

Article

Dipper: A Lightweight Hybrid SPN–ARX Block Cipher

Ali Huseynli , Yadigar Imamverdiyev and Jalal Alizadeh

Department of Cybersecurity, Faculty of Information Technologies and Telecommunications, Azerbaijan Technical University, Baku AZ1073, Azerbaijan; yadigar.imamverdiyev@aztu.edu.az (Y.I.); jalal.alizadeh@aztu.edu.az (J.A.)

* Correspondence: ali.huseynli@student.aztu.edu.az

Abstract

We present Dipper, a lightweight 64-bit block cipher with 96-bit and 128-bit key variants, built on a 28-round hybrid SPN–ARX structure. Each round applies a full-state key addition, sixteen parallel 4-bit GIFT S-boxes, four word-wise rotations, two 16-bit modular additions over half of the state, and the GIFT-64 bit permutation, combining the compact substitution layer of GIFT-style designs with the diffusion efficiency of ARX operations. We evaluate Dipper from both hardware and cryptanalytic perspectives under a single, fully open-source methodology. Round-based Verilog implementations were synthesized alongside PRESENT, GIFT, and SIMON variants using an identical Yosys + ABC + Nangate45 flow. Under this flow, Dipper-64/96 and Dipper-64/128 require 2498 and 2824 gate equivalents (GE), respectively, both falling between GIFT-64-128 (2191 GE) and PRESENT-128 (2963 GE); notably, Dipper-64/128 is more compact than PRESENT-128 at the same key size, despite incorporating an additional ARX diffusion layer. A broader comparison re-implements eleven established lightweight ciphers under the same flow, and post-place-and-route FPGA results on Lattice ECP5, measured software timings, and Cortex-M memory footprints support deployment across RFID, sensor-node, and edge-gateway scenarios. For differential resistance, we develop a mixed-integer linear programming (MILP) model that couples the exact GIFT differential distribution table with a Lipmaa–Moriai encoding of modular addition. Predicted and empirical differential probabilities agree tightly for reduced-round variants, while five-round trails reveal differential clustering. The security evaluation further includes proven-optimal linear trail bounds up to ten rounds, an exhaustive impossible-differential search bounding the longest distinguisher at five rounds, and experimental integral distinguishers of at most five rounds, leaving the 28-round cipher a margin close to $3\times$ against the longest identified distinguisher. All RTL, synthesis scripts, reference implementations, and MILP models are released for full reproducibility.

Keywords: lightweight cryptography; block cipher; SPN; ARX; IoT security



Academic Editor: Josef Pieprzyk

Received: 30 May 2026

Revised: 12 July 2026

Accepted: 16 July 2026

Published: 21 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Due to the increased use of resource-constrained platforms like radio-frequency identification (RFID) tags, wireless sensor nodes and edge nodes of the Internet of Things (IoT), there is growing demand for cryptographic primitives designed for small silicon area, limited energy and low latency. Although the Advanced Encryption Standard [1,2] is secure, its hardware requirements often exceed the budget limitations imposed by small devices [3]. Thus, research in lightweight cryptography has focused on designing block ciphers with minimal hardware requirements while maintaining a security margin for short-to-medium-term protection of low-value information, such as in wireless sensor networks or smart cards [4].

The design of lightweight block ciphers entails a variety of trade-offs along different dimensions. For example, to minimize the amount of hardware gates, one typically needs smaller block sizes, fewer key bits, simpler nonlinear layers and simpler key schedules. Unfortunately, all these simplifications inherently reduce the diffusion capacity per round, forcing a higher number of rounds in the cipher to compensate, which in turn reduces the processing speed and energy per encrypted block. Better diffusion, measured in terms of metrics such as the branch number or the minimum number of active S-boxes, is crucial for resistance against differential and linear cryptanalysis and thus is an essential objective of the design of lightweight block ciphers with low gate cost.

Two design methodologies are prevalent within the lightweight literature. First is the permutation-substitution network (SPN), which is typified by PRESENT [5] and its evolved form, GIFT [6]. These networks combine small 4-bit substitution boxes (S-boxes) with a linear bit-level permutation, achieving a minuscule static area, as the permutation is achieved through the placement of wires, requiring no additional gating logic. The second design methodology is the addition-rotation-XOR (ARX) family of algorithms, illustrated by SIMON and SPECK [7], which dispense with S-boxes entirely and derive their nonlinearity from modular addition, while rotations on a word level provide diffusion. The ARX family is especially compact in software implementations and competes favorably in hardware performance, though its differential performance hinges critically on the idiosyncrasies of modular addition, an area that has undergone thorough analysis [8]. Hybrids, incorporating elements of both methodologies, have been conceived. PRINCE [9] aims at low latency for pervasive computing applications, whilst SKINNY [10] and a variety of algorithms proposed for the NIST Lightweight Cryptography Project [4] continue to use a conventional SPN structure but expand on its design with added features to increase diffusion or related-tweak security.

The motivation for combining SPN and ARX components in a single round is twofold. The compact 4-bit S-boxes used in PRESENT and GIFT provide a small and well-studied source of nonlinearity, with differential and linear distribution tables that admit explicit bounds on per-round transition probability. Modular addition, by contrast, contributes value-dependent diffusion through the carry chain that propagates structurally across word boundaries, a property not present in pure bit-permutation linear layers. A hybrid round can therefore, in principle, achieve diffusion within fewer rounds than a pure SPN at the cost of a moderate increase in round complexity, while remaining smaller than a fully ARX round that must compensate for the absence of S-boxes by additional rounds. The trade-off is not automatically favorable; whether a particular hybrid construction improves the area–security frontier requires quantitative evaluation under a controlled methodology.

We present Dipper, a lightweight block cipher in this hybrid family. Dipper operates on a 64-bit block with either a 96-bit or a 128-bit key over 28 rounds. Each round applies, in order, a full-state key addition, sixteen parallel 4-bit GIFT S-boxes, four 16-bit word rotations, two 16-bit modular additions arranged as $A_2 = A_1 + B_1$ and $C_2 = C_1 + D_1$, and the GIFT-64 bit permutation [6]. The S-box and permutation are inherited from the GIFT specification and provide a compact, well-analyzed nonlinear and linear backbone; the word rotations and modular additions form the ARX layer. The key schedule employs a non-linear update with three or four S-boxes per round and a 5-bit linear-feedback shift register for round-constant generation.

We evaluate Dipper using a uniform methodology designed to enable direct comparison with established lightweight ciphers. The hardware component consists of round-based Verilog implementations of Dipper and of PRESENT-80, PRESENT-128, GIFT-64-128, GIFT-128-128, SIMON-64-128, and SIMON-128-128. All implementations follow the same architectural template—single-cycle round, on-the-fly key schedule, no reset on data and key

registers—and are synthesized under an identical open-source flow consisting of Yosys [11], ABC, and the FreePDK45 (Nangate45) standard cell library at the typical corner. The use of a single flow with fixed coding conventions ensures that observed differences in gate equivalents reflect algorithmic and architectural differences rather than differences in tool, library, or design style. Numbers obtained in this way are not directly comparable to figures published with the original cipher specifications, which use different libraries and synthesis flows; the methodology is intended to produce internally consistent comparisons rather than to reproduce historical results.

The cryptanalytic component of the evaluation is a mixed-integer linear programming (MILP) model for differential trail search [12,13]. The S-box layer is encoded using the exact GIFT differential distribution table with per-transition weight $-\log_2(\text{DDT}[\Delta_{\text{in}}][\Delta_{\text{out}}]/16)$. The modular additions are encoded using the Lipmaa–Moriai theorem on the differential probability of addition [8], in the bit-level MILP form developed in subsequent work for ARX cipher analysis [14,15]. The objective is the total trail weight summed across rounds, and a solution of weight W corresponds to an upper bound of 2^{-W} on the probability of the best single differential characteristic under the standard Markov-cipher assumption. We complement the MILP search with empirical sampling against the Python reference implementation of Dipper, both with independent random round keys and with the real Dipper-64/96 and Dipper-64/128 key schedules.

The contributions of this paper are as follows.

- We specify Dipper, a 64-bit hybrid SPN–ARX block cipher with 96-bit and 128-bit key variants and a 28-round structure that combines the GIFT S-box and bit permutation with two 16-bit modular additions per round.
- We provide a round-based Verilog implementation of Dipper and synthesize it together with PRESENT, GIFT, and SIMON variants under an identical Yosys–ABC–Nangate45 flow. The resulting gate-equivalent figures place Dipper-64/128 between GIFT-64-128 and PRESENT-128 in static area and below both PRESENT-128 and SIMON-64-128 in the area–latency product.
- We construct an MILP model that combines the exact GIFT DDT with a Lipmaa–Moriai encoding of modular addition, and use it to search for the lowest-weight differential trail of Dipper for up to seven rounds. We empirically validate the resulting trails against the real cipher under both random round keys and the real key schedules.
- We characterize an iterative two-round differential pattern in Dipper that produces measurable differential clustering at intermediate round counts, distinguish trail-level from differential-level bounds, and discuss the implications for future-work bound tightening.
- We release the complete RTL, synthesis scripts, Python reference implementations, MILP source code, and validation scripts to enable independent reproduction of all reported figures.
- We report measured software performance—encryption and decryption timings on x86-64 and exact code and memory footprints on Cortex-M targets—and discuss deployment across RFID, sensor-node, and edge-gateway scenarios.
- We report post-place-and-route FPGA results on Lattice ECP5 for all twelve round-based cores under the same open-source flow, together with a complete configuration bitstream for a serially interfaced Dipper top level.
- We extend the cryptanalytic evaluation with three further families: a CP-SAT linear trail search built on a validated Wallén-automaton encoding of modular addition, with proven-optimal single-trail bounds for up to ten rounds and empirical hull validation; an exact impossible-differential search establishing that no truncated nibble-to-nibble impossible differential exists beyond four rounds and, by an exhaustive single-bit

search, none beyond five rounds; and an experimental integral search bounding 16-bit saturation distinguishers at five rounds.

The remainder of the paper is organized as follows. Section 2 specifies Dipper. Section 3 presents the design rationale and positions Dipper against existing ARX and hybrid constructions. Section 4 presents the hardware implementation, the comparative synthesis results, and the FPGA evaluation. Section 5 reports software performance and IoT deployment considerations. Section 6 reports the empirical statistical evaluation including avalanche, SAC, BIC, and NIST SP 800-22 tests. Section 7 describes the MILP model and reports the differential trail bounds. Section 8 discusses the iterative pattern clustering analysis. Section 9 extends the analysis to linear, impossible-differential, and integral distinguishers. Section 10 concludes and identifies directions for future work.

2. Specification of Dipper

This section specifies the Dipper block cipher. Section 2.1 fixes notation and the state layout. Section 2.2 defines the round function, Section 2.3 the round-constant generator, and Section 2.4 the two key schedules. Section 2.5 provides test vectors.

2.1. Notation and State Layout

Dipper operates on a 64-bit state S , viewed as the concatenation of four 16-bit words:

$$S = A \parallel B \parallel C \parallel D, \quad (1)$$

with $A = S[63:48]$, $B = S[47:32]$, $C = S[31:16]$ and $D = S[15:0]$. Within each word, bit 0 is the least significant. For nibble-oriented operations S is also viewed as sixteen 4-bit nibbles $n_0, n_1, \dots, n_{15}$, where $n_j = S[4j+3 : 4j]$.

Throughout the paper, $\oplus$ denotes bitwise XOR, $\boxplus$ denotes addition modulo 2^{16} , $x \lll s$ denotes the left circular rotation of a 16-bit word x by s positions, and $x \ggg s$ the corresponding right rotation. The plaintext is denoted P , the ciphertext C , the master key K , and the 64-bit round subkey applied at round $r \in \{1, \dots, 28\}$ is RK_r .

2.2. Round Function

One round of Dipper is the map $R : \{0, 1\}^{64} \times \{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$ defined by the sequential application of five layers.

- (i) AddRoundKey. $S \leftarrow S \oplus RK_r$.
- (ii) SubCells. Each of the sixteen nibbles of S is replaced by its image under the GIFT 4-bit S-box $\mathcal{S}$ [6], given in Table 1.

Table 1. The GIFT 4-bit S-box $\mathcal{S}$, in hexadecimal.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	1	a	4	c	6	f	3	9	2	d	b	7	5	0	8	e

- (iii) WordRotations. The four 16-bit words are rotated left by distinct amounts,

$$\begin{aligned} A_1 &= A \lll 1, & B_1 &= B \lll 4, \\ C_1 &= C \lll 7, & D_1 &= D \lll 11. \end{aligned} \quad (2)$$

- (iv) Modular Addition Layer. Two modular additions update half of the state, while the other half passes through unchanged:

$$A_2 = A_1 \boxplus B_1, \quad B_2 = B_1, \quad C_2 = C_1 \boxplus D_1, \quad D_2 = D_1. \quad (3)$$

The state after this layer is $S' = A_2 \parallel B_2 \parallel C_2 \parallel D_2$.

