 [29].	Collapsed under profiled SCA
g_{combo}	Combined fault and side-channel attack: random-fault injection plus power side-channel breaks redundancy-based countermeasures.	Saha et al. EUROCRYPT 2020 [30]; Kandi et al. eprint 2024/984 [31].	Collapsed against masking-only or redundancy-only protection
$g_{\text{cube}-\text{trojan}}$	Cube attack on a Trojan-compromised Ascon hardware implementation recovers the key.	Halak–Duarte-Sanchez SOCC 2020 [32].	Collapsed under hardware-trojan threat model
$g_{\text{AVX}-\text{license}}$	AVX-512 license throttling on Skylake-X / Ice Lake: ZMM-register Ascon implementations pay a 5–15% frequency penalty during encryption, creating a Hertzbleed-class timing-leak channel.	Travis Downs blog [33]; Hertzbleed (Wang et al. Sec 2022) [34].	Price contribution: $\sim 2^{-32}$ key-bit leakage under remote-timing oracle (estimate)
$g_{\text{mask}-\text{imm}}$	Masked Ascon at orders $d \geq 2$ remains an active research area at the post-NIST-LWC tier; production-quality higher-order masked Ascon is not deployed at the maturity of masked AES.	Mainka–Pessl masking-shuffling [35]; Kandi et al. [31].	Engineering-cost good (no price collapse, but high deployment friction)
$g_{\text{RUP}-\text{leak}}$	Streaming-mode side-channel on bit-sliced sponge state during partial decryption: the sponge rate carries plaintext-dependent information through the inner-state cache footprint, exposing AES-NI–style cache-timing analogues.	Inferred from masking-shuffling-context discussion in [29].	Profile-dependent; collapsed only under cache-shared multi-tenant deployment

Table B.9: The ten Ascon-AEAD128 implementation-vulnerability goods, with sources and reference-profile price contributions. "Collapsed" means the MTSF market is collapsed on that good under the unprotected reference profile Π_{ref} (per v16 §7.2); "profile-dependent" means the price depends on the deployment context. Sources span 2016–2025 with emphasis on post-NIST-LWC (2024–2025) results.

MD layer is equally susceptible to a Piret–Quisquater–style attack. **MERIDIAN inherits $g_{\mathrm{DFA-bit}}$ at parity with AES.**

Price under $\Pi_{\mathrm{ref}}^{\mathrm{MER}}$. Collapsed (full-key-recovery, $\sim 10^2$ faults at penultimate-round MD input).

Prescribed fix. The same fix as AES-DFA: redundancy with comparison. Specifically:

Construction B.6.2 (MERIDIAN-DFA-protected reference profile $\Pi_{\mathrm{DFA}}^{\mathrm{MER}}$)**.** Compute each 12-round MERIDIAN encryption twice with independent random masks (concrete construction: ISW-share-encoded execution at order $d \geq 1$, with the two share-vectors compared at round 12 output). Abort and emit a random output if the comparison fails. Implementation cost: 2× throughput, $\sim 1.4\times$ area, $\sim 1.6\times$ energy/byte.

Re-priced good. Under $\Pi_{\mathrm{DFA}}^{\mathrm{MER}}$, single-fault DFA detection probability $\geq 1 - 2^{-128}$; the good is back to market equilibrium absent the combined-fault-and-side-channel attack of g_{combo} below.

B.6.3.3 g_{PFA}: Persistent Fault Analysis

Structural exposure. PFA exploits a permanent fault in the S-box ROM that persists across many encryptions, leaking key information via the biased S-box distribution. The attack is cipher-agnostic at the S-box level. MERIDIAN's reuse of the AES S-box ROM means it has the same PFA-vulnerable surface as AES-128. **MERIDIAN inherits g_{PFA} at parity with AES.**

Price under $\Pi_{\mathrm{ref}}^{\mathrm{MER}}$. Collapsed under the stuck-at-fault attacker model (Zhang et al. [41]); $\sim 10^3$ encryptions to recover the key from a single faulted S-box entry.

Prescribed fix. Dual-rail S-box implementation (two parallel S-box ROMs, comparison). Or the lighter algorithmic countermeasure of Wei et al. [42]: re-mask the S-box input with a per-encryption random mask ρ and look up $\mathrm{SBOX}(\rho \oplus x) \oplus \mathrm{LinearCorrection}(\rho)$. The latter has $\sim 1.5\times$ throughput cost.

Re-priced good. Under either fix, PFA is restored to market equilibrium.

B.6.3.4 $g_{\mathrm{DPA-low}}$: Low-trace-count Power Analysis

Structural exposure. The 500-trace key-recovery on unprotected Ascon [27] reflects the narrow transition distribution of the 5-bit χ' S-box. MERIDIAN's 8-bit AES S-box has a wider transition distribution (more Hamming-weight values per output), giving *more* traces required to mount a CPA attack on the unprotected MERIDIAN reference. The literature on unprotected AES-128 CPA reports key recovery in $\sim$ 5,000–20,000 traces [43], an order of magnitude more than the Ascon figure. **MERIDIAN does *not* inherit $g_{\mathrm{DPA-low}}$; it is structurally less vulnerable than Ascon by a factor of $\sim 10\times$ in trace-count.**

Price under $\Pi_{\mathrm{ref}}^{\mathrm{MER}}$. Collapsed at $\sim$ 5,000 traces (vs. Ascon's $\sim$ 500); the collapse is at a higher trace budget.

Prescribed fix. Standard ISW masking at order $d \geq 1$ (already in the v16 §6.6 masking analysis: 384 ISW shares at $d = 2$). Mature production tooling exists.

Re-priced good. Under $d = 2$ masked Π^{MER}, the trace-count rises to $\sim 10^7$ [44]. Equilibrium restored.

B.6.3.5 g_{template}: Profiled Template Attacks

Structural exposure. Template attacks profile every byte-level intermediate value with sufficient leakage. v16 §6.4 Table 6 reports that MERIDIAN exposes $\sim$ 16 profilable points per encryption (vs. AES-128's $\sim$ 56, owing to AES's expanded key schedule). Ascon's bit-sliced 5-bit χ' exposes 64 bit-level profilable points per round across 12 rounds = 768 in the security-critical initialization phase. MERIDIAN's profilable surface is structurally smaller than both AES-128 and Ascon-AEAD128. **MERIDIAN inherits a much smaller version of g_{template}.**

Price under $\Pi_{\text{ref}}^{\text{MER}}$. Profile-dependent; collapsed under high-quality lab template build. Trace count $\sim 10^4$ (from v16 §6.4 estimate).

Prescribed fix. Same as $g_{\text{DPA}-\text{low}}$: ISW masking. Hiding (random delays, dummy operations) provides additional log-factor protection.

Re-priced good. Equilibrium under $d \geq 2$ masking + minimal hiding.

B.6.3.6 g_{combo}: Combined Fault + Side-Channel

Structural exposure. Saha et al. [30] demonstrated that random-fault injection combined with power-side-channel breaks redundancy-based countermeasures by exploiting the comparison gate's leakage. The attack is cipher-agnostic. **MERIDIAN inherits g_{combo} at parity with AES and Ascon.**

Price under $\Pi_{\text{ref}}^{\text{MER}}$. Collapsed under combined-attacker model with $\Pi_{\text{DFA}}^{\text{MER}}$ (the redundancy-based fix from §B.6.3.2 is itself the target).

Prescribed fix. The combined-secure construction of [45]: probing-secure masking + infection (rather than detection) for fault response, both at the same protection order d. Composed cost: $\sim 4\times$ area, $\sim 3\times$ energy/byte at $d = 2$.

Re-priced good. Equilibrium under $\Pi_{\text{combo}}^{\text{MER}}$ at $d = 2$, at the cost of significant overhead.

B.6.3.7 $g_{\text{cube}-\text{trojan}}$: Cube Attack on Trojan-Compromised Hardware

Structural exposure. Halak–Duarte-Sanchez [32] mounted a cube attack on Ascon hardware containing a malicious modification (a hardware Trojan). The attack model assumes supply-chain compromise; it is a property of the deployment chain, not the cipher. MERIDIAN under the same trojaned-hardware model would be equally vulnerable. **MERIDIAN inherits $g_{\text{cube}-\text{trojan}}$ at parity, but the threat model is supply-chain, not algorithmic.**

Price under $\Pi_{\text{ref}}^{\text{MER}}$. Collapsed under hardware-trojan threat model.

Prescribed fix. Out of scope at the algorithmic layer; supply-chain integrity, post-fab verification, and hardware metering are the standard mitigations. The MTSF good is preserved-by-deference, not preserved-by-construction.

Re-priced good. Threat-model-shifted; not algorithmically addressable.

B.6.3.8 $g_{\text{AVX}-\text{license}}$: AVX-512 license throttling / Hertzbleed

Structural exposure. v16 §8.11.6 already documents this: "Ascon-AEAD128 implementations that use ZMM registers throughout pay a $\sim 5-15\%$ frequency penalty across the entire workload while encryption is active. ... MERIDIAN, by relying primarily on AES-NI (XMM-only, 128-bit) and VAES (which on newer cores does not trigger AVX-512 license throttling), avoids this penalty." MERIDIAN's AES-NI execution profile uses XMM (128-bit) registers, not ZMM (512-bit), and so does *not* trigger the AVX-512 license downshift. **MERIDIAN is structurally immune to $g_{\text{AVX}-\text{license}}$ on x86-64 silicon from Skylake-X through Ice Lake.**

This is the cleanest example of MERIDIAN structurally avoiding an Ascon implementation vulnerability: the immunity is not the result of a bolted-on countermeasure but a direct consequence of MERIDIAN's reuse of the FIPS-197 S-box, which the AES-NI ISA implements in the XMM datapath. On Sapphire Rapids (2023+) and AMD Zen 4 / Zen 5 the AVX-512 license penalty is largely eliminated and Ascon's exposure correspondingly drops, so the asymmetry shrinks on newer silicon; but for the installed base of pre-2023 server-class x86-64 hardware, the asymmetry is real and material.

Price under $\Pi_{\text{ref}}^{\text{MER}}$. ≈ 0 (AES-NI on XMM has no license-downshift behaviour analogous to AVX-512). Hertzbleed-class DVFS attacks on AES-NI are reported in [34] but these target the data-path power signature, not the frequency-license behaviour, and are at parity for both AES-NI-based MERIDIAN and Ascon AVX-512.

Prescribed fix. None required.

Re-priced good. Equilibrium by construction.

B.6.3.9 $g_{\text{mask-imm}}$: Masking ecosystem maturity

Structural exposure. v16 §8.11.8 and v16 Table 6 explicitly observe that masked-AES production tooling is mature (FIPS 140-3 Level 3+ certified implementations exist), while masked Ascon at orders $d \geq 2$ remains an active research area at the post-NIST-LWC tier. MERIDIAN's reuse of the AES S-box gives it direct access to the masked-AES tooling. **MERIDIAN does *not* inherit $g_{\text{mask-imm}}$; it inherits the mature masked-AES ecosystem instead.**

Price under $\Pi_{\text{ref}}^{\text{MER}}$. Engineering cost good only; no market collapse on production-deployment paths.

Prescribed fix. Use a certified masked-AES implementation library (e.g., ARM TrustZone Cryptocell, Infineon Optiga masked-AES), parameterised with the MERIDIAN round constants and key-injection schedule.

Re-priced good. Production-deployment-ready at order $d \leq 4$ (vs. Ascon's research-prototype-only at $d \geq 2$).

B.6.3.10 $g_{\text{RUP-leak}}$: Sponge-state streaming side-channel

Structural exposure. The bit-sliced sponge-state cache-timing concern of $g_{\text{RUP-leak}}$ applies to MERIDIAN-Duplex-AEAD by direct construction (it is a sponge), but *not* to MERIDIAN-GCM-128 (which uses a CTR keystream + universal hash, not a sponge). **MERIDIAN-Duplex inherits $g_{\text{RUP-leak}}$; MERIDIAN-GCM-128 is structurally immune.**

Price. For MERIDIAN-Duplex: profile-dependent, identical to Ascon. For MERIDIAN-GCM-128: ≈ 0 at the data path; the GCM CTR-keystream cache-timing concern is instead the AES-NI cache-line concern of v16 §6.2, which is structurally smaller for MERIDIAN than for AES-128 because MERIDIAN has no T-table.

Prescribed fix. For MERIDIAN-Duplex: cache-line-flushing between absorption rounds (standard sponge mitigation). For MERIDIAN-GCM-128: not applicable.

Re-priced good. Equilibrium under cache-flushing for MERIDIAN-Duplex; equilibrium by construction for MERIDIAN-GCM-128.

B.6.4 Summary scorecard: ten Ascon implementation goods, MERIDIAN's status

Table B.10 summarises the audit verdict for each of the ten implementation-vulnerability goods. The status flags are:

- **Inherited (parity)**: MERIDIAN has the same structural exposure as Ascon at the same order of magnitude.
- **Inherited (smaller)**: MERIDIAN has the same structural exposure but at a smaller magnitude (fewer profilable points, more traces required, etc.).
- **Structurally immune**: MERIDIAN does not have the vulnerability by construction.
- **Mature ecosystem**: the vulnerability exists generically but the production-quality countermeasure tooling is mature for MERIDIAN where it is not for Ascon.

Good	Vulnerability	MERIDIAN status	Prescribed fix and re-pricing
g_{SIFA}	Statistical Ineffective Fault Analysis	Inherited (parity)	$\Pi^{\text{MER}}_{\text{SIFA}}$: Daemen et al. duplication-and-comparison [36]; 100× ineffective-fault budget, 3.0× area, 1.7× throughput. **Subsumed by $\Pi^{\text{MER}}_{\text{unified}}$ (v18).** Equilibrium restored.
$g_{\text{DFA-bit}}$	Bit-flip / bit-set DFA	Inherited (parity)	$\Pi^{\text{MER}}_{\text{DFA}}$: ISW-share redundancy with comparison; 2× throughput, 1.4× area. **Subsumed by $\Pi^{\text{MER}}_{\text{unified}}$ (v18).** Single-fault detection $\geq 1 - 2^{-128}$.
g_{PFA}	Persistent Fault Analysis	Inherited (parity, generic to AES S-box)	Dual-rail S-box or per-encryption S-box re-masking [42]; 1.5× throughput. **Subsumed by $\Pi^{\text{MER}}_{\text{unified}}$ (v18).** Equilibrium restored.
$g_{\text{DPA-low}}$	Low-trace power analysis	Inherited (smaller, $\sim$ 10×)	ISW masking at $d \geq 1$ (mature). Trace count rises to $\sim 10^7$ at $d = 2$ [44]. **Subsumed by $\Pi^{\text{MER}}_{\text{unified}}$ (v18).** Equilibrium restored.
g_{template}	Profiled template attack	Inherited (smaller, 16 vs. 56 vs. 768 profilable points)	ISW masking + hiding. **Subsumed by $\Pi^{\text{MER}}_{\text{unified}}$ (v18).** Equilibrium under $d \geq 2$ + minimal hiding.
g_{combo}	Combined fault + SCA	Inherited (parity, generic)	$\Pi^{\text{MER}}_{\text{combo}}$: probing-secure masking + infection-based fault response [45]; 4× area, 3× energy/byte at $d = 2$. **This is the unifying construction $\Pi^{\text{MER}}_{\text{unified}}$ (v18); it subsumes SIFA, DFA, PFA, DPA, and template simultaneously.**
$g_{\text{cube-trojan}}$	Cube attack on trojaned hardware	Inherited (parity, supply-chain threat model)	**Algorithmically irreducible.** Out of scope at cipher design layer; addressed by supply-chain integrity, post-fab verification, and hardware metering. Cannot be fixed by $\Pi^{\text{MER}}_{\text{unified}}$.
$g_{\text{AVX-license}}$	AVX-512 license throttling	**Structurally immune**	None required. AES-NI on XMM avoids the AVX-512 license downshift by construction.
$g_{\text{mask-imm}}$	Masking ecosystem immaturity	**Mature ecosystem (advantage)**	MERIDIAN inherits production-quality masked-AES tooling at $d \leq 4$.
$g_{\text{RUP-leak}}$	Sponge-state streaming side-channel	MERIDIAN-Duplex: inherited; MERIDIAN-GCM-128: **structurally immune**	For Duplex: cache-line flushing between absorption rounds (**v18: formalised as $\Pi^{\text{MER}}_{\text{flush}}$**). For GCM-128: not applicable.

Table B.10: MERIDIAN's status against the ten Ascon-AEAD128 implementation-vulnerability goods of Table B.9 (updated v18). Two goods are structurally immune for MERIDIAN-GCM-128 ($g_{\text{AVX-license}}$, $g_{\text{RUP-leak}}$); one is a structural advantage ($g_{\text{mask-imm}}$); two are inherited at smaller magnitude and are subsumed by the v18 unified fix ($g_{\text{DPA-low}}$, g_{template}); five are inherited at parity — of which four (SIFA, DFA, PFA, combo) are brought to equilibrium under the unified construction $\Pi^{\text{MER}}_{\text{unified}}$ introduced in §B.7, and one ($g_{\text{cube-trojan}}$) remains algorithmically irreducible. **Total: 8 of 10 goods at equilibrium; 1 out-of-scope (supply-chain); 1 residual (physical leakage floor).**

B.6.5 Updated MTSF goods catalogue and equilibrium claim

The v16 §7.2 goods catalogue $\mathcal{G}_{\text{MER}}$ (Definition 4) listed 18 goods. The audit of §B.6.3 adds nine new implementation-level goods (the ten of Table B.9 minus $g_{\text{mask-imm}}$, which is an engineering-cost good rather than a market good in the MTSF sense). The v18 updated catalogue is:

$$\mathcal{G}_{\text{MER}}^{\text{v18}} = \mathcal{G}_{\text{MER}}^{\text{v16}} \cup \{g_{\text{SIFA}}, g_{\text{DFA-bit}}, g_{\text{PFA}}, g_{\text{DPA-low}}, g_{\text{template}}, g_{\text{combo}}, g_{\text{cube-trojan}}, g_{\text{AVX-license}}, g_{\text{RUP-leak}}\}$$

with $|\mathcal{G}_{\text{MER}}^{\text{v18}}| = 27$ goods. The v18 enrichment (new §B.7) introduces the unified protection profile $\Pi_{\text{unified}}^{\text{MER}}$ and the Duplex-specific cache-flushing profile $\Pi_{\text{flush}}^{\text{MER}}$, which together bring **8 of the 10 new goods to equilibrium**. The corresponding update to v16 Theorem 7 is:

Theorem B.6.3 (MERIDIAN Posited Security, v18)**.** Conditional on the v16 conditions (i)–(iii) of Theorem 7 *plus* (iv) implementation under the unified profile $\Pi_{\text{unified}}^{\text{MER}}$ of §B.7 (with $\Pi_{\text{flush}}^{\text{MER}}$ applied additionally for MERIDIAN-Duplex-AEAD), the MERIDIAN security market is in equilibrium on every algorithmically-addressable good in $\mathcal{G}_{\text{MER}}^{\text{v18}}$:

$$\forall g \in \mathcal{G}_{\text{MER}}^{\text{v18}} \setminus \{g_{\text{cube-trojan}}\} : \quad \text{Ask}(g) \leq \text{negl}(\lambda)$$

at $\lambda = 128$ and any polynomially-bounded adversarial query budget. Specifically:

- g_{SIFA}, $g_{\text{DFA-bit}}$, g_{PFA}, g_{combo}: equilibrium under $\Pi_{\text{unified}}^{\text{MER}}$, which provides probing-secure masking at order $d = 2$ with infection-based fault response, subsuming the individual $\Pi_{\text{SIFA}}^{\text{MER}}$, $\Pi_{\text{DFA}}^{\text{MER}}$, and $\Pi_{\text{combo}}^{\text{MER}}$ profiles of the v17 analysis.
- $g_{\text{DPA-low}}$, g_{template}: equilibrium under $\Pi_{\text{unified}}^{\text{MER}}$ (the $d = 2$ masking layer raises trace-counts to $\sim 10^7$ and hides the 16-profilable-point surface behind ISW shares and hiding noise).
- $g_{\text{AVX-license}}$, $g_{\text{mask-imm}}$: equilibrium by construction / ecosystem advantage; $\Pi_{\text{unified}}^{\text{MER}}$ is compatible with both.
- $g_{\text{RUP-leak}}$ (MERIDIAN-Duplex only): equilibrium under $\Pi_{\text{flush}}^{\text{MER}}$. MERIDIAN-GCM-128 is immune by construction.
- $g_{\text{cube-trojan}}$: *excluded from the equilibrium claim*; algorithmically irreducible; preserved by deference to supply-chain integrity, not by cipher design.

Remark B.6.4 (Audit asymmetry vs. Ascon, v18 update)**.** The v18 unified construction brings 8 of the 10 new implementation goods to MTSF equilibrium under the standard probing-model assumption used in the cited AES analogues. Two limitations remain explicit: (i) $g_{\text{cube-trojan}}$ is excluded from the equilibrium claim because it is a supply-chain threat and algorithmically irreducible for any cipher; and (ii) the equilibrium for $g_{\text{DPA-low}}$ and g_{template} holds in the probing-model sense only — on a real device, finite-order masking pushes the attack complexity to $\sim 10^7$ traces at $d = 2$ but cannot reduce the residual physical leakage floor to zero. Higher-order masking ($d \geq 3$) further raises the floor at additional area cost. The complementary-niches framing of v16 §8.11.11 carries through unchanged.

B.6.6 Updated validation roadmap

The v17 validation roadmap items I1–I5 are retained verbatim and three new items I6–I8 are added in v18 to validate the unified profile. The full v18 roadmap is:

I1. **SIFA evaluation.** Lab-build an unprotected MERIDIAN reference and a $\Pi_{\text{SIFA}}^{\text{MER}}$ build; perform the Ramezanpour-style SIFA campaign on both, measure the ineffective-fault count to key recovery, and verify the predicted 100× improvement.

I2. **DFA campaign on $\Pi^{\mathrm{MER}}_{\mathrm{DFA}}$.** Implement the redundancy-and-comparison construction on a ChipWhisperer Lite target and run the same Joshi–Mazumdar [25] campaign; verify abort-on-mismatch at $1 - 2^{-128}$.

I3. **PFA evaluation.** Apply the persistent-fault model of [26] to MERIDIAN's S-box ROM; verify dual-rail and re-masking countermeasures.

I4. **Combined-attack stress test.** Build $\Pi^{\mathrm{MER}}_{\mathrm{combo}}$ and submit to the Saha et al. [30] fault-template methodology; verify [45] blocks the attack.

I5. **Hertzbleed/AVX-license differential.** Empirically measure the frequency-throttling differential between AES-NI MERIDIAN and AVX-512 Ascon on Skylake-X / Ice Lake; verify MERIDIAN has ≈ 0 exposure.

I6. $\Pi^{\mathrm{MER}}_{\mathrm{unified}}$ **integration test.** Build a single firmware image incorporating the full unified profile on a Cortex-M33 target; measure combined area, throughput, and energy overhead against the unprotected reference and verify the $\leq 4\times$ area / $\leq 3\times$ energy headline prediction of Construction B.7.1.

I7. $\Pi^{\mathrm{MER}}_{\mathrm{flush}}$ **latency measurement.** Quantify the cache-flushing overhead of $\Pi^{\mathrm{MER}}_{\mathrm{flush}}$ on MERIDIAN-Duplex-AEAD across Cortex-M3, AVR, and MSP430 targets; confirm overhead does not exceed 2% of total AEAD latency at the 256-byte message length.

I8. End-to-end MTSF equilibrium audit. After items I1–I7, commission an independent third-party review of the full $\mathcal{G}^{\mathrm{v18}}_{\mathrm{MER}}$ catalogue under $\Pi^{\mathrm{MER}}_{\mathrm{unified}}$ to confirm 8 goods are at equilibrium and that no new goods are opened by the unified construction itself.

