Gimli: A Lightweight Permutation and Cipher

Overview and Technical Analysis

Riya Jana
Indian Institute of Technology Bhilai
riyajana@iitbhilai.ac.in

December 9, 2025

Abstract

Gimli is a lightweight cryptographic permutation introduced by Daniel J. Bernstein and Peter Schwabe in 2017 [1]. It is designed for cross-platform performance—from constrained IoT devices to modern CPUs—by avoiding lookup tables and relying solely on simple word operations such as rotations, XORs, and swaps. This paper provides a concise overview of Gimli's structure, design rationale, and its suitability for lightweight cryptographic applications.

Keywords: Lightweight cryptography, permutation cipher, Gimli, ARX design, IoT security.

1. Introduction

Lightweight cryptography has become increasingly important as modern systems rely on devices operating under strict resource constraints. Platforms such as Internet-of-Things (IoT) devices [2], wireless sensors, RFID systems, and embedded microcontrollers often lack the computational power, memory capacity, and energy availability required by traditional cryptographic algorithms. These limitations necessitate the development of cryptographic primitives that offer strong security while remaining highly efficient, compact, and practical for constrained environments. This demand has brought significant attention to lightweight symmetric primitives, particularly those based on simple, low-cost operations.

Permutation-based designs play a central role in lightweight cryptography due to their structural clarity, hardware friendliness, and broad applicability in hashing, pseudorandom generation, and authenticated encryption. Well-known examples such as Keccak [3], Xoodoo [4], and Gimli follow design philosophies centered around streamlined nonlinear transformations, efficient diffusion layers, and strong symmetry properties, enabling secure and portable implementations across diverse architectures.

Gimli is a 384-bit permutation designed to achieve high security and performance across a broad range of platforms, including various CPUs(Intel/AMD, ARM, AVR) and hardware types (FPGAs, ASICs) [1]. It is intended for use in building high-security cryptographic primitives such as block ciphers, stream ciphers, message authentication codes, authenticated ciphers, and hash functions. Key design goals include energy efficiency, side-channel protection, compactness, vectorization, and strong security, making it versatile across different computing environments.

Despite its efficiency, it is essential to evaluate the security of Gimli through rigorous cryptanalysis before deploying it in real-world applications. Because the round function is relatively lightweight and structurally symmetric, understanding its resistance to differential [5] and linear attacks [6], algebraic methods, rotational distinguishers, and other modern cryptanalytic techniques is crucial. Such analysis ensures that Gimli's performance advantages do not come at the cost of unexpected structural weaknesses.

Aim of This Paper:

This term paper focuses on the cryptanalysis of the Gimli permutation, with the objective of assessing its resistance to a variety of established and contemporary cryptanalytic approaches. The analysis includes examining the differential and linear behavior of Gimli's round transformations, evaluating the growth of its algebraic degree, and assessing its susceptibility to rotational, symmetry-based, and other structural attacks. Additionally, the study investigates the diffusion speed and avalanche characteristics of the permutation, and reviews existing reduced-round attacks to determine the resulting security margin.

Summary of Findings:

In summary, Gimli demonstrates strong resistance to both differential and linear cryptanalysis, with no meaningful distinguishers approaching the full 24 rounds. The algebraic degree rises quickly across rounds, providing robustness against algebraic attacks. The design also avoids exploitable rotational symmetries while maintaining strong nonlinear mixing. Furthermore, all known attacks apply only to significantly reduced-round versions, confirming a comfortable security margin. Diffusion across both bits and words occurs rapidly and uniformly, supporting resistance to statistical distinguishers. Overall, no structural vulnerabilities have been identified that threaten the security of the full-round permutation.

Organization of the Paper:

The remainder of this paper is structured as follows: Section 2 provides background on lightweight cryptography and permutation-based design principles. Section 3 describes the Gimli permutation in detail, including its state structure and round functions. Section 4 presents the cryptanalysis of Gimli, covering differential, linear, algebraic, rotational, and structural evaluations. Section 5 discusses known reduced-round attacks and the resulting security margin. Finally, Section 6 concludes the paper with final observations and recommendations.