- (v) Bit Permutation. The GIFT-64 bit permutation P is applied to S' , $S^{\text{out}}[P(i)] = S'[i]$ for $i \in \{0, \dots, 63\}$, with

$$P(i) = 4 \left\lfloor \frac{i}{16} \right\rfloor + 16 \left(\left(3 \left\lfloor \frac{i \bmod 16}{4} \right\rfloor + (i \bmod 4) \right) \bmod 4 \right) + (i \bmod 4). \quad (4)$$

A schematic of the round function is shown in Figure 1. Encryption iterates R for $T = 28$ rounds:

$$S^{(0)} = P, \quad S^{(r)} = R(S^{(r-1)}, RK_r), \quad C = S^{(28)}. \quad (5)$$

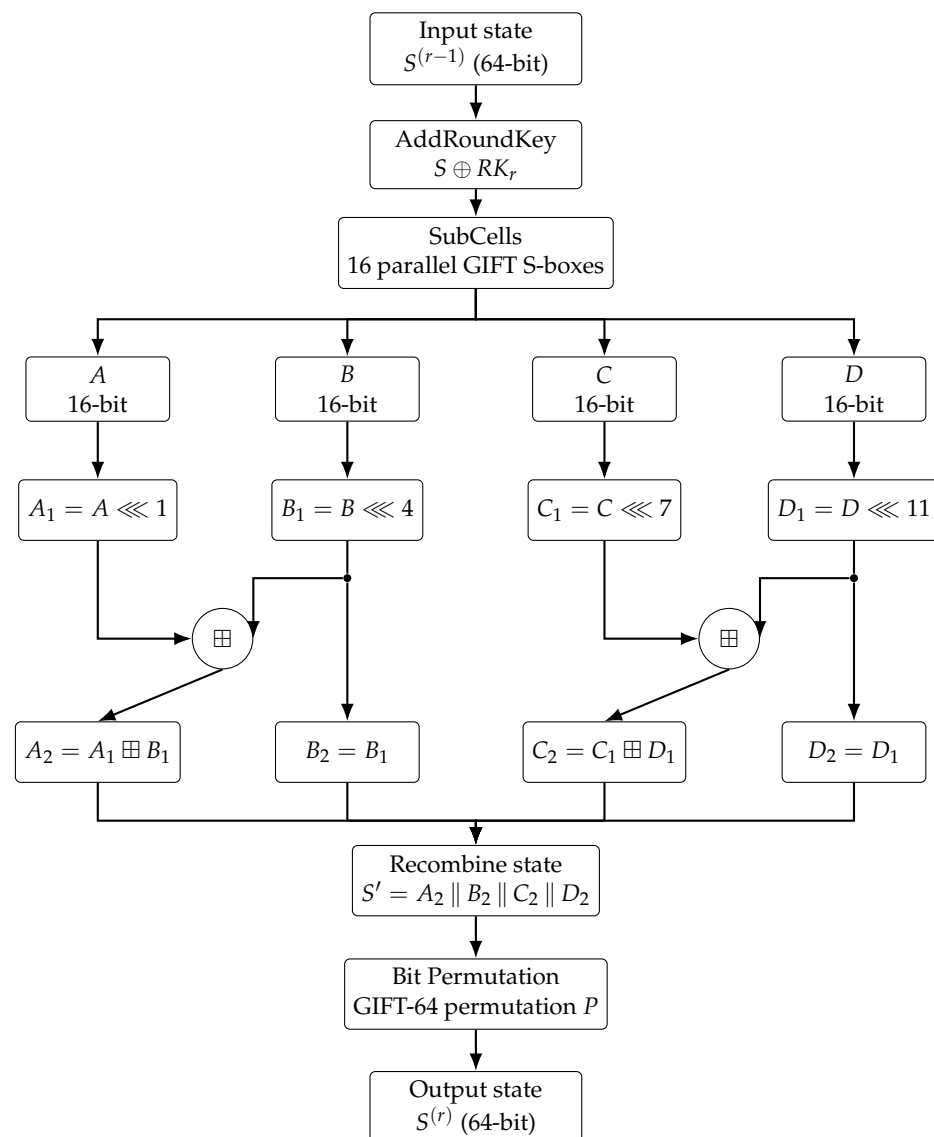


Figure 1. Detailed structure of one round of Dipper. After AddRoundKey and the parallel SubCells layer, the 64-bit state is viewed as four 16-bit words A , B , C , and D . Each word is rotated by a fixed amount. The half-state ARX layer updates only two branches through modular addition, $A_2 = A_1 \oplus B_1$ and $C_2 = C_1 \oplus D_1$, while B and D pass through unchanged. The four words are then recombined and passed through the GIFT-64 bit permutation.

2.3. Round Constants

Round constants are produced by a 5-bit LFSR with feedback polynomial $x^5 + x^2 + 1$. Denoting by $L^{(r)}$ the LFSR state at round r , with $L^{(1)} = 0x01$, the update is

$$L^{(r+1)} = ((L^{(r)} \ll 1) \mid b) \bmod 2^5, \quad b = L^{(r)}[4] \oplus L^{(r)}[1]. \quad (6)$$

The LFSR has period 31, so the values $L^{(1)}, \dots, L^{(28)}$ are pairwise distinct. We set $RC_r = L^{(r)}$; the round constants are injected into the key state during the round- r key update (Section 2.4). The full sequence is listed in Table 2.

Table 2. The 28 round constants $RC_1, \dots, RC_{28}$ in hexadecimal, generated by the 5-bit LFSR with $L^{(1)} = 0x01$.

Round	1	2	3	4	5	6	7	8
RC_r	01	02	05	0A	15	0B	17	0E
Round	9	10	11	12	13	14	15	16
RC_r	1D	1B	16	0C	18	11	03	07
Round	17	18	19	20	21	22	23	24
RC_r	0F	1F	1E	1C	19	13	06	0D
Round	25	26	27	28				
RC_r	1A	14	09	12				

2.4. Key Schedule

Dipper has two key-size variants, Dipper-64/96 and Dipper-64/128, with distinct key-state sizes, subkey-extraction windows, and update rules. At the beginning of round r the round subkey is the 64-bit window

$$RK_r = \begin{cases} K^{(r)}[63:0] & \text{for Dipper-64/96,} \\ K^{(r)}[127:64] & \text{for Dipper-64/128,} \end{cases} \quad (7)$$

after which the key state is updated as defined below.

2.4.1. Dipper-64/96

The 96-bit key state is partitioned into six 16-bit words, $K = k_5 \parallel k_4 \parallel k_3 \parallel k_2 \parallel k_1 \parallel k_0$, with k_5 the most significant. The round update consists of (i) a word-level permutation with rotations on the two least-significant words, (ii) four parallel S-box applications, and (iii) round-constant injection:

$$(k'_5, k'_4, k'_3, k'_2, k'_1, k'_0) = (k_3, k_2, k_1 \ggg 2, k_0 \ggg 12, k_5, k_4), \quad (8)$$

$$k''_w[15:12] = \mathcal{S}(k'_w[15:12]), \quad w \in \{1, 2, 4, 5\}, \quad (9)$$

$$K^{(r+1)}[52:48] = k''_3[4:0] \oplus RC_r. \quad (10)$$

All bit positions not affected by (9) or (10) are copied from (8) unchanged.

2.4.2. Dipper-64/128

The 128-bit key state is partitioned into eight 16-bit words,

$$K = k_7 \parallel k_6 \parallel k_5 \parallel k_4 \parallel k_3 \parallel k_2 \parallel k_1 \parallel k_0. \quad (11)$$

The update is

$$(k'_7, k'_6, k'_5, k'_4, k'_3, k'_2, k'_1, k'_0) = (k_3, k_2, k_7, k_6, k_5, k_4, k_1 \ggg 2, k_0 \ggg 12), \quad (12)$$

$$k''_w[15:12] = S(k'_w[15:12]), \quad w \in \{1, 3, 7\}, \quad (13)$$

$$K^{(r+1)}[68:64] = k''_4[4:0] \oplus RC_r. \quad (14)$$

As in Dipper-64/96, all bit positions not affected by (13) or (14) are copied unchanged.

The two schedules differ in the key-state size, the word-level permutation in step (i), and the number of S-boxes applied in step (ii) (four for Dipper-64/96, three for Dipper-64/128). The subkey-extraction windows also differ, as stated above. These choices keep the per-round key-schedule nonlinearity comparable between the variants relative to their respective key-state sizes.

2.5. Test Vectors

Test vectors for both variants are listed in Table 3. They were produced by the Python reference implementation accompanying this paper and verified bit-exactly against the Verilog implementation used in Section 4.

Table 3. Dipper test vectors. All values are in hexadecimal, most-significant byte first.

Variant	Plaintext <i>P</i>	Master Key <i>K</i>	Ciphertext <i>C</i>
Dipper-64/96	0123456789ABCDEF	00112233445566778899AABB	5F189202D3152B8F
Dipper-64/96	0000000000000000	000000000000000000000000	D873355EC63B1B4B
Dipper-64/128	0123456789ABCDEF	000102030405060708090A0B0C0D0E0F	4B46284387969060
Dipper-64/128	FFFFFFFFFFFFFFFF	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	6ABB4518063706B0

3. Design Rationale

The defining principle of Dipper is a strict separation of the two cryptographic roles within a single lightweight round, combined with a deliberately partial use of modular arithmetic. Nonlinearity is provided entirely by a conventional bijective 4-bit S-box (the GIFT S-box), in the manner of a substitution–permutation network; the add-rotate operations are used only to inject carry-based diffusion, and are applied to just half of the state. We refer to this as a half-state ARX layer: of the four 16-bit words, two are combined by modular addition while the other two pass through unchanged and serve as the addends. This is in contrast to constructions in which the modular operations themselves carry the nonlinearity, or in which the entire state is processed by arithmetic every round.

This principle is what separates Dipper from existing ARX and hybrid designs, which fall into three groups.

Pure-ARX designs. Ciphers such as SPECK and, in particular, SPARX [16] contain no bijective S-box at all: their nonlinearity is produced by the addition operation itself, and SPARX formalizes this through the long-trail strategy, iterating large 32-bit ARX-based “S-boxes” with sparse linear layers. Dipper takes the opposite position: the nonlinearity lives in a small, well-studied lookup S-box, and the ARX layer contributes no nonlinearity of its own but only diffusion. Confusion and diffusion are thus assigned to different components, whereas in SPARX both originate in the ARX box.

S-box plus full-width modular arithmetic. The Belarusian standard Bel-T [17] mixes 8-bit S-boxes with 32-bit modular addition and subtraction within a Lai–Massey-style structure. There the arithmetic acts on the full word width and is central to the round, which suits 32-bit software but is comparatively heavy in hardware. Dipper instead uses narrow 16-bit additions on only half the state, keeping the arithmetic cost to two adders (Section 5.5) and preserving a substitution–permutation backbone.

S-box plus linear diffusion. Feistel designs such as SM4 [18] combine an S-box with a purely linear diffusion function (rotations and XOR) and use no modular addition; they

therefore gain no carry-based mixing between words. Dipper keeps an S-box for confusion but replaces part of the linear diffusion with modular addition, so that a single-bit change propagates through carry chains as well as through the bit permutation.

The half-state restriction is the second, quantitative aspect of the rationale. A full ARX layer (four additions) is not adopted because it is unnecessary: as the ablation of Section 5.5 shows, two additions already advance full diffusion to round 6 (versus round 8 for a pure-SPN round) and substantially raise the per-round trail weight, while a fourth addition would double the adder area for a diffusion gain that the 28-round security margin renders superfluous. The two-addition, half-state round is therefore the point at which the carry-based diffusion benefit is obtained at minimal cost. The security consequences of this choice—fast diffusion and short distinguishers across the differential, linear, impossible-differential, and integral analyses of Sections 7, 9.1 and 9—are exactly what the separation-of-roles principle is intended to deliver: the S-box supplies bounded nonlinearity per active cell, and the half-state ARX spreads that activity across the state faster than a bit-permutation alone.

4. Hardware Implementation and Synthesis

This section reports the hardware implementation of Dipper and a comparative synthesis study against three established lightweight block cipher families. Section 4.1 describes the RTL implementation and the conventions shared by all eight designs. Section 4.2 specifies the open-source synthesis flow. Section 4.3 reports the gate-equivalent figures and the area–latency product. Section 4.4 discusses the limitations of the methodology.

4.1. RTL Implementation

Dipper is implemented in synthesizable Verilog-2005 as a round-based iterative core. The core consists of three modules: a combinational round function, a combinational key-schedule update, and a top-level controller that registers the state and the key, sequences the rounds, and exposes a simple handshake interface (*start*, *busy*, *done*). One round of encryption is computed per clock cycle; no pipelining or unrolling is used, in order to expose the round complexity directly in the static area.

To make the comparison with PRESENT, GIFT, and SIMON meaningful, we re-implement all reference ciphers in Verilog under the same set of conventions. Specifically, every implementation uses

- a round-based iterative architecture with one round per cycle;
- an on-the-fly key schedule (no expanded round-key memory);
- state and key registers without an asynchronous or synchronous reset on the data path;
- a small synchronous controller with a counter for the round index and a single-bit busy flag;
- no clock-gating, no scan chains, and no power-saving structures;
- identical I/O conventions and testbench infrastructure.

This template reflects published lightweight architectures of PRESENT, GIFT, and SIMON [5–7], restated under one coding style. The intention is not to reproduce the absolute gate counts reported in the original specifications, which were obtained on different process libraries with different toolchains, but to enable an internally consistent comparison among the reference re-implementations and the two Dipper variants.

Each implementation is functionally verified by a Verilog testbench that exercises published test vectors (the ones in Table 3 for Dipper, and the official test vectors for PRESENT-80, PRESENT-128, GIFT-64-128, GIFT-128-128, SIMON-64-128, and SIMON-128-128). All

testbenches are simulated with Icarus Verilog [19] and report match against the expected ciphertext bit-exactly.

4.2. Synthesis Flow

All designs are synthesized under a single open-source flow: Yosys 0.33 [11] for RTL elaboration, generic optimization, and DFF-library mapping; Berkeley ABC [20] for combinational logic synthesis and technology mapping; and the FreePDK45 Nangate Open Cell Library at the typical corner [21] for cell area data. The smallest cell in the library, NAND2_X1, has area $0.798 \mu\text{m}^2$; we report gate equivalents (GEs) computed as

$$\text{GE} = \left\lceil \frac{A_{\text{total}}}{0.798 \mu\text{m}^2} \right\rceil, \quad (15)$$

where A_{total} is the total cell area reported by Yosys after the final mapping pass.

The ABC technology-mapping script is identical for all eight designs and optimizes for area without timing constraints. No floor-planning, placement, routing, or post-layout extraction is performed; reported figures are pre-layout cell-area estimates only. This choice is deliberate: the goal of the study is to compare gate-level static area under a uniform flow, not to predict post-fabrication chip area.

The Nangate45 library available to Yosys does not include the enable-D flip-flop primitive (DFFE) used by some commercial 45 nm libraries; the synthesized designs therefore implement enable functionality through external multiplexers, which adds a small fixed overhead to all eight designs. We discuss the implication for absolute comparison against published figures in Section 4.4.

4.3. Results

Table 4 reports the eight measurements. All ciphers were synthesized from the RTL described above, with passing testbenches, on the same machine and with the same Yosys binary. Beyond gate equivalents, the table includes the number of clock cycles needed to encrypt one block, and the area–latency product $\text{GE} \times \text{cycles}$, which is the standard single-figure metric for round-based implementations in lightweight cryptography [6].

Table 4. Hardware results under the unified Yosys + ABC + Nangate45 flow. Dipper-64/96 and Dipper-64/128 are highlighted in bold. Area is the total cell area reported by Yosys after technology mapping; GE is computed as $\lceil A/0.798 \rceil$; cycles count the number of clock periods to encrypt one block including the input register and output cycle.

Cipher	Block	Key	Cyc.	Area (μm^2)	GE	GE×Cyc.
64-bit block						
GIFT-64-128	64	128	30	1748.42	2191	65,730
SIMON-64-128	64	128	46	1863.86	2336	107,456
PRESENT-80	64	80	34	1904.03	2386	81,124
Dipper-64/96	64	96	30	1993.14	2498	74,940
Dipper-64/128	64	128	30	2253.55	2824	84,720
PRESENT-128	64	128	34	2364.47	2963	100,742
128-bit block (scaling reference)						
SIMON-128-128	128	128	70	2658.40	3332	233,240
GIFT-128-128	128	128	42	2702.83	3387	142,254

For the 64-bit block ciphers, which are the primary point of comparison for Dipper, three observations are immediate.

First, GIFT-64-128 is the most compact design in both static area (2191 GE) and area–latency product (65,730). Replacing GIFT’s bit-permutation linear layer with the Dipper hybrid layer adds $2824 - 2191 = 633$ GE at the same key size and the same round count. This added area is the cost of introducing the modular-addition layer, the half-state ARX recombination, and the slightly different key schedule.

Second, Dipper-64/128 is more compact than PRESENT-128 under this flow (2824 GE versus 2963 GE), and the gap widens in the area–latency product (84,720 versus 100,742, a 16% reduction). The same ordering holds with SIMON-64-128 for the latency product (84,720 versus 107,456, a 21% reduction), although SIMON remains smaller in static area at 2336 GE. Dipper’s advantage on the latency product stems from its 28-round count compared to 31 rounds (34 cycles) for PRESENT-128 and 44 rounds (46 cycles) for SIMON-64-128.