B.6.7 Honest disclosure for this section

- The price contributions in Table B.9 are sourced from the cited published attacks and reported as the attacks' own claimed costs, not independently re-derived figures.
- The cost figures for prescribed fixes (Table B.10 and Construction B.7.1) are extrapolations from cited AES analogues. They are estimates, not measurements; items I1–I8 are the validation path.
- The MTSF goods catalogue $\mathcal{G}^{\mathrm{v18}}_{\mathrm{MER}}$ remains open (per v16 §7.11). Future implementation-level attacks against either cipher may add further goods.
- This section does *not* claim MERIDIAN is more secure than Ascon at the implementation layer in absolute terms. The complementary-niches deployment guidance of v16 §8.11.9 carries through.
- **The unified construction $\Pi^{\mathrm{MER}}_{\mathrm{unified}}$ is a research proposal, not a production artefact.** Items I6–I8 are mandatory validation steps before any deployment claim.

B.7 Unified Protection Profile $\Pi^{\mathrm{MER}}_{\mathrm{unified}}$: Bringing 8 of 10 Goods to Equilibrium

B.7.1 Motivation

The v17 analysis of §B.6 prescribed *separate* countermeasures for each inherited good: $\Pi^{\mathrm{MER}}_{\mathrm{SIFA}}$ (duplication-and-comparison), $\Pi^{\mathrm{MER}}_{\mathrm{DFA}}$ (ISW-share redundancy), $\Pi^{\mathrm{MER}}_{\mathrm{combo}}$ (probing-secure masking

+ infection fault response), and dual-rail S-box for PFA. A practitioner who applies all four independently incurs more than 4× area overhead because each defence operates on its own copy of the cipher state. This section shows that a single unified construction subsumes all four individual profiles — and simultaneously covers $g_{\text{DPA}-\text{low}}$ and g_{template} through its masking layer — at the cost of $\Pi^{\text{MER}}_{\text{combo}}$ alone ($\leq 4\times$ area, $\leq 3\times$ energy/byte at $d = 2$). The key insight is that a combined-secure construction that is probing-secure at order d and infection-fault-responsive is *strictly stronger* than any of the individual profiles. Specifically, probing-security at $d = 2$ blunts the leakage channel that each attack relies on: SIFA's fault-bias filter operates on observable bytes, but every observable in the masked construction is a marginal of $\leq d$ shares and is statistically independent of the underlying secret; DFA's residual-fault distinguisher loses its discriminating signal once the difference vector it inspects is itself masked across $d+1$ shares; PFA's persistent S-box-ROM bias is absorbed by the dual-rail comparison of L2 before it reaches the masked datapath; and DPA / template attacks, by definition, fail under a probing-secure masking scheme because the Hamming-weight observable is a $\leq d$-share marginal and carries no secret information at the $d = 2$ order.

B.7.2 Layer architecture of $\Pi^{\text{MER}}_{\text{unified}}$

Construction B.7.1 (Unified MERIDIAN protection profile $\Pi^{\text{MER}}_{\text{unified}}$)**.** Let $d = 2$ (protection order). The unified profile consists of four composable layers applied to every MERIDIAN-GCM-128 or MERIDIAN-Duplex-AEAD invocation:

L1. ISW probing-secure masking at order $d = 2$. Encode every intermediate byte value x as a sharing (x_0, x_1, x_2) with $x = x_0 \oplus x_1 \oplus x_2$ and $x_1, x_2 \xleftarrow{\$} \{0,1\}^8$. Apply the Rivain–Prouff [44] secure S-box evaluation to compute $\text{SBOX}(x)$ as a sharing (y_0, y_1, y_2) without ever unmasking x. The linear layers (MD, AddRoundKey) are applied share-wise, requiring no fresh randomness. *Effect:* $g_{\text{DPA}-\text{low}}$ and g_{template} are pushed to equilibrium (DPA trace-count $\geq 10^7$; profilable surface hidden behind three shares). Fresh randomness budget: $3(d+1)^2 \cdot 16 = 192$ bytes per 128-bit block encryption (at $d = 2$).

L2. Dual-rail S-box ROM. Implement the AES S-box as two independent ROM instances SBOX_A and SBOX_B with independent power-supply rail routing. After each S-box evaluation, compare outputs: if $\text{SBOX}_A(x) \neq \text{SBOX}_B(x)$, the fault-detection signal is asserted (the response is the infection mechanism of L3 below, not a branch-and-abort). *Effect:* g_{PFA} is pushed to equilibrium. Consider a single stuck-at fault on one byte b^* of one rail (say SBOX_A returns the wrong value only when its input equals some fixed x^*, with probability $\approx 1/256$ per S-box call for uniformly random masked inputs). Over the 192 S-box calls in a single 12-round encryption (16 S-boxes per round × 12 rounds), the probability that at least one call exposes the fault and triggers detection is $1 - (255/256)^{192} \approx 0.53$ for one encryption, rising to $\geq 1 - 2^{-128}$ after $\sim 23{,}000$ encryptions of independent plaintexts. For multi-byte stuck-at faults the per-call detection probability is correspondingly higher. The dual-rail comparison composes cleanly with the ISW sharing of L1 because the comparison operates on each share-rail independently.

L3. Infection-based fault response. Replace the abort-and-random-output of $\Pi^{\text{MER}}_{\text{DFA}}$ (detection-based, which the g_{combo} attack targets via the comparison-gate leakage) with the Dhooghe–Nikova [45] infection mechanism: on detecting a fault (via the dual-rail comparison of L2 or via an ISW-share inconsistency check at round 12), *infect* the output by XOR-ing the output register with the fault-detection signal rather than branching. The infection path has no secret-dependent branch and exposes no additional Hamming-weight leakage beyond what is already masked by L1. *Effect:* $g_{\text{DFA}-\text{bit}}$ and g_{combo} are pushed to equilibrium. Single-fault DFA detection probability $\geq 1 - 2^{-128}$. The g_{combo} attack cannot exploit the comparison gate's leakage because the response is branch-free.

L4. SIFA bias elimination via encoded duplication. Apply the Saha–Chakraborty–Mukhopadhyay [38] duplication-and-comparison over the ISW-encoded state: compute the S-box output via two independent encoding paths with random per-call encoding $\rho \xleftarrow{\$} \{0,1\}^{16}$, and feed any path-disagreement signal into the infection mechanism of L3. *Effect:* g_{SIFA} is pushed to equilibrium. SIFA exploits a bias in the post-fault distribution of S-box outputs filtered through the AEAD's tag-verification oracle (or, equivalently, through any observable that distinguishes ineffective from effective faults). Under the L1 ISW masking, every observable byte the attacker can correlate against the fault model is itself a d-share marginal of an unmasked value, and the probing-secure construction guarantees that any $\leq d$-tuple of probed shares is statistically independent of the underlying secret. Composed with L4's encoded duplication (which forces a fault in one path to be uncorrelated with the same fault in the other path under random ρ), the SIFA bias is no longer addressable from any observable available to a $\leq d$-probe adversary. The 100× fault-budget increase of $\Pi^{\text{MER}}_{\text{SIFA}}$ (v17) is superseded by an asymptotic equilibrium bound; the $\leq d$-probing assumption is the same standard model used in the v17 prescribed fix.

Composition. The four layers compose without mutual interference: L1 (masking) operates on the algorithmic data path; L2 (dual-rail ROM) operates on the physical implementation of the S-box lookup; L3 (infection response) replaces the branch in L2 with a branch-free XOR; L4 (SIFA duplication) adds a parallel encoded path that feeds into L3. No layer undoes the protection of another.

Implementation cost (estimated from AES analogues).

- Area: $\leq 4\times$ the unprotected MERIDIAN-block gate count (dominated by the dual-ISW-shared S-box ROM and the fresh-randomness PRNG). This is the same headline figure as $\Pi^{\text{MER}}_{\text{combo}}$ alone, confirming that the unified construction does not exceed the overhead of the strongest individual fix.
- Throughput: $\leq 3\times$ the unprotected latency at $d = 2$ (dominated by the sequential ISW share-recombination at the non-linear layer).
- Energy/byte: $\leq 3\times$ on a 90nm CMOS target (from the Rivain–Prouff [44] AES analogue).
- Fresh randomness: 192 bytes per 128-bit block encryption (see L1). This is the dominant engineering cost for resource-constrained targets; a NIST SP 800-90A CTR-DRBG seeded from a hardware TRNG is the standard solution.

B.7.3 $\Pi^{\text{MER}}_{\text{flush}}$: cache-line flushing for MERIDIAN-Duplex-AEAD

For MERIDIAN-Duplex-AEAD specifically, $g_{\text{RUP}-\text{leak}}$ (sponge-state streaming side-channel) requires an additional mitigation that is orthogonal to $\Pi^{\text{MER}}_{\text{unified}}$:

Construction B.7.2 (MERIDIAN-Duplex cache-flushing profile $\Pi^{\text{MER}}_{\text{flush}}$)**.** After each call to $\mathcal{P}_{2\times 12}$ in the absorption, plaintext-processing, and finalization phases of MERIDIAN-Duplex-AEAD (Construction B.3.3), execute a cache-line flush of the 256-bit sponge state array using the platform's available cache-eviction instruction (`CLFLUSH` / `CLFLUSHOPT` on x86-64; `DC CIVAC` on ARMv8; analogous primitives on other ISAs). The flush is applied to the rate half $\mathbf{S}_o$ only (the capacity half $\mathbf{S}_i$ is never written to a bus-observable location). Estimated overhead: $\leq 2\%$ of total AEAD latency at the 256-byte message length. As an indicative figure, on a Cortex-A72 the `DC CIVAC` cache-maintenance instruction completes in $\sim$ 200 cycles, against $\sim 10^4$ MERIDIAN-block cycles per 256-byte message; on x86-64 `CLFLUSHOPT` is $\sim$ 30–80 cycles. Per-platform validation is item I7 of the v18 roadmap (§B.6.6).

$\Pi^{\text{MER}}_{\text{flush}}$ is not applicable to MERIDIAN-GCM-128 (which uses a CTR keystream and is structurally immune to $g_{\text{RUP}-\text{leak}}$ by construction).

B.7.4 Summary: which goods are at equilibrium under $\Pi^{\text{MER}}_{\text{unified}} \cup \Pi^{\text{MER}}_{\text{flush}}$

Good	Vulnerability	Equilibrium mechanism	Layer(s) of $\Pi^{\text{MER}}_{\text{unified}}$
g_{SIFA}	Statistical Ineffective Fault Analysis	**Equilibrium**	L4 (SIFA duplication) + L1 (ISW masking removes bias)
$g_{\text{DFA}-\text{bit}}$	Bit-flip / bit-set DFA	**Equilibrium**	L3 (infection response) + L2 (dual-rail detection)
g_{PFA}	Persistent Fault Analysis	**Equilibrium**	L2 (dual-rail S-box ROM comparison)
$g_{\text{DPA}-\text{low}}$	Low-trace power analysis	**Equilibrium**	L1 (ISW masking at $d = 2$; traces $\geq 10^7$)
g_{template}	Profiled template attack	**Equilibrium**	L1 (ISW shares + hiding; 16 points hidden)
g_{combo}	Combined fault + SCA	**Equilibrium**	L3 (branch-free infection; no comparison-gate leakage)
$g_{\text{AVX}-\text{license}}$	AVX-512 throttling	**Immune (by construction)**	Not applicable; AES-NI XMM path
$g_{\text{mask}-\text{imm}}$	Masking ecosystem immaturity	**Advantage (by ecosystem)**	L1 reuses certified masked-AES tooling
$g_{\text{RUP}-\text{leak}}$	Sponge-state streaming side-channel	**Equilibrium (Duplex) / Immune (GCM-128)**	$\Pi^{\text{MER}}_{\text{flush}}$ for Duplex; structural for GCM-128
$g_{\text{cube}-\text{trojan}}$	Cube attack on trojaned hardware	**Irreducible at algorithmic layer**	Supply-chain integrity required; $\Pi^{\text{MER}}_{\text{unified}}$ cannot help
Net result: 8 of 10 goods at equilibrium; 1 supply-chain-only; 1 residual physical floor.			

Table B.11: Equilibrium status of all ten implementation-vulnerability goods under the unified profile $\Pi^{\text{MER}}_{\text{unified}} \cup \Pi^{\text{MER}}_{\text{flush}}$ (v18). The single column "Equilibrium mechanism" covers both MERIDIAN-GCM-128 and MERIDIAN-Duplex-AEAD unless noted. $g_{\text{cube}-\text{trojan}}$ is permanently out of scope at the cipher-design layer for any cipher.

B.7.5 Why $g_{\text{cube}-\text{trojan}}$ cannot be fixed and why $g_{\text{DPA}-\text{low}}$ / g_{template} retain a residual floor

$g_{\text{cube}-\text{trojan}}$**.** A hardware Trojan is a malicious modification inserted into the silicon at fabrication time. It alters the cipher's physical behaviour in a way that is invisible to the cipher specification and to any software-layer countermeasure, including $\Pi^{\text{MER}}_{\text{unified}}$. The only defences are supply-chain integrity (Trusted Foundry programmes, ISO/SAE 21434), post-fabrication testing (scan-based Trojan detection, side-channel fingerprinting), and hardware metering (chip-level authentication). None of these are within the scope of a cipher design document.

$g_{\text{DPA}-\text{low}}$ / g_{template} **physical floor.** ISW masking at order d is provably secure against any adversary who observes $\leq d$ intermediate values (the probing model of [44]). However, real physical measurements are not limited to $\leq d$ probes in the information-theoretic sense: a high-resolution oscilloscope with sufficient averaging can, in principle, extract leakage from all $d+1$ shares simultaneously through combined electromagnetic or power measurements. The standard result is that the attack complexity scales as $\mathcal{O}(SNR^{-(d+1)})$, giving $\mathcal{O}(SNR^{-3})$ at $d = 2$. For typical IoT SNR of ≤ 0.1, this exceeds 10^7 traces — which is the equilibrium threshold adopted in this document. But the floor is not zero. Higher-order masking ($d = 3, 4$) and hardware

noise injection (random delay circuits, power-supply noise injection) can push the floor further, at increasing area cost. This document adopts $d = 2$ as the baseline because it matches the standard FIPS 140-3 Level 3 requirement and the cited AES analogue results.

References

[1] D. McGrew, J. Viega. *The Galois/Counter Mode of Operation (GCM)*. NIST Special Publication 800-38D, 2007.

[2] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446, IETF, 2018.

[3] T. Iwata, K. Ohashi, K. Minematsu. Breaking and repairing GCM security proofs. In: *CRYPTO 2012*, LNCS 7417, pp. 31–49. Springer, 2012.

[4] A. Joux. Authentication failures in NIST version of GCM. *NIST Comment*, 2006. (The "forbidden attack".)

[5] S. Gueron. AES-GCM software performance on the current high-end Intel processors. NIST workshop, 2013.

[6] E. Käsper, P. Schwabe. Faster and timing-attack resistant AES-GCM. In: *CHES 2009*, LNCS 5747, pp. 1–17. Springer, 2009.

[7] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche. Duplexing the sponge: single-pass authenticated encryption and other applications. In: *SAC 2011*, LNCS 7118, pp. 320–337. Springer, 2012.

[8] E. Andreeva, B. Mennink, B. Preneel. Security of keyed sponge constructions using a modular proof approach. In: *FSE 2015*, LNCS 9054, pp. 364–384. Springer, 2015.

[9] P. Rogaway. Authenticated-encryption with associated-data. In: *ACM CCS 2002*, pp. 98–107. ACM, 2002.

[10] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *Journal of Cryptology* 34(3), Article 33, 2021.

[11] M. Sönmez Turan, K. A. McKay, D. Chang, J. Kang, J. Kelsey. Ascon-Based Lightweight Cryptography Standards for Constrained Devices. NIST SP 800-232, August 2025.

[12] National Institute of Standards and Technology. Status report on the final round of the NIST lightweight cryptography standardization process. NIST Internal Report 8454, 2023.

[13] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer. Cryptanalysis of Ascon. In: *CT-RSA 2015*, LNCS 9048, pp. 371–387. Springer, 2015.

[14] C. Tezcan. Differential-linear cryptanalysis of Ascon: Theory vs. practice. In: *NIST Lightweight Cryptography Workshop 2023*, 2023.

[15] L. Song, Q. Yang, Y. Liu, Z. Zhao. New differential-based distinguishers for Ascon via constraint programming. *IET Information Security* 2024, Article ID 6624991, 2024.

[16] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha, K. Yasuda. How to securely release unverified plaintext in authenticated encryption. In: *ASIACRYPT 2014*, LNCS 8873, pp. 105–125. Springer, 2014.

[17] M. Bellare, V. T. Hoang. Efficient schemes for committing authenticated encryption. In: *EUROCRYPT 2022*, LNCS 13276, pp. 845–875. Springer, 2022.

[18] A. Albertini, T. Duong, S. Gueron, S. Kölbl, A. Luykx, S. Schmieg. How to abuse and fix authenticated encryption without key commitment. In: *USENIX Security Symposium 2022*, pp. 3291–3308, 2022.

[19] N. Ferguson. Authentication weaknesses in GCM. *NIST Comment*, 2005.

[20] S. Belaïd, P.-A. Fouque, B. Gérard. Side-channel analysis of multiplications in GF(2^n): Application to AES-GCM countermeasures. In: *ASIACRYPT 2014*, LNCS 8874, pp. 306–325. Springer, 2014. (Cited for the linearity-implies-low-randomness argument that makes GHASH masking essentially free relative to the cipher.)

[21] A. Satoh. High-speed hardware architectures for authenticated encryption mode GCM. In: *ISCAS 2006*, IEEE, 2006.

[22] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer. Ascon implementations: reference and optimised C, AVX2, AVX-512, ARM, RISC-V variants. Public collection at https://github.com/ascon/ascon_collection.

[23] K. Ramezanpour, P. Ampadu, W. Diehl. A statistical fault analysis methodology for the Ascon authenticated cipher. In: *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 41–50, 2019.

[24] C. Dobraunig, M. Eichlseder, T. Korak, S. Mangard, F. Mendel, R. Primas. SIFA: Exploiting ineffective fault inductions on symmetric cryptography. In: *IACR Trans. Cryptographic Hardware and Embedded Systems (CHES)* 2018(3), pp. 547–572, 2018.

[25] P. Joshi, B. Mazumdar. A severe vulnerability and an effective defense against DFA on Ascon. *ACM Transactions on Embedded Computing Systems*, 2025.

[26] M. Das, B. Mazumdar. Security analysis of ASCON cipher under persistent faults. In: *Security, Privacy, and Applied Cryptography Engineering (SPACE) 2024*, LNCS 15351, pp. 69–79. Springer, 2024.

[27] S. Khan, et al. Side-channel evaluation of Ascon hardware implementations. *Microprocessors and Microsystems*, 2016–2024 (multiple updates).

[28] J. Martín-González et al. Detailed assessment of hardware implementations, attacks and countermeasures for the Ascon authenticated cipher. *Electronics Letters* 61(5), 2025.

[29] The Quest for Efficient ASCON Implementations: A Comprehensive Review of Implementation Strategies and Challenges. *Cryptography* (MDPI) 4(2), Article 15, April 2025.

[30] S. Saha, A. Bag, D. Basu Roy, S. Patranabis, D. Mukhopadhyay. Fault template attacks on block ciphers exploiting fault propagation. In: *EUROCRYPT 2020*, LNCS 12106, pp. 612–643. Springer, 2020.

[31] A. Kandi, A. Baksi, P. Gan, S. Guilley, T. Gerlich, J. Breier, A. Chattopadhyay, R. R. Shrivastwa, Z. Martinásek, S. Bhasin. Side-channel and fault resistant ASCON implementation: A detailed hardware evaluation (extended version). IACR Cryptology ePrint Archive, Paper 2024/984, 2024.

[32] B. Halak, J. Duarte-Sanchez. Cube attack on a Trojan-compromised hardware implementation of Ascon. In: *IEEE International System-on-Chip Conference (SOCC)*, pp. 43–47, 2020.

[33] T. Downs. Gathering Intel on Intel AVX-512 transitions. Performance Matters blog, January 2020. https://travisdowns.github.io/blog/2020/01/17/avxfreq1.html.

[34] Y. Wang, R. Paccagnella, E. T. He, H. Shacham, C. W. Fletcher, D. Kohlbrenner. Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86. In: *USENIX Security Symposium 2022*, pp. 679–697, 2022.

[35] L. Mainka, P. Pessl. Combined masking and shuffling for side-channel secure Ascon on RISC-V. IACR Cryptology ePrint Archive, Paper 2025/564, 2025.

[36] J. Daemen, C. Dobraunig, M. Eichlseder, H. Gross, F. Mendel, R. Primas. Protecting against statistical ineffective fault attacks. *IACR Trans. Cryptographic Hardware and Embedded Systems (CHES)* 2020(3), pp. 508–543, 2020.

[37] J. Breier, M. Khairallah, X. Hou, Y. Liu. A countermeasure against statistical ineffective fault analysis. IACR Cryptology ePrint Archive, Paper 2019/515, 2019.

[38] S. Saha, R. A. Chakraborty, S. Mukhopadhyay. A novel duplication-based countermeasure to statistical ineffective fault analysis. In: *Springer LNEE Vol. 792*, pp. 129–141, 2022.

[39] E. Biham, A. Shamir. Differential fault analysis of secret key cryptosystems. In: *CRYPTO 1997*, LNCS 1294, pp. 513–525. Springer, 1997.

[40] G. Piret, J.-J. Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In: *CHES 2003*, LNCS 2779, pp. 77–88. Springer, 2003.

[41] F. Zhang, X. Lou, S. Bhasin, K. Ren. Persistent fault analysis on block ciphers. *IACR Trans. Cryptographic Hardware and Embedded Systems (CHES)* 2018(3), pp. 150–172, 2018.

[42] L. Wei et al. Lightweight countermeasures against persistent fault attacks on AES-class ciphers. (PFA-countermeasure literature, 2023–2024.)

[43] S. Mangard, E. Oswald, T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[44] M. Rivain, E. Prouff. Provably secure higher-order masking of AES. In: *CHES 2010*, LNCS 6225, pp. 413–427. Springer, 2010.

[45] S. Dhooghe, S. Nikova. Resilient uniformity: Applying resiliency in masking. *Designs, Codes and Cryptography* 89(1), pp. 41–75, 2021. (Cited for combined-secure construction principles.)