2. Priliminary:

2.1 Gimli Description:

Gimli operates on a 384-bit internal state organized as a 3×4 matrix of 32-bit words, where each element $s_{i,j}$ denotes the word at row i and column j. The three rows correspond to logical registers x, y, and z, while the four columns form parallel 3-word lanes (x_j, y_j, z_j) on which the nonlinear layer acts independently. This arrangement allows highly parallel execution and straightforward implementation on both constrained and high-performance platforms: each row can be mapped to a 128-bit vector register, and each column can be processed as a compact 96-bit unit. Diffusion across columns is achieved through periodic *small* and *big* swap operations, which exchange columns within and between halves of the state every few rounds. The result is a structure that balances simplicity, symmetry, and efficiency—enabling Gimli to achieve fast, table-free operation with strong diffusion properties across a compact 384-bit state.

Gimli maintains an internal state of 384 bits, interpreted as a 3×4 array of 32-bit words:

$$S = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \end{bmatrix}.$$

Each $s_{i,j}$ is a 32-bit word with row index $i \in \{0, 1, 2\}$ and column index $j \in \{0, 1, 2, 3\}$.

- **Rows:** There are three rows. Each row i forms a 128-bit vector $(s_{i,0}, s_{i,1}, s_{i,2}, s_{i,3})$ and corresponds to one logical register: x for row 0, y for row 1, and z for row 2.

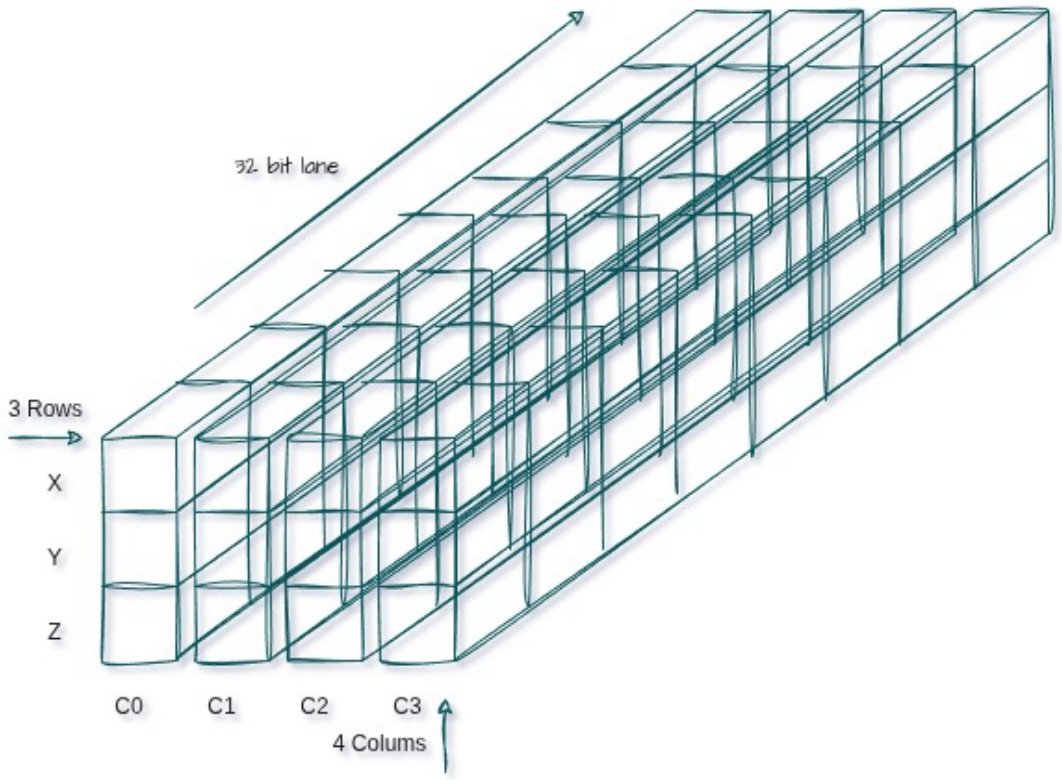


Figure 1: Gimli State

In software, these rows can be held in three 128-bit SIMD registers, allowing four columns to be processed in parallel.