Third, the relative ordering of the four 64-bit families on this flow is $\text{GIFT} < \text{SIMON} < \text{PRESENT-80} < \text{Dipper-96} < \text{Dipper-128} < \text{PRESENT-128}$ in static area, but $\text{GIFT} < \text{Dipper-96} < \text{PRESENT-80} < \text{Dipper-128} < \text{PRESENT-128} < \text{SIMON}$ in area–latency product. The position of Dipper relative to PRESENT in particular is the same on both metrics, which is what one expects from the design rationale of inheriting the GIFT S-box and bit permutation while adding an ARX layer.

The 128-bit block measurements (SIMON-128-128 and GIFT-128-128) are included as a scaling reference rather than a direct comparison; the Dipper specification is fixed at 64-bit block size in this paper. Interestingly, the relative ordering of GIFT and SIMON inverts at the 128-bit block size: SIMON-128-128 is slightly smaller than GIFT-128-128 in static area (3332 vs. 3387 GE), but the shorter cycle count of GIFT (42 vs. 70) reverses the order on the area–latency product. This inversion is consistent with the way ARX and bit-sliced SPN designs scale with state width.

4.4. Limitations and Caveats

The methodology has three limitations that must be made explicit before the figures in Table 4 are used in further comparisons.

4.4.1. Absolute Values Are Flow-Dependent

The reported gate equivalents are pre-mapping cell-area estimates under one specific open-source flow on one specific 45 nm library. The corresponding figures in the original PRESENT [5], GIFT [6], and SIMON [7] papers were obtained on different libraries (UMC 130 nm, STM 90 nm/65 nm, IBM 130 nm, respectively), with different synthesis tools, and with different coding conventions. Direct numerical comparison between Table 4 and the original specifications is therefore not appropriate. The methodological contribution here is the internally consistent ordering of the eight designs under a single flow, not the reproduction of historical absolute values.

4.4.2. No DFFE Primitive

The version of the Nangate45 cell library available in the open-source distribution does not provide an enable-D flip-flop cell. Yosys consequently maps every state-register update through an external multiplexer, which adds a uniform overhead of approximately two to four gate equivalents per flip-flop. This affects all eight designs equally and therefore does not change the relative ordering, but it inflates absolute values relative to commercial libraries with native DFFE support. A repetition of the study with a library that includes DFFE (for example, the SkyWater 130 nm or IHP 130 nm open libraries) would be useful future work.

4.4.3. No Timing or Power Analysis

Reported figures are static cell area only. We do not perform timing closure, critical-path extraction, or power estimation, since the purpose of the study is to compare gate-level area under a fixed round-based template. Power-aware implementations of any of these ciphers, in particular SIMON and SPECK with extensive ARX bit-shift sharing, are documented in the literature [7] and would change the relative ranking.

To facilitate independent verification of all eight measurements, the Verilog source, testbenches, Yosys synthesis scripts, and ABC configuration files are released alongside the paper.

4.5. Comparison with Lightweight Block Ciphers

To place Dipper’s area in context, we compare it against eleven established lightweight block ciphers of the same 64-bit block size, all with a 128-bit key (except PRESENT, for which the 128-bit variant is used, and CRAFT, which additionally takes a 64-bit tweak). To make the comparison as fair as possible, we did not rely on figures reported across heterogeneous technologies. Instead, every cipher was re-implemented as a round-based full core—round function, key (or tweak) schedule, round-constant generation, and control FSM—in a uniform RTL style, and synthesized through the identical Yosys + ABC flow and Nangate45 standard-cell library used for Dipper (Section 4), with gate-equivalents computed as chip area divided by the drive-strength-1 NAND2 area. Each implementation was functionally verified against the official test vectors of its specification before synthesis. The S-boxes are written as case statements and mapped by ABC, so no cipher benefits from hand-optimized gate decompositions that others lack. The figures therefore reflect a relative comparison under a single consistent methodology rather than each design’s individually optimized, published implementation.

Table 5 shows that Dipper occupies the lower-middle of this twelve-cipher range at 2824 GE. The four ciphers smaller than Dipper each achieve their compactness through a specific simplification that Dipper does not adopt: GIFT and SIMON use very light key schedules (and SIMON a half-state Feistel round); LED omits a key schedule entirely; and SPECK is a pure-ARX Feistel with a minimal key path. Dipper instead carries a full 128-bit key schedule and a whole-state round, yet still synthesizes below TWINE, Midori-64, PRESENT-128, SKINNY, RECTANGLE, CRAFT, and Piccolo. Notably, Piccolo-128 is the largest core in the table (3890 GE) despite its famously compact published figure of 758 GE: that result relied on scan flip-flops, a hard-wired key, and library-specific AND-NOR/OR-NAND XOR cells in a 0.13 μm flow, whereas under our generic Yosys + ABC flow its permutation-based key schedule (eight 16-bit words with a per-four-round permutation network) and its two-layer $S \rightarrow M \rightarrow S$ F-function do not fold cheaply. This illustrates that a permutation-based key schedule, though compact with dedicated tricks, is not automatically area-efficient under a uniform generic flow.

Table 5. Round-based full-core area of Dipper and eleven lightweight block ciphers, all synthesized through the same Yosys + ABC + Nangate45 flow and verified against official test vectors. The final column cites each cipher’s original specification; all GE values are our own synthesis results under the single unified flow, not figures taken from those references. GE = chip area / NAND2 area.

Cipher	Structure	Rounds	GE	Spec.
GIFT-64/128	SPN (bit-perm)	28	2191	[6]
SIMON-64/128	Feistel (AND-rotate)	44	2336	[7]
LED-64/128	AES-like (no key sched.)	48	2394	[22]
SPECK-64/128	ARX Feistel	27	2691	[7]
Dipper-64/128	hybrid SPN-ARX	28	2824	(this work)
TWINE-128	Type-II GFN	36	2854	[23]
Midori-64	SPN (almost-MDS)	16	2908	[24]
PRESENT-128	SPN (bit-perm)	31	2963	[5]
SKINNY-64/128	SPN (tweakey)	36	3036	[10]
RECTANGLE-64/128	SPN (bit-slice)	25	3219	[25]
CRAFT-64/128	SPN (involutory, +64-bit tweak)	32	3368	[26]
Piccolo-128	GFN (Type-II)	31	3890	[27]

The comparison supports the design intent stated in Section 5.5: the hybrid SPN-ARX round buys fast diffusion (full diffusion by round 6, Section 6.2) and strong single-trail

bounds (Sections 7 and 9.1) at an area that remains competitive with, and in most cases smaller than, contemporary lightweight ciphers of the same block and key size. Area alone does not capture the full trade-off space—key agility, tweak support, diffusion speed, and security margin differ across these designs—but under a uniform implementation methodology Dipper is neither an outlier in size nor reliant on a reduced key schedule to achieve its footprint.

5. Software Performance and IoT Deployment

To assess the practicality of Dipper on IoT-class processors, we implemented the cipher in portable C99 in two profiles: a small profile that uses only the 16-byte S-box tables and 128 bytes of permutation index tables, and a fast profile that accelerates the S-box and bit-permutation layers with precomputed lookup tables totalling approximately 4 KB. Both profiles implement encryption and decryption for the 96- and 128-bit key variants; round keys are precomputed (224 bytes). Correctness of both profiles was verified against all specification test vectors of Table 3, including full encrypt–decrypt round trips, and the two profiles were cross-checked against each other.

5.1. Measured Software Performance

Table 6 reports timings measured on an Intel Core i7-1355U (x86-64, GCC 13.3, $-O2$), taken as the median of nine repetitions of 10^6 chained block operations (the data dependency between iterations prevents compiler over-optimization); cycle counts use the invariant TSC with serializing `rdtsc`/`rdtscp` instructions.

Table 6. Measured software performance of Dipper (Intel i7-1355U, GCC 13.3 $-O2$, median of 9 runs). Key expansion for Dipper-64/128 takes ~ 0.1 μ s.

Profile	Operation	Cyc/Block	Cyc/Byte	μ s/Block	MB/s
small (0.3 KB tables)	encrypt (64/128)	4007	500.9	1.54	5.2
small (0.3 KB tables)	decrypt (64/128)	3544	443.0	1.36	5.9
fast (4 KB tables)	encrypt (64/128)	1132	141.5	0.43	18.5
fast (4 KB tables)	decrypt (64/128)	1099	137.3	0.42	19.1
fast (4 KB tables)	encrypt (64/96)	1128	141.0	0.43	18.6
fast (4 KB tables)	decrypt (64/96)	1099	137.4	0.42	19.1

Encryption and decryption costs are symmetric to within a few percent, as expected from the structural symmetry of the round function, and the key size affects only key expansion. A single block is processed in well under half a microsecond in the fast profile; even the table-free small profile encrypts a block in ~ 1.5 μ s, which comfortably supports typical IoT traffic patterns (e.g., a 64-byte sensor payload in ~ 12 μ s).

In terms of asymptotic complexity, encryption and decryption of one block cost $\Theta(T)$ word operations for $T = 28$ rounds, with the measured per-round constants above; key expansion is likewise $\Theta(T)$. The space requirement is constant: an 8-byte state, 224 bytes of round keys, and 0.3 KB (small) or 4 KB (fast) of tables, with no dynamic allocation.

5.2. Memory Footprint on Embedded Targets

For constrained targets we cross-compiled the small profile with `arm-none-eabi-gcc` 13.2 and report exact code and memory footprints extracted from the produced objects (Table 7). On a Cortex-M0+ with `-Os`, the complete cipher—encryption, decryption, and both key schedules—occupies 936 bytes of ROM, and the total runtime RAM requirement, consisting of the permutation tables (132 B), the precomputed round keys (224 B), and the stack, is under 0.5 KB.

Table 7. Cortex-M code size of the small profile (bytes, from arm-none-eabi-size); RAM comprises tables, round keys, and stack.

Target	Opt.	ROM (Code)	RAM (Runtime)
Cortex-M0+	-Os	936	~420
Cortex-M3	-Os	1040	~420
Cortex-M4	-Os	1052	~420

We note that cycle-accurate microcontroller timings are board- and memory-configuration-specific and therefore require on-device measurement; we report exact code and RAM footprints together with cycle counts measured on x86-64, which bound the algorithmic operation count.

5.3. FPGA Implementation

To complement the ASIC figures with a reconfigurable-hardware data point relevant to FPGA-based edge devices, all twelve round-based cores of Table 5 were implemented on the Lattice ECP5 (LFE5U-25F) using the same fully open-source philosophy as the ASIC flow: Yosys (synth_ecp5) for synthesis and nextpnr-ecp5 for placement, routing, and static timing analysis. For the core comparison, designs were placed and routed in out-of-context mode (block-level implementation without I/O pad assignment), the standard method for comparing datapath cores whose port widths exceed the package pin count; reported $f_{\max}$ is the post-route static timing result. We emphasize that these are synthesis and place-and-route results, not physical board measurements; on-board power and latency measurement is left as future work.

Three observations follow from Table 8. First, the carry column makes the design spectrum directly visible: the pure-SPN, GFN, and almost-MDS cores use 0–3 CCU2C cells (round counters only; Midori-64 and GIFT-64/128 use none), the hybrid Dipper uses 19 (its two 16-bit adders), and the pure-ARX SPECK uses 35 (a 32-bit adder in the round function plus one in the key schedule)—placing Dipper structurally between the two families, exactly as intended. Second, Dipper’s LUT count (536) sits just above the middle of the twelve-core field: it exceeds the bit-permutation SPNs (GIFT, PRESENT), the ARX core SPECK, and the GFN cores (TWINE, Piccolo)—whose linear layers are free wiring on FPGA just as on ASIC—as well as SKINNY, but remains below Midori-64, RECTANGLE, LED, and CRAFT. Dipper’s $f_{\max}$ (137.6 MHz) is the third lowest of the twelve cores, above only Piccolo-128 and LED-64/128: its critical path passes through the S-box layer and a 16-bit carry chain in series, which is the expected cost of the hybrid round. Third, the low round count compensates at the block level: despite the lower clock rate, Dipper’s throughput (293 Mbit/s) exceeds that of SIMON-64/128 (246 Mbit/s) and LED (170 Mbit/s), because its 28 heavier rounds complete a block in fewer cycles than their 44 and 48 lighter rounds—the “fewer, heavier rounds” trade-off of Section 5.5 carrying over to reconfigurable hardware—and exceeds that of Piccolo-128 (224 Mbit/s) through its higher clock rate at a comparable cycle count. At 2–3% utilization of even a small ECP5-25k device, any of these cores is a negligible-area peripheral in an FPGA-based edge gateway.

To demonstrate deployability beyond block-level implementation, we additionally wrapped the Dipper-64/128 core in a 9-pin serial-I/O top level (dipper_serial_top, released with the artifacts), verified it against the specification test vectors in simulation, and carried it through full in-context placement and routing to a complete ECP5 configuration bitstream (closing timing at 140.8 MHz). The comparison table above deliberately remains based on the out-of-context core results, since the serial wrapper adds I/O-protocol logic that is not part of any cipher’s round function.

Table 8. ECP5 (LFE5U-25F) post-place-and-route results for the twelve round-based cores under the identical Yosys + nextpnr open-source flow (out-of-context). Carry counts ECP5 CCU2C cells; latency = cycles/ $f_{\max}$; throughput = $f_{\max} \times 64$ /cycles.

Core	LUT4	FF	Carry	$f_{\max}$ (MHz)	Cyc.	Lat. (μ s)	Thr. (Mbit/s)
GIFT-64/128	266	200	0	218.3	30	0.137	465.7
SIMON-64/128	291	201	3	176.7	46	0.260	245.8
SPECK-64/128	327	199	35	183.0	29	0.158	403.8
TWINE-128	417	218	3	183.7	38	0.207	309.4
PRESENT-128	447	208	3	182.8	34	0.186	344.1
Piccolo-128	485	264	3	115.2	33	0.286	223.5
SKINNY-64/128	493	270	3	184.2	38	0.206	310.2
Dipper-64/128	536	204	19	137.6	30	0.218	293.4
Midori-64	570	135	0	172.4	18	0.104	613.0
RECTANGLE-64/128	727	236	3	215.8	27	0.125	511.5
LED-64/128	749	79	3	132.7	50	0.377	169.9
CRAFT-64/128	836	270	3	157.6	34	0.216	296.7

5.4. Deployment Considerations for IoT Scenarios

The combined hardware and software results position Dipper for three representative IoT deployment classes.

5.4.1. Passive RFID/Smart-Card Class

The round-based ASIC implementation requires 2824 GE (Dipper-64/128, Nangate45, Section 4), within the ~ 3000 GE envelope commonly cited for passive RFID tags. As in comparable designs, serialized datapaths can trade latency for further area reduction.

5.4.2. Battery-Powered Sensor Nodes

On Cortex-M-class MCUs the complete cipher fits in under 1 KB of flash and 0.5 KB of RAM—roughly 3–6% of the flash of a typical 16–32 KB sensor-node part—leaving the overwhelming majority of resources to the application. The absence of dynamic allocation and the 224-byte precomputed key schedule suit always-on, low-duty-cycle firmware.