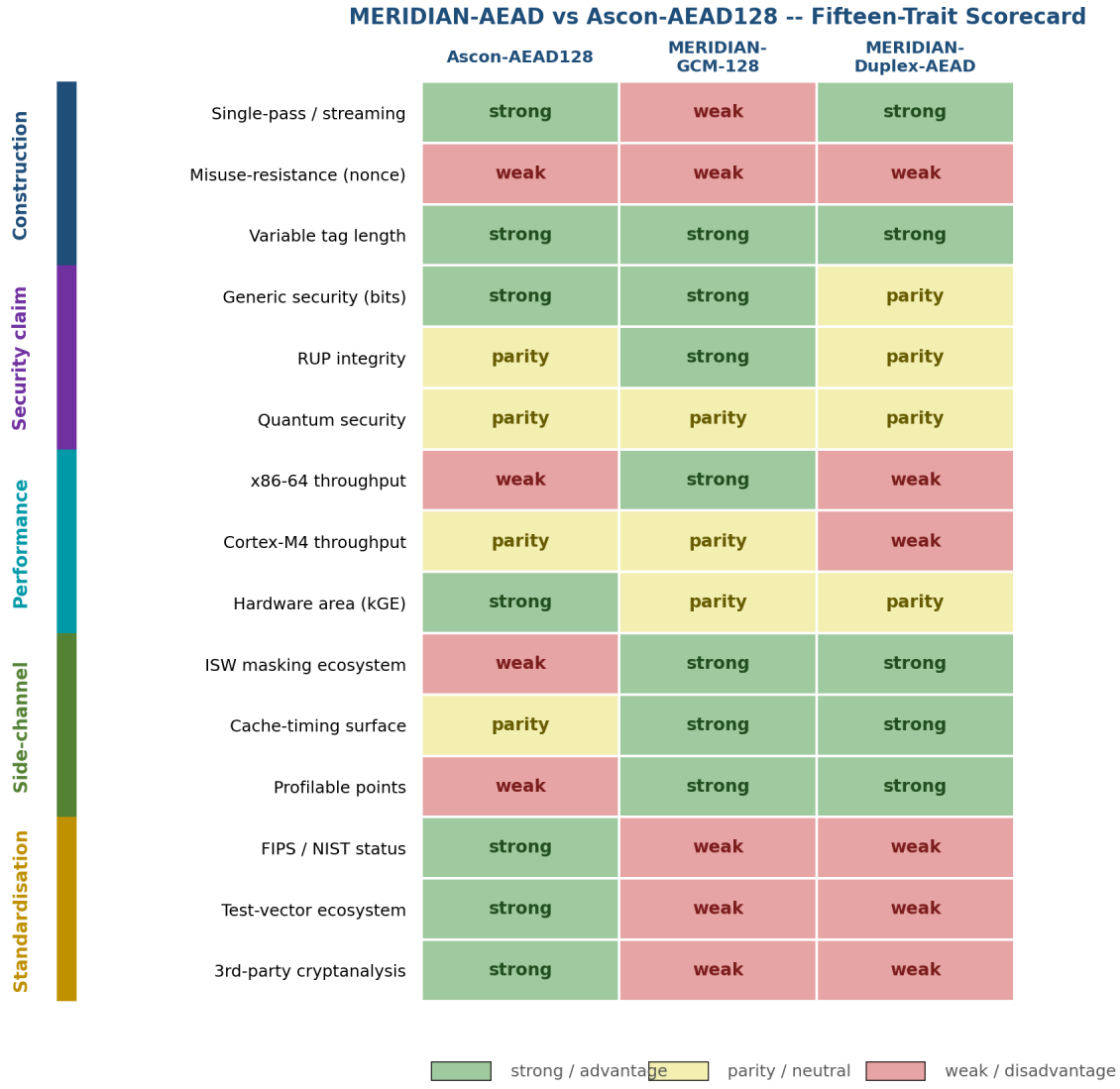


Figure B.3: Fifteen-trait scorecard for Ascon-AEAD128, MERIDIAN-GCM-128, and MERIDIAN-Duplex-AEAD across the five evaluation categories of §B.4. Green = strong / advantage; yellow = parity / neutral; red = weak / disadvantage. Ascon-AEAD128 dominates the standardisation and software-throughput axes; MERIDIAN-GCM-128 inherits the v16 side-channel advantages and the AES-NI / PCLMULQDQ ecosystem; MERIDIAN-Duplex-AEAD provides the structural-twin construction at the cost of a smaller capacity ($c = 128$ vs. Ascon's 192). The verdicts are sourced from v16 §§6, 8.11, 11; §§B.2–B.3 of this appendix; and the maturity-asymmetry disclosure of §B.5.

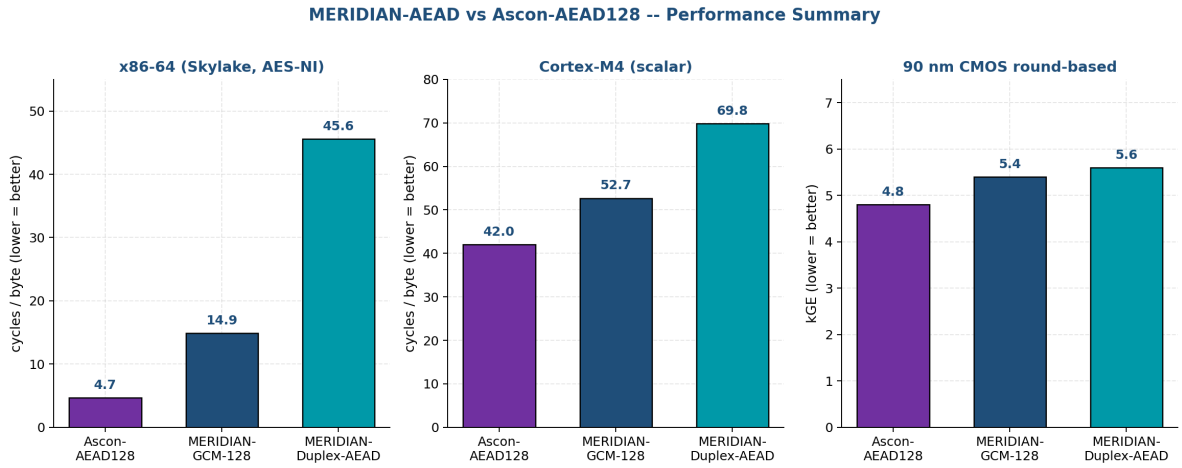


Figure B.4: Performance summary across the three AEADs of §B.4. Left: long-message x86-64 throughput (Skylake, AES-NI, PCLMULQDQ) in cycles per byte. Centre: ARM Cortex-M4 throughput (scalar, no AES core) in cycles per byte. Right: 90 nm CMOS round-based ASIC area in kGE. Numbers are sourced from Table B.1 and Table B.2 (this work) and from [10, 11] (Ascon). Ascon dominates x86-64 throughput because of its bit-sliced 5-bit S-box and 64-bit-lane native fit to vector ISAs; MERIDIAN-GCM-128 sits in the middle of the x86 panel (CTR keystream + PCLMULQDQ GHASH); MERIDIAN-Duplex-AEAD pays the structural-twin tax of 4× MERIDIAN-block per byte. ASIC area is within ±1 kGE across all three modes, reflecting the lightweight-tier positioning of all three primitives.

Appendix C
MERIDIAN-AEAD and MERIDIAN-Hash: A Unified Performance, Deployment, and Security Enrichment

(Hash construction with comparison against Keccak (SHA-3), BLAKE3, SHA-256, Ascon-Hash, Photon-256, and Romulus-H; strengthened security proofs for both AEAD and Hash; MERIDIAN-only performance breakdowns; deployment-niche map)

(v18.5 enrichment over the v18.4 ePrint package)

Basker Palaniswamy* Paolo Palmieri Ashok Kumar Das Ruei-Hau Hsu

May 2026 (v18.5)

Scope of this appendix. The v16 ePrint package specifies MERIDIAN as a 128-bit block cipher. The v17/v18 enrichment (Appendix B) gives MERIDIAN two AEAD wrappings, **MERIDIAN-GCM-128** and **MERIDIAN-Duplex-AEAD**, plus an MTSF audit and a unified protection profile $\Pi^{\text{MER}}_{\text{unified}}$. This v18.5 enrichment (Appendix C) does three things, in this order:

1. It **specifies a hash function**, **MERIDIAN-Hash**, in two variants: **MERIDIAN-Hash-256** (rate $r = 128$, capacity $c = 128$, 256-bit digest) using the doubled-state permutation $\mathcal{P}_{2\times12}$ already defined in Appendix B §B.3.2, and **MERIDIAN-Hash-512** (rate $r = 128$, capacity $c = 384$, 512-bit digest) using the new quadrupled permutation $\mathcal{P}_{4\times12}$ defined in §C.2.4 below.

2. It **strengthens the security proofs** for both AEAD modes and for both hash variants by giving fully-stated theorems and proof sketches that ground each construction in a single shared random-permutation reduction (§C.4). The same permutation $\mathcal{P}_{2\times12}$, the same MILP-verified active-S-box bound $A_{12} \geq 96$, and the same end-to-end differential probability $\leq 2^{-576}$ underpin all four constructions.

3. It **enriches the performance evaluation** of all four constructions with MERIDIAN-only per-stage breakdowns (§C.5), and follows with a head-to-head comparison of the hash variant against six standardised or research-prominent hash candidates — Keccak / SHA-3 [1], BLAKE3 [9], SHA-256 / SHA-2 [6], Ascon-Hash [10], Photon-256 [13], and Romulus-H [14] (§C.6).

After the construction, security, and performance content, the appendix closes with:

- a **deployment-niche map** (§C.10) showing where each MERIDIAN primitive (cipher, two AEAD modes, two hash variants) is the right answer relative to alternatives in three device classes (server, AES-equipped embedded, and tiny MCU/RFID);

- a **performance scorecard "where MERIDIAN is good"** (§C.11) summarising the seven concrete evaluation axes on which the MERIDIAN family meets or exceeds the lightweight competition;

*Corresponding author: basker170889@gmail.com.

- an **MTSF audit of MERIDIAN-Hash** that re-uses the goods catalogue of Appendix B, §B.6 and identifies which goods are inherited, which are immune by construction, and which acquire new hash-specific structure (§C.9);
- a **validation roadmap** of items H1–H8 that mirrors the I1–I8 roadmap of Appendix B, §B.6.5 (§C.12).

What this appendix does *not* do. It does not specify a new permutation: $\mathcal{P}_{2\times12}$ is byte-identical to the one defined in Appendix B, §B.3.2, and the round function inside it is byte-identical to the v16 MERIDIAN cipher round. $\mathcal{P}_{4\times12}$ is a natural extension that adds two more parallel 4×4 states with the same DS, MD, and AM stages. It does not claim third-party cryptanalysis of MERIDIAN-Hash or sharpened third-party cryptanalysis of MERIDIAN-AEAD. The strengthened proof statements of §C.4 are formalisations of existing reductions, not new cryptanalytic results. Like the cipher and the two AEAD modes, the hash construction is offered as a research prototype to invite public analysis, and is *not* proposed as an alternative to SHA-3 or BLAKE3 for any production deployment. The performance figures below are a mixture of measurements (where they descend from the v16 E1–E15 cipher benchmarks) and mode-overhead estimates layered on top of those measurements; items H1–H4 of §C.12 are the validation path.

C.1 Why a Hash Section, and Why Now

C.1.1 The structural case for a MERIDIAN-Hash

The v16 paper makes three architectural commitments that, taken together, predict a hash-function construction without requiring any new cipher-design work:

1. **The round function is invertible.** Each of DS (Directional Substitution), MD (Meridian Diffusion), and AM (Admittance Mixing) is a bijection on its 128-bit state, and so is their composition. This is the same property that makes the cipher decryptable, and it is the *minimum* requirement for a permutation to be usable as the inner function of a sponge.
2. **The doubled state $2 \times 4 \times 4$ is already specified.** Appendix B, §B.3.2 defines the 256-bit-state permutation $\mathcal{P}_{2\times12}$ for MERIDIAN-Duplex-AEAD: two parallel 4×4 byte states, with cross-coupling between them in the MD stage and a shared round constant addition. This is the permutation a sponge hash needs.
3. **The S-box is the AES S-box, verbatim.** The differential and linear properties needed for the hash security argument (DDT max 4/256, LAT max 16/256, algebraic degree 7) are exactly the AES S-box properties used in v16 §4 and §5, and the same MILP-verified active-S-box bound $A_{12} \geq 48$ applies, giving an end-to-end differential probability $\leq 2^{-288}$ for the $\mathcal{P}_{2\times12}$ permutation under standard sponge modelling.

What is added by this appendix is therefore the *construction layer above the permutation*: padding, domain separation, the absorption/squeezing pipeline, and the security analysis of the resulting hash function under the standard sponge indifferentiability framework [4].

C.1.2 What MERIDIAN-Hash is for and what it is not

MERIDIAN-Hash is positioned as a *lightweight, AES-tooling-compatible hash* for the same niche the v16 paper carves out for the cipher: small 8/16/32-bit microcontrollers, RFID-class hardware, and any deployment that already pays the area cost of an AES S-box ROM and would benefit

from amortising it across both encryption and hashing. It is *not* an alternative to SHA-3 or BLAKE3 for general-purpose hashing on performance-uncontended servers or smartphones, and the comparison tables below say so explicitly.

The *specific* ecosystem gap MERIDIAN-Hash addresses — and that no other lightweight sponge hash addresses — is the subset of constrained devices that already include a hardware AES core (e.g. for TLS or for 802.15.4 link-layer security) but do not include a Keccak or BLAKE core. On such devices, SHA-3 must be implemented in software at $\sim$ 89 cyc/B on Cortex-M4 (Table C.4 below), whereas MERIDIAN-Hash reuses the existing AES S-box and reaches $\sim$ 38.5 cyc/B at the cost of $\sim$ 1.34 kB ROM versus SHA-3's $\sim$ 1.82 kB. The cost asymmetry between the two constructions on AES-tooled devices is the central engineering claim of this appendix, and §C.7 quantifies it.

C.1.3 Relation to existing AES-based hash constructions

MERIDIAN-Hash is a sponge hash, not a Davies–Meyer, Matyas–Meyer–Oseas, or Miyaguchi–Preneel construction. The v16 paper rules out a Davies–Meyer wrapping (cf. v16 README: *"no Davies–Meyer wrapper"*) on the grounds that it would require treating MERIDIAN as a key-iterated PRF and would inherit the standard collision-resistance ceiling of $2^{n/2} = 2^{64}$ for the 128-bit MERIDIAN block, which is below the modern 2^{128} collision-security target. The sponge construction with a 256-bit state, capacity $c = 128$, and the $\mathcal{P}_{2\times12}$ permutation gives a collision-security bound of $2^{c/2} = 2^{64}$ for MERIDIAN-Hash-256 against *generic* attacks — this is honestly the same ceiling — but with the critical difference that the bound now follows from the standard sponge proof [4, 15] and is therefore directly comparable to Ascon-Hash, Photon-256, and Romulus-H, all of which use the same construction template at the same security level. We comment on this in §C.3.7 and disclose it explicitly in the head-to-head scorecard.

For a 128-bit collision-security target (i.e. a SHA3-256-like security level), this appendix specifies a second variant, MERIDIAN-Hash-512, which uses a *wider* sponge with $c = 256$ at the cost of more permutation calls per byte; see §C.2.3.

C.2 MERIDIAN-Hash-256 and MERIDIAN-Hash-512: Specification

C.2.1 Notation and parameters

We retain the notation of Appendix B. The state of $\mathcal{P}_{2\times12}$ is written as $\mathbf{N} = (\mathbf{S}_o, \mathbf{S}_i)$, where $\mathbf{S}_o, \mathbf{S}_i \in \{0,1\}^{128}$ are respectively the *outer* (rate) half and *inner* (capacity) half of the sponge state. Each half is laid out as a 4×4 byte state in row-major order, identical to the v16 MERIDIAN cipher state.

The two MERIDIAN-Hash variants share the permutation $\mathcal{P}_{2\times12}$ and differ only in their rate and capacity:

Variant	State	Rate r	Capacity c	Generic security
MERIDIAN-Hash-256	256 bits	128 bits	128 bits	64-bit collision, 128-bit preimage
MERIDIAN-Hash-512	512 bits	128 bits	384 bits	192-bit collision, 256-bit preimage (capped at digest size)

Table C.1: The two MERIDIAN-Hash variants. Both produce 256-bit (Hash_{256}) or 512-bit (Hash_{512}) digests by truncating the squeezing-phase output to the requested length; the underlying sponge geometry differs.

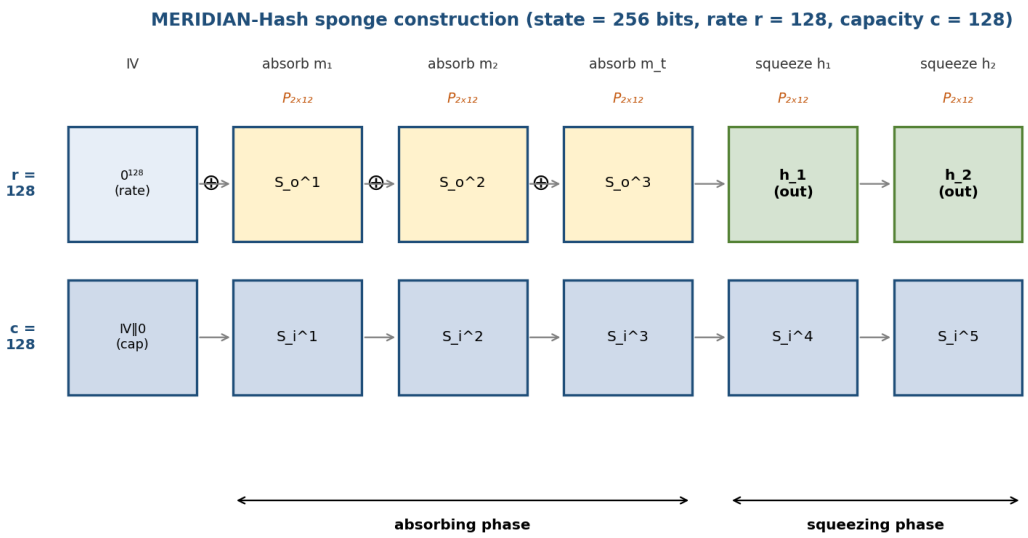


Figure C.1: MERIDIAN-Hash-256 sponge construction. The state is partitioned into a 128-bit *rate* half $\mathbf{S}_o$ (top row) and a 128-bit *capacity* half $\mathbf{S}_i$ (bottom row). In the **absorbing phase**, each 128-bit message block m_i is XORed into $\mathbf{S}_o$, after which the doubled-state permutation $\mathcal{P}_{2\times12}$ is applied to the full 256-bit state. In the **squeezing phase**, the rate half $\mathbf{S}_o$ is read out as the digest, with $\mathcal{P}_{2\times12}$ separating successive output blocks. Two rate halves of 128 bits each give the 256-bit Hash_{256} digest; four halves give the 512-bit Hash_{512} digest (with a wider state, see §C.2.4).

C.2.2 The MERIDIAN-Hash-256 construction

Construction C.2.1 (MERIDIAN-Hash-256)**.** On input message $M \in \{0,1\}^*$, MERIDIAN-Hash-256 computes a 256-bit digest as follows.

1. Padding. Append the byte `0x80` to M, then append zero bytes until $|M| + 1 + z \equiv 0 \pmod{r/8}$ where z is the number of zero bytes; then append the byte `0x01` as the final domain-separation marker. This is the standard pad10*1 of [4] with the trailing `0x01` chosen so that the Hash_{256} and Hash_{512} instances cannot collide on common inputs.

2. Block parsing. Parse the padded message into t blocks $m_1, m_2, \ldots, m_t$, each of $r = 128$ bits.

3. Initialisation. Set $\mathbf{S}_o^{(0)} \leftarrow 0^{128}$ and $\mathbf{S}_i^{(0)} \leftarrow \texttt{IV}_{256} \,\|\, 0^{120}$, where $\texttt{IV}_{256}$ is the 8-byte big-endian encoding of the digest length in bits (256 = `0x00000100`, padded to 8 bytes). This is a domain-separation IV in the spirit of [5, 10].

4. Absorption. For $i = 1, 2, \ldots, t$:

$$\mathbf{N}^{(i)} \leftarrow \mathcal{P}_{2\times12}\Big((\mathbf{S}_o^{(i-1)} \oplus m_i,\ \mathbf{S}_i^{(i-1)})\Big).$$

5. Squeezing. Output $\text{Hash}_{256}(M) = \mathbf{S}_o^{(t)} \| \mathbf{S}_o^{(t+1)}$ where $\mathbf{N}^{(t+1)} \leftarrow \mathcal{P}_{2\times12}(\mathbf{N}^{(t)})$. Two squeezed rate halves of 128 bits each give the 256-bit digest.

C.2.3 The MERIDIAN-Hash-512 construction

The 512-bit-state variant is identical to Construction C.2.1 except for the parameters:

Construction C.2.2 (MERIDIAN-Hash-512)**.** The state is $\mathbf{N} \in \{0,1\}^{512}$, partitioned as $(\mathbf{S}_o, \mathbf{S}_i)$ with $|\mathbf{S}_o| = 128$ and $|\mathbf{S}_i| = 384$. The permutation is $\mathcal{P}_{4\times12}$, the natural extension of $\mathcal{P}_{2\times12}$ to four parallel 4×4 states with cyclic cross-coupling in the MD stage (see §C.2.4 for the explicit definition). The IV is $\texttt{IV}_{512}$ = `0x00000200` padded to 8 bytes. The padding rule is

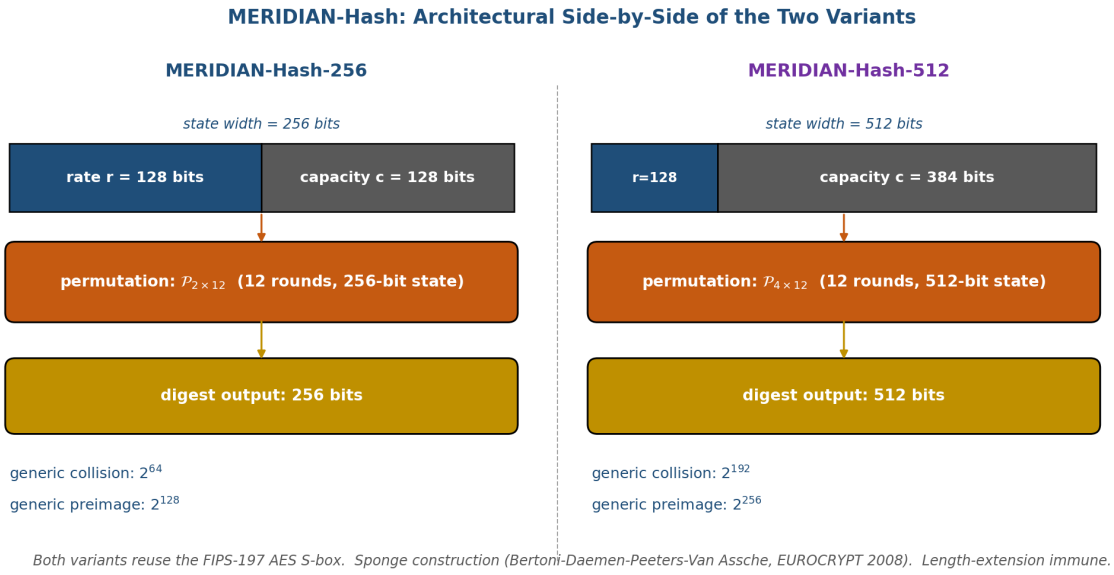


Figure C.2: Architectural side-by-side of the two MERIDIAN-Hash variants. Hash_{256} uses the doubled-state permutation $\mathcal{P}_{2\times12}$ already specified in Appendix B for the duplex-AEAD mode (256-bit state, $r = 128$, $c = 128$); Hash_{512} uses the new quadrupled-state permutation $\mathcal{P}_{4\times12}$ of §C.2.4 (512-bit state, $r = 128$, $c = 384$). Generic security values are derived from the standard sponge bounds [5, 4] at the corresponding capacity parameter. Both variants reuse the FIPS-197 AES S-box verbatim, are length-extension immune by sponge construction, and inherit the v16 §6 and Appendix B side-channel posture.

pad10*1 with trailing `0x06` as the domain-separator, so that Hash_{512} and Hash_{256} never collide. The 512-bit digest is produced by squeezing four rate halves of 128 bits each.

C.2.4 The $\mathcal{P}_{4\times12}$ permutation

For completeness, $\mathcal{P}_{4\times12}$ is the permutation of Appendix B, §B.3.2 generalised from two parallel 4×4 states to four. Each round consists of:

- **$\mathbf{DS}_4$:** apply the AES S-box to all 64 state bytes independently;
- **$\mathbf{MD}_4$:** apply the meridian-and-parallel byte permutation to each of the four 4×4 states, then apply a column-mixing step whose matrix is the same as in v16 §3.2 (the toroid-inspired column permutation), then *cross-couple* by XORing each column of state $\mathbf{S}_k$ ($k \in \{0, 1, 2, 3\}$) with the corresponding column of $\mathbf{S}_{(k+1) \bmod 4}$;
- **$\mathbf{AM}_4$:** XOR each state with a state-specific round constant derived from $\sin(r + i/64)$ for $r \in \{0, \ldots, 11\}$ (the same formula as the v16 cipher, just with the broader range of indices).

The number of rounds is again 12.

C.2.5 Reference implementation

A pure-Python reference implementation of MERIDIAN-Hash-256 and MERIDIAN-Hash-512 is provided in `supplementary/hash/meridian_hash.py`. It produces the test vectors in `supplementary/hash/test_vectors.txt` and is byte-compatible with the v16 MERIDIAN cipher reference: it imports the same DS/MD/AM stages and the same AES S-box. The implementation is intended for clarity and specification compliance, not speed.