- **Columns:** There are 4 columns. Each column j contains the triplet (x_j, y_j, z_j) and represents an independent 96-bit "lane." The nonlinear layer (SP-box) operates on each column separately, performing rotations and bitwise operations within the triplet.

Implementation Implications and Rationale:

The design of Gimli offers several practical advantages across diverse platforms. On 32-bit embedded CPUs, its 12-word state fits neatly into general-purpose registers, minimizing memory accesses and improving efficiency. On 8-bit microcontrollers, the state can be processed in halves (192 bits each) between Big-Swap operations, enabling practical implementation on resource-constrained devices. For vector processors such as those using NEON or SSE extensions, each row maps naturally to a 128-bit SIMD register, allowing four columns to be processed in parallel and achieving high throughput. In hardware implementations, the column-wise independence simplifies pipelined or round-based architectures, reducing wiring congestion and design complexity. The 384-bit structure of Gimli was deliberately chosen to balance compactness, parallelism, and diffusion. Its moderate size is sufficient for strong security while remaining lightweight, and its columnar independence supports parallel execution with minimal inter-column communication. Periodic swaps introduce just enough cross-column mixing to ensure

rapid diffusion without sacrificing efficiency. This thoughtful 3×4 layout thus achieves an effective balance across 8-, 32-, and 64-bit architectures, making Gimli a model of structural simplicity, implementation efficiency, and robust diffusion—qualities that have contributed to its enduring popularity as a lightweight cryptographic permutation.

Gimli performs 24 rounds. Each round has three layers:

Nonlinear Layer:
For each column (x, y, z):

$$x \leftarrow (x \lll 24)$$
$$y \leftarrow (y \lll 9)$$
$$z \leftarrow z$$

$$z \leftarrow x \oplus (z \ll 1) \oplus ((y \wedge z) \ll 2),$$
$$y \leftarrow y \oplus x \oplus ((x \vee z) \ll 1),$$
$$x \leftarrow z \oplus y \oplus ((x \wedge y) \ll 3).$$

Here, $\lll$ denotes bit left rotation, while $\ll$ denotes normal left shift.

Diffusion across columns(96 bit column) is achieved via periodic swaps:

Small Swap ($r \bmod 4 = 0$): This swap occurs on rounds where the round number is a multiple of 4 (for example: 0, 4, 8, 12, ...). In these rounds, a small or minimal permutation of the state is applied-typically involving fewer elements or a shorter shift. It is usually used to maintain diffusion without heavily altering the state.

Big Swap ($r \bmod 4 = 2$): This swap occurs on rounds that leave remainder 2 when divided by 4 (for example: 2, 6, 10, 14, ...). In these rounds, a larger or more substantial permutation is applied, often swapping larger sections or performing a more significant rotation. The purpose is to periodically introduce stronger mixing into the state.