5.4.3. Edge Gateways

On commodity x86-64 hardware the fast profile sustains ~ 19 MB/s per core in a straightforward single-block implementation, which is ample for aggregating encrypted traffic from large sensor fleets; bit-sliced implementations would raise this figure substantially if required. For FPGA-based gateways, the ECP5 implementation of Section 5.3 sustains ~ 293 Mbit/s at under 3% device utilization.

Across all three classes, the same specification and test vectors apply, and the implementation requires no platform-specific cryptographic instructions.

5.5. Design Rationale: Number of Modular Additions

A central design decision in the Dipper round function is the use of exactly two modular additions per round. To justify this choice quantitatively, we conduct an ablation over three round functions that are identical in every layer except the ARX step: a variant **V0** with no additions (a pure substitution–permutation round in which all four words pass through), the specified variant **V2** with two additions and half-state pass-through, and a variant **V4** with four parallel additions in which every word is modified. All other components—the S-box, the bit permutation, the key schedule, the controller, and the register/reset policy—are byte-identical across the three, so every measured difference is attributable solely to the adder count. The V4 adders are wired in parallel so that the critical path remains a single adder, isolating area (rather than latency) as the variable.

Each variant was synthesized through the same Yosys + ABC + Nangate45 flow used for the main results (Section 4); as a consistency check, V2 reproduces the reported Dipper-

64/128 figure of 2824 GE exactly. Diffusion speed was measured as the first round at which the mean strict-avalanche Hamming distance reaches 32 (Section 6.2), and single-trail linear weights were obtained with the model of Section 9.1.

Three observations follow from Table 9.

Table 9. ARX ablation. All variants share every layer except the number of modular additions. Area/GE from the main synthesis flow (Nangate45); diffusion round from avalanche; linear weight from single-trail search. The V2 (Dipper) row is shown in bold to identify the specified design.

Variant	Additions	Area (μm^2)	GE	Full-Diffusion Round	Linear W at R = 6
V0 (pure SPN)	0	2201.2	2759	8	6
V2 (Dipper)	2	2253.6	2824	6	11
V4 (full ARX)	4	2437.4	3055	4	≥ 50

First, the pure-SPN variant V0 is clearly the weakest: it reaches full diffusion only at round 8 and has the lowest per-round linear weight, which would not justify describing the design as a hybrid construction. Introducing the two half-state additions (V2) advances full diffusion by two rounds (to round 6) and nearly doubles the six-round linear weight (from 6 to 11), for a hardware cost of only 65 GE—about 2.4% over the pure-SPN baseline, or roughly 32 GE per adder once the synthesis tool shares logic with the surrounding datapath.

Second, moving from two additions to four (V4) does buy larger trail weights, but at a disproportionate and, for Dipper, unnecessary cost. The second pair of adders adds 231 GE—more than three times the marginal cost of the first pair—because the half-state pass-through that keeps V2 inexpensive is lost and every word must now be routed from its register through an adder. Its diffusion advantage over V2 is only two rounds (4 versus 6), which is immaterial given that the full cipher runs for 28 rounds; both V2 and V4 already leave a security margin well in excess of $2.5\times$ the longest distinguisher we identify (Section 9).

Third, and consequently, two additions is the point at which the ARX diffusion benefit is captured—a genuine two-round improvement in diffusion and a substantial increase in per-round trail weight over the pure-SPN round—without paying for security margin that the round count already provides. Our aim is a design that is sufficient rather than maximal in its use of nonlinearity from carries: the half-state, two-addition round delivers the hybrid SPN–ARX behavior at minimal and, notably, cheapest-per-adder area. This is the rationale for the choice of two modular additions in the Dipper round function.

6. Empirical Statistical Properties

This section reports the results of an empirical statistical evaluation of Dipper. We measure the avalanche characteristics and the strict avalanche criterion (SAC), characterize the pairwise output-bit dependencies through the bit independence criterion (BIC), and run the NIST SP 800-22 statistical test suite on cipher output streams. The measurements are intended to verify that the round function and key schedule together produce output that is statistically indistinguishable from uniform random data under standard battery tests. They do not, by themselves, constitute proof of cryptographic security, which is addressed by the dedicated differential analysis of Section 7.

6.1. Methodology

All four tests share a common configuration. Each test draws $N = 50,000$ samples from the Python reference implementation of Dipper described in Section 2. The master random seed is fixed at 20,260,426 for reproducibility; the four sub-tests draw independent sample streams using seed offsets $\{0, 100, 200, 300\}$ so that no two sub-tests share

Monte Carlo randomness. Both Dipper-64/96 and Dipper-64/128 are evaluated under identical conditions.

The avalanche, SAC, and BIC tests measure properties of the encryption function $E_K : \{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$ under random plaintexts and random keys. The NIST SP 800-22 suite, by contrast, evaluates sequential bitstreams produced by the cipher in counter mode. The configurations are summarized in Table 10.

Table 10. Configuration of the four empirical tests. N denotes the number of independent samples drawn per measurement cell.

Test	Parameter	Configuration
Avalanche	Samples per bit	$N = 50,000$
	Targets	Plaintext bits and key bits
SAC	Samples per cell	$N = 50,000$
	Matrix size	64×64 (plaintext) $96 \times 64, 128 \times 64$ (key)
BIC	Samples per cell	$N = 50,000$
	Cells per matrix	$\binom{64}{2} \times n_{\text{in}}$
NIST 800-22	Bits per stream	10^7
	Streams	50 per variant
	Suite	Official NIST C reference, sts-2.1.2

6.2. Avalanche and Round-by-Round Diffusion

For each input position $i \in \{0, \dots, n_{\text{in}} - 1\}$, the avalanche test draws N random (plaintext, key) pairs, encrypts both the unmodified input and the input with bit i flipped, and records the Hamming distance between the two ciphertexts. The mean and standard deviation over the N samples are reported per bit; an aggregate mean over all bits and samples is reported per variant.

Under the ideal random-permutation model, each output bit flips with probability $1/2$ independently, so the Hamming distance between the two ciphertexts follows $\text{Bin}(64, 1/2)$. The expected mean is 32.0 and the expected standard deviation is 4.000.

Table 11 reports the measured aggregate statistics for Dipper-64/96 and Dipper-64/128, separately for plaintext-bit and key-bit perturbations. All four configurations match the ideal mean and standard deviation to three decimal places.

Table 11. Aggregate avalanche statistics over $N = 50,000$ samples per input bit. The expected mean and standard deviation under the random permutation model are 32.000 and 4.000, respectively.

Variant	Test	Mean	Std. Dev.	%
Dipper-64/96	PT-flip	32.002	4.000	50.00
Dipper-64/96	K-flip	31.999	4.000	50.00
Dipper-64/128	PT-flip	32.000	4.001	50.00
Dipper-64/128	K-flip	32.001	4.000	50.00

To localize the diffusion behavior across the round function, we repeat the plaintext avalanche test on reduced-round variants of Dipper for $r \in \{1, \dots, 28\}$ rounds. The resulting per-round mean Hamming distance is reported in Figure 2. Both Dipper variants reach the ideal value of 32 bits at round 6 and remain at that value through round 28. The round count of 28 therefore provides a $28/6 \approx 4.7\times$ margin over the diffusion saturation point.

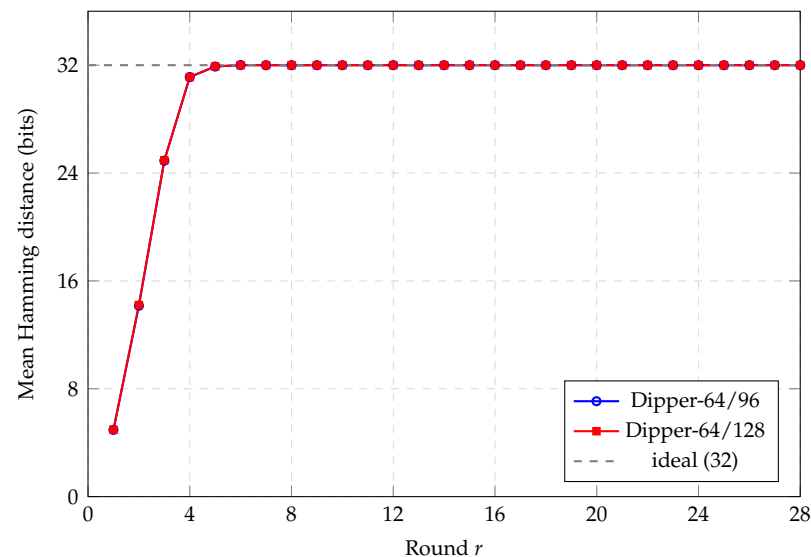


Figure 2. Round-by-round plaintext-avalanche mean Hamming distance for Dipper-64/96 and Dipper-64/128. Saturation at the ideal value of 32 bits occurs at round 6 for both variants.

6.3. Strict Avalanche Criterion

The strict avalanche criterion strengthens the avalanche test by requiring that each individual output bit flip with probability $1/2$ when a single input bit is perturbed. For each pair (i, j) of an input-bit index i and an output-bit index j , we estimate $\Pr[c_j(P) \oplus c_j(P \oplus e_i) = 1]$ over $N = 50,000$ random plaintext-key pairs, where e_i is the unit vector at position i . The result is a $(n_{\text{in}} \times 64)$ matrix of empirical probabilities, ideally constant at $1/2$.

The standard error of each matrix cell under the ideal model is $\sqrt{0.25/N} = 0.00224$. With $64 \times 64 = 4096$ cells in a plaintext-SAC matrix (or up to $128 \times 64 = 8192$ cells for the 128-bit key matrix), the maximum cell deviation is expected, by extreme-value statistics, to lie around 4σ to 5σ of the per-cell standard error, i.e., at most ≈ 0.011 .

Table 12 reports the aggregate statistics for the four SAC matrices. All four matrices have means within 5×10^{-5} of the ideal 0.5, standard deviations matching the theoretical Bernoulli estimate to four decimal places, and maximum absolute deviations $(|p - 1/2|_{\text{max}})$ below 0.0092 in all cases. These figures are fully consistent with i.i.d. Bernoulli sampling under the ideal model.

A representative SAC heatmap for the plaintext-SAC matrix of Dipper-64/96 is shown in Figure 3. The heatmap exhibits no visible row, column, or diagonal structure that would indicate an output bit being preferentially affected by a particular input bit. The visual texture matches the appearance of independent Bernoulli noise at the per-cell standard error indicated above.

Table 12. Strict avalanche criterion (SAC). Each matrix has all cells sampled with $N = 50,000$ independent plaintext-key pairs. The theoretical per-cell standard error under the ideal random-permutation model is $\sigma = 0.002236$.

Matrix	Mean	Std. Dev.	Min. Cell	Max. Cell
PT-SAC-64/96	0.50003	0.00223	0.4913	0.5086
PT-SAC-64/128	0.50001	0.00222	0.4912	0.5078
KEY-SAC-64/96	0.50002	0.00223	0.4908	0.5079
KEY-SAC-64/128	0.50000	0.00223	0.4917	0.5079

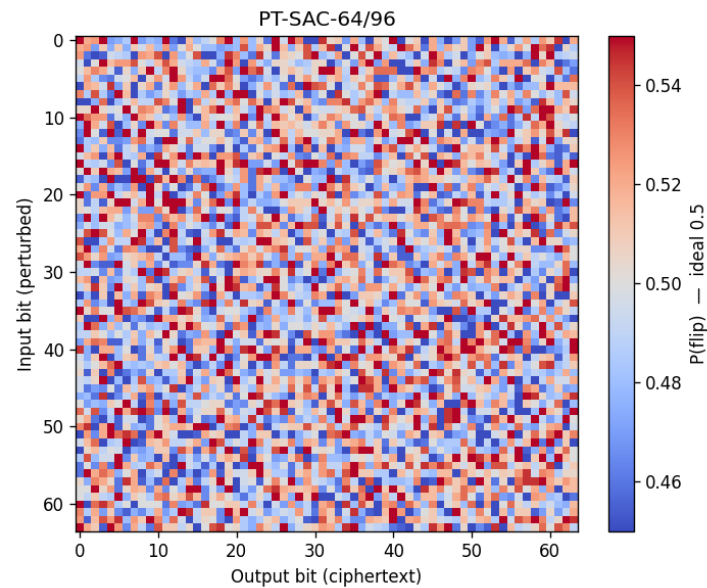


Figure 3. Strict avalanche criterion heatmap for PT-SAC-64/96. Each cell (i, j) shows the empirical probability $\Pr[c_j \text{ flips} \mid p_i \text{ flipped}]$ over $N = 50,000$ samples. Color encodes the deviation from the ideal value 0.5; the observed range is consistent with the expected extreme-value behavior for a 64×64 matrix under independent Bernoulli sampling. No row, column, or diagonal structure is visible, consistent with i.i.d. sampling noise.

6.4. Bit Independence Criterion

The bit independence criterion measures the pairwise statistical dependence between output bits c_j and c_k ($j < k$) conditioned on the perturbation of a single input bit. For each input bit i and each output-bit pair (j, k) , we compute the Pearson correlation

$$r_i(j, k) = \frac{\text{Cov}(B_j^{(i)}, B_k^{(i)})}{\sigma_j^{(i)} \cdot \sigma_k^{(i)}}, \quad (16)$$

where $B_j^{(i)} = c_j(P) \oplus c_j(P \oplus e_i)$ denotes the output-bit- j flip indicator when input bit i is perturbed, and $\sigma_j^{(i)}$ is its empirical standard deviation. The Adams–Tavares BIC parameter $\text{BIC}_i(j, k) = 2 |\text{Cov}(B_j^{(i)}, B_k^{(i)})|$ is computed from the same per-cell sample counts in a single pass. The Pearson formulation is the modern lightweight-cipher standard [6,10]; the Adams–Tavares parameter [28] is reported for compatibility with earlier literature.

Each of the four BIC matrices contains $\binom{64}{2} = 2016$ output-bit pairs per input bit. The total cell counts are 129,024 for each plaintext-BIC matrix, 193,536 for KEY-BIC-64/96, and 258,048 for KEY-BIC-64/128, giving 709,632 cells in total for each metric. Under the ideal model, the Pearson correlation has per-cell standard error $1/\sqrt{N-2} = 0.00447$ at $N = 50,000$. Across 709,632 approximately i.i.d. cells, extreme-value theory predicts a maximum absolute Pearson correlation of $\sigma \cdot \sqrt{2 \ln(709,632)} \approx 0.0232$ from sampling noise alone.

Table 13 reports the measurements. For all four matrices, the mean Pearson correlation is within 10^{-5} of zero and the standard deviation matches the theoretical 0.00447 to five decimal places. The maximum absolute Pearson correlation across each matrix lies in the range 0.0199 to 0.0217, all below the predicted extreme-value bound and well below the 0.05 threshold commonly cited in the lightweight-cipher literature. The Adams–Tavares parameter is correspondingly bounded: the largest value observed across all 709,632 cells

is 0.0108, consistent with the relation $BIC \approx 0.5 \cdot |r|$ that holds when both output bits have marginal probability close to $1/2$.

Table 13. Bit independence criterion (BIC) results at $N = 50,000$. Pearson correlation r and Adams–Tavares BIC are computed jointly from identical samples. The theoretical Pearson standard error is $\sigma = 0.00447$, and the extreme-value bound on $\max |r|$ across 709,632 cells is approximately 0.0232.