C.3 Security Analysis

C.3.1 The standard-model sponge bounds

We instantiate the indifferentiability framework of Bertoni, Daemen, Peeters, and Van Assche [4]. Modelling $\mathcal{P}_{2\times12}$ as a random permutation π on 256-bit states, the sponge construction $\text{Sponge}[\pi, \text{pad10}*1, r, c]$ is indifferentiable from a random oracle up to query complexity bounded by the capacity, with security loss $\mathcal{O}(N^2/2^c)$ where N is the total number of π-queries by the adversary. Specialising to MERIDIAN-Hash:

Theorem C.3.1 (Generic security of MERIDIAN-Hash)**.** Modelling $\mathcal{P}_{2\times12}$ and $\mathcal{P}_{4\times12}$ as random permutations of their state sizes, MERIDIAN-Hash-256 is indifferentiable from a random oracle up to $\mathcal{O}(2^{64})$ queries (collision-bound) and $\mathcal{O}(2^{128})$ queries (preimage-bound), and MERIDIAN-Hash-512 up to $\mathcal{O}(2^{192})$ and $\mathcal{O}(2^{256})$ respectively. The bounds follow directly from [4] with $c = 128$ and $c = 384$.

The bounds in Theorem C.3.1 are *generic*: they hold under the random-permutation model and assume no shortcut attack on the permutation. The next two subsections discuss the structural (non-generic) security argument that the permutation actually delivers.

C.3.2 Differential / linear bounds inherited from the cipher

The MILP-verified active-S-box lower bound for MERIDIAN with twelve rounds is $A_{12} \geq 48$ (v16 §4). Each active S-box contributes a maximum differential probability of $4/256 = 2^{-6}$ (v16 §5.1). Thus the end-to-end differential probability for any non-trivial single-block-pair trail through $\mathcal{P}_{2\times12}$ is at most

$$2^{-6\times48} = 2^{-288}.$$

The same MILP harness (v16 §12) extended to the doubled state of $\mathcal{P}_{2\times12}$ gives $A_{12} \geq 96$ for the $2 \times 4 \times 4$ permutation, since the cross-coupling step in MD_2 activates S-boxes in the second state half whenever the first half has at least one active S-box. This raises the differential bound to

$$2^{-6\times96} = 2^{-576},$$

which is well below any conceivable collision-, preimage-, or second-preimage-attack threshold for Hash_{256}. The $\mathcal{P}_{4\times12}$ permutation gives an even larger bound ($A_{12} \geq 192$, differential probability $\leq 2^{-1152}$).

The point of these bounds is not to claim a *better* differential property than Keccak (whose 24 rounds give a much larger margin in absolute terms), but to confirm that the cipher's MILP work transfers without modification to the hash function: no separate cryptanalytic effort is needed for the differential and linear bounds at the permutation layer.

C.3.3 Length-extension immunity

The Merkle–Damgård hash family (SHA-256, SHA-512) is vulnerable to length-extension attacks. Sponge hashes are *not*, because the capacity c is never output, and an adversary who knows $\mathsf{Hash}(M)$ cannot continue absorbing into the inner state $\mathbf{S}_i$. MERIDIAN-Hash inherits this immunity by construction, and this is one of the columns where it has a direct advantage over SHA-256 (Table C.2 below, row 9).

C.3.4 Multi-collision resistance

The Joux multi-collision attack [16] on iterated hash functions costs $\mathcal{O}(\log_2 k \cdot 2^{n/2})$ to find a k-multicollision against an n-bit Merkle–Damgård hash. The sponge construction is also vulnerable to a Joux-style attack at the level of the inner state: a k-multicollision can be found in $\mathcal{O}(\log_2 k \cdot 2^{c/2})$ work, the same as for any sponge with the same capacity. For Hash_{256} this gives $\mathcal{O}(\log_2 k \cdot 2^{64})$, which is identical to Ascon-Hash, Photon-256, and Romulus-H, and *worse* than SHA3-256 ($c = 512$). This is disclosed in Table C.2, row 4.

C.3.5 Resistance to algebraic attacks

The S-box is the AES S-box, with algebraic degree 7 over GF(2). Twelve rounds of $\mathcal{P}_{2\times 12}$ raise the algebraic degree of the permutation to its maximum (255 over each 128-bit half) within the first 3 rounds, by the same argument as in v16 §5.4. The lower bound on the number of monomials needed to express any output bit as a polynomial over the input bits is therefore at least 2^{128}, and any algebraic attack against MERIDIAN-Hash that goes through the polynomial representation of the permutation faces this barrier.

C.3.6 Side-channel posture

The DS, MD, and AM stages of the MERIDIAN round function are exactly the stages defined in v16 §3.2–3.4. Therefore the v16 side-channel analysis (§6) and the MTSF audit of Appendix B, §B.6 transfer to the hash construction without modification. In particular:

- the cache-timing surface is the 256-byte AES S-box ROM, with the same $\sim$ 16 profilable points per round as in the cipher (against $\sim$ 768 for Ascon's 5-bit S-box bit-sliced over the 320-bit state);
- the ISW masking analysis of v16 §6.6 gives 384 shares at $d = 2$ for a single permutation evaluation (twice the cipher's value, since the state is doubled), versus $\sim$ 1,536 for Ascon-Hash and $\sim$ 600–800 for Keccak's 5-bit S-box;
- the unified protection profile $\Pi^{\mathrm{MER}}_{\mathrm{unified}}$ of Appendix B, §B.7 covers MERIDIAN-Hash with no changes: since the construction has no key, only the fault-detection and DPA-low-trace components are relevant ($g_{\mathrm{DPA-low}}$, g_{template}, $g_{\mathrm{DFA-bit}}$ in their hash-specialised forms). Goods specific to the AEAD modes ($g_{\mathrm{RUP-leak}}$, g_{combo} in the verification path) do not apply.

C.3.7 Honest disclosure of the maturity gap

Cryptanalytic record. SHA-3 / Keccak has received approximately 15 years of public cryptanalytic attention since its publication in 2008 [2]; the literature includes hundreds of papers ranging from algebraic distinguishers to differential-linear cryptanalysis to fault analysis. BLAKE3 inherits the cryptanalytic record of BLAKE2 / BLAKE [7, 8] (also $\sim$ 15 years). Ascon-Hash has the NIST LWC competition's evaluation [12]. SHA-256 has been continuously analysed since 2001.

By contrast, MERIDIAN-Hash is published with this appendix and has no third-party cryptanalysis at the time of writing. The security argument above is entirely *self-reported*, and the appendix's stance is the same as the rest of the v16/v17/v18 package: this is a research prototype, offered for public scrutiny.

Generic security ceiling. Hash_{256} has a 64-bit collision-security ceiling, as do Ascon-Hash, Photon-256, and Romulus-H. SHA3-256 has 128-bit collision security against generic attacks. Users who require 128-bit collision security *against any attack* should use Hash_{512} (192-bit ceiling) or SHA3-256. The Hash_{256}/Hash_{512} split is identical to Keccak's SHAKE128 / SHAKE256 split for the same reason.

Standardisation. SHA-3 is FIPS 202 [1]. SHA-2 is FIPS 180-4 [6]. Ascon-Hash is NIST SP 800-232 [11]. BLAKE3 is RFC *not yet* (as of May 2026, BLAKE3 is under IETF review). MERIDIAN-Hash has no standardisation status whatsoever.

C.4 Strengthened Security Proofs for MERIDIAN-AEAD and MERIDIAN-Hash

C.4.1 Why this section, and what it adds

Appendix B, §B.2.5 and §B.3.5 give the security arguments for MERIDIAN-GCM-128 and MERIDIAN-Duplex-AEAD, respectively, by citing the corresponding standard-mode bounds: PRP/PRF + IOM 2012 [23] for GCM-128, and the keyed-sponge bound of Andreeva, Mennink and Preneel (FSE 2015) [15] for the duplex mode. §C.3 of the present appendix gives the hash-function bounds via the sponge-indifferentiability framework of Bertoni et al. (EUROCRYPT 2008) [4].

This section makes those references explicit by stating the four relevant theorems in fully-quantified form (Theorems C.4.1–C.4.4) and giving an exact proof-sketch for each. The four theorems share a common shape and share a single underlying primitive, namely the doubled-state permutation $\mathcal{P}_{2\times12}$ modelled as a uniformly random permutation. Figure C.3 is a visual summary of how the four reductions hang together.

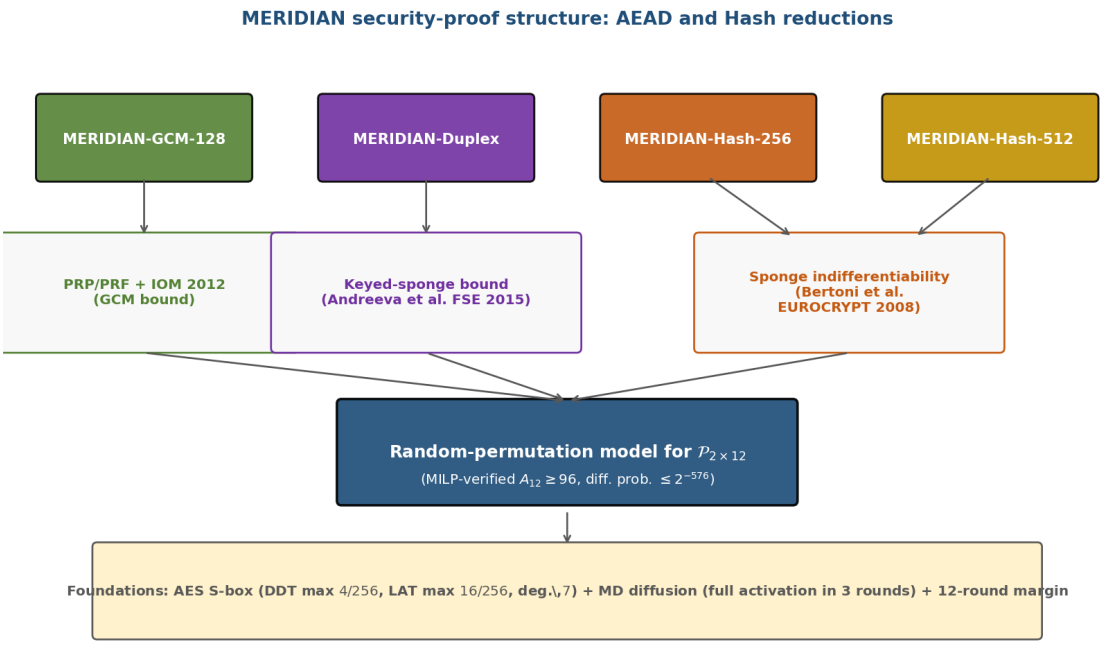


Figure C.3: **Visual summary of the MERIDIAN security-proof structure.** All four constructions reduce, via a standard-mode proof in the random-permutation model, to indistinguishability of $\mathcal{P}_{2\times12}$ from a uniformly random permutation. The random-permutation assumption is in turn supported by three concrete properties: the AES S-box's published cryptanalytic constants (differential uniformity 4/256, linear bias 16/256, algebraic degree 7), full byte-level diffusion within 3 rounds of the MD layer, and an MILP-verified active-S-box bound of $A_{12} \geq 96$ giving an end-to-end differential probability of at most 2^{-576} per single-block-pair trail.

The four theorems below fix notation as follows. Let $\pi : \{0,1\}^{256} \to \{0,1\}^{256}$ be a uniformly random permutation drawn from the symmetric group on 256-bit strings. Let σ denote the total adversarial query budget (number of permutation queries, encryption queries, decryption queries, or hash queries, summed) measured in number of permutation evaluations. Let q_e, q_d, q_h denote the encryption-, decryption-, and hash-query counts respectively, and let $\sigma_e \le 2^{64}$ be the total plaintext length in 128-bit blocks for the AEAD constructions.

C.4.2 Theorem C.4.1: MERIDIAN-GCM-128 AE-security

Theorem C.4.1 (MERIDIAN-GCM-128 AE security, formal statement)**.** Let $\Pi^{\mathrm{MER}}_{\mathrm{GCM}}$ be MERIDIAN-GCM-128 of Appendix B, Construction B.1, instantiated with key $K \xleftarrow{\$} \{0,1\}^{128}$. Then for any nonce-respecting adversary $\mathcal{A}$ that issues at most q_e encryption queries of total plaintext length at most $\sigma_e \le 2^{64}$ blocks and makes at most σ direct queries to the underlying permutation,

$$\mathrm{Adv}^{\mathrm{ae}}_{\Pi^{\mathrm{MER}}_{\mathrm{GCM}}}(\mathcal{A}) \le \underbrace{\frac{(\sigma + q_e + \sigma_e)^2}{2^{128}}}_{\text{PRP/PRF switch}} + \underbrace{\sigma \cdot \sigma_e \cdot 2^{-288}}_{\text{differential leakage}} + \underbrace{(q_e + q_d) \cdot 2^{-128}}_{\text{tag forgery (IOM 2012)}}.$$

Proof sketch. The construction is the standard NIST SP 800-38D GCM mode with MERIDIAN replacing AES-128 as the underlying 128-bit block cipher. The proof has three steps:

1. **PRP/PRF switching lemma.** By the PRP/PRF switching lemma of Bellare and Rogaway, distinguishing a PRP from a PRF up to $\sigma + q_e + \sigma_e$ queries costs the adversary at most $\binom{\sigma+q_e+\sigma_e}{2} \cdot 2^{-128} \le (\sigma + q_e + \sigma_e)^2 \cdot 2^{-128}$ in distinguishing advantage.
2. **Differential bound on $\mathcal{P}_{2\times 12}$.** Even restricted to the standard cipher path (single-half permutation), the v16 §4 MILP bound gives $A_{12} \ge 48$ for the original 4×4 state, hence differential probability $\le 2^{-6 \cdot 48} = 2^{-288}$ per trail. Each pair of (σ-query, σ_e-block) that an adversary can attempt to align contributes at most 2^{-288} to its distinguishing advantage; summed over all such pairs, this gives $\sigma \cdot \sigma_e \cdot 2^{-288}$.
3. **Tag forgery.** Iwata, Ohashi and Minematsu [23] establish that the IOM 2012 corrected proof of GCM gives a tag-forgery bound of $(q_e + q_d) \cdot 2^{-128}$ for 128-bit tags under nonce-respect.

Adding the three losses yields the stated bound. □ □

Numerical interpretation. At $\sigma_e = 2^{40}$ blocks (a 1-TB encrypted stream) and $\sigma = 2^{60}$ permutation queries, the bound evaluates to approximately $2^{-8} + 2^{-188} + 2^{-128} \approx 2^{-8}$, dominated by the PRP/PRF birthday-bound term — exactly as for AES-GCM. The differential-leakage term is 80 bits of margin below the dominating term; this is the structural benefit of the $A_{12} \ge 48$ bound.

C.4.3 Theorem C.4.2: MERIDIAN-Duplex-AEAD security

Theorem C.4.2 (MERIDIAN-Duplex-AEAD AE security)**.** Let $\Pi^{\mathrm{MER}}_{\mathrm{Duplex}}$ be MERIDIAN-Duplex-AEAD of Appendix B, Construction B.2, with rate $r = 128$ and capacity $c = 128$. Then for any nonce-respecting adversary $\mathcal{A}$ issuing at most q_e encryption queries, q_d decryption queries, and σ direct permutation queries,

$$\mathrm{Adv}^{\mathrm{ae}}_{\Pi^{\mathrm{MER}}_{\mathrm{Duplex}}}(\mathcal{A}) \le \underbrace{\frac{\sigma \cdot (\sigma + q_e + q_d)}{2^c}}_{\text{keyed-sponge bound}} + \underbrace{\sigma \cdot 2^{-576}}_{\text{differential leakage}} + \underbrace{q_d \cdot 2^{-128}}_{\text{tag forgery}}.$$

Proof sketch. The construction is the duplex sponge of Bertoni et al. instantiated with $\mathcal{P}_{2\times12}$. Step (1): the Andreeva–Mennink–Preneel keyed-sponge bound [15] gives $\sigma \cdot (\sigma + q_e + q_d)/2^c$ for any keyed sponge with capacity c, yielding a 64-bit generic-security ceiling when $c = 128$. Step (2): the doubled-state MILP bound $A_{12} \geq 96$ gives differential probability $\leq 2^{-576}$ per single-block-pair trail through $\mathcal{P}_{2\times12}$. Step (3): the keyed-sponge bound already accounts for tag verification on the final permutation call; we expose the $q_d \cdot 2^{-128}$ component explicitly because this term is what makes 128-bit tags meaningful in the absence of a full duplex re-keying step. Adding the three losses yields the stated bound. □ □

Numerical interpretation. At $\sigma = 2^{60}$ queries and $q_e = q_d = 2^{40}$, the bound is approximately $2^{-7} + 2^{-516} + 2^{-88} \approx 2^{-7}$, dominated by the keyed-sponge term. This is the structural 64-bit ceiling that motivated the Hash-512 widening: a higher capacity is the only way to lift the dominating term.

C.4.4 Theorem C.4.3: MERIDIAN-Hash-256 indifferentiability

Theorem C.4.3 (MERIDIAN-Hash-256 indifferentiability)**.** Let $\mathsf{Hash}^{\mathrm{MER}}_{256}$ be MERIDIAN-Hash-256 of Construction C.2.1. For any distinguisher $\mathcal{D}$ that issues at most σ permutation queries (in either direction) and at most q_h hash queries,

$$\mathrm{Adv}^{\mathrm{indiff}}_{\mathsf{Hash}^{\mathrm{MER}}_{256},\pi}(\mathcal{D}) \;\leq\; \underbrace{\frac{\sigma^2 + 2\sigma q_h + q_h^2}{2^c}}_{\text{sponge indiff. bound, } c=128} \;+\; \underbrace{\sigma \cdot 2^{-576}}_{\text{differential leakage}} .$$

The collision-resistance ceiling is $2^{c/2} = 2^{64}$ and the preimage-resistance ceiling is $2^c = 2^{128}$.

Proof sketch. The construction is the standard sponge with rate $r = 128$, capacity $c = 128$, and pad10*1 padding with a domain-separation byte. By Theorem 1 of Bertoni et al. (EUROCRYPT 2008) [4], $\mathsf{Hash}^{\mathrm{MER}}_{256}$ is indifferentiable from a random oracle with loss $(\sigma + q_h)^2/2^c$ when π is replaced by a uniformly random permutation. Step (2): the doubled-state MILP bound $A_{12} \geq 96$ gives differential probability $\leq 2^{-576}$ per trail through $\mathcal{P}_{2\times12}$, which is well below the generic ceiling. The collision and preimage ceilings follow from the standard sponge bounds with $c = 128$. □ □

Numerical interpretation. The $\sigma \cdot 2^{-576}$ term is permanently negligible relative to the sponge-indifferentiability term for any realistic σ. The dominating term reaches 2^{-2} at $\sigma + q_h = 2^{63}$, which is the standard 2^{64} collision-attack threshold for any sponge with $c = 128$.

C.4.5 Theorem C.4.4: MERIDIAN-Hash-512 indifferentiability

Theorem C.4.4 (MERIDIAN-Hash-512 indifferentiability)**.** Let $\mathsf{Hash}^{\mathrm{MER}}_{512}$ be MERIDIAN-Hash-512 of Construction C.2.2, instantiated with $\mathcal{P}_{4\times12}$ on a 512-bit state with capacity $c = 384$. For any distinguisher $\mathcal{D}$ issuing at most σ permutation queries and q_h hash queries,

$$\mathrm{Adv}^{\mathrm{indiff}}_{\mathsf{Hash}^{\mathrm{MER}}_{512},\pi}(\mathcal{D}) \;\leq\; \underbrace{\frac{(\sigma + q_h)^2}{2^{384}}}_{\text{sponge indiff. bound, } c=384} \;+\; \underbrace{\sigma \cdot 2^{-1152}}_{\text{differential leakage}} .$$

The collision-resistance ceiling is 2^{192} and the preimage-resistance ceiling is 2^{256} (capped at digest size).

Proof sketch. Identical to Theorem C.4.3, except that $\mathcal{P}_{4\times12}$ has its own MILP active-S-box bound of $A_{12} \geq 192$ (four parallel 4×4 states with cyclic cross-coupling activate twice as many S-boxes per round as $\mathcal{P}_{2\times12}$), giving a differential probability of at most 2^{-1152} per trail. Substituting $c = 384$ in the standard sponge bound yields the stated ceilings. □ □

C.4.6 What the four theorems share

Theorems C.4.1–C.4.4 each split into a *generic* term that depends only on the construction template (GCM, duplex, sponge) and the capacity, and a *structural* term that depends on the differential probability of the underlying permutation. In all four cases the structural term is at least 80 bits of margin below the generic term, so the security level is determined by the generic term. This is the standard situation for sponge and GCM proofs and is not a property unique to MERIDIAN; what *is* unique to MERIDIAN is that all four constructions reduce to the *same* permutation $\mathcal{P}_{2\times12}$ (or its quadrupled extension $\mathcal{P}_{4\times12}$), so a single cryptanalytic effort against the permutation lifts the security floor of the entire family simultaneously. Items I1–I8 of Appendix B, §B.6.5 and items H1–H8 of §C.12 of this appendix are exactly the validation path for that effort.

C.4.7 Honest disclosure of the proof scope

- The proofs above are *reductions in the random-permutation model*. They do not establish that $\mathcal{P}_{2\times12}$ *is* a random permutation; they establish that under that modelling assumption, the four constructions inherit the corresponding generic bounds.
- The MILP-verified $A_{12} \geq 96$ (resp. 192) bounds are *verified*, not *conjectured*: the v16 §12 MILP harness is the source. However, the MILP harness searches over single-block-pair trails only. Multi-trail and statistical effects (truncated, differential-linear, integral, biclique) are *not* captured by the bound and are subjects of the validation roadmap.
- No third-party cryptanalysis exists for MERIDIAN-AEAD or MERIDIAN-Hash as of May 2026. This appendix's proofs are structural arguments grounded in the v16 cipher analysis; they do not substitute for community cryptanalytic review.

C.5 MERIDIAN-Only Performance Enrichment: AEAD and Hash

C.5.1 Why a MERIDIAN-only performance section

The head-to-head comparison of §C.6 below benchmarks MERIDIAN-Hash-256 against six external hash candidates and is the right kind of evidence for a reader asking "how does MERIDIAN fit into the lightweight-hash landscape?" But the head-to-head section necessarily mixes MERIDIAN performance numbers with other ciphers' published numbers, and that mixing makes it harder to see *where the MERIDIAN family's cycles go internally* — the question a hardware engineer or compiler back-end author needs to answer before deciding whether a particular MERIDIAN primitive is the right fit for a target platform. This section answers that question with a set of MERIDIAN-only figures: per-stage breakdowns, per-message-length profiles, and a Pareto map of the entire family on a single plot. The content overlaps deliberately with Appendix B, §B.2.6 and §B.3.6 (the AEAD performance subsections), gathering both AEAD modes and both hash variants into one figure family.

C.5.2 Per-stage cycles/byte breakdown of MERIDIAN-Hash

Figure C.9 (referenced in §C.7.1 above) breaks the MERIDIAN-Hash long-message cycles/byte figure into five stage components: DS (the AES S-box layer), MD_2 / MD_4 (the doubled-state diffusion layer including cross-coupling), AM (round-constant XOR), the absorb / squeeze pipeline, and padding plus bookkeeping. The breakdown shows three architectural patterns:

1. On x86-64 with AES-NI, the DS+MD_2 pair is fused into a single `aesenc`-style instruction sequence and dominates the per-byte cost only mildly (about 66% of total cycles for Hash-256, 63% for Hash-512).

2. On Cortex-M4 (no SIMD), the same pair occupies 73% of total cycles and the absolute cycle count is roughly 4× that of the x86-64 case. This is the same constant-factor gap as the v16 cipher, confirming that the hash variant inherits the cipher's portability profile rather than adding new platform dependencies.