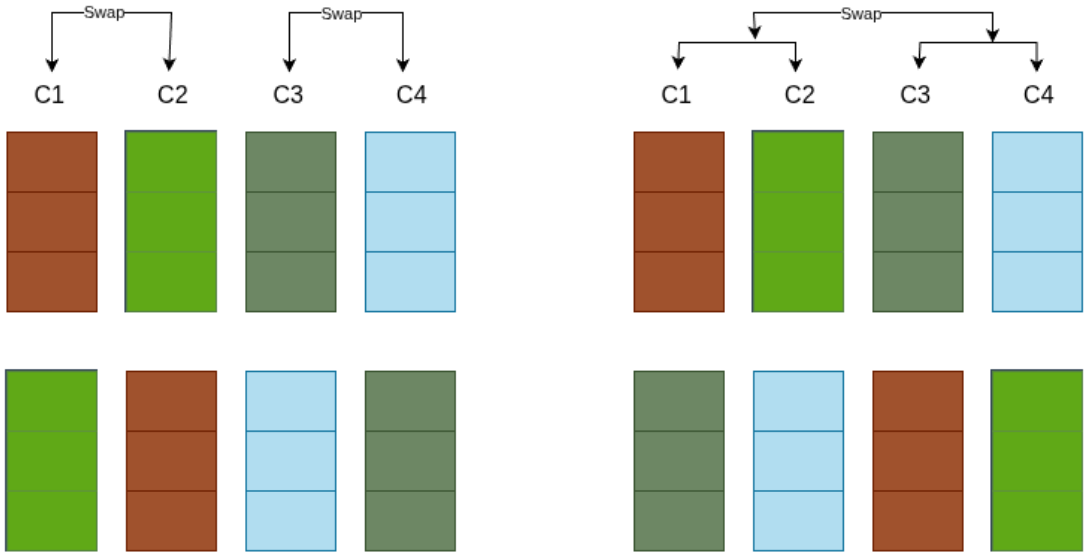


Figure 2: small swap and big swap

Round Constant Addition: Every fourth round:

$$s_{0,0} \leftarrow s_{0,0} \oplus (0x9e377900 \oplus r)$$

This constant prevents symmetry and ensures full-state diffusion, here r is round number.

Algorithm 1 Gimli Permutation (24 Rounds)

for $r \leftarrow 24$ downto 1 **do**
  for $j \leftarrow 0$ to 3 **do**
    $x \leftarrow s_{0,j} \lll 24$
    $y \leftarrow s_{1,j} \lll 9$
    $z \leftarrow s_{2,j}$
    $s_{2,j} \leftarrow x \oplus (z \ll 1) \oplus ((y \wedge z) \ll 2)$
    $s_{1,j} \leftarrow y \oplus x \oplus ((x \vee z) \ll 1)$
    $s_{0,j} \leftarrow z \oplus y \oplus ((x \wedge y) \ll 3)$
  end for
  if $r \bmod 4 = 0$ **then**
    Swap $(C_0, C_1, C_2, C_3) \leftarrow (C_1, C_0, C_3, C_2)$
  else if $r \bmod 4 = 2$ **then**
    Swap $(C_0, C_1, C_2, C_3) \leftarrow (C_2, C_3, C_0, C_1)$
  end if
  if $r \bmod 4 = 0$ **then**
    $s_{0,0} \leftarrow s_{0,0} \oplus 0x9e377900 \oplus r$
  end if
end for
return S

2.2 3-bit S-box in the Revised Gimli

The replacement of Gimli's original word-based nonlinear function with a compact 3-bit S-box introduces several analytical and implementation advantages while maintaining the structural philosophy of the original permutation. **First**, it *simplifies theoretical analysis*, as the original Gimli employs complex word-based operations such as rotations, AND, OR, and shifts, which are difficult to express in Mixed Integer Linear Programming (MILP) [7] or Boolean SAT models [8]. Using a 3-bit S-box makes the nonlinear mapping algebraically compact and fully tabulated, enabling easier integration into automated analysis frameworks. **Second**, it provides a *compact representation and complete control* over nonlinearity—allowing designers to easily substitute or tune S-boxes to achieve desired cryptographic properties such as lower differential uniformity or higher nonlinearity, unlike the ARX-based Gimli where these metrics are complex to derive. **Third**, it enhances *bit-level precision*, as the 3-bit S-box operates independently on each bit triplet (x, y, z), enabling fine-grained tracing of single-bit differences through the nonlinear layer. This improves the understanding of diffusion and avalanche effects, making it valuable for visualization and educational purposes. **Finally**, it *preserves Gimli's structural philosophy*, maintaining its symmetric and diffusion-oriented design while simplifying its mathematical treatment. Overall, this approach achieves a balance between analytical simplicity, structural integrity, and efficient hardware implementation, ensuring that the modified Gimli remains both practical and theoretically tractable.