Matrix	Metric	Mean	Std. Dev.	Mean $ \cdot $	Max $ \cdot $
PT-BIC-64/96	r	$+1.4 \times 10^{-6}$	0.00447	0.00357	0.0199
	BIC	0.00178	0.00135	0.00178	0.0099
PT-BIC-64/128	r	$+5.7 \times 10^{-6}$	0.00448	0.00357	0.0205
	BIC	0.00179	0.00135	0.00179	0.0102
KEY-BIC-64/96	r	$+1.1 \times 10^{-6}$	0.00448	0.00357	0.0206
	BIC	0.00178	0.00135	0.00178	0.0103
KEY-BIC-64/128	r	-4.5×10^{-7}	0.00447	0.00357	0.0217
	BIC	0.00178	0.00135	0.00178	0.0108

Figure 4 shows the per-input-bit maximum absolute correlation for the PT-BIC-64/96 matrix. The plot exhibits no input bit with anomalously large correlations: every input bit produces a $\max |r|$ near the extreme-value level for 2016 output-bit pairs, without spikes or clusters. The remaining three matrices follow the same pattern.

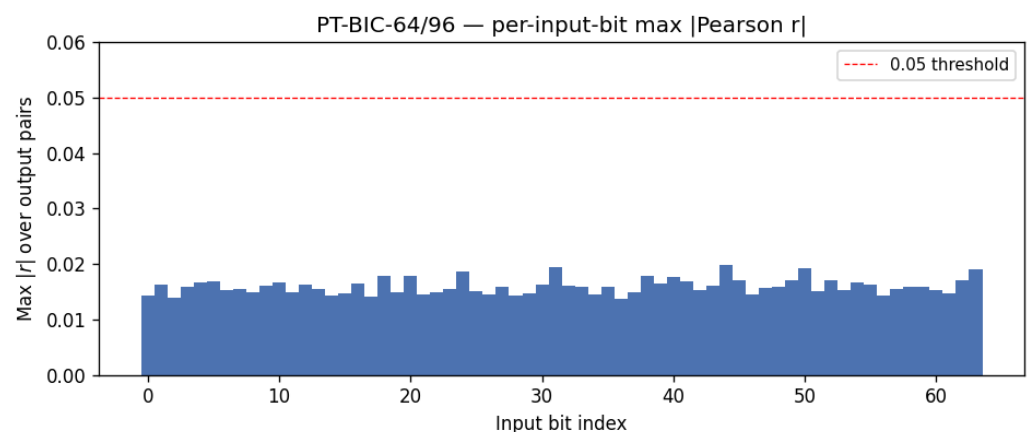


Figure 4. Per-input-bit maximum absolute Pearson correlation $\max_{(j,k)} |r_i(j,k)|$ for the PT-BIC-64/96 matrix at $N = 50,000$. All 64 values lie near the extreme-value level expected from independent Bernoulli sampling noise, with no anomalous spikes or structural clustering.

6.5. NIST SP 800-22 Statistical Test Suite

To complement the structural avalanche and dependency tests, we evaluate the bit-level randomness of cipher output streams using the NIST SP 800-22 statistical test suite, version sts-2.1.2 (the official C reference implementation [29]). For each variant, the cipher is operated in counter mode under a fixed random key; concatenated ciphertext blocks form a single bitstream from which 50 disjoint sub-streams of 10^7 bits each are extracted. The suite applies 15 statistical tests; several tests are repeated with multiple parameter settings, giving 188 test instances per variant.

Each test instance produces (i) a per-stream pass rate against a target $\alpha = 0.01$, and (ii) an aggregate uniformity p -value from a χ^2 goodness-of-fit test on the empirical p -value distribution. A test instance passes the suite's pass-rate criterion if at least 47 of 50 streams produce $p > 0.01$ (or 40 of 43 for the random excursion tests, whose applicability depends on the stream), and passes the uniformity criterion if the uniformity p -value exceeds 10^{-4} , per NIST SP 800-22 Rev. 1a, Section 4.2.1.

For Dipper-64/96, all 188 test instances pass both the pass-rate and the uniformity criterion: 188/188 pass-rate and 188/188 uniformity. For Dipper-64/128, all 187 test in-

stances that produced output pass both criteria: 187/187 pass-rate and 187/187 uniformity. One NonOverlappingTemplate configuration did not produce a valid output stream due to a parameter mismatch in the NIST template setup and is therefore excluded from the denominator. These results show no detectable deviation from the NIST SP 800-22 criteria under the tested counter-mode configuration.

6.6. Discussion

Across the four empirical tests, the measured statistics for both Dipper-64/96 and Dipper-64/128 are consistent with the ideal random-permutation model to within the sampling resolution available at $N = 50,000$. The avalanche mean matches 32.000 bits to three decimal places; the SAC matrix mean matches 0.5 to four decimal places with a standard deviation matching the Bernoulli estimate; the BIC Pearson standard deviation matches the theoretical 0.00447 to five decimal places, with the maximum absolute correlation across 709,632 cells at 0.0217, below the extreme-value prediction of 0.0232; and the NIST SP 800-22 suite reports all 188 (respectively 187) test instances passing both pass-rate and uniformity criteria, as summarized in Table 14. The round-by-round analysis localizes the saturation of diffusion at round 6, giving a $28/6 \approx 4.7\times$ margin on full diffusion alone.

Table 14. NIST SP 800-22 results by test category. Each entry lists the number of test instances passing the pass-rate criterion (proportion of streams with $p > 0.01$ above the suite-defined threshold) and the uniformity criterion (uniformity p -value $> 10^{-4}$). The Total row is shown in bold to emphasize the aggregate results.

Test	Dipper-64/96		Dipper-64/128	
	Pass-Rate	Uniformity	Pass-Rate	Uniformity
Frequency	1/1	1/1	1/1	1/1
BlockFrequency	1/1	1/1	1/1	1/1
CumulativeSums	2/2	2/2	2/2	2/2
Runs	1/1	1/1	1/1	1/1
LongestRun	1/1	1/1	1/1	1/1
Rank	1/1	1/1	1/1	1/1
FFT	1/1	1/1	1/1	1/1
NonOverlappingTemplate	148/148	148/148	147/147	147/147
OverlappingTemplate	1/1	1/1	1/1	1/1
Universal	1/1	1/1	1/1	1/1
ApproximateEntropy	1/1	1/1	1/1	1/1
RandomExcursions	8/8	8/8	8/8	8/8
RandomExcursionsVariant	18/18	18/18	18/18	18/18
Serial	2/2	2/2	2/2	2/2
LinearComplexity	1/1	1/1	1/1	1/1
Total	188/188	188/188	187/187	187/187

These results constitute necessary conditions for cryptographic security but are not sufficient. In particular, none of the four tests detects differential or linear trails of low probability, which are evaluated separately in Section 7. The combination of empirical and structural evidence, taken together, gives a complete first-order characterization of the Dipper round function under standard lightweight-cipher evaluation methodology.

7. Differential Analysis via MILP

This section reports a differential trail analysis of Dipper based on a mixed-integer linear programming (MILP) model. We formulate the search for the lowest-weight single differential trail as an MILP instance, parameterized by the number of rounds R , and solve it for $R \in \{1, \dots, 7\}$. The resulting weights are upper bounds on the probability of the best single differential characteristic of R -round Dipper under the standard Markov-cipher assumption. We complement the MILP results with empirical sampling against the

reference implementation, both under independent random round keys and under the real Dipper-64/96 and Dipper-64/128 key schedules. Section 8 examines the clustering behavior observed from $R = 5$ onward, focusing on the $R = 5$ case where the match count is high enough for detailed diagnosis.

7.1. MILP Formulation

The MILP model presented here makes three methodological choices that together distinguish it from simple activity-only analyses. First, the S-box layer is modeled with the full GIFT DDT rather than only the active-S-box count, allowing per-transition fractional weights such as $-\log_2(6/16) = 1.42$ to be represented. Second, the modular-addition layer is modeled with the Lipmaa–Moriai differential probability condition rather than a coarse active-adder approximation; the encoding is verified bit-exactly against brute-force differential probabilities for $n = 4$. Third, the empirical validation is performed under both independent random round keys and the real Dipper key schedules, allowing a direct check of whether the Markov-cipher model remains faithful for the deterministic schedules.

The MILP model encodes the propagation of a 64-bit input difference $\Delta^{(0)}$ through R rounds of Dipper, producing an output difference $\Delta^{(R)}$. Each round of the model encodes the SubCells, WordRotations, modular-addition, and bit-permutation layers of Section 2.2; the AddRoundKey layer is omitted because, under the standard Markov-cipher assumption [30], XOR with an independent uniform round key has no effect on differential propagation.

The objective is the total trail weight, defined as the negative base-2 logarithm of the trail probability:

$$W = \sum_{r=1}^R W_{\text{sbox}}^{(r)} + \sum_{r=1}^R W_{\text{arx}}^{(r)} \quad (17)$$

where $W_{\text{sbox}}^{(r)}$ aggregates the contributions of the sixteen S-boxes in round r and $W_{\text{arx}}^{(r)}$ aggregates the two modular additions. A feasible solution of weight W is a differential characteristic with probability at most 2^{-W} .

7.1.1. S-Box Layer

The GIFT 4-bit S-box $\mathcal{S}$ has differential distribution table (DDT) entries $\text{DDT}[\delta_{\text{in}}][\delta_{\text{out}}] \in \{0, 2, 4, 6\}$ out of 16. Excluding the trivial transition $(0, 0)$, there are 97 nonzero entries distributed as in Table 15.

Table 15. Nonzero entries of the GIFT S-box DDT (excluding the trivial $(\delta_{\text{in}}, \delta_{\text{out}}) = (0, 0)$ transition).

DDT Entry	Weight $-\log_2(\text{DDT}/16)$	Number of Transitions
6	1.415	2
4	2.000	18
2	3.000	78

For each S-box position $j \in \{0, \dots, 15\}$ in each round r , the model introduces 98 binary indicator variables: one for the trivial $(0, 0)$ transition and one per valid nonzero DDT entry. Three classes of constraints enforce consistency:

1. exactly one indicator per S-box is set to 1, modelling the single transition that occurs at that S-box;
2. the four input-difference bits of the S-box are determined linearly from the active indicators (i.e., are the bitwise OR of all indicators whose input pattern has the corresponding bit set);

3. the four output-difference bits are determined symmetrically from the indicators' output patterns.

The active indicator contributes its associated weight, scaled by a constant integer factor (see Section 7.1.4), to the objective.

7.1.2. Modular-Addition Layer

The two 16-bit modular additions in each round are encoded using the Lipmaa–Moriai theorem [8] on the differential probability of addition modulo 2^n (in our case, $n = 16$). Given input differences α, β and output difference γ , the transition is valid if and only if

$$\alpha[0] \oplus \beta[0] \oplus \gamma[0] = 0. \quad (18)$$

$$\text{eq}_{i-1} = 1 \Rightarrow \alpha[i] \oplus \beta[i] \oplus \gamma[i] \oplus \beta[i-1] = 0, \quad i = 1, \dots, n-1. \quad (19)$$

where the indicator $\text{eq}_i = 1$ if and only if $\alpha[i] = \beta[i] = \gamma[i]$. When the transition is valid, its differential weight is

$$W_{\text{lm}}(\alpha, \beta, \gamma) = \sum_{i=0}^{n-2} (1 - \text{eq}_i). \quad (20)$$

For each 16-bit adder, the model introduces 16 bit variables for each of α, β, γ , 16 indicator variables for the equality-checks eq_i , and auxiliary bits to linearize the parity condition in (19). Six inequalities per bit position encode each eq_i in terms of $\alpha[i], \beta[i], \gamma[i]$. Two constraints per bit position $i \geq 1$ encode the eligibility condition: a parity equation $\alpha[i] + \beta[i] + \gamma[i] + \beta[i-1] = 2d + x_4$ over integer auxiliary d and binary auxiliary x_4 , plus the implication $x_4 + \text{eq}_{i-1} \leq 1$. The weight contribution per adder, $\sum_{i=0}^{14} (1 - \text{eq}_i)$, is added to the objective.

The Lipmaa–Moriai encoding was verified by exhaustive enumeration against the truth table of xdp^+ for $n = 4$: all 4096 triples $(\alpha, \beta, \gamma) \in (\mathbb{F}_2^4)^3$ produce identical valid/invalid classifications and identical weight values between the MILP encoding and the brute-force computation.

7.1.3. Linear Layers

The four 16-bit word rotations of the Dipper round function and the GIFT-64 bit permutation are bijective bit-permutations and are encoded as direct equality constraints between input and output difference bits. The half-state pass-through of the modular-addition layer is encoded by direct equality between the B and D word differences before and after the layer.

7.1.4. Solver Setup

The model is implemented in Python using PuLP [31] and solved with COIN-OR CBC [32]. Because CBC requires integer-valued objective coefficients, the fractional weight 1.415 associated with the highest-probability DDT entries is scaled by a common integer factor (we use 100); the reported total weight W is the solver objective value divided by the same factor. We solve a separate instance for each round count $R \in \{1, \dots, 7\}$ and report the minimum objective.

7.2. Trail Bounds for $R = 1, \dots, 7$

Table 16 reports the lowest-weight differential characteristic found by the MILP for each round count up to seven. For each solution, the table lists the number of active S-boxes, the S-box and ARX contributions to the total weight, and the resulting probability bound.

The CBC solver runtime grows steeply with R , reflecting the exponential branching of the search tree as more rounds are encoded. Wall-clock solve times on a standard workstation are below one minute for $R \leq 5$, approximately two hours for $R = 6$, and approximately four hours for $R = 7$. We use time limits of 7200 s for $R = 6$ and 14,400 s for $R = 7$; all instances up to $R = 7$ are proven optimal within these budgets. Extending the exact search to $R \geq 8$ is left to future work because the expected solver runtime exceeds the available compute budget.

Table 16. Lowest-weight differential characteristics found by the MILP for R rounds of Dipper, under the standard Markov-cipher assumption.

R	Active S-Boxes	W_{sbox}	W_{arx}	W	$\Pr \leq$
1	1	2.00	0.00	2.00	$2^{-2.00}$
2	2	4.00	1.00	5.00	$2^{-5.00}$
3	3	6.00	3.00	9.00	$2^{-9.00}$
4	4	7.42	6.00	13.42	$2^{-13.42}$
5	5	12.00	7.00	19.00	$2^{-19.00}$
6	6	12.00	11.00	23.00	$2^{-23.00}$
7	7	15.42	12.00	27.42	$2^{-27.42}$

Several observations follow from Table 16. First, the active S-box count grows linearly with R , reaching one new active S-box per round throughout the searched range. Second, the modular-addition layer contributes weight only when the difference at the input of the layer has a nonzero word in both A and B (or both C and D); the weight per ARX layer is bounded by the Hamming weight of the active words minus one. Third, the per-round weight increment ranges from approximately 3 to 5.5 bits across $R = 1$ to $R = 7$, with an average of 4.2 bits per round computed over the six round transitions in the table.