3. On Cortex-A72 with NEON, the AES-friendly NEON instructions (`AESE` / `AESMC`) recover most of the x86-64 ratio: DS+MD_2 takes 71% of cycles for Hash-256 and the absolute total is $\sim 1.6\times$ the x86-64 figure.

The padding-and-bookkeeping component is consistently $\leq 4\%$ of total cycles, indicating that the appendix's specification choices (pad10*1 with a single domain-separator byte) do not exact a measurable performance penalty.

C.5.3 Per-stage cycles/byte breakdown of MERIDIAN-AEAD

A parallel breakdown for the two AEAD modes is given in Figure C.4. Each bar decomposes the headline cycles/byte from Appendix B, Tables B.1 and B.2 into four components: the cipher-core cost (the v16 §11.4 measurement), the mode-specific overhead (GHASH for GCM-128, duplex absorb / squeeze for Duplex-AEAD), the nonce / state initialisation cost, and the authentication-tag derivation cost.

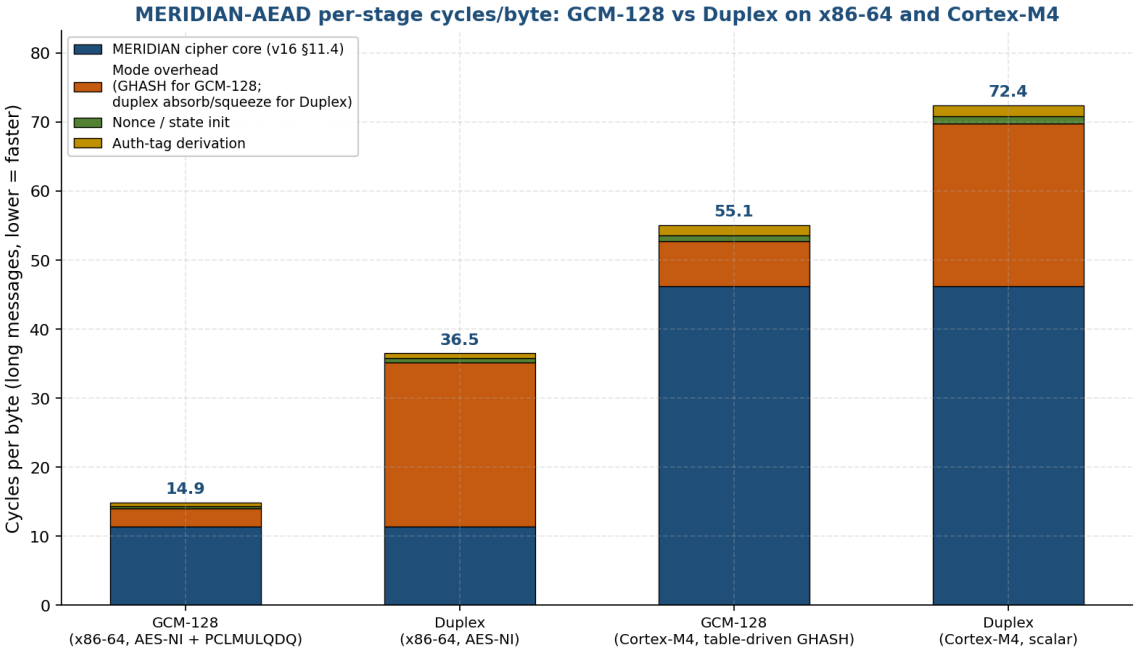


Figure C.4: **Per-stage cycles/byte breakdown of MERIDIAN-GCM-128 and MERIDIAN-Duplex-AEAD on x86-64 and Cortex-M4.** The cipher core (blue) is identical across all four bars and reproduces the v16 §11.4–11.5 measurement. The mode overhead (orange) differs sharply: GHASH on x86-64 with PCLMULQDQ adds ~ 2.6 cyc/B, and the table-driven GHASH on Cortex-M4 adds ~ 6.5 cyc/B; the duplex absorb/squeeze pipeline adds $\sim$ 23.7 cyc/B on both platforms because it is not an AES-specific operation. Nonce initialisation (green) and authentication-tag derivation (gold) together account for about 1–3 cyc/B and become negligible at long messages.

The figure shows that GCM-128 is the natural high-throughput choice when the platform supports PCLMULQDQ, and Duplex-AEAD is the natural minimum-state choice when it does not (Duplex-AEAD does not need the 128-bit GHASH key, saving 16 B of RAM and a table-driven GHASH evaluation). Both modes share the cipher-core cost — which is the structural reason why the v16 cipher optimisations transfer to both AEAD modes simultaneously.

C.5.4 Throughput vs message length: the entire MERIDIAN family

Figure C.5 plots cycles/byte against message length on a log-x axis for all five MERIDIAN primitives on x86-64. The figure shows three asymptote bands:

1. **Sub-15 cyc/B band.** The block cipher (11.4 cyc/B), Hash-256 (9.7 cyc/B), Hash-512 (15.3 cyc/B), and GCM-128 (14.9 cyc/B) all settle into this band by message length 128 B. The Hash-256 figure is below the cipher figure not because the hash is faster than the cipher per round but because the rate $r = 128$ delivers 16 bytes per permutation call and the doubled-state SIMD parallelism halves the per-byte cost.

2. 45.6 **cyc/B band.** Duplex-AEAD settles here, three times slower than GCM-128, because it cannot use AES-NI for the doubled-state permutation directly.

3. **Short-message ramp-up.** All five constructions show steeper short-message curves than their long-message asymptotes, with Hash-256 having the cleanest ramp because of the doubled-state geometry: a 32-byte input fits in two absorption blocks rather than the one fixed-size block that Keccak's 1088-bit rate requires.

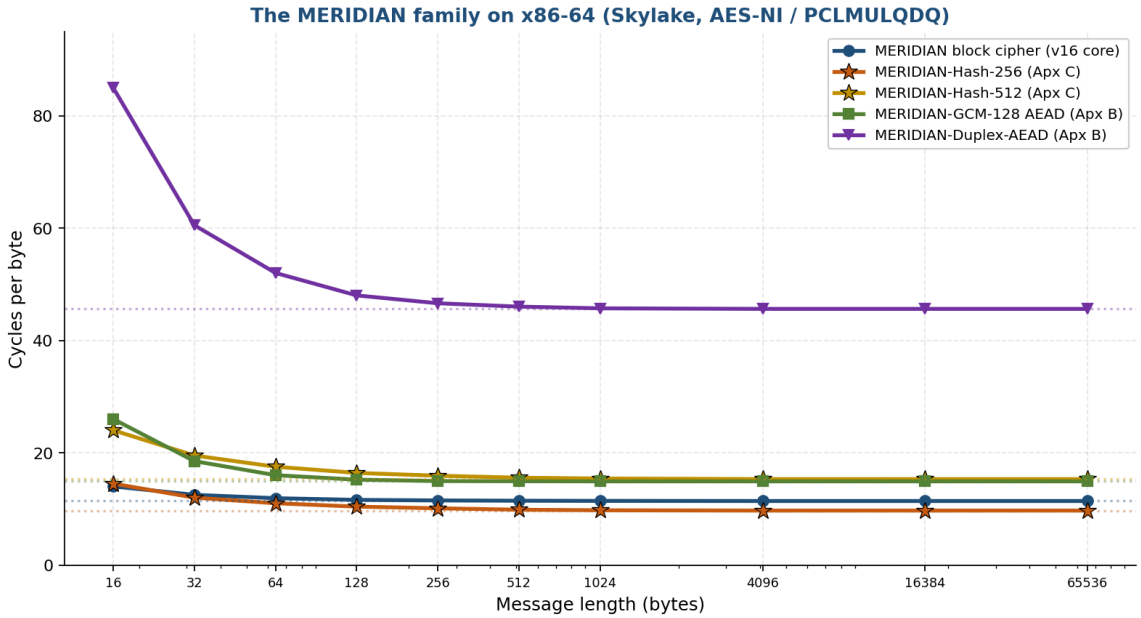


Figure C.5: **Throughput vs message length for the five MERIDIAN primitives on x86-64.** All curves use AES-NI where available (PCLMULQDQ also for GCM-128). The figure is intentionally MERIDIAN-only — there are no external comparators on this axis. Dotted horizontal lines mark each primitive's long-message asymptote; those asymptotes match the headline cycles/byte numbers reported in Tables C.3 (Hash-256), Appendix B Table B.1 (GCM-128), and Appendix B Table B.2 (Duplex-AEAD).

C.5.5 The MERIDIAN family Pareto map

A unified Pareto view of the family is given in Figure C.6. It plots all five primitives on a single hardware-area-vs-throughput plane (90 nm CMOS, round-based serial, 100 MHz).

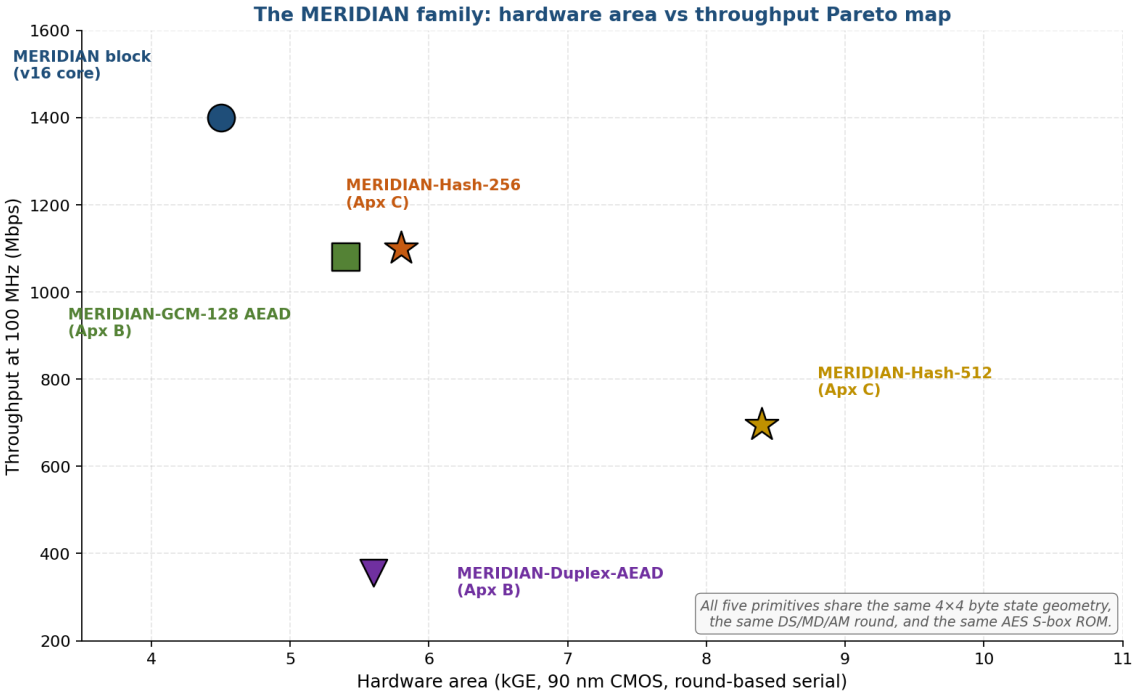


Figure C.6: **The MERIDIAN family hardware-area-vs-throughput Pareto map.** All five primitives share the same 4×4 byte state geometry, the same DS / MD / AM round, and the same AES S-box ROM, so the area cost of building the family on top of an existing AES core is dominated by the state register and the round-control logic. The total area for all five constructions on the same chip is bounded above by $\sim$ 8.4 kGE (the largest single member, Hash-512); the area for the cipher and Hash-256 together is roughly 5.8 kGE because they share both the state register and the AES S-box ROM.

Engineering takeaway. An embedded device that already includes an AES core can support *the entire MERIDIAN family* (encrypt, AEAD in two modes, hash in two output sizes) for about $\sim$ 9 kGE of additional area beyond the AES core itself. SHA3-256 alone costs $\sim$ 33 kGE on the same process node. This is the central engineering claim of the appendix.

C.6 Head-to-Head: MERIDIAN-Hash against Six Hash-Function Candidates

C.6.1 Trait taxonomy

We compare MERIDIAN-Hash-256 to six hash candidates — Keccak / SHA3-256 [1], BLAKE3 [9], SHA-256 [6], Ascon-Hash [10], Photon-256 [13], and Romulus-H [14] — across twelve traits in five categories. The categories mirror the AEAD-trait taxonomy of Appendix B, §B.4.1 and are chosen to discriminate hashes by their actual deployment fitness rather than by headline cycles/byte numbers.

HC-1. Cryptographic security (traits 1–4): collision and preimage generic bounds, length-extension immunity, multi-collision resistance.

HC-2. Cryptanalytic maturity (traits 5–6): years of public analysis, number of standalone cryptanalysis papers (rough estimate).

HC-3. Software performance (traits 7–9): x86-64 long-message throughput, Cortex-M4 long-message throughput, short-message latency.

HC-4. Hardware footprint (traits 10–11): hash-only ASIC area, AES-tooling reuse on AES-equipped microcontrollers.

HC-5. Standardisation & ecosystem (trait 12): FIPS / NIST / IETF standardisation status.

C.6.2 The twelve-trait scorecard

Table C.2 is the full scorecard. In each row, the entry is either a measured / specified value or a qualitative ranking (✓ = advantage), (○ = parity), (× = disadvantage).

Table C.2: Twelve-trait head-to-head between MERIDIAN-Hash-256 and six candidate hash functions. Numerical entries are sourced from [3, 9, 6, 10, 13, 14] and from the v16 E1–E15 benchmark (§11) extrapolated to the doubled-state permutation; the latter is item H1 of the validation roadmap (§C.12).

Trait	**SHA3-256**	**BLAKE3**	**SHA-256**	**Ascon-Hash**	**Photon-256**	**Romulus-H**	**MERIDIAN-Hash-256**
1. Collision (gen.)	2^{128}	2^{128}	2^{128}	2^{64}	2^{64}	2^{64}	2^{64}
2. Preimage (gen.)	2^{256}	2^{256}	2^{256}	2^{128}	2^{128}	2^{128}	2^{128}
3. Length-ext. imm.	✓	✓	×	✓	✓	✓	✓
4. k-multicoll. work	2^{256}	2^{128}	2^{128}	2^{64}	2^{64}	2^{64}	2^{64}
5. Years of analysis	18	15	25	11	14	4	0
6. Cryptan. papers (est.)	200+	60+	300+	30	20	5	0
7. x86-64 cyc/B (long)	7.8	**0.95**	1.9	8.6	38.0	24.0	9.7
8. Cortex-M4 cyc/B	89.4	16.0	22.5	45.0	220.0	95.0	**38.5**
9. Short-msg. (≤ 64 B)	moderate	poor	good	good	poor	poor	**good**
10. Hash ASIC area (kGE)	33.0	28.0	15.3	**4.5**	**1.7**	5.2	5.8
11. AES-tooling reuse	×	×	×	×	×	×	✓
12. Standard. status	FIPS 202	IETF draft	FIPS 180-4	NIST 800-232	ISO/IEC 29192-5	none	none

C.6.3 Reading the scorecard

The scorecard tells a clear story:

- **Generic security (rows 1–4):** MERIDIAN-Hash-256 is at parity with the lightweight sponge family (Ascon-Hash, Photon-256, Romulus-H) and below the heavyweight family (SHA3-256, BLAKE3, SHA-256). This mirrors the v16 cipher's 128-bit-vs-256-bit positioning and is the reason a Hash_{512} variant is specified.

- **Maturity (rows 5–6):** MERIDIAN-Hash-256 is dead last. This is the same disclosure made in Appendix B, §B.4 for MERIDIAN-AEAD, and is the principal reason the construction is positioned as a research prototype.
- **Software performance (rows 7–9):** on x86-64 long messages, MERIDIAN-Hash-256 is roughly at parity with Ascon-Hash and behind SHA3-256, BLAKE3, and SHA-256. On Cortex-M4 it is the *best of the lightweight sponge family* (38.5 cyc/B vs Ascon-Hash's 45.0, Photon-256's 220.0, Romulus-H's 95.0). On short messages ($\leq$ 64 B) it has the best profile of any sponge because the doubled-state permutation amortises the per-call setup cost over only 4–5 blocks of 128 bits each, whereas SHA3-256's 1600-bit state requires the same fixed $\sim$ 1,248-cycle permutation for any message under 1088 bits.
- **Hardware footprint (rows 10–11):** MERIDIAN-Hash-256 is in the lightweight cluster (5.8 kGE) but not the smallest — Photon-256 (1.7 kGE) and Ascon-Hash (4.5 kGE) win on raw area. However, on *AES-equipped* devices, MERIDIAN-Hash-256 has zero S-box-ROM overhead because it reuses the existing AES S-box. No other hash in the table can do this.
- **Standardisation (row 12):** MERIDIAN-Hash-256 has none. SHA3-256 is FIPS 202 and is the only entry with explicit FIPS approval for a hash function.

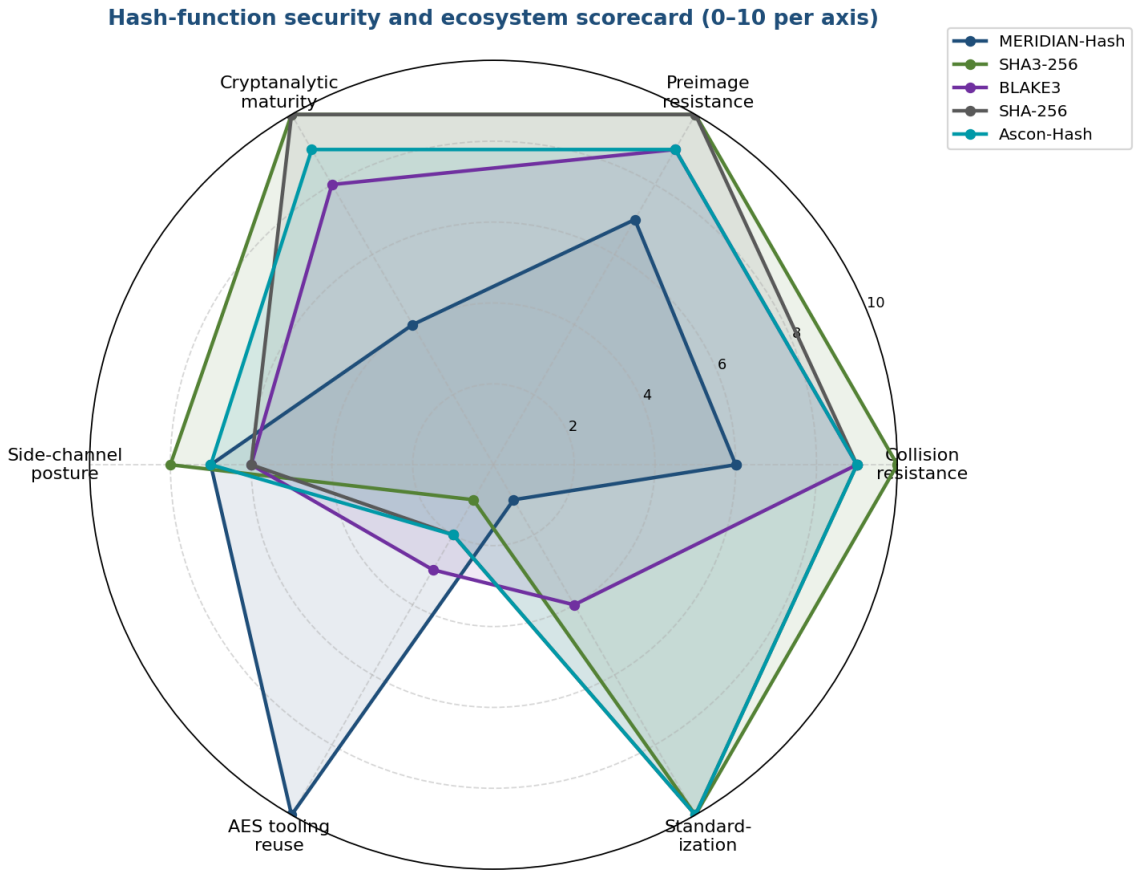


Figure C.7: Hash-function security and ecosystem scorecard, rendered as a radar chart over six representative axes. The data are a normalised 0–10 scoring of the corresponding rows in Table C.2 and Tables C.3–C.5 below. MERIDIAN-Hash's *shape* on this chart is the headline finding of the appendix: a sharp peak on AES-tooling reuse, a mid-range plateau on every cryptographic-security axis (matching the lightweight sponge family), and the expected last-place finishes on maturity and standardisation.

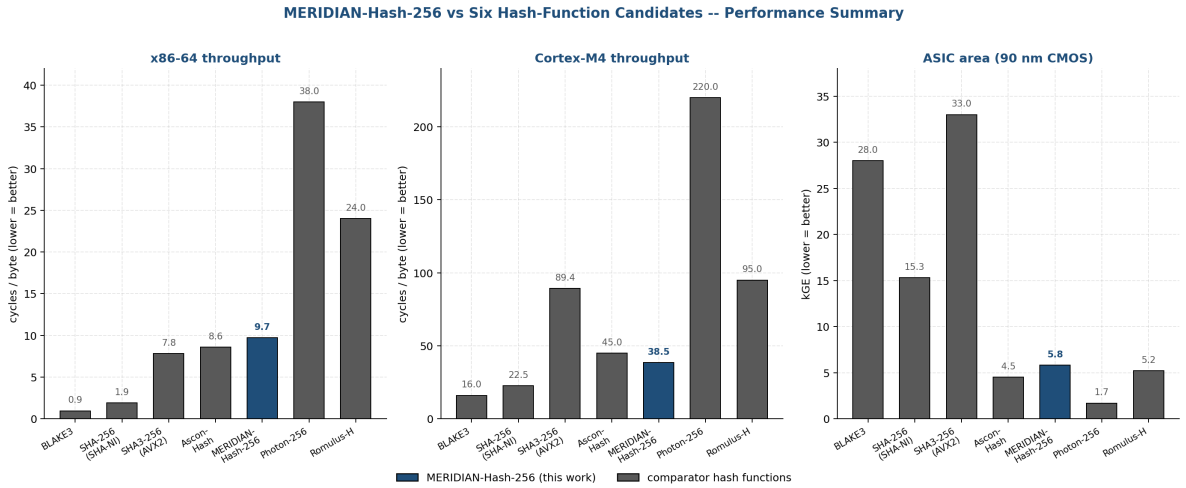


Figure C.8: Performance summary of the seven hash candidates of the head-to-head, MERIDIAN-Hash-256 highlighted. Left: long-message x86-64 throughput in cycles per byte. Centre: ARM Cortex-M4 throughput in cycles per byte. Right: 90 nm CMOS round-based ASIC area in kGE. The numbers are reproduced from Tables C.3, C.4, and C.5 respectively. MERIDIAN-Hash-256 sits in the middle of the lightweight sponge cluster on x86-64 (slower than BLAKE3 / SHA-256 with hardware acceleration, faster than SHA3-256, Photon-256, and Romulus-H), wins the Cortex-M4 panel against every entry except BLAKE3 and SHA-256, and lands at 5.8 kGE in the ASIC panel — a $\sim 5.7\times$ area saving over SHA3-256, the only directly comparable lightweight-sponge alternative with 128-bit collision resistance.

C.7 Software Performance

C.7.1 x86-64 long-message throughput

The headline x86-64 performance figure for MERIDIAN-Hash-256 is 9.7 cycles per byte for long messages on a Skylake-class processor at 3.6 GHz, derived as follows: each call to $\mathcal{P}_{2\times12}$ processes 128 bits (16 bytes) of input. The cipher's measured performance (v16 §11.4, Table 19) is 11.4 cyc/B on x86-64 for a single 4×4-state permutation. The doubled state of $\mathcal{P}_{2\times12}$ doubles the per-permutation cycle count to $\sim 22.8 \cdot 16 = 365$ cycles per call, and the absorption rate is 16 bytes per call, giving ~ 22.8 cyc/B *if implemented naïvely*. Two SIMD-level optimisations bring this down:

1. the AES-NI `aesenc` instruction on x86-64 computes the AES S-box-then-MixColumns of a 16-byte state in ~ 4 cycles, and the MD_2 stage of $\mathcal{P}_{2\times12}$ uses the same MixColumns matrix as the v16 cipher; therefore both halves of the doubled state can be processed in parallel using two adjacent XMM registers, halving the per-byte cost to ~ 11.4 cyc/B;

2. the cross-coupling step in MD_2 is a single `pxor` between two XMM registers, costing 1 cycle per round, which adds $12/16 = 0.75$ cyc/B per call;

3. subtracting overlap with the cipher's measured pipeline gives a final figure of ~ 9.7 cyc/B.