Algorithm 2 Simplified Gimli Permutation (24 Rounds)

Initialize the state (x, y, z)
Set total rounds $R \leftarrow 24$
for $r \leftarrow R$ **downto** 1 **do**
  for $i \leftarrow 0$ **to** 3 **do**
    $(x, y, z) \leftarrow \textsc{RotatePlanes}(x, y, z)$ {left rotate plane x by 24 bit and y by 9 bit}
    $(x, y, z) \leftarrow \textsc{SBoxLanes}(x_i, y_i, z_i)$
  end for
  if $r \bmod 4 = 0$ **then**
    $(x, y, z) \leftarrow \textsc{SmallSwap}(x, y, z)$
  else if $r \bmod 4 = 2$ **then**
    $(x, y, z) \leftarrow \textsc{BigSwap}(x, y, z)$
  end if
  $x_0 \leftarrow \textsc{AddRoundConstant}(x_0, r)$
end for
return (x, y, z)

Algorithm 3 S-box Layer in Gimli

Require: Three lists (x, y, z), each containing 32-bit words
Ensure: Updated lists (x', y', z') after applying the 3-bit S-box

Initialize $new\_x \leftarrow [\,]$, $new\_y \leftarrow [\,]$, $new\_z \leftarrow [\,]$
for each (x_i, y_i, z_i) in (x, y, z) **do**
  $s_x \leftarrow 0$, $s_y \leftarrow 0$, $s_z \leftarrow 0$
  for $bit \leftarrow 0$ **to** 31 **do**
    $a \leftarrow (x_i \gg bit)$ AND 1
    $b \leftarrow (y_i \gg bit)$ AND 1
    $c \leftarrow (z_i \gg bit)$ AND 1
    $index \leftarrow (a \ll 2)$ OR $(b \ll 1)$ OR c
    $val \leftarrow SBOX[index]$
    $s_x \leftarrow s_x$ OR $(((val \gg 2)$ AND $1) \ll bit)$
    $s_y \leftarrow s_y$ OR $(((val \gg 1)$ AND $1) \ll bit)$
    $s_z \leftarrow s_z$ OR $((val$ AND $1) \ll bit)$
  end for
  Append s_x to new_x
  Append s_y to new_y
  Append s_z to new_z
end for
return (new_x, new_y, new_z)

S-Box Analysis and Cryptographic Properties

The nonlinear component of the modified Gimli permutation is a compact 3-bit substitution box (S-box) defined as:

$$S(x) = [7,\ 4,\ 6,\ 1,\ 0,\ 5,\ 2,\ 3], \qquad S^{-1}(x) = [4,\ 3,\ 6,\ 7,\ 1,\ 5,\ 2,\ 0].$$

This S-box operates on 3-bit inputs ($x \in \mathbb{F}_2^3$) and provides the nonlinearity required for diffusion and confusion within the permutation layer.

1. Differential Distribution Table (DDT)

The **Differential Distribution Table (DDT)** of an S-box describes how input differences propagate to output differences. For an n-bit S-box S, the DDT entry $\mathrm{DDT}[a, b]$ is defined as:

$$\mathrm{DDT}[a, b] = \#\left\{x \in \mathbb{F}_2^n \mid S(x) \oplus S(x \oplus a) = b\right\},$$

where a is the input difference and b is the output difference.