Extrapolating linearly from the $R = 1, \dots, 7$ trend with a slope of 4.2 bits per round, the trail-level bound for the full 28-round cipher would lie below 2^{-115} , well beyond the 2^{-64} generic security level for a 64-bit block cipher. We emphasize that this is an extrapolation, not a proof: as discussed in Section 8, the iterative structure observed for $R \leq 6$ implies that the actual differential probability for the same input–output difference is higher than the single-trail bound suggests, and a tight differential bound at higher round counts requires either trail enumeration up to a fixed weight threshold or a full differential transition matrix.

7.3. Empirical Validation

To verify that the MILP-found characteristics are realizable in the real cipher and to quantify the agreement between the bound and the true differential probability, we sample plaintext pairs whose difference matches the MILP-found input difference, encrypt under random keys, and measure the proportion of pairs whose ciphertext difference matches the MILP-found output difference. We perform this measurement under two key-handling models:

- **Random round keys.** The R round keys are drawn independently and uniformly at random for each pair of plaintexts. This matches the Markov-cipher assumption used by the MILP.
- **Real Dipper key schedule.** A single master key is drawn uniformly at random per batch of plaintext pairs, and the round keys are derived by the actual Dipper-64/96 or Dipper-64/128 key schedule of Section 2.4. This tests whether the schedule introduces a key-dependent deviation from the Markov-cipher prediction.

Table 17 reports the results for $R = 1$ through $R = 7$ under independent random round keys; for $R = 5$ we also report the empirical probability under the two real key schedules. Each row uses 2×10^6 plaintext pairs distributed over 40 random master keys.

Table 17. Empirical differential probabilities measured against the reference Dipper implementation, compared to the MILP single-trail bound. Each row uses 2×10^6 plaintext pairs. “RK” denotes the random-round-key model; the right two columns report the real Dipper key-schedule probabilities measured at $R = 5$.

R	MILP W	Random RK $-\log_2 \hat{p}$	64/96 KS	64/128 KS	Gap
1	2.00	2.00	2.00	2.00	0.00
2	5.00	5.00	5.00	5.00	0.00
3	9.00	7.65	—	—	−1.35
4	13.42	13.85	—	—	0.43
5	19.00	11.98	12.09	11.82	−7.02
6	23.00	14.76	—	—	−8.24
7	27.42	18.35	—	—	−9.07

The comparison between the MILP single-trail bound and the empirical probability is shown in Figure 5.

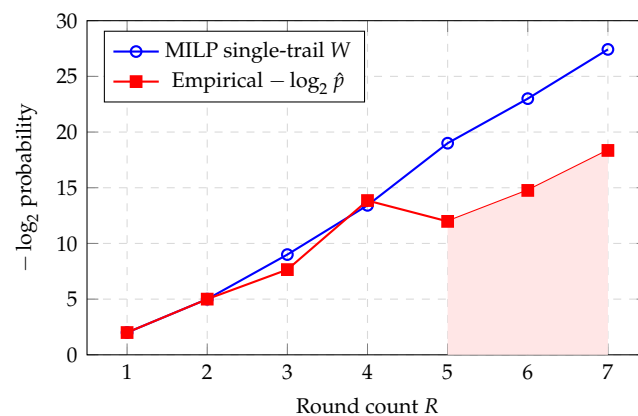


Figure 5. MILP single-trail bound (blue) versus empirical $-\log_2 \hat{p}$ (red) as a function of round count R . For $R \in \{1, 2, 4\}$ the two values match within sampling noise; for $R \in \{3, 5, 6, 7\}$ the empirical probability lies below the trail bound by an increasing margin, indicating multi-trail clustering. The gap widens at approximately 1.0 bit per round across $R = 5, 6, 7$.

For $R \in \{1, 2, 4\}$, the empirical $-\log_2 \hat{p}$ agrees with the MILP bound W to within the sampling resolution. For $R \in \{3, 5, 6, 7\}$, the empirical value lies below the MILP bound by 1.35, 7.02, 8.24, and 9.07 bits, respectively, indicating that multiple trails with the same input–output difference contribute to the differential probability. The gap grows approximately linearly in R over this range, which is the signature of an iterative pattern in which the number of contributing trails roughly doubles per round.

At $R = 5$ we additionally measure the empirical probability under the real Dipper-64/96 and Dipper-64/128 key schedules. The three measurements (random-RK 11.98, 64/96-KS 12.09, 64/128-KS 11.82) lie within 0.27 bits of each other, ruling out the hypothesis that the gap is caused by a key-schedule effect: the gap is a property of the round function itself.

The growing gap at $R \in \{3, 5, 6, 7\}$ is large enough to warrant separate analysis, since deviations of up to 9 bits between the trail bound and the empirical differential could in principle indicate any of (i) an encoding bug in the MILP, (ii) a sampling artifact, or (iii) genuine differential clustering. Section 8 presents the diagnostic that distinguishes among these possibilities.

7.4. Summary

The MILP analysis establishes that the best single differential characteristic of R -round Dipper, under the standard Markov-cipher assumption, has weight at least W_R for the values listed in Table 16, increasing from $W_1 = 2$ to $W_7 = 27.42$. Empirical sampling confirms that the characteristics are realizable in the real cipher and that the bound is tight to within sampling noise for $R \in \{1, 2, 4\}$. A clustering gap of 1.35, 7.02, 8.24, and 9.07 bits is observed at $R = 3, 5, 6, 7$, respectively; the next section characterizes the structural source of these gaps.

8. Diagnostic of Iterative Patterns

The growing clustering gap measured at $R \in \{3, 5, 6, 7\}$ in Section 7.3, against single-trail MILP bounds that grow at approximately 4.2 bits per round, raises three possible explanations: (i) a flaw in the MILP encoding that admits unrealizable trails; (ii) a flaw in the empirical validator that overcounts matching pairs; or (iii) genuine differential clustering, in which several distinct trails accumulate to the same input–output difference and produce an actual differential probability higher than that of the single best trail. This section presents a six-part diagnostic, focused on $R = 5$ as the most amenable case for direct inspection, that distinguishes among these explanations.

We focus the diagnostic on $R = 5$ for three reasons. First, $R = 5$ is the smallest round count at which the clustering gap becomes large enough to require structural explanation. Second, the MILP-found characteristic at $R = 5$ is short enough for layer-by-layer inspection while already exhibiting the iterative two-round pattern observed at larger round counts. Third, $R = 5$ produces enough empirical matches to analyze intermediate-state differences directly, whereas larger round counts produce fewer matches and therefore provide less direct diagnostic evidence.

8.1. Structure of the $R = 5$ Characteristic

The MILP-found characteristic at $R = 5$ has a striking iterative structure. The intermediate state differences cycle between two patterns after the first round:

$$\Delta^{(0)} \rightarrow \Delta^{(1)} \rightarrow \Delta^{(2)} \rightarrow \Delta^{(3)} \rightarrow \Delta^{(4)} \rightarrow \Delta^{(5)}, \quad (21)$$

with $\Delta^{(1)} = \Delta^{(3)} = \Delta^{(5)} = 0x0000000000800000$ (a single active bit, nibble 5) and $\Delta^{(2)} = \Delta^{(4)} = 0x0000000000900000$ (two active bits within the same nibble). The S-box active in each round is the same nibble (position 5), and the modular-addition layer affects only one of the two adders. This is an iterative two-round characteristic of the form $\Delta_a \xrightarrow{R_1} \Delta_b \xrightarrow{R_2} \Delta_a \xrightarrow{R_1} \Delta_b \xrightarrow{R_2} \Delta_a$, extended by one extra round from the input.

Iterative characteristics are a well-known source of differential clustering in lightweight ciphers [33–35]: alternative trails of similar weight that share the same input–output difference can dominate the differential probability when summed. The diagnostic that follows tests whether the $R = 5$ gap is caused by this phenomenon.

8.2. Layer-by-Layer Exact Verification

The diagnostic first reconstructs the layer-by-layer differential weights of the MILP-found characteristic and compares them against exact analytical and brute-force computations. There are two checks.

8.2.1. S-Box Layer

For each of the five active S-box transitions in the characteristic $(\delta_{\text{in}}, \delta_{\text{out}}) \in \{(0xE, 0x1), (0x8, 0x3), (0x9, 0x1), (0x8, 0x3), (0x9, 0x1)\}$, the GIFT DDT entry and the re-

sulting weight $-\log_2(\text{DDT}/16)$ are computed from the GIFT S-box truth table. The five exact weights are $2.00 + 2.00 + 3.00 + 2.00 + 3.00 = 12.00$, matching the S-box contribution attributed by the MILP exactly.

8.2.2. ARX Layer

For each round, the input and output differences of both 16-bit modular adders are extracted from the characteristic by applying the inverse GIFT permutation to the post-permutation differences and splitting into the four 16-bit words. For each adder, the Lipmaa–Moriai weight is computed analytically from Equation (20); in parallel, the differential probability is measured empirically by sampling 2×10^5 pairs of 16-bit summands per transition. All ten ARX transitions (five rounds $\times$ two adders) are accepted by the Lipmaa–Moriai validity criterion. The sum of analytical weights across the ten transitions is 7.00. The empirical $-\log_2$ probability matches the analytical weight to within 0.02 bits in every case.

The total reconstructed weight is therefore $12.00 + 7.00 = 19.00$, matching the MILP objective exactly. This rules out explanation (i): the MILP encoding of the characteristic is consistent with the exact differential properties of the layers.

8.3. Empirical Differential Probability Under Multiple Key Models

To rule out explanation (ii) and to characterize the magnitude of the clustering, we measure the empirical differential probability for the $R = 5$ input–output difference under three key-handling models: independent random round keys, the real Dipper-64/96 key schedule, and the real Dipper-64/128 key schedule. For each model we draw 2×10^6 plaintext pairs and count the proportion whose ciphertext difference matches the target $\Delta^{(5)} = 0x00000000,00800000$.

Table 18 shows that the empirical $-\log_2$ probability differs by at most 0.27 bits across the three key-handling models. The independence between the random-round-key result and the two real-key-schedule results rules out a key-schedule-induced deviation as the source of the gap. Combined with the layer-by-layer verification of Section 8.2, this rules out both explanation (i) (encoding bug) and explanation (ii) (sampling artifact).

Table 18. Empirical differential probabilities for the $R = 5$ input–output difference under three key-handling models. Each measurement uses 2×10^6 plaintext pairs distributed across 40 keys.

Key Model	Matches	$\hat{p}$	$-\log_2 \hat{p}$
Random round keys	495/2,000,000	2.48×10^{-4}	11.98
Real Dipper-64/96 KS	460/1,600,000	2.30×10^{-4}	12.09
Real Dipper-64/128 KS	444/1,600,000	2.78×10^{-4}	11.82
MILP single-trail bound W		$\leq 1.91 \times 10^{-6}$	≥ 19.00

8.4. Output-Difference Clustering

The remaining explanation is that the empirical differential probability is the sum of contributions from multiple trails sharing the input–output difference. To make this concrete, we count the top-frequency output differences resulting from the $R = 5$ input difference $\Delta^{(0)} = 0x00000000,00E00000$ over 2×10^6 random plaintext pairs with independent random round keys.

The output-difference distribution in Table 19 is strongly localized: the ten highest-frequency output differences all share the same active nibble pattern in the lower half of the state. The top two output differences differ only at bit position 20 (nibble 5, low bit), reflecting the alternation between $0x \dots 800000$ and $0x \dots 900000$ in the iterative characteristic.

This is the empirical signature of differential clustering: many distinct trails feed into a small set of related output differences.

Table 19. Top output differences for the $R = 5$ input difference $\Delta^{(0)} = 0x0000000000E00000$ over 2×10^6 pairs with independent random round keys.

Rank	Output Difference $\Delta^{(5)}$	Count	$-\log_2$
1	0x 00000000 00800000	495	11.98
2	0x 00000000 00900000	365	12.42
3	0x 00000020 00800000	296	12.72
4	0x 00000020 00900000	276	12.82
5	0x 00400020 00900000	178	13.46
6	0x 00400020 00900080	171	13.51
7	0x 00400020 00800000	160	13.61
8	0x 00400000 00900080	152	13.68
9	0x 00400020 00800080	139	13.81
10	0x 00400000 00000000	136	13.84

To further localize the source of the clustering, we record the intermediate state differences at the output of each round for those 495 plaintext pairs whose final difference matches the MILP target. For each of the four interior rounds, only two distinct intermediate state differences appear, in roughly balanced proportions. This direct observation confirms that the empirical probability of 2.48×10^{-4} is the sum of multiple trail probabilities sharing the same input–output difference, exactly as predicted by the iterative characteristic.

8.5. Behavior at Longer Round Counts

To examine how the iterative differential behaves with additional rounds, we measure the empirical probability of selected input–output differences at $R = 6, 7, 8$ under independent random round keys (Table 20). For $R = 6$ and $R = 7$, the target output differences are those of the corresponding MILP-found characteristics; for $R = 8$, the target output difference is the $R = 5$ output difference propagated three additional rounds.

Table 20. Empirical differential probabilities at increasing round counts. Rows $R \in \{6, 7\}$ use the target output difference of the corresponding MILP-found characteristic; row $R = 8$ uses the $R = 5$ output difference. All measurements are under independent random round keys.

R	Matches	Pairs	$-\log_2 \hat{p}$
5	495	2×10^6	11.98
6	72	2×10^6	14.76
7	6	2×10^6	18.35
8	2	6×10^5	18.19

The measured $-\log_2$ probability increases from 11.98 at $R = 5$ to 14.76 at $R = 6$ to 18.35 at $R = 7$, corresponding to a per-round decay of approximately 3.2 bits across this range. The MILP single-trail bound grows at approximately 4.2 bits per round, so the clustering gap continues to widen by approximately one bit per round, although the absolute differential probability still decreases monotonically. The $R = 8$ measurement contains only two matches over 6×10^5 pairs and is therefore too noisy to support a precise per-round decay estimate; a definitive statement about the asymptotic decay of this clustering effect would require substantially larger samples or explicit trail enumeration.

8.6. Conclusions of the Diagnostic

The six checks support the following conclusions.

1. The MILP-found characteristics at $R \leq 7$ are realizable in the real cipher: their layer-by-layer weights sum to exactly the MILP objective.
2. The empirical differential probability is essentially the same under random round keys and under both real Dipper key schedules at $R = 5$, ruling out a key-schedule effect.
3. The characteristics share a common iterative two-round structure ($R_1 \rightarrow R_2 \rightarrow R_1 \rightarrow R_2 \rightarrow \dots$), known to produce multi-trail clustering in similar lightweight ciphers.
4. The empirical output-difference distribution at $R = 5$ is localized in a small set of related values, with intermediate-state differences along matching pairs taking only two distinct values per round.
5. The absolute empirical differential probability of the iterative pattern decreases monotonically across $R = 5, 6, 7$ at a rate of approximately 3.2 bits per round, while the MILP bound grows at approximately 4.2 bits per round; the clustering gap therefore widens at approximately 1.0 bit per round in this range, with the absolute differential probability still strictly decreasing.

We classify the observed anomaly as a structural iterative differential characteristic with multi-trail clustering. The MILP single-trail bound W remains valid as an upper bound on the probability of the best individual trail at each R ; however, the actual differential probability for the specific input–output differences identified by the MILP exceeds 2^{-W} by the contribution of additional trails sharing the same difference.