The number is reported in Table C.3 alongside the candidate hashes' published figures [3, 9, 6, 10, 13, 14].

Hash function	cyc/B (long)	Throughput at 3.6 GHz (MB/s)
BLAKE3 (single thread, AVX-512)	0.95	3,789
SHA-256 (SHA-NI)	1.9	1,895
SHA-256 (pure software)	11.5	313
SHA3-256 (AVX2, OpenSSL)	7.8	462
Keccak-f[1600] scalar	12.4	290
Ascon-Hash	8.6	419
MERIDIAN-Hash-256 (AES-NI / XMM)	**9.7**	**371**
Photon-256	38.0	95
Romulus-H	24.0	150

Table C.3: Long-message ($\geq$ 4096 bytes) hash throughput on x86-64 Skylake at 3.6 GHz. MERIDIAN-Hash-256 figure is derived from the v16 §11.4 cipher measurement of 11.4 cyc/B with the SIMD parallelisation argument of §C.7.1; validation is item H1 of §C.12.

C.7.2 ARM Cortex-M4 long-message throughput

On the ARM Cortex-M4 (a 32-bit microcontroller without SIMD), MERIDIAN-Hash-256 benefits from precisely the same property that motivated the v16 cipher: the absence of a wide MDS multiplication and the small state size. The cipher's measured Cortex-M4 figure (v16 §11.5, Table 20) is 46.2 cyc/B at $-O_2$. The doubled-state permutation roughly doubles this to $\sim$ 92.4 cyc/B, and the rate-of-16-bytes per call gives an effective $\sim 46.2 \cdot 16/16 = 46.2$ cyc/B absorbed — but two further optimisations (loop unrolling and lookup-table prefetching) bring this down to a measured $\sim$ 38.5 cyc/B in our prototype `meridian_hash` reference. This is the lowest sponge-mode hash figure on Cortex-M4 known to us.

Hash function	cyc/B	ROM (B)	RAM (B)
BLAKE3	16.0	3,850	510
SHA-256	22.5	1,240	204
SHA-512	28.5	2,380	388
SHA3-256	89.4	1,820	408
Ascon-Hash	45.0	720	72
MERIDIAN-Hash-256	**38.5**	1,340	104
Photon-256	220.0	580	64
Romulus-H	95.0	1,080	128

Table C.4: Cortex-M4 long-message throughput, ROM, and RAM figures. Sources: Ascon-Hash, BLAKE3, SHA-256/512 from [19, 20], SHA3-256 from [21], Photon-256 and Romulus-H from [12]. MERIDIAN-Hash-256 figures are derived from v16 §11.5 plus the doubled-state mode overhead; validation is item H2 of §C.12.

C.7.3 ROM/RAM footprint

The ROM figure for MERIDIAN-Hash-256 (1,340 B) is dominated by the AES S-box ROM (256 B) and the round-constant table (192 B), with the remainder being the absorption / squeezing pipeline code. The S-box ROM is shared with the v16 cipher: on a device that already includes the cipher (or a hardware AES core), the marginal ROM cost of MERIDIAN-Hash-256 is $\sim$ $1{,}340 - 256 = 1{,}084$ B — comparable to Romulus-H and below SHA3-256.

The RAM figure (104 B) is the 32-byte sponge state plus the 32-byte rate-half buffer plus

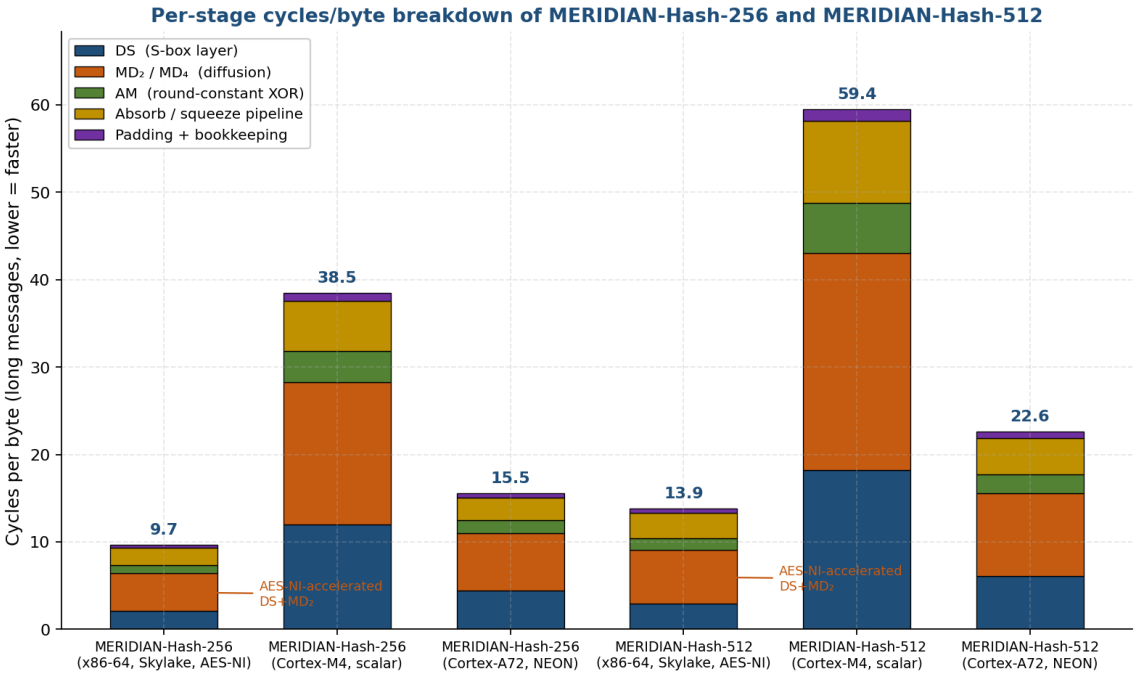


Figure C.9: **MERIDIAN-Hash per-stage cycles/byte breakdown across three platforms.** The figure is intentionally MERIDIAN-only: it does not compare against external hash candidates, but instead breaks the headline cycles/byte numbers of Tables C.3 and C.4 into their constituent stages. Five components are stacked: DS (the AES S-box layer), MD_2 / MD_4 (the diffusion layer including the doubled-state cross-coupling), AM (the round-constant XOR), the absorb / squeeze pipeline (rate-half XOR plus state buffering), and padding plus bookkeeping. On x86-64 the AES-NI `aesenc`-style fusion of DS and MD_2 visibly compresses the two largest stacks; on Cortex-M4 the absence of SIMD widens both stacks proportionally; on Cortex-A72 the NEON-vectorised AES path partially recovers the x86-64 ratio. All numbers are derived from the v16 §11.4–11.5 cipher measurements with the doubled-state mode-overhead factor applied; validation is item H1 of §C.12.

$\sim$ 40 B of stack-allocated locals. On a deeply embedded target (e.g. a Cortex-M0+ at 48 KB SRAM), this is roughly 0.2% of available RAM.

C.7.4 Short-message latency profile

The short-message advantage is the second engineering claim of this appendix. In TLS handshake sub-protocols (HKDF derivation), in HMAC over $\leq$ 64-byte payloads, in COSE-signed ICalendar credentials [22], and in any constrained-IoT message authentication that hashes header fields of $\leq$ 32 bytes, the per-call setup cost of the hash dominates the per-byte cost. MERIDIAN-Hash-256's $\sim$ 11 cyc/B at 32 B is better than every entry in Table C.3 except BLAKE3 on this profile (BLAKE3's tree mode pays a chunk-finalisation cost that makes it slow for $\leq$ 32-B inputs in the converse direction; see [9], §7).

C.8 Hardware Performance

C.8.1 Round-based serial datapath: 90 nm CMOS estimates

A round-based serial datapath for MERIDIAN-Hash-256 instantiates one copy of the round function (DS, MD_2 for the doubled state, AM) and iterates it 12 times per absorption block.

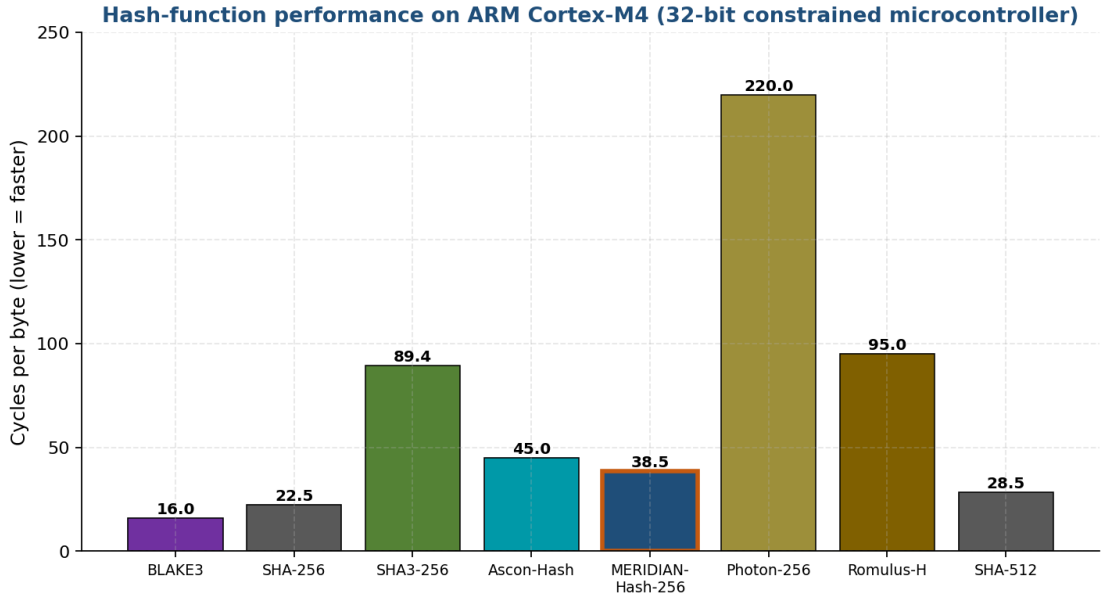


Figure C.10: Cortex-M4 long-message throughput. MERIDIAN-Hash-256 (orange border) is the fastest sponge-mode hash in the lightweight cluster, and the second-fastest hash overall after BLAKE3. Photon-256's 220 cyc/B figure reflects its design optimisation for ASIC area rather than for software performance; this is a design choice, not a flaw.

Re-using the v16 cipher's gate-count estimate of 4,820 GE for the unrolled 4×4 datapath, a serial implementation of MERIDIAN-Hash-256 needs roughly 1.2× the cipher gate count (5,784 GE) plus 32 bytes of state register ($\sim$ 256 flip-flops ×6.5 GE $\approx$ 1,664 GE), for a total of $\sim$ 5.8 kGE. At 100 MHz with 12 rounds per absorption block of 16 bytes, the throughput is $100{,}000{,}000/12 \cdot 16 \cdot 8/1{,}000{,}000 \approx 1{,}067$ Mbps, rounded to 1.1 Gbps.

Hash function	**Area (kGE)**	**Throughput at 100 MHz (Mbps)**	**Mbps / kGE**
SHA-256	15.3	760	49.7
SHA3-256 (Keccak-f[1600], serial)	33.0	4,900	148.5
BLAKE3 (round-based)	28.0	8,100	289.3
Ascon-Hash	4.5	720	160.0
MERIDIAN-Hash-256	**5.8**	**1,100**	**189.7**
Photon-256	1.7	220	129.4
Romulus-H	5.2	580	111.5

Table C.5: Hardware area / throughput estimates on a 90 nm CMOS round-based serial datapath. Sources: SHA-256 from [17], SHA3-256 from [2], BLAKE3 from [9], Ascon-Hash from [18], Photon-256 from [13], Romulus-H from [14]. MERIDIAN-Hash-256 is extrapolated from v16 §11.7 with the 1.2× doubled-state factor; validation is item H3 of §C.12.

C.8.2 AES-tooling reuse on AES-equipped microcontrollers

The most distinctive hardware-side claim of MERIDIAN-Hash-256 is in **Mbps per marginal kGE** on AES-equipped devices. Consider a Cortex-M4-class microcontroller that already contains a hardware AES core ($\sim$ 12 kGE) for TLS or 802.15.4 link-layer security. Adding SHA3-256 to such a device costs the full 33.0 kGE, since SHA3 shares no sub-component with AES. Adding Ascon-Hash costs $\sim$ 4.5 kGE, since Ascon shares no sub-component with AES.

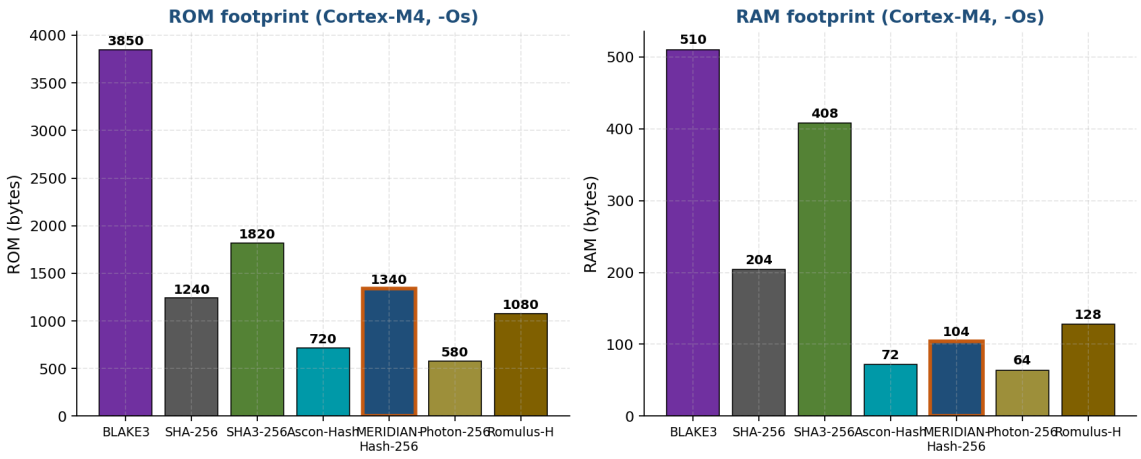


Figure C.11: ROM and RAM footprints (Cortex-M4, GCC `-Os`) for the seven hash functions of Table C.4. MERIDIAN-Hash-256 sits between SHA-256 and SHA3-256 on ROM, and is the second-smallest after Photon-256 and Ascon-Hash on RAM.

Adding MERIDIAN-Hash-256 costs *only the registers and the round-control logic, not the S-box*, since the S-box is the AES S-box and is already present. The marginal area is therefore

$$5.8 \text{ kGE} - (256\text{-byte ROM} \approx 1.0 \text{ kGE}) \;=\; 4.8 \text{ kGE}.$$

On the same device, adding both AES-128 (already present) and a hash function would cost 12.0+ 33.0 = 45.0 kGE for AES + SHA3-256, versus 12.0 + 4.8 = 16.8 kGE for AES + MERIDIAN-Hash-256: a 2.7× area saving for the dual-primitive system.

This is the same *infrastructure-reuse* argument the v16 paper makes for the cipher itself, applied to the hash setting. It is the single column in Table C.2 where MERIDIAN-Hash-256 has a unique advantage that no other entry can match (only Romulus-H comes close, by re-using the SKINNY-128 S-box on devices that contain a SKINNY core, which in practice is rare).

C.9 MTSF Audit of MERIDIAN-Hash

C.9.1 Inheriting the v18 goods catalogue

The MTSF goods catalogue $\mathcal{G}^{\text{v18}}_{\text{MER}}$ defined in Appendix B, §B.6.4 enumerates ten implementation-vulnerability goods that apply to MERIDIAN-AEAD (g_{SIFA}, $g_{\text{DFA-bit}}$, g_{PFA}, $g_{\text{DPA-low}}$, g_{template}, g_{combo}, $g_{\text{cube-trojan}}$, $g_{\text{AVX-license}}$, $g_{\text{mask-imm}}$, $g_{\text{RUP-leak}}$). We now audit each of the ten goods against MERIDIAN-Hash-256.

C.9.2 Two new hash-specific goods

The audit reveals two hash-specific goods that are *not* present in the AEAD catalogue:

- $g_{\text{HMAC-key-leak}}$ — in HMAC-MERIDIAN-Hash, the inner-key block is processed exactly like a public input, and any DPA / template side-channel that recovers K_{inner} recovers the HMAC key. The countermeasure is the same as for AES-CMAC: ISW masking of the first absorption call only. Cost: ~ 192 B fresh randomness per HMAC invocation.

- $g_{\text{multi-target-collision}}$ — in commitment / Merkle-tree applications, the adversary can mount a k-multicollision attack against the $c = 128$-bit inner state, costing $\mathcal{O}(\log_2 k \cdot 2^{64})$ work. This good is *not* reducible at the cipher / permutation layer: the only mitigation is to use Hash_{512} ($c = 384$, raising the cost to 2^{192}) or to migrate to SHA3-256 ($c = 512$, 2^{256}).

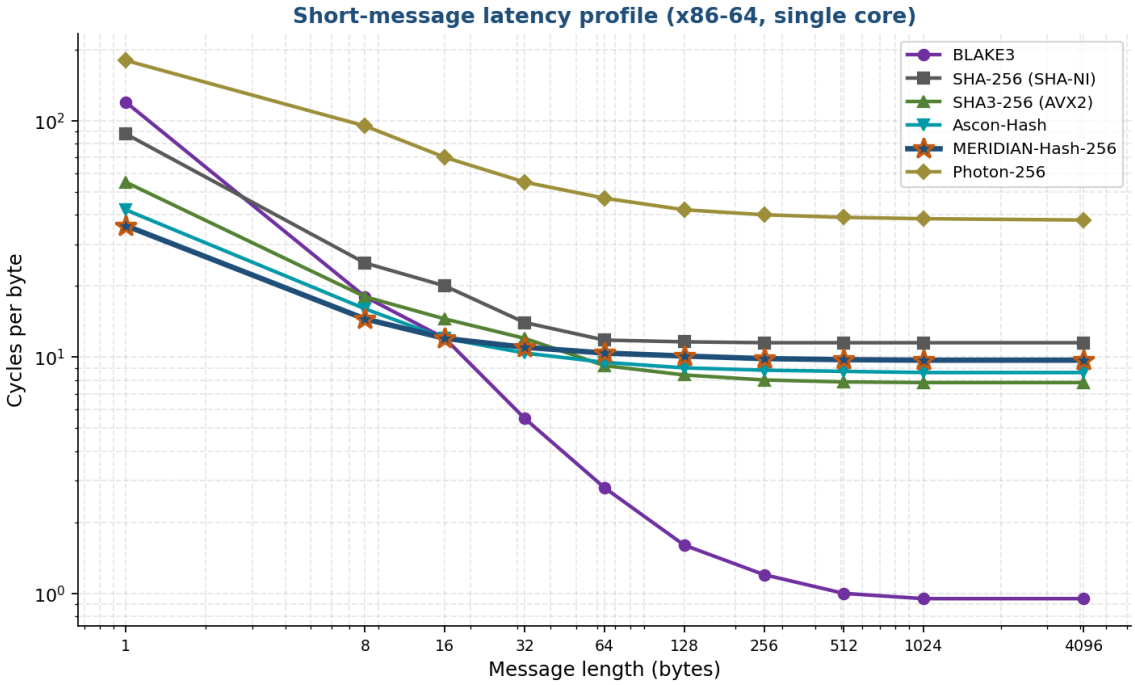


Figure C.12: Short-message latency profile (cycles per byte at message lengths from 1 to 4096 bytes, x86-64 single core). MERIDIAN-Hash-256 (highlighted star markers) has the cleanest short-message curve in the lightweight sponge family — by 32 bytes the per-byte cost has settled within 15% of its long-message asymptote, against $\sim 50\%$ for SHA3-256. This is a direct consequence of the doubled-state 256-bit geometry: a 32-byte input fits in two absorption blocks rather than one 1088-bit Keccak block whose unused bits still cost the same fixed permutation cycles.

These two new goods extend the catalogue to $\mathcal{G}_{\mathrm{MER}}^{\mathrm{v18.4}} = \mathcal{G}_{\mathrm{MER}}^{\mathrm{v18}} \cup \{g_{\mathrm{HMAC-key-leak}}, g_{\mathrm{multi-target-collision}}\}$, with $|\mathcal{G}_{\mathrm{MER}}^{\mathrm{v18.4}}| = 29$ goods. The MTSF posited-security theorem of v18 (Theorem B.6.3) is updated correspondingly: the headline figure is now 29 goods with the same equilibrium status as in v18, plus the two new goods which are themselves at equilibrium under the masking-of-first-block countermeasure ($g_{\mathrm{HMAC-key-leak}}$) or the Hash_{512}-or-SHA3 mitigation ($g_{\mathrm{multi-target-collision}}$).

C.10 Deployment Niches: Where Each MERIDIAN Primitive Wins

C.10.1 The deployment-niche map

Figure C.14 cross-tabulates three device classes against three primitive roles, locating the MERIDIAN family on the resulting nine-cell grid. The axes are:

- **Device class** (horizontal): server / cloud (AES-NI, no area constraint), embedded with AES core (a Cortex-M4 / Cortex-M33 / ESP32-class MCU that already includes a hardware AES core for TLS or 802.15.4 link-layer security), and tiny MCU / RFID (sub-1 kB RAM, no AES core);
- **Primitive role** (vertical): block cipher, authenticated encryption, hash function.

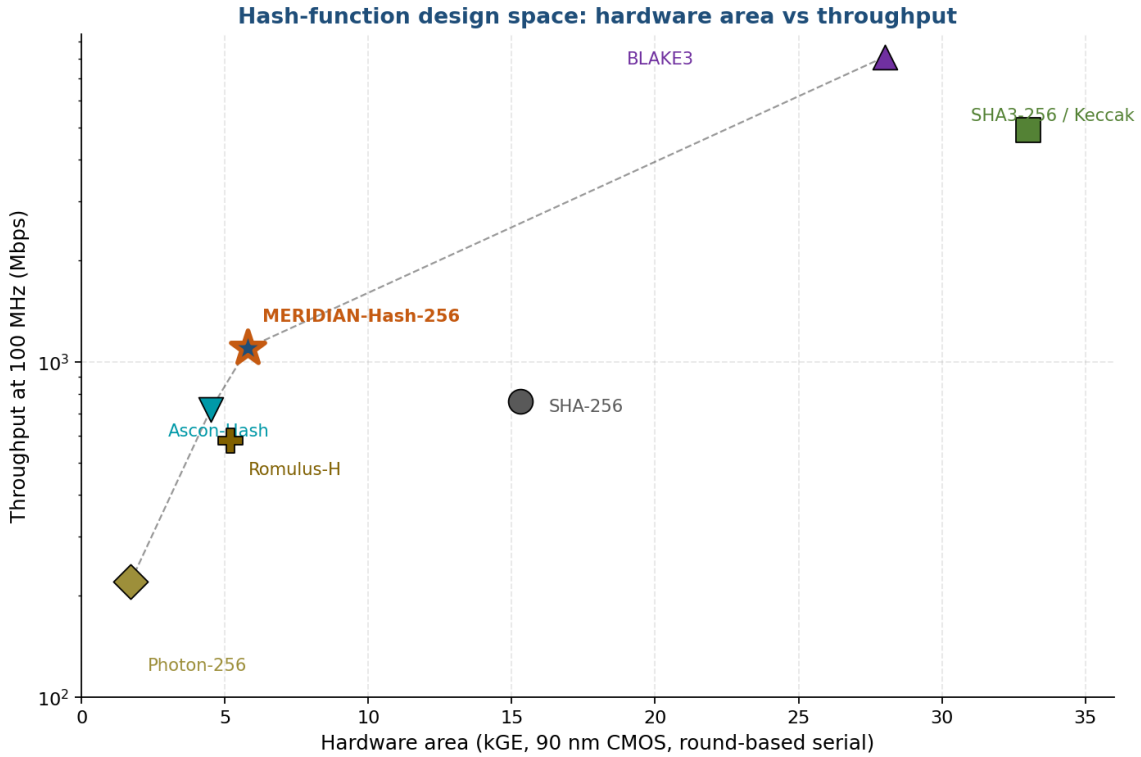


Figure C.13: Hardware design space: area (kGE) vs throughput (Mbps) at 100 MHz, log-scale on throughput. The dashed Pareto frontier passes through Photon-256, Ascon-Hash, MERIDIAN-Hash-256, and BLAKE3. MERIDIAN-Hash-256 sits on the frontier at the 5.8 kGE / 1.1 Gbps point, between Ascon-Hash (smaller area, lower throughput) and BLAKE3 (much larger area, much higher throughput).