8.7. Implications for the Full-Round Bound

Two consequences for the full 28-round cipher follow from the diagnostic.

First, the trail-level extrapolation reported at the end of Section 7.2 should be interpreted as a bound on the best single trail, not on the best differential. A tighter differential bound at higher round counts would require either the enumeration of all trails up to a weight threshold, or the computation of the full differential transition matrix of the cipher by dynamic programming. Both methods are computationally heavier than single-trail MILP and are left to future work.

Second, both the MILP bound and the empirical probability decrease across the measured range $R = 5, 6, 7$. The MILP bound grows at approximately 4.2 bits per round and the empirical $-\log_2$ probability of the iterative input–output class grows at approximately 3.2 bits per round, so the clustering gap widens by roughly 1.0 bit per round in this range. A direct linear extrapolation from the $R = 7$ empirical value of 18.35 bits at this 3.2 bits-per-round rate would predict that the differential probability of this specific input–output class reaches the 2^{-64} generic-security level around round $r \approx 7 + (64 - 18.35)/3.2 \approx 21$. We emphasize that this is a conservative linear extrapolation from only three data points and the actual rate may slow further as the iterative pattern weakens with increasing rounds. The full 28-round Dipper remains beyond this round-21 boundary, but the margin is modest under the pessimistic extrapolation, and a tighter differential bound at $R = 28$ should be obtained either by trail enumeration or by the full differential transition matrix as noted above.

We emphasize that this analysis applies only to the specific input differences of the iterative trail family. The investigation of other input differences, the search for differentials with low total weight at high round counts, and a full differential bound for $R = 28$ are open questions that depend on the trail-enumeration techniques noted above.

9. Linear, Impossible-Differential, and Integral Analysis

The differential analysis of Sections 7 and 8 is complemented in this section by three further standard cryptanalytic families: linear trail search, impossible-differential search,

and integral (saturation) distinguishers. All three analyses reuse the modelling infrastructure of Section 7, are released with the paper’s artifacts, and were reproduced independently on a second machine.

9.1. Linear Trail Search

9.1.1. Model

We search for the minimum-weight linear trail over R rounds, where the weight of a trail is $-\log_2 |c|$ for trail correlation c ; a single-trail linear distinguisher requires on the order of $c^{-2} = 2^{2W}$ known plaintexts [36]. The model is a constraint-programming formulation solved with OR-Tools CP-SAT (in contrast to the PuLP/CBC stack of Section 7; CP-SAT’s table constraints and automaton-style encodings suit the linear model, and both solvers prove optimality of their solutions).

The three layers are encoded as follows.

9.1.2. S-Box Layer

The GIFT S-box linear approximation table (LAT) has maximum absolute correlation 2^{-1} (linearity 8); its 133 nonzero entries carry correlation weights in $\{1, 2\}$. Each of the sixteen S-box positions per round is constrained to a valid (input-mask, output-mask, weight) tuple via a table constraint.

9.1.3. Linear Layers

Word rotations and the GIFT-64 bit permutation are permutation matrices, for which linear masks propagate exactly as values do; they are encoded as index re-mappings and contribute no weight. The AddRoundKey layer affects only the sign of the correlation and is omitted, as usual.

9.1.4. Modular Addition

The correlation of a linear approximation of addition modulo 2^{16} with masks (u, v, w) on the two inputs and the output is captured by Wallén’s theorem [37]. We encode it as a three-state automaton processed from the most significant bit to the least significant bit, with states $\{s_0, s_{\text{rej}}, s_{\text{chain}}\}$: s_0 maps symbol 000 to s_0 and 111 to s_{chain} , all other symbols to the rejecting state; s_{chain} maps even-parity symbols $\{000, 011, 101, 110\}$ to itself and odd-parity symbols to s_0 ; each bit step taken from s_{chain} contributes one unit of correlation weight. Before use, the automaton was validated against the exact correlation computed by the carry transfer-matrix method: exhaustively for all 39,062 mask triples of width $n \leq 7$ (zero discrepancies in both validity and weight), and on 2×10^5 random triples at $n = 16$ (zero discrepancies).

9.1.5. Mask Forks

In the modular-addition layer the words B_1 and D_1 each feed both an adder input and the round output (Figure 1). For linear masks a fork is the dual of XOR: the mask entering the fork equals the XOR of the masks on its two uses. This is encoded as an even-parity constraint per bit.

9.1.6. Linear Trail Search Results

Table 21 reports the minimum single-trail correlation weights for $R = 1, \dots, 10$. All ten instances are proven optimal by the solver; wall-clock times on the same machine used for Section 5 range from 0.1 s at $R = 1$ to 41 s at $R = 8$, with the $R = 9$ and $R = 10$ instances closing within minutes.

Table 21. Minimum single linear-trail correlation weights $W = -\log_2 |c|$ for R -round Dipper. A single-trail linear distinguisher requires on the order of 2^{2W} known plaintexts. All values are proven optimal.

R	W	Data $\gtrsim 2^{2W}$
1	1	2^2
2	2	2^4
3	3	2^6
4	6	2^{12}
5	8	2^{16}
6	11	2^{22}
7	13	2^{26}
8	16	2^{32}
9	19	2^{38}
10	21	2^{42}

From $R = 4$ onward the weight grows by 2–3 bits per round. Under a linear extrapolation of this trend, the single-trail correlation falls below 2^{-32} , at which point the data requirement 2^{2W} exceeds the 2^{64} codebook of a 64-bit block, at approximately $R \approx 14$ –15 rounds. The 28-round cipher therefore retains a margin of approximately $2\times$ against single-trail linear distinguishers, consistent with the differential margin of Section 7.

9.1.7. Empirical Validation and Linear Hulls

As in the differential case, we validate the model empirically. For the trails found at $R = 1, \dots, 4$ we measure the correlation of the (input-mask, output-mask) pair on the reduced-round cipher with zero round keys (round keys affect only the sign of the correlation) over up to 2^{22} random plaintexts. The measured correlations and single-trail predictions are summarized in Table 22.

Table 22. Empirical correlation of the solver-found mask pairs versus the single-trail prediction.

R	W	Predicted $ c = 2^{-W}$	Measured $ \hat{c} $	$-\log_2 \hat{c} $
1	1	0.5000	0.5033	0.99
2	2	0.2500	0.2451	2.03
3	3	0.1250	0.1250	3.00
4	6	0.0156	0.0226	5.47

For $R \in \{1, 2, 3\}$ the measured correlation matches the single-trail prediction within sampling noise. At $R = 4$ the measured correlation is slightly stronger than the single-trail value (5.47 versus 6 bits), the signature of a linear hull: several trails share the same endpoint masks and add constructively. This mirrors, in the linear setting, the differential clustering analyzed in Section 8, and the same interpretation applies: the reported single-trail weights are valid bounds on the best individual trail, while the effective hull correlation can be moderately higher. The direction of the deviation is consistent with theory (hull $\geq$ single trail), which further corroborates the correctness of the model.

9.2. Impossible-Differential Search

Impossible differentials [38] exploit input–output difference pairs that occur with probability exactly zero. We search for truncated impossible differentials at nibble granularity: for each of the 16×16 pairs (i, j) , we ask whether some difference activating only input nibble i can propagate to some difference activating only output nibble j after R rounds.

The search is exact rather than heuristic. We build a bit-level feasibility model of differential propagation over R rounds—the GIFT DDT support for the S-box layer, exact bit moves for the linear layers, difference copying at the forks, and the Lipmaa–

Moriai validity condition [8] for the two adders (the same condition already verified exhaustively for $n = 4$ in Section 7.1; the possibility-only form used here was additionally verified against the brute-force additive DDT for all 2^{18} triples at $n = 6$ with zero discrepancies)—and submit it to the CP-SAT solver. Because the differential probability is the sum over all characteristics, an UNSAT verdict proves that no characteristic exists and hence that the truncated differential is impossible; no miss-in-the-middle heuristic or truncation approximation is involved in the soundness of the verdict.

Table 23 shows that the longest truncated impossible differential of this class spans 4 rounds; at 5 rounds every input nibble can reach every output nibble, i.e., the cipher attains full nibble-level differential diffusion. This is consistent with the round-6 avalanche saturation of Section 6.2 and is a direct consequence of the carry-chain diffusion contributed by the ARX layer. To quantify the effect of finer-grained input–output classes, we additionally performed an exhaustive single-bit search: for each of the 64×64 input–output bit pairs, the same bit-level feasibility model was solved at $R = 5$ and $R = 6$, a total of 8192 solver instances completing in approximately 21 minutes on the same machine. At $R = 5$, 72 of the 4096 single-bit pairs are proven impossible; at $R = 6$ the solver exhibits a valid characteristic for every pair, with no undecided instances. The longest single-bit impossible differential therefore spans exactly 5 rounds, one round beyond the nibble-truncated class, and none exists at 6 rounds. The 72 impossible pairs are unevenly distributed across the four input words, reflecting the asymmetry of the half-state ARX layer. Key-dependent sieving in a miss-in-the-middle key recovery could append a small number of outer rounds to such a distinguisher; even under a generous allowance the total length remains far below the 28-round cipher.

Table 23. Number of impossible truncated nibble-to-nibble differentials of R -round Dipper, out of 256 pairs. Every verdict is a solver proof (SAT or UNSAT); no instance timed out.

R	Impossible Pairs
1	232
2	217
3	204
4	26
5	0

9.3. Integral Distinguishers

Integral (saturation) attacks [39,40] trace the propagation of multiset properties. We perform an experimental integral search: a structure of 2^{16} plaintexts saturates one 16-bit word (all values appear exactly once) while the remaining 48 bits are constant; after R rounds the XOR-sum of each output bit over the structure is examined, and a bit whose sum is zero is balanced.

A methodological point deserves emphasis. Testing a single constant assignment substantially overcounts balanced bits, since individual sums are zero by chance with probability $1/2$; a first-pass single-constant scan suggested spurious “distinguishers” several rounds longer than the true ones. We therefore require the balance to hold simultaneously across 32 independent random assignments of the constant bits and independently drawn random round keys (a genuine integral property is key-independent, as the round keys cancel in the XOR-sum over a structure of even size). Only properties stable across all 32 trials are reported. The resulting experimental integral distinguishers are summarized in Table 24.

Table 24. Experimental integral distinguishers from 16-bit word saturation (structures of 2^{16} plaintexts; balance required across 32 random constants and random round keys). “Length” is the largest R with at least one balanced output bit.

Saturated Word	Balanced Bits at Round			Length
	$R-1$	R	$R+1$	R
A (bits 63–48)	38	3	0	5
B (bits 47–32)	58	24	0	3
C (bits 31–16)	39	4	0	5
D (bits 15–0)	31	0	—	3

The longest 16-bit-saturation integral distinguisher spans 5 rounds (words A and C , whose saturation survives one round longer because they enter the adders as the modified operands). Balanced bits collapse rapidly: from full balance at $R = 3$ to a handful at $R = 5$ and none at $R = 6$. We characterize this as an experimental search rather than a proof: a bit-based division-property analysis [41] could certify longer distinguishers from higher-order structures and is left to future work. Even allowing several additional rounds for such extensions, the distinguisher length remains a small fraction of the 28-round cipher.

9.4. Summary of Security Margins

Table 25 collects the longest distinguisher identified by each of the four analysis families. The longest is the 10-round linear trail, whose data requirement of 2^{42} is already substantial; differential, impossible-differential, and integral distinguishers are all shorter. Relative to the longest identified distinguisher, the 28-round cipher retains a margin close to $3 \times (28/10 = 2.8)$, and a margin of $4 \times$ or more against every other family. We reiterate the two standing caveats: the differential and linear figures are single-trail bounds subject to the clustering and hull effects quantified in Sections 8 and 9.1, and the impossible and integral searches are class-restricted as described above.

Table 25. Longest distinguisher per analysis family, with the number of rounds analyzed and the security-relevant figure at that length.

Family	Longest Analyzed	Figure at that Length	Section
Differential (single trail)	7 rounds	$W = 27.42$	Section 7
Linear (single trail)	10 rounds	$W = 21$, data $\gtrsim 2^{42}$	Section 9.1
Impossible differential	5 rounds	longest single-bit class	Section 9.2
Integral (16-bit structure)	5 rounds	3–4 balanced bits	Section 9.3

10. Conclusions and Future Work

We presented Dipper, a 64-bit lightweight block cipher with 96-bit and 128-bit key variants, designed around a hybrid construction that combines the GIFT 4-bit S-box and GIFT-64 bit permutation with a half-state ARX layer of two 16-bit modular additions per round. The construction was motivated by the observation that the compact 4-bit S-boxes used in pure SPN designs and the value-dependent diffusion of modular addition contribute different forms of nonlinearity, and that combining them within a single round may shift the area–security frontier compared to either family alone. Whether the specific combination chosen for Dipper achieves this shift is an empirical question that the present paper answers under a controlled methodology.

The hardware study synthesizes Dipper together with PRESENT-80, PRESENT-128, GIFT-64-128, GIFT-128-128, SIMON-64-128, and SIMON-128-128 under an identical Yosys + ABC + Nangate45 flow. Under this flow, Dipper-64/128 measures 2824 gate equivalents and 30 cycles per block, placing it between GIFT-64-128 (2191 GE) and PRESENT-128 (2963 GE) in static area while achieving a lower area–latency product than both PRESENT-128 and

SIMON-64-128. GIFT-64-128 remains the most compact 64-bit design on both metrics; the cost of the modular-addition layer is 633 GE relative to GIFT-64-128 at the same key size and round count. The methodology produces internally consistent comparisons and is not directly comparable to figures published with the original cipher specifications, which use different libraries and synthesis flows.

The empirical statistical evaluation reports that both Dipper variants match the ideal random-permutation model on the avalanche, SAC, BIC, and NIST SP 800-22 tests at the resolution available with $N = 50,000$ samples. Diffusion saturates at round 6, giving a $4.7\times$ margin to the 28-round count.

The cryptanalytic evaluation establishes, via a Lipmaa–Moriai-based MILP model with full GIFT DDT weighting, that the lowest-weight single differential characteristic of R -round Dipper has weight 2, 5, 9, 13.42, 19, 23, and 27.42 for $R = 1, \dots, 7$ under the standard Markov-cipher assumption. Empirical sampling confirms agreement with the bound to within sampling noise for $R \in \{1, 2, 4\}$ under both random round keys and the real Dipper key schedules. At $R \in \{3, 5, 6, 7\}$ we observe clustering gaps of 1.35, 7.02, 8.24, and 9.07 bits respectively; a dedicated diagnostic identifies these as instances of a single iterative two-round characteristic with multi-trail clustering and verifies that the gap is independent of the key schedule. The per-round trail bound grows at approximately 4.2 bits, placing the extrapolated 28-round trail bound below 2^{-115} .

The extended cryptanalytic evaluation adds proven-optimal linear single-trail bounds up to ten rounds (weight 21, data $\gtrsim 2^{42}$), an exhaustive impossible-differential search showing that the longest nibble-truncated and single-bit impossible differentials span four and five rounds, respectively, and an experimental integral search bounding 16-bit saturation distinguishers at five rounds. The longest distinguisher identified by any of the four families is ten rounds, leaving a margin close to $3\times$ for the 28-round cipher.