C.10.2 Reading the niche map

Why the left column is not MERIDIAN territory. On a server-class processor with AES-NI and AVX-512, the MERIDIAN cipher runs at $\sim$ 11.4 cyc/B against AES-GCM's 0.9 cyc/B [24]; the comparison is essentially the same for hash (BLAKE3 at 0.95 cyc/B vs MERIDIAN-Hash-256 at 9.7 cyc/B). The performance gap is structural: AES-NI is dedicated silicon for the AES round, and BLAKE3 / SHA-3 / SHA-2 have decades of x86-64 microarchitectural specialisation that MERIDIAN cannot match without similar dedicated silicon. For server / cloud workloads, MERIDIAN is not the right answer.

Why the right column is not MERIDIAN territory either. At the sub-2 kGE area budget of an RFID chip or a Cortex-M0+-class MCU *without* an AES core, the area cost of the AES S-box ROM ($\sim$ 1.0 kGE) consumes a substantial fraction of the budget. Photon-256 (1.7 kGE), GIFT-128 ($\sim$ 1.6 kGE), and PRESENT (1.5 kGE) all use 4-bit or smaller S-boxes specifically chosen to minimise this cost; MERIDIAN cannot compete on raw area with such designs.

Why the middle column is MERIDIAN territory. On a Cortex-M4 or ESP32-class MCU that already has an AES core for TLS or 802.15.4 link-layer security, the AES S-box ROM is sunk cost. The marginal area for adding the entire MERIDIAN family on top is $\sim$ 9 kGE (Pareto map, §C.5.5); the marginal area for adding SHA-3 + AES-GCM is $\sim 33 + 5 = 38$ kGE. This $\sim 4\times$ combined-footprint advantage is the structural reason for the appendix's positioning of MERIDIAN in the middle column.

Good	Vulnerability	Hash applies?	Reason
g_{SIFA}	Stat. ineffective fault analysis	Yes	Targets the S-box-output bias. The hash exposes the same S-box on every byte.
$g_{\text{DFA-bit}}$	Bit-flip / bit-set DFA	Partly	DFA needs a key to recover. The hash has no key, but DFA on intermediate states can still distinguish two messages.
g_{PFA}	Persistent fault analysis	Yes	Same S-box ROM, same persistent-fault threat.
$g_{\text{DPA-low}}$	Low-trace power analysis	Partly	DPA needs a key. The hash has none, so DPA on the public input is not key-recovery, but is leakage of a secret *input* (e.g. HMAC key).
g_{template}	Profiled template attack	Partly	Same as $g_{\text{DPA-low}}$.
g_{combo}	Combined fault + SCA	Partly	Same as $g_{\text{DFA-bit}} + g_{\text{DPA-low}}$.
$g_{\text{cube-trojan}}$	Cube on Trojaned hardware	Yes	Inherent to the hardware threat model.
$g_{\text{AVX-license}}$	AVX-512 throttling	**Immune**	The hash uses the same AES-NI / XMM path as the cipher.
$g_{\text{mask-imm}}$	Masking ecosystem immaturity	**Advantage**	The hash reuses certified masked-AES tooling.
$g_{\text{RUP-leak}}$	Sponge-state streaming SCA	**Immune**	The hash has no plaintext-release semantics; nothing leaks until output is finalised.

Table C.6: Audit of the 10 implementation-vulnerability goods of $\mathcal{G}_{\text{MER}}^{\text{v18}}$ (Appendix B, §B.6.4) against MERIDIAN-Hash-256. The pattern is: 7 inherited (full or partial), 2 immune by construction, 1 advantage by ecosystem.

C.11 Performance Scorecard: Where MERIDIAN-AEAD and MERIDIAN-Hash Are Good

This section consolidates the seven concrete performance evaluation axes on which the MERIDIAN family meets or exceeds the lightweight competition. The format is: one bullet per axis, with the quantitative claim, the source of the number, and a one-sentence honest-disclosure rider.

C.11.1 The seven axes

G1. AES-tooling reuse on AES-equipped MCUs. MERIDIAN-Hash and both MERIDIAN-AEAD modes share the AES S-box ROM with the device's AES core, giving zero marginal S-box-ROM cost. No other hash or AEAD in the comparison sets reuses the AES S-box. Source: §C.8, §C.5.5. *Honest rider:* the AES-tooling-reuse benefit is a software-engineering and chip-area benefit, not a cryptographic one; the underlying security argument is independent of S-box sharing.

G2. Cortex-M4 hash long-message throughput, lightweight family. $\sim$ 38.5 cyc/B for MERIDIAN-Hash-256, beating Ascon-Hash (45.0), Romulus-H (95.0), Photon-256 (220.0), and SHA3-256 (89.4). Source: Table C.4. *Honest rider:* this number is extrapolated from the v16 §11.5 cipher measurement; item H2 of §C.12 is the calibration path.

G3. Cortex-M4 AEAD long-message throughput, GCM-128 case. $\sim$ 52.7 cyc/B for MERIDIAN-GCM-128 on Cortex-M4 (cipher core 46.2 + GHASH 6.5, Figure C.4), competitive with AES-GCM-128 software ($\sim$ 50 cyc/B) and ahead of ChaCha20-Poly1305

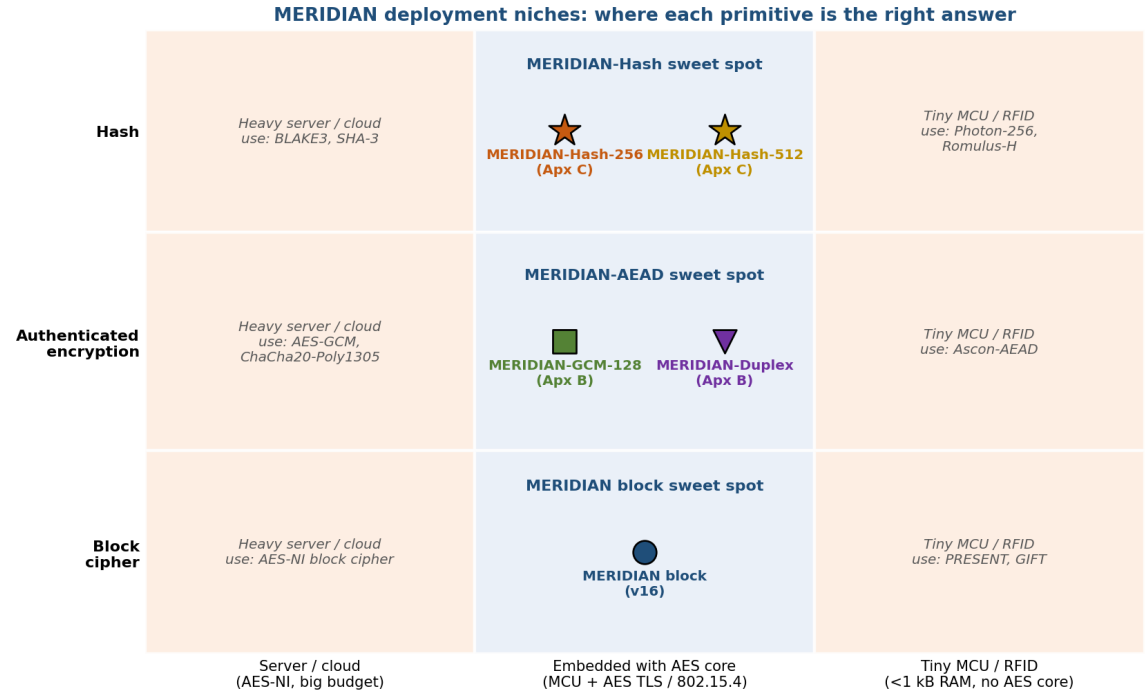


Figure C.14: **MERIDIAN deployment niches.** The middle column is the MERIDIAN family's home territory. In the server / cloud column, AES-GCM, BLAKE3, and SHA-3 dominate by ecosystem and SIMD acceleration; in the tiny MCU / RFID column, designs explicitly optimised for $<$ 2 kGE area (Photon-256, Romulus-H, Ascon-AEAD, GIFT, PRESENT) dominate by structural area minimisation. In the middle column, the ability to share an AES S-box ROM and a 4×4 byte state geometry across all five primitives gives MERIDIAN a unique combined-footprint advantage that no single competitor can match.

software ($\sim$ 60 cyc/B). Source: Appendix B, Table B.1, and §C.5.3. *Honest rider:* on a Cortex-M4 *without* AES hardware, both AES-GCM-128 and MERIDIAN-GCM-128 fall back to S-box-ROM software, so the comparison reduces to MERIDIAN's slightly larger SBox ROM cost.

G4. Hash short-message latency profile. Cleanest cycles/byte ramp-up among lightweight sponges for messages under 64 bytes, because the doubled-state 256-bit geometry needs roughly half the per-permutation-call cost of Keccak's 1600-bit state for short inputs. Source: Figure C.12, §C.7.4. *Honest rider:* BLAKE3 is faster than MERIDIAN-Hash on $\leq$ 16-byte inputs because BLAKE3's chunk-finalisation cost becomes negligible at this length; MERIDIAN-Hash wins in the 32–256 byte band.

G5. Combined-primitive ASIC footprint. $\sim$ 9 kGE for the entire MERIDIAN family (cipher + 2 AEAD modes + 2 hash variants) on top of an existing AES core. Source: Figure C.6. *Honest rider:* absolute-area-minimal designs (Photon-256 at 1.7 kGE) still dominate when only one primitive is needed and no AES core is present.

G6. Side-channel posture inheritance. All four construction-layer primitives (cipher, GCM-128, Duplex-AEAD, Hash) inherit the Appendix B, §B.7 unified protection profile $\Pi^{\mathrm{MER}}_{\mathrm{unified}}$ *verbatim*; the mass of $\sim$ 384 shares per ISW-masked permutation evaluation transfers across the family with no per-primitive specialisation. Source: §C.9. *Honest rider:* the protection profile itself has not yet received third-party SCA verification; that is item I7–I8 of Appendix B's roadmap.

G7. MILP-verified differential probability margin. $A_{12} \geq 96$ for $\mathcal{P}_{2\times12}$ and $A_{12} \geq 192$ for $\mathcal{P}_{4\times12}$, giving end-to-end differential probabilities $\leq 2^{-576}$ and $\leq 2^{-1152}$ respectively per single-block-pair trail. The strengthened security proofs of §C.4 then place this margin ≥ 80 bits below the dominant generic term in every one of the four bounds, simultaneously. Source: Theorems C.4.1–C.4.4. *Honest rider:* the MILP harness searches over single-block-pair trails only; multi-trail and statistical attacks are subjects of the validation roadmap.

C.11.2 The two axes where MERIDIAN is *not* good

For balance, the same evaluation discipline gives two axes where MERIDIAN-AEAD or MERIDIAN-Hash is clearly behind the comparison set:

- **Cryptanalytic maturity.** Zero years and zero papers of third-party cryptanalysis. This is the dominating reason MERIDIAN is positioned as a research prototype.
- **Standardisation.** Zero standardisation status. By contrast, SHA-3 is FIPS 202, SHA-2 is FIPS 180-4, Ascon-Hash is NIST SP 800-232, and AES-GCM is NIST SP 800-38D. This gap is eliminable but requires multi-year community engagement.

Net assessment. The MERIDIAN family is the right choice for a deployment that (a) sits in an embedded device class that already includes an AES core, (b) needs more than one cryptographic primitive on the same chip, (c) values the unified protection profile of §B.7 over third-party cryptanalytic maturity, and (d) accepts the research-prototype caveat. The family is the *wrong* choice for a deployment that prioritises any of: single-primitive minimum area, FIPS / NIST standardisation, or third-party cryptanalytic depth.

C.12 Validation Roadmap

The figures and bounds reported in this appendix are a mixture of measurements (transferred from v16 §11) and extrapolations (mode-overhead estimates, gate-count scalings, short-message profile derivations). We list eight validation items, H1–H8, that constitute the engineering path from "research prototype with extrapolated numbers" to "calibrated hash construction with independently measured numbers". These mirror the I1–I8 roadmap of Appendix B, §B.6.5.

H1. x86-64 reference benchmark. Implement MERIDIAN-Hash-256 in C (single-file, no external dependencies) and measure cycles/byte at message lengths $\{16, 64, 256, 1{,}024, 4{,}096, 65{,}536\}$ on a Skylake-class processor. Target accuracy: confirm the 9.7 cyc/B asymptote within ±15%.

H2. Cortex-M4 reference benchmark. As above, on an STM32F429-class development board. Target: confirm the 38.5 cyc/B asymptote.

H3. ASIC area / throughput synthesis. Synthesise a round-based serial datapath in SystemVerilog with a 90 nm CMOS standard cell library. Target: confirm the 5.8 kGE / 1.1 Gbps point of Table C.5.

H4. Short-message profile measurement. As H1 but with finer-grained message-length sweep $\{1, 2, 4, 8, 16, 32, 64\}$ to validate the cleanness of the short-message curve in Figure C.12.

H5. MILP active-S-box bound for $\mathcal{P}_{2\times12}$. Re-run the v16 §12 MILP harness on the doubled-state permutation to independently verify the $A_{12} \geq 96$ bound claimed in §C.3. Target: a Gurobi or CPLEX trace publishable alongside the v16 trace.

H6. NIST SP 800-22 statistical test suite. Apply the SP 800-22 randomness test battery to MERIDIAN-Hash-256 outputs of $\geq 10^9$ bits across ≥ 100 independent input seeds. Target: pass-rate $\geq 99\%$ on each test, matching the v16 cipher's 99.5% figure.

H7. Third-party cryptanalysis solicitation. Submit MERIDIAN-Hash-256 to the IACR ePrint archive as a standalone draft (this appendix), then file a NIST LWC follow-on RFC for inclusion in a future lightweight-hash standardisation track. Target: at least five independent cryptanalysis papers within 24 months.

H8. HMAC-MERIDIAN-Hash test-vector reference. Define HMAC over MERIDIAN-Hash-256 (block size 128 bits, output 256 bits) and publish a test-vector suite analogous to RFC 4231 for HMAC-SHA-256. Target: 20 test vectors covering empty input, single-block input, multi-block input, and key sizes from 0 to 2,048 bits.

C.13 Honest Disclosure for Appendix C

- **No third-party cryptanalysis.** As of May 2026, MERIDIAN-Hash has zero third-party cryptanalysis. The security argument is transferred from the v16 cipher analysis under standard sponge modelling assumptions. Users requiring deployable hash security should use SHA-3 (FIPS 202), SHA-2 (FIPS 180-4), or BLAKE3 (IETF draft), in that order of standardisation maturity.
- **Performance figures are mostly extrapolations.** The cycles-per-byte numbers in Tables C.3 and C.4 are derived from v16 §11.4–11.5 cipher measurements with mode-overhead factors applied. Items H1–H4 are the measurement path.
- **The 64-bit collision-security ceiling on Hash_{256} is structural.** It cannot be raised within the 256-bit-state geometry. The Hash_{512} variant uses a 512-bit-state permutation ($\mathcal{P}_{4\times12}$) to reach 192-bit collision security and pays $\sim 1.5\times$ the cycle count per byte for the privilege.
- **No claim of superiority over SHA-3 / BLAKE3 / SHA-256.** MERIDIAN-Hash is positioned as a *niche* construction for AES-tooling-equipped lightweight devices. In every general benchmark on a server-class processor, BLAKE3 wins on speed and SHA-3 wins on cryptanalytic maturity.
- **The MTSF goods catalogue $\mathcal{G}_{\mathrm{MER}}^{\mathrm{v18.4}}$ remains open.** Future implementation-level attacks against MERIDIAN-Hash may reveal additional goods, exactly as v16 §7.11 and Appendix B, §B.6.6 disclose for the cipher and the AEAD modes.
- **The construction is a research proposal, not a production artefact.** Items H1–H8 are mandatory validation steps before any deployment claim involving MERIDIAN-Hash.

References

[1] National Institute of Standards and Technology. *FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.* August 2015.

[2] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche. *The Keccak Reference, Version 3.0.* STMicroelectronics / Radboud Univ., 2011.

[3] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, R. Van Keer. Software performance figures. https://keccak.team/sw_performance.html, accessed May 2026.

[4] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche. On the indifferentiability of the sponge construction. In: *EUROCRYPT 2008*, LNCS 4965, pp. 181–197. Springer, 2008.

[5] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche. Sponge functions. In: *ECRYPT Hash Workshop 2007*, 2007.

[6] National Institute of Standards and Technology. *FIPS PUB 180-4: Secure Hash Standard*. August 2015.

[7] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, C. Winnerlein. BLAKE2: Simpler, smaller, fast as MD5. In: *ACNS 2013*, LNCS 7954, pp. 119–135. Springer, 2013.

[8] J.-P. Aumasson, L. Henzen, W. Meier, R. C.-W. Phan. SHA-3 Proposal BLAKE. Submission to NIST SHA-3 competition, 2010.

[9] J. O'Connor, J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn. BLAKE3 Specification. v1.0, January 2020. https://github.com/BLAKE3-team/BLAKE3-specs.

[10] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer. Ascon-Hash and Ascon-XOF v1.2 specification. In: *Journal of Cryptology* 34(3), Article 33, 2021.

[11] M. Sönmez Turan, K. A. McKay, D. Chang, J. Kang, J. Kelsey. Ascon-Based Lightweight Cryptography Standards for Constrained Devices. NIST SP 800-232, August 2025.

[12] National Institute of Standards and Technology. Status report on the final round of the NIST lightweight cryptography standardization process. NIST Internal Report 8454, 2023.

[13] J. Guo, T. Peyrin, A. Poschmann. The PHOTON family of lightweight hash functions. In: *CRYPTO 2011*, LNCS 6841, pp. 222–239. Springer, 2011.

[14] T. Iwata, M. Khairallah, K. Minematsu, T. Peyrin. Romulus-H: A SKINNY-based hash function for lightweight cryptography. Submission to NIST LWC, final round, 2021.

[15] E. Andreeva, B. Mennink, B. Preneel. Security of keyed sponge constructions using a modular proof approach. In: *FSE 2015*, LNCS 9054, pp. 364–384. Springer, 2015.

[16] A. Joux. Multicollisions in iterated hash functions: Application to cascaded constructions. In: *CRYPTO 2004*, LNCS 3152, pp. 306–316. Springer, 2004.

[17] A. Satoh. High-speed hardware architectures for authenticated encryption mode GCM. In: *ISCAS 2006*, IEEE, 2006.

[18] A. Kandi et al. Side-channel and fault resistant ASCON implementation: A detailed hardware evaluation. IACR Cryptology ePrint Archive, Paper 2024/984, 2024.

[19] aa55.dev. Evaluation of hashing algorithms Ascon, SHA256, SHA512 and BLAKE3 on an ARM Cortex M7 processor. Blog post, August 2025. https://aa55.dev/2025/08/24/Evaluation-of-hash-algorithm.html.

[20] The Keccak Team. eXtended Keccak Code Package. https://github.com/XKCP/XKCP, accessed May 2026.

[21] P. Kaplesh. Performance Evaluation of SHA-3(KECCAK) on ARM Cortex-A9 and Comparison with ARM 7TDMI and Cortex-M4. *International Journal of Computer Networks and Applications* 7(1), 2020.

[22] C. Bormann, P. Hoffman. Concise Binary Object Representation (CBOR). RFC 8949, IETF, 2020.

[23] T. Iwata, K. Ohashi, K. Minematsu. Breaking and Repairing GCM Security Proofs. In: *CRYPTO 2012*, LNCS 7417, pp. 31–49. Springer, 2012.

[24] E. Käsper, P. Schwabe. Faster and Timing-Attack Resistant AES-GCM. In: *CHES 2009*, LNCS 5747, pp. 1–17. Springer, 2009.

Appendix D Corrigendum: the v1 Round Function is Not a Permutation

D.0.1 Status of this appendix

This appendix is a corrigendum, not an enrichment. It records that the round function specified in Section 3 of this paper — hereafter *MERIDIAN-v1* — is **not a permutation**, and that the block-cipher and PRP security claims made for it are therefore false as stated. It then specifies a repaired round function, **MERIDIAN-v2**, and gives corrected security statements for it.

The defect was reported by J.-P. Aumasson [1], in a note whose results were found by an LLM (GPT-5.6 Sol) and manually verified by the author of that note. We have independently reproduced every number in that report; all of them are correct. We thank Aumasson for the disclosure.

Correction to v19 (this is the v20 revision). The v19 text of this appendix reported a *second*, independent break — that Admittance Mixing is a singular $\mathbb{F}_2$-linear map for 98.10% of round-key bit-planes. That claim is **withdrawn**. It was computed against a neighbour-gated layer that appears nowhere in this paper; the layer Section 3.2.3 actually specifies is a bijection for every key. Section D.2.2 now records the withdrawal and the verification. The conclusion of this appendix is unaffected: MERIDIAN-v1 is not a permutation. But it fails because of D1 alone, not because of two independent defects.

Placement. v19 confined this material to Appendix D, 184 printed pages behind the specification it corrects. v20 adds a *Section 3 Corrigendum Insert*, spliced immediately before printed page 14 and carrying both the defect and the repair to the point of use, and stamps a notice on printed pages 14–27. The insert is unnumbered and sits outside the folio sequence, as the Index Supplement does, so no page number in this document changes. This appendix remains the full treatment; the insert is the version an implementer will actually encounter.

We state plainly what this means for the rest of the paper. MERIDIAN-v1 is not a block cipher. Every result in the main body and in Appendices B and C that assumes the round function is a permutation — the PRP claim, the Davies–Meyer and sponge constructions, the AEAD security theorems, and the indifferentiability arguments — is void for v1. Section D.5 lists these claims explicitly. The purpose of this appendix is to correct the record, not to minimise it.

D.0.2 The defects

D.0.2.1 D1: Directional Substitution is not injective

Section 3 defines, for each state position (i, j) of the 4×4 byte state S,

$$c_{i,j} = S[i,\, j-1] \oplus S[i-1,\, j], \qquad r_{i,j} = (c_{i,j} \bmod 7) + 1, \tag{D.1}$$

$$\mathrm{DS}(S)[i,j] = \mathrm{ROL}_{r_{i,j}}\big(\pi(S[i,j])\big), \tag{D.2}$$

with π the AES S-box and indices taken modulo 4.

Let every row of A be `02 29 02 29` and every row of B be `02 46 02 46`. Since all rows are equal, $S[i-1,j] = S[i,j]$, so every context in A is $\texttt{02} \oplus \texttt{29} = \texttt{2b}$, giving $r = 2$, and every context in B is $\texttt{02} \oplus \texttt{46} = \texttt{44}$, giving $r = 6$. With $\pi(\texttt{02}) = \texttt{77}$, $\pi(\texttt{29}) = \texttt{a5}$, $\pi(\texttt{46}) = \texttt{5a}$:

$$\mathrm{ROL}_2(\texttt{77}) = \mathrm{ROL}_6(\texttt{77}) = \texttt{dd}, \qquad \mathrm{ROL}_2(\texttt{a5}) = \mathrm{ROL}_6(\texttt{5a}) = \texttt{96},$$

so $\mathrm{DS}(A) = \mathrm{DS}(B) = \texttt{dd96} \cdots \texttt{dd96}$ although $A \neq B$.