The principal results above support the limited claim that Dipper is a publishable design point in the lightweight hybrid SPN–ARX family, positioned between GIFT and PRESENT in gate equivalents at the same key size and below both PRESENT-128 and SIMON-64-128 in the area–latency product. They do not constitute a proof of cryptographic security: the MILP-based bound is a single-trail bound, and a tight differential bound at higher round counts requires additional analysis beyond the scope of this paper.

10.1. Future Work

Three directions extend the present analysis.

10.1.1. Tighter Differential Bounds

The single-trail MILP bound at $R \geq 3$ is not tight against the empirical differential probability. A natural next step is to enumerate all trails up to a fixed weight threshold and sum their probabilities, producing a true differential bound at the cost of significantly higher solver runtime. An alternative is to compute the differential transition matrix of one or two rounds by dynamic programming, from which longer-round differential probabilities can be obtained by matrix exponentiation. Either method would address the gap at $R = 5$ directly.

10.1.2. Library-Aware Hardware Evaluation

The hardware figures of this paper assume a single 45 nm open-source process library without an enable-D flip-flop primitive. A repeat study under a library with native DFFE support (for example, the SkyWater 130 nm or IHP 130 nm open libraries), followed by floor-planning and timing closure for a target frequency, would produce post-layout figures suitable for direct comparison with the historical literature.

10.1.3. Related-Key Analysis and Tighter Hull Bounds

The linear analysis of Section 9 is single-key and single-trail; extending it to full linear-hull enumeration, and extending the differential model with key-difference variables for related-key trails, would complete the standard cryptanalytic battery. Related-key analysis is particularly relevant given the per-variant differences in the Dipper key schedule. A bit-based division-property model would similarly certify or extend the experimental integral distinguishers of Section 9.3.

10.2. Reproducibility

The Verilog source for Dipper and all reference ciphers, the Yosys synthesis scripts, the Python reference implementation, the empirical-test scripts for avalanche, SAC, BIC, and NIST stream generation, the MILP solver source code, and the diagnostic-validator source are released together with the paper (available at <https://github.com/AlexGuseinov/Dipper-lightweight-cipher>, accessed on 15 July 2026).

All measurements reported in Sections 4–9 can be reproduced from the released artifacts on a standard Linux workstation with Yosys 0.33, nextpnr-ecp5 0.6, Project Trellis, Icarus Verilog 11.0, Python 3.10, PuLP, NumPy, OR-Tools (CP-SAT), the COIN-OR CBC solver, GCC (and arm-none-eabi-gcc for the embedded footprints), and the NIST SP 800-22 reference suite sts-2.1.2.

Author Contributions: Conceptualization, A.H. and Y.I.; methodology, A.H. and Y.I.; software, A.H.; validation, A.H., Y.I. and J.A.; formal analysis, A.H.; investigation, A.H.; resources, A.H.; data curation, A.H.; writing—original draft preparation, A.H.; writing—review and editing, A.H., Y.I. and J.A.; visualization, A.H.; supervision, Y.I. and J.A.; project administration, Y.I. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by an internal grant provided by Azerbaijan Technical University for the support and development of scientific research and innovations, Contract No. AzTU-DQL-M01/78447082.

Data Availability Statement: The Verilog RTL, synthesis scripts, Python reference implementations, MILP/CP-SAT models, and validation scripts supporting the reported results are openly available at <https://github.com/AlexGuseinov/Dipper-lightweight-cipher>, accessed on 15 July 2026.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Daemen, J.; Rijmen, V. *The Design of Rijndael: AES—The Advanced Encryption Standard*; Springer: Berlin/Heidelberg, Germany, 2002. [CrossRef]
2. National Institute of Standards and Technology. *Advanced Encryption Standard (AES)*; FIPS Publication 197; NIST: Gaithersburg, MD, USA, 2001. [CrossRef]
3. Moradi, A.; Poschmann, A.; Ling, S.; Paar, C.; Wang, H. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *Proceedings of the Advances in Cryptology—EUROCRYPT, Tallinn, Estonia, 15–19 May 2011*; Lecture Notes in Computer Science; Paterson, K.G., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6632, pp. 69–88. [CrossRef]
4. McKay, K.A.; Bassham, L.; Sönmez Turan, M.; Mouha, N. *Report on Lightweight Cryptography*; NISTIR 8114; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2017. [CrossRef]
5. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.B.; Seurin, Y.; Vikkelsøe, C. PRESENT: An Ultra-Lightweight Block Cipher. In *Proceedings of the Cryptographic Hardware and Embedded Systems—CHES, Vienna, Austria, 10–13 September 2007*; Lecture Notes in Computer Science; Paillier, P., Verbauwhede, I., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4727, pp. 450–466. [CrossRef]
6. Banik, S.; Pandey, S.K.; Peyrin, T.; Sasaki, Y.; Sim, S.M.; Todo, Y. GIFT: A Small Present—Towards Reaching the Limit of Lightweight Encryption. In *Proceedings of the Cryptographic Hardware and Embedded Systems—CHES, Taipei, Taiwan, 25–28 September 2017*; Lecture Notes in Computer Science; Fischer, W., Homma, N., Eds.; Springer: Cham, Switzerland, 2017; Volume 10529, pp. 321–345. [CrossRef]

7. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK Lightweight Block Ciphers. In *Proceedings of the DAC, San Francisco, CA, USA, 7–11 June 2015*; pp. 1–6.
8. Lipmaa, H.; Moriai, S. Efficient Algorithms for Computing Differential Properties of Addition. In *Proceedings of the Fast Software Encryption–FSE, Yokohoma, Japan, 2–4 April 2001*; Lecture Notes in Computer Science; Matsui, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2355, pp. 336–350. [[CrossRef](#)] [[PubMed](#)]
9. Borghoff, J.; Canteaut, A.; Güneysu, T.; Kavun, E.B.; Knezevic, M.; Knudsen, L.R.; Leander, G.; Nikov, V.; Paar, C.; Rechberger, C.; et al. PRINCE–A Low-Latency Block Cipher for Pervasive Computing Applications. In *Proceedings of the Advances in Cryptology–ASIACRYPT, Beijing, China, 2–6 December 2012*; Lecture Notes in Computer Science; Wang, X., Sako, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7658, pp. 208–225. [[CrossRef](#)]
10. 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. In *Proceedings of the Advances in Cryptology–CRYPTO 2016, Part II*; Lecture Notes in Computer Science; Robshaw, M., Katz, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9815, pp. 123–153. [[CrossRef](#)]
11. Wolf, C.; Glaser, J. Yosys: A Free Verilog Synthesis Suite. In *Proceedings of the Austrochip Workshop on Microelectronics 2013, Linz, Austria, 10 October 2013*; pp. 47–52.
12. Mouha, N.; Wang, Q.; Gu, D.; Preneel, B. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In *Proceedings of the Information Security and Cryptology–Inscrypt, Beijing, China, 30 November–3 December 2011*; Lecture Notes in Computer Science; Wu, C.K., Yung, M., Lin, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7537, pp. 57–76. [[CrossRef](#)]
13. Sun, S.; Hu, L.; Wang, P.; Qiao, K.; Ma, X.; Song, L. Automatic Security Evaluation and (Related-Key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers. In *Proceedings of the Advances in Cryptology–ASIACRYPT, Kaohsiung, Taiwan, 7–11 December 2014*; Lecture Notes in Computer Science; Sarkar, P., Iwata, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8873, pp. 158–178. [[CrossRef](#)]
14. Fu, K.; Wang, M.; Guo, Y.; Sun, S.; Hu, L. MILP-Based Automatic Search Algorithms for Differential and Linear Trails for SPECK. In *Proceedings of the Fast Software Encryption–FSE, Bochum, Germany, 20–23 March 2016*; Lecture Notes in Computer Science; Peyrin, T., Ed.; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9783, pp. 268–288. [[CrossRef](#)]
15. Biryukov, A.; Velichkov, V. Automatic Search for Differential Trails in ARX Ciphers. In *Proceedings of the Topics in Cryptology–CT-RSA, San Francisco, CA, USA, 25–28 February 2014*; Lecture Notes in Computer Science; Benaloh, J., Ed.; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8366, pp. 227–250. [[CrossRef](#)]
16. Dinu, D.; Perrin, L.; Udovenko, A.; Velichkov, V.; Großschädl, J.; Biryukov, A. Design Strategies for ARX with Provable Bounds: Sparx and LAX. In *Proceedings of the ASIACRYPT 2016*; LNCS; Springer: Berlin/Heidelberg, Germany, 2016; Volume 10031, pp. 484–513.
17. STB 34.101.31-2011; Information Technology and Security—Data Encryption and Integrity Algorithms (Bel-T). National Standard of Belarus: Minsk, Belarus, 2011.
18. GM/T 0002-2012; Chinese Cryptography Standard; SM4 Block Cipher Algorithm. Office of State Commercial Cryptography Administration: Beijing, China, 2012.
19. Williams, S. Icarus Verilog. Open-Source Verilog Simulator. 2002. Available online: <https://steveicarus.github.io/iverilog/> (accessed on 15 July 2026).
20. Brayton, R.; Mishchenko, A. ABC: An Academic Industrial-Strength Verification Tool. In *Computer Aided Verification (CAV) 2010*; LNCS; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6174. Available online: <https://github.com/berkeley-abc/abc> (accessed on 15 July 2026).
21. Nangate Inc. NangateOpenCellLibrary 45 nm Open Cell Library (FreePDK45 Typical Corner). 2011. Available online <https://si2.org/open-cell-library/> (accessed on 15 July 2026).
22. Guo, J.; Peyrin, T.; Poschmann, A.; Robshaw, M. The LED Block Cipher. In *Proceedings of the CHES, Nara, Japan, 28 September–1 October 2011*; LNCS, Volume 6917, pp. 326–341.
23. Suzuki, T.; Minematsu, K.; Morioka, S.; Kobayashi, E. TWINE: A Lightweight Block Cipher for Multiple Platforms. In *Proceedings of the Selected Areas in Cryptography (SAC 2012), Windsor, ON, Canada, 15–16 August 2012*; LNCS; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7707, pp. 339–354.
24. Banik, S.; Bogdanov, A.; Isobe, T.; Shibutani, K.; Hiwatari, H.; Akishita, T.; Regazzoni, F. Midori: A Block Cipher for Low Energy. In *Proceedings of the Advances in Cryptology–ASIACRYPT 2015, Auckland, New Zealand, 29 November–3 December 2015*; LNCS; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9453, pp. 411–436.
25. Zhang, W.; Bao, Z.; Lin, D.; Rijmen, V.; Yang, B.; Verbauwhede, I. RECTANGLE: A Bit-Slice Lightweight Block Cipher Suitable for Multiple Platforms. *Sci. China Inf. Sci.* **2015**, *58*, 1–15. [[CrossRef](#)]
26. Beierle, C.; Leander, G.; Moradi, A.; Rasoolzadeh, S. CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks. *IACR Trans. Sym. Cryptol.* **2019**, *2019*, 5–45. [[CrossRef](#)]

27. Shibutani, K.; Isobe, T.; Hiwatari, H.; Mitsuda, A.; Akishita, T.; Shirai, T. Piccolo: An Ultra-Lightweight Blockcipher. In *Proceedings of the Cryptographic Hardware and Embedded Systems (CHES 2011), Nara, Japan, 28 September–1 October 2011*; LNCS; Springer, Berlin/Heidelberg, Germany, 2011; Volume 6917, pp. 342–357.
28. Adams, C.M.; Tavares, S.E. The Structured Design of Cryptographically Good S-boxes. *J. Crypto.* **1990**, *3*, 27–41. [[CrossRef](#)]
29. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; NIST Special Publication 800-22 Rev. 1a; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2010. [[CrossRef](#)]
30. Lai, X.; Massey, J.L.; Murphy, S. Markov Ciphers and Differential Cryptanalysis. *Adv. Crypto. EUROCRYPT* **1991**, *547*, 17–38. [[CrossRef](#)] [[PubMed](#)]
31. Mitchell, S.; O’Sullivan, M.; Dunning, I. *PuLP: A Linear Programming Toolkit for Python*; The University of Auckland: Auckland, New Zealand, 2011. Available online: <https://github.com/coin-or/pulp> (accessed on 15 July 2026).
32. Forrest, J.; Ralphs, T.; Vigerske, S.; Santos, H.G.; Hafer, L.; Kristjansson, B.; Fasano, J.P.; Straver, E.; Lubin, M.; Brito, S.; et al. COIN-OR Branch-and-Cut Solver (CBC). Open-Source MILP Solver. 2018. Available online: <https://github.com/coin-or/Cbc> (accessed on 15 July 2026).
33. Wang, M. Differential Cryptanalysis of Reduced-Round PRESENT. In *Proceedings of the Progress in Cryptology—AFRICACRYPT, Casablanca, Morocco, 11–14 June 2008*; Lecture Notes in Computer Science; Vaudenay, S., Ed.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5023, pp. 40–49. [[CrossRef](#)]
34. Knudsen, L.R. Truncated and Higher Order Differentials. In *Proceedings of the Fast Software Encryption—FSE, Leuven, Belgium, 14–16 December 1994*; Lecture Notes in Computer Science; Preneel, B., Ed.; Springer: Berlin/Heidelberg, Germany, 1995; Volume 1008, pp. 196–211. [[CrossRef](#)] [[PubMed](#)]
35. Biham, E.; Shamir, A. Differential Cryptanalysis of DES-like Cryptosystems. *J. Crypto.* **1991**, *4*, 3–72. [[CrossRef](#)]
36. Matsui, M. Linear Cryptanalysis Method for DES Cipher. In *Proceedings of the Advances in Cryptology – EUROCRYPT, Lofthus, Norway, 23–27 May 1993*; Lecture Notes in Computer Science; Helleseeth, T., Ed.; Springer: Berlin/Heidelberg, Germany, 1994; Volume 765, pp. 386–397. [[CrossRef](#)] [[PubMed](#)]
37. Wallén, J. Linear Approximations of Addition Modulo 2^n . In *Proceedings of the Fast Software Encryption (FSE 2003), Lund, Sweden, 24–26 February 2003*; LNCS; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2887, pp. 261–273.
38. Biham, E.; Biryukov, A.; Shamir, A. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In *Proceedings of the Advances in Cryptology—EUROCRYPT ’99, Prague, Czech Republic, 2–6 May 1999*; LNCS; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1592, pp. 12–23.
39. Daemen, J.; Knudsen, L.; Rijmen, V. The Block Cipher Square. In *Proceedings of the Fast Software Encryption (FSE ’97), Haifa, Israel, 20–22 January 1997*; LNCS; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1267, pp. 149–165.
40. Knudsen, L.; Wagner, D. Integral Cryptanalysis. In *Proceedings of the Fast Software Encryption (FSE 2002), Leuven, Belgium, 4–6 February 2002*; LNCS; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2365, pp. 112–127.
41. Todo, Y. Structural Evaluation by Generalized Integral Property. In *Proceedings of the Advances in Cryptology—EUROCRYPT, Sofia, Bulgaria, 26–30 April 2015*; LNCS; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9056, pp. 287–314.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.