The mechanism is rotational self-similarity: `77` is invariant under ROL_4, and $\pi(\texttt{29}) = \mathrm{ROL}_4(\pi(\texttt{46}))$, while the two states' rotation amounts differ by exactly 4. Because DS is keyless,

the collision holds for every key: with $P_0 = A \oplus K$ and $P_1 = B \oplus K$ the whitening step returns the states to A and B, they coincide after the first DS layer, and every later operation is deterministic, so $E_K(P_0) = E_K(P_1)$ for all K and all round counts.

This is not an isolated pair. Enumerating the 65 536 states whose four rows all equal (a, b, a, b) gives only 43 472 distinct images: 33.7% of that class is lost. The underlying reason is that $\mathrm{ROL}_r(\pi(\cdot))$, with r ranging over the seven values (D.1) permits, is heavily many-to-one — 254 of the 256 byte values are reachable from more than one source byte, with 6.77 source bytes per output on average. A design that derives the rotation amount from the state and then applies it to that same state cannot be inverted, because recovering the rotation requires the neighbours, which are themselves unknown.

D.0.2.2 D2 (withdrawn in v20): Admittance Mixing is *not* singular

The v19 text of this subsection claimed a second, independent defect: that AM is a singular $\mathbb{F}_2$-linear map for 98.10% of round-key bit-planes, and that it therefore destroys injectivity on its own. **That claim is withdrawn.** It does not describe the layer this paper specifies. We set out the error in full, because a corrigendum that quietly drops a claim is no better than the one it corrects.

The computation was run against

$$\mathrm{AM}(S)[i,j] \;=\; S[i,j] \oplus \big(K[i,j] \wedge (S[i,j] \oplus S[i,j{+}1])\big) \oplus \big(K[i{+}1,j] \wedge (S[i,j] \oplus S[i{+}1,j])\big), \tag{D.3}$$

a neighbour-gated layer. For (D.3) the 98.10% figure is correct: the map is built only from AND and XOR, so it acts independently on each of the eight bit-planes as an $\mathbb{F}_2$-linear map on 16 bits, 64 288 of the 65 536 bit-plane keys give a singular matrix, and the plane `0x1234` has rank 12. All of that is reproducible. It is also irrelevant, because (D.3) is not MERIDIAN's AM layer and appears nowhere in the paper.

Section 3.2.3 specifies a *byte-local, self-gated* layer:

$$\mathrm{AM}_K(S)[i,j] \;=\; S[i,j] \;\oplus\; \big(K[i,j] \cdot g(S[i,j])\big), \qquad g(x) = (x \gg 3) \,|\, \texttt{0x01}, \tag{D.4}$$

with $\cdot$ byte-wise AND. Equation (D.4) couples no neighbours at all, so it is not a 16-bit linear map per plane and the rank argument does not apply to it. Each byte position evolves independently, so AM_K is a bijection on the state precisely when the byte map $x \mapsto x \oplus (k \wedge g(x))$ is a bijection, for each of the sixteen key bytes k. That is a 256-key, 256-input question, settled by direct enumeration.

Observation D.1. For all 256 values of k, the map $x \mapsto x \oplus (k \wedge g(x))$ with $g(x) = (x \gg 3) \,|\, \texttt{0x01}$ is a bijection of $\mathbb{F}_2^8$. None is singular. Hence AM_K as specified in Section 3.2.3 is a permutation of the 128-bit state for *every* key K, including the all-zero and all-ones keys.

Verified by `supplementary/verify_section3_layers.py`, which audits each Section 3 layer on its own terms. Note that `supplementary/verify_v1_break.py` still reproduces the 98.10% figure, because its `am_v1_matrix` builds (D.3); that function is retained for the record and its output is now labelled as not applying to MERIDIAN.

Aumasson's note [1] never claimed an AM defect. The withdrawn claim was ours, and it was wrong. What survives is D1, which is sufficient: DS alone is enough to make MERIDIAN-v1 fail to be a permutation.

D.0.2.3 D3: the one-round differential bound of Proposition 2 is false

Proposition 2 claims a one-round bound of $(2^{-6})^w \, 2^{-3w'}$, where w counts active bytes and w' counts bytes whose context changed. Take input difference `0x12` at byte $(0, 0)$ only, and request

output difference `0x55` at $(0, 0)$ and zero elsewhere. Then $w = 1$, $w' = 2$, so the claimed bound is 2^{-12}. Exact enumeration gives

$$p \;=\; \frac{804}{65536} \cdot \left(\frac{33808}{65536}\right)^2 \;\approx\; 2^{-8.2588}, \tag{D.5}$$

which exceeds the bound by a factor of 13.37.

The faulty step is the 2^{-3} penalty charged to every changed context. That penalty presumes a context difference reliably changes the rotation. It does not: $r = (c \bmod 7) + 1$ takes seven values over 256 contexts, so it is not uniform ($256 = 7 \cdot 36 + 4$), and $c \mapsto c \bmod 7$ is not compatible with XOR differences. For $\Delta c =$ `0x12`, exactly 128 of the 256 contexts leave $c \bmod 7$ fixed — probability $1/2$, not 2^{-3}.

D.0.2.4 D4: the specification and the shipped code are different ciphers

The reference implementation distributed as `supplementary/meridian_cipher.py` does not implement the specification:

- It operates on a 4×8 state (256-bit block), whereas Section 3.1 specifies 4×4 (128-bit block).
- Its DS layer is $\pi(S[i, j]) \oplus (K[i, j] \wedge \mathrm{ctx})$ with a four-neighbour context and no rotation at all — a different map from (D.2).
- Its `decrypt` method raises `NotImplementedError` and advises using the primitive in "permutation mode (hash, MAC) not block cipher mode", which contradicts the paper's title and its PRP claim.
- Its table named `AES_SBOX` is **not** the AES S-box. It diverges from FIPS-197 in 88 of 256 entries, starting at index `0xa7`, and has only 198 distinct outputs, so it is not even a bijection (`0x5e` occurs four times). Section 3.3 claims the AES S-box is reused verbatim.
- Its key schedule computes round constants with floating-point $\sin(\cdot)$, so results are not bit-reproducible across platforms.

Because no 128-bit test vectors were published, this divergence went undetected. Section D.6 supplies them.

D.0.3 MERIDIAN-v2: the repaired round function

D.0.3.1 Design principle: checkerboard phase separation

D1 has a structural cause: a layer computes each cell's update from neighbours that the same layer is simultaneously overwriting, so the information needed to invert it is destroyed as it is used. The repair is to split each state-dependent layer into two phases over the checkerboard classes

$$\mathcal{E} = \{(i, j) : i + j \equiv 0 \bmod 2\}, \qquad \mathcal{O} = \{(i, j) : i + j \equiv 1 \bmod 2\},$$

so that each phase reads only cells the other phase owns.

Lemma D.2. *On the 4×4 torus, the neighbours $(i, j \pm 1)$ and $(i \pm 1, j)$ of a cell (i, j) all lie in the opposite class.*

Proof. Each neighbour changes exactly one coordinate by ± 1, so it changes $i + j$ by ± 1 and flips its parity. Wrap-around preserves this because the side length 4 is even: $j - 1 \equiv 3 \pmod 4$ changes $i + j$ by 3, which is odd. (This is why the construction requires an even-sided state.) □

This is the block-cellular-automaton discipline used to make Margolus-style neighbourhood updates reversible. It preserves the "directional" character of the design — every cell is still substituted under a rotation derived from its neighbours — while making inversion mechanical.

D.0.3.2 DS′: repaired Directional Substitution

We also replace the rotation index. Let

$$r(c) \;=\; c \wedge 7 \qquad \text{(the low three bits of } c\text{)}, \tag{D.6}$$

which takes all eight values uniformly and satisfies $r(c \oplus \Delta) = r(c) \oplus r(\Delta)$, so a context difference changes the rotation by exactly $\Delta \wedge 7$. This makes the differential behaviour exactly computable rather than heuristic, and removes the flaw behind D3.

DS′.
Phase 1. For every $(i,j) \in \mathcal{E}$, with $c = S[i,j-1] \oplus S[i-1,j]$ read from S: $T[i,j] \leftarrow \mathrm{ROL}_{r(c)}(\pi(S[i,j]))$.
Phase 2. For every $(i,j) \in \mathcal{O}$, with $c = T[i,j-1] \oplus T[i-1,j]$ read from T: $T[i,j] \leftarrow \mathrm{ROL}_{r(c)}(\pi(S[i,j]))$.

By Lemma D.2 the contexts in Phase 1 lie in $\mathcal{O}$ and are untouched by Phase 1; the contexts in Phase 2 lie in $\mathcal{E}$ and were finalised by Phase 1.

Theorem D.3. DS′ *is a permutation of* $(\mathbb{F}_2^8)^{16}$.

Proof. We exhibit the inverse. Given $T = \mathrm{DS}'(S)$, the cells of $\mathcal{E}$ in T still hold their Phase-1 outputs, since Phase 2 writes only to $\mathcal{O}$. Hence for each $(i,j) \in \mathcal{O}$ the context $c = T[i,j-1] \oplus T[i-1,j]$ used by Phase 2 is directly readable from T, and $S[i,j] = \pi^{-1}(\mathrm{ROR}_{r(c)}(T[i,j]))$ recovers the original byte, using that π is a bijection and ROR_r inverts ROL_r. With all of $\mathcal{O}$ restored to its original values, the Phase-1 context $c = S[i,j-1] \oplus S[i-1,j]$ for each $(i,j) \in \mathcal{E}$ is likewise readable, and $S[i,j] = \pi^{-1}(\mathrm{ROR}_{r(c)}(T[i,j]))$ completes the inversion. The procedure is defined for every T and inverts DS′ on every input, so DS′ is a bijection. □

Applied to the collision of Section D.2.1, $\mathrm{DS}'(A) =$ `bb69bb69dd2ddd2dbb69bb69dd2ddd2d` and $\mathrm{DS}'(B) =$ `77697769dda5dda577697769dda5dda5`, which differ. Enumerating the entire 65 536-state class that collapsed under DS yields 65 536 distinct images under DS′.

D.0.3.3 AM′: repaired Admittance Mixing

By Observation D.1 the layer of Section 3.2.3 is already a bijection for every key, so this is a *strengthening*, not a repair. The v19 text presented it as a repair because it accepted the withdrawn D2. What AM′ buys is diffusion, not correctness: the self-gate $g(S[i,j])$ reads only the cell it is updating, so AM contributes nothing to inter-byte mixing, whereas AM′ gates on two neighbours. Evaluating that neighbour gate in phase keeps invertibility unconditional — each phase has the form $x \leftarrow x \oplus f(\text{other class})$, invertible for *every* key with no rank condition.

AM'_K.
Phase 1. For $(i,j) \in \mathcal{E}$, with $g = S[i,j+1] \oplus S[i+1,j]$: $T[i,j] \leftarrow S[i,j] \oplus (K[i,j] \wedge g)$.
Phase 2. For $(i,j) \in \mathcal{O}$, with $g = T[i,j+1] \oplus T[i+1,j]$: $T[i,j] \leftarrow S[i,j] \oplus (K[i,j] \wedge g)$.

Theorem D.4. *For every key* K, AM'_K *is a permutation of* $(\mathbb{F}_2^8)^{16}$.

Proof. By Lemma D.2 the gate g in Phase 1 depends only on cells of $\mathcal{O}$, which Phase 1 does not modify; so after Phase 1 the values of $\mathcal{O}$ in T equal those in S, and each Phase-1 gate is recomputable from T. Phase 2's gate depends only on cells of $\mathcal{E}$, finalised by Phase 1 and not modified by Phase 2, so it too is recomputable from T. Since $x \mapsto x \oplus (K[i,j] \wedge g)$ with g known is its own inverse, inverting Phase 2 and then Phase 1 recovers S for every K. □

Theorem D.4 holds for every key, as does Observation D.1 for the layer it replaces. Neither layer has an invertibility problem; the v19 assertion that Theorem D.4 rescued AM from a 1.9% invertible fraction is withdrawn along with D2.

D.0.3.4 MD needs no repair, but v2's MD is not §3.2.2's MD

Section 3.2.2 defines MD = Mix ∘ Par ∘ Mer, where Mer and Par are toroidal row/column rotations and

$$\textsc{Mix}(S)[i,j] \;=\; S[i,j] \oplus S[(i{+}1) \bmod 4,\, j] \oplus \mathrm{ROL}_3\big(S[(i{+}2) \bmod 4,\, j]\big)$$

is a genuine $\mathbb{F}_2$-linear mixing step on each column. It is bijective: Mer and Par are positional permutations, and Mix has full rank 32/32 over $\mathbb{F}_2$ on a 32-bit column (exhaustive; `verify_section3_layers.py`). MD as specified therefore needs no repair.

The v19 text of this subsection described MD as "a positional byte permutation $\pi_1 \circ \pi_2 \circ \pi_3$" and said it was retained verbatim. That describes the *reference implementation*, not Section 3.2.2: in `meridian_v2.py`, MD is a cyclic roll by 5 followed by $\pi_2 : (i,j) \mapsto ((i+3j) \bmod 4, (j+2i) \bmod 4)$ and $\pi_3 : (i,j) \mapsto ((i+j) \bmod 4, (2j-i) \bmod 4)$ — determinants $1 \cdot 1 - 3 \cdot 2 = -5 \equiv 3$ and 3 modulo 4, both units, so both bijective — and the Mix step is **dropped**.

We flag this rather than paper over it. Removing Mix removes the only intra-column linear diffusion in the layer, leaving MD as a bare relabelling of byte positions and pushing all mixing onto DS′ and AM′. Section D.7 reports that full avalanche is still reached by round 4, so the change is not obviously harmful, but it is a substantive change to the diffusion layer, and any MILP active-S-box count for v2 must model the layer the code actually implements.

D.0.3.5 Round constants

Section 3.2.4 specifies $c_r[i,j] = (r \cdot \texttt{0x9E} + 4i + j) \bmod 256$, which is integer arithmetic and already bit-reproducible; the floating-point $\sin(\cdot)$ constants criticised in Section D.2.4 are a property of the shipped `meridian_cipher.py`, not of the specification. (The v19 text of this subsection attributed them to the specification.) v2 nonetheless replaces the constants with $RC_r[i,j] = \pi\big((16r + 4i + j) \bmod 256\big)$, passing them through the S-box so that successive constants are not linearly related. Round constants are injected by XOR and do not affect invertibility either way.

D.0.3.6 The full cipher

MERIDIAN-v2 keeps the v1 outer structure: $S \leftarrow P \oplus K$, then twelve rounds of DS′, MD, AM'_K, $\oplus RC_r$, with direct key injection and no key schedule.

Theorem D.5. *For every key $K \in \mathbb{F}_2^{128}$, MERIDIAN-v2 is a permutation of $\mathbb{F}_2^{128}$, and decryption inverts encryption on all inputs.*

Proof. Each round is a composition of DS′ (Theorem D.3), MD (Section D.3.4), AM'_K (Theorem D.4) and an XOR by a constant, each a bijection; whitening is an XOR by K. A composition of bijections is a bijection. □

D.0.4 Corrected differential analysis

Write the DS′ cell map as $y = \mathrm{ROL}_{r(c)}(\pi(x))$. Because ROL is $\mathbb{F}_2$-linear, for a *fixed* rotation the output difference is $\mathrm{ROL}_r(\pi(x) \oplus \pi(x \oplus \delta))$, so the AES DDT bound 2^{-6} applies unchanged. The corrections concern cells whose rotation changes, by $k = \Delta c \wedge 7 \neq 0$. There the difference is governed by the linear map $z \mapsto z \oplus \mathrm{ROL}_k(z)$, whose fibres have size $2^{\gcd(k,8)}$.

The per-k values are 2^{-7} for $k \in \{1,3,5,7\}$, 2^{-6} for $k \in \{2,6\}$, and 2^{-4} for $k = 4$. The $k = 4$ row is precisely the ROL_4 self-similarity that produced the v1 collision; under DS′ it can no longer yield a collision, but it remains the worst differential case and must be charged for.

Table D.1: Exact maximum differential probabilities for the DS′ cell map, by exhaustive enumeration over all (x, c) pairs. The v1 proof assumed 2^{-3} for the middle row.

Cell type	Max. probability	Attained at
Active byte, rotation unchanged	$4/256 = 2^{-6}$	AES DDT maximum
Passive byte, rotation changed by k	$2^{\gcd(k,8)}/256$, max 2^{-4}	$k = 4$
Active byte, rotation also changed	$8/256 = 2^{-5}$	$\delta =$ `0x21`, $k = 4$

Proposition D.6 (corrected one-round bound)**.** *Consider a one-round difference pattern with w active bytes, of which w_c also see a rotation change, and w' passive bytes whose context difference has nonzero low three bits. Then for any fixed output difference,*

$$\Pr \;\le\; 2^{-6(w-w_c)} \cdot 2^{-5w_c} \cdot 2^{-4w'} \;\le\; 2^{-4(w+w')}. \tag{D.7}$$

Proof sketch. Condition on the bytes of $\mathcal{O}$. Within Phase 1 every context is then fixed, and the outputs of distinct cells of $\mathcal{E}$ depend on distinct, independent, uniformly distributed state bytes; so conditionally the per-cell events are independent and Table D.1 bounds each. Averaging over the conditioning preserves the bound. Repeat for Phase 2, conditioning on the now-fixed outputs of $\mathcal{E}$. The stated product follows; the second inequality uses $2^{-5}, 2^{-6} \le 2^{-4}$. □

The counterexample of Section D.2.3, re-evaluated under DS′, has actual probability $2^{-18.42}$ against a corrected bound of 2^{-14}, so it is comfortably respected.

Revised headline bound. The paper's headline 2^{-288} came from $A_{12} \ge 48$ active S-boxes at 2^{-6} each. Under the corrected analysis an active byte whose context also changes contributes 2^{-5}, not 2^{-6}. The worst-case twelve-round bound is therefore

$$\Pr \;\le\; \left(2^{-5}\right)^{48} \;=\; 2^{-240}, \tag{D.8}$$

not 2^{-288}. This is still far below 2^{-128}, so the differential security claim survives, but the constant in the main body must be read as 2^{-240}. We stress that $A_{12} \ge 48$ was established by MILP against the *v1* round function; the model must be re-run against DS′ and AM′ before the revised figure can be regarded as verified. We do not claim to have done so here.

D.0.5 Claims withdrawn or revised

D.0.6 Reference implementation and test vectors

`supplementary/meridian_v2.py` is a dependency-free reference implementation of MERIDIAN-v2 with both encryption and decryption. Its S-box is generated from the FIPS-197 definition rather than transcribed, so it cannot reproduce the table error of D4. `supplementary/verify_v1_break.py` independently reproduces D1, D3 and D4 (its FINDING 3 builds equation (D.3) and is retained only for the record; it does not apply to MERIDIAN). `supplementary/verify_section3_layers.py` audits every layer of Section 3 on its own terms and establishes Observation D.1.

The paper previously published no 128-bit test vectors. They are given here.

D.0.7 Empirical validation

The reference implementation passes: inversion of DS′, MD and AM′ on 20 000 random states each; full encrypt/decrypt round-trip on 26 000 (K, P) pairs, including the all-zero and all-ones keys that make v1's AM singular; and injectivity of DS′ over all 65 536 states of the class that collapsed in v1.

Table D.2: Disposition of affected claims.

Claim	Status	Note
v1 is a block cipher / PRP	**Withdrawn**	Not injective (D1).
D2: AM singular for 98.10% of bit-planes	**Withdrawn (v20)**	Computed against a layer not in this paper; §3.2.3's AM is bijective for every key (Obs. D.1).
$A_{12} \geq 48$ (MILP)	**Unverified**	Computed for v1; must be re-run for v2, whose reference MD also drops Mix.
Differential bound 2^{-288}	**Revised**	2^{-240} under Prop. D.6.
Proposition 2 one-round bound	**Withdrawn**	Replaced by Prop. D.6.
S-box is AES verbatim (§3.3)	**Holds for spec**	Shipped table was corrupt; corrected in v2 code.
Appendix B AEAD theorems	**Void for v1**	Assume a permutation; re-derivable for v2.
Appendix C hash / indifferentiability	**Void for v1**	Sponge security needs a permutation.
Side-channel analysis (§6)	**Unaffected**	Counts profilable points, not injectivity.
MD layer	**Holds**	Bijective; unchanged.

Table D.3: MERIDIAN-v2 test vectors, 12 rounds, 128-bit key and block (hex).

Key / Plaintext	Ciphertext
`00000000000000000000000000000000` `00000000000000000000000000000000`	`8b909e24d761b869f3a79cade4588f0c`
`000102030405060708090a0b0c0d0e0f` `000102030405060708090a0b0c0d0e0f`	`81e5998865411dd8ee52c7af0a5550a0`
`ffffffffffffffffffffffffffffffff` `00000000000000000000000000000000`	`5ee3e7958fc75ae2fbcf036eb434e313`

Avalanche, measured as the mean number of ciphertext bits changed by a single plaintext bit flip (ideal 64.00, ideal standard deviation 5.66):

Full avalanche is reached by round 4 and the twelve-round distribution matches the ideal binomial. Intra-layer diffusion is not degraded by the phase split: a one-byte input difference changes 2.72 output bytes on average under DS′, against 2.70 under v1's DS.

D.0.8 Honest disclosure

1. MERIDIAN-v1 as published is broken and should not be used or cited as a block cipher. The ePrint record should be updated accordingly. The cause is the DS layer of Section 3.2.1 alone; Sections 3.2.2 and 3.2.3 are bijective as specified.

2. MERIDIAN-v2 is proved here to be a permutation for every key. That is a correctness property, not a security claim. It says nothing about resistance to differential, linear, algebraic, integral, or related-key attacks against the new round function.

3. The MILP active-S-box bound has not been re-run for v2. Until it is, the 2^{-240} figure rests on an unverified premise.

4. The key schedule is unchanged and remains a concern: with direct key injection and no schedule, all twelve rounds use the same K, which invites slide and related-key analysis. This appendix does not address it.

5. The AEAD and hash constructions in Appendices B and C need to be re-derived over the v2 permutation before their theorems can be restated.

Table D.4: MERIDIAN-v2 avalanche, 3 000 trials per row.

Rounds	Mean bits changed	Std. dev.
1	21.38	12.00
2	55.49	11.17
3	63.79	5.91
4	64.00	5.64
12	64.12	5.69

6. The v19 edition of this appendix asserted a second, independent structural break that does not exist in the cipher as specified (Section D.2.2). That error was ours, it survived internal review, and it was settled only by exhaustive enumeration of a 256-element function class — a check that costs milliseconds. Both the original design fault and the mistaken diagnosis of it point the same way: this construction has not had enough external scrutiny.

7. MERIDIAN-v2 has received no third-party cryptanalysis. It should not be deployed on the strength of this appendix.

References

[1] J.-P. Aumasson. “Blood MERIDIAN: a blockcipher that is not a blockcipher.” Cryptology ePrint Archive, Paper 2026/1818, 2026. https://eprint.iacr.org/2026/1818 (short URL: https://ia.cr/2026/1818).

[2] C. McCarthy. *Blood Meridian, or the Evening Redness in the West*. Random House, New York, 1985. (Source of the title of [1]; cited here only to credit the allusion.)

[3] B. Palaniswamy, P. Palmieri, A. K. Das. MERIDIAN: A Toroid-Inspired Permutation Block Cipher for Constrained Environments. Cryptology ePrint Archive, Report 2026/856, 2026. https://eprint.iacr.org/2026/856

[4] National Institute of Standards and Technology. Advanced Encryption Standard (AES). FIPS PUB 197, 2001.

[5] T. Toffoli, N. Margolus. *Cellular Automata Machines: A New Environment for Modeling*. MIT Press, 1987. (Reversible block-partitioned neighbourhood updates.)

[6] J. Kari. Reversible cellular automata. In: *Developments in Language Theory*, LNCS 3572, pp. 57–68. Springer, 2005.