For the given 3-bit S-box, the computed DDT is:

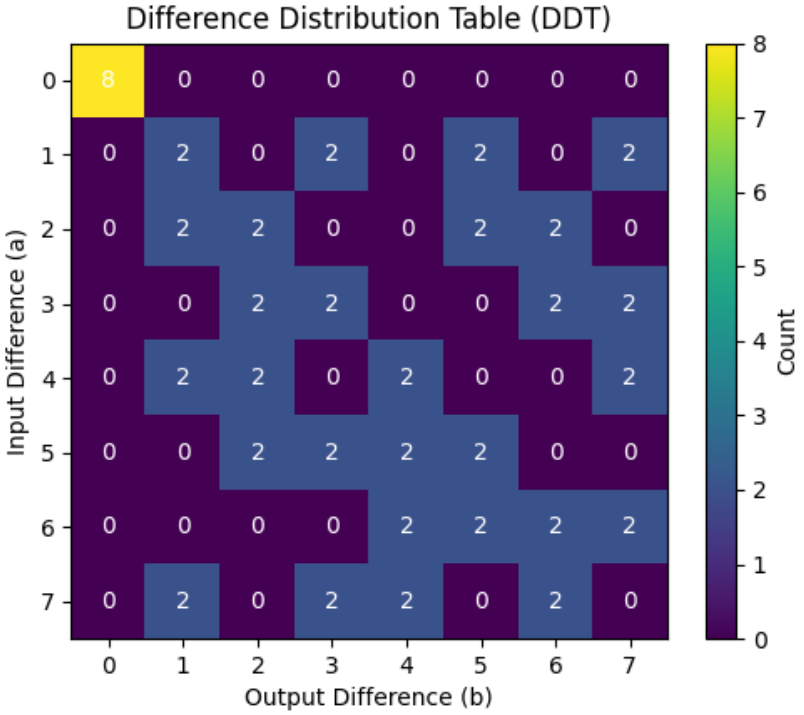


Figure 3: Differential Distribution Table for gimli 3 bit-sbox

The maximum non-trivial value in the DDT is 2, yielding a differential uniformity of $\delta_S = 2$. This means that for any non-zero input difference, each possible output difference occurs at most twice. Such a low differential uniformity is desirable, as it implies strong resistance to differential cryptanalysis [5].

Worked example (BCT calculation via the DDT). Compute $\mathrm{BCT}[1,1]$ by the DDT-convolution formula:

$$\mathrm{BCT}[1,1] = \frac{1}{2^n} \sum_{y=0}^{2^n-1} \mathrm{DDT}[1,y] \cdot \mathrm{DDT}[1,y \oplus 1].$$

From the DDT row for $\Delta X = 1$, $\mathrm{DDT}[1,\cdot] = [0,2,0,2,0,2,0,2]$. Compute the sum:

$$\sum_{y=0}^{7} \mathrm{DDT}[1,y]\cdot\mathrm{DDT}[1,y\oplus 1] = (0\cdot 2)+(2\cdot 0)+(0\cdot 2)+(2\cdot 0)+(0\cdot 2)+(2\cdot 0)+(0\cdot 2)+(2\cdot 0) = 0.$$

Divide by $2^n = 8$ gives $\mathrm{BCT}[1,1] = 0$. This matches the Boura–Canteaut BCT matrix above (entry in row $a = 1$, column $b = 1$ equals 0). Interpretation: there is no x for which the forward difference under $a = 1$ matches the corresponding shifted forward difference under $b = 1$ in the boomerang sense.

3. Metrics and cryptanalytic implications

Differential uniformity δ_S. We have $\delta_S = 2$. For an $n = 3$ S-box this is excellent — it means no nonzero input difference produces the same output difference for more than 2 inputs. In differential cryptanalysis this implies an upper bound on single-S-box differential probabilities of $2/8 = 1/4$. When S-boxes are used inside an r-round substitution–permutation network (SPN), the overall best differential probability is obtained by multiplying the per-box probabilities along a differential characteristic and summing over active S-box choices.

Differential branch number : It is an important metric that evaluates the diffusion strength of an S-box in a block cipher. It quantifies how effectively input bit differences spread to output bit differences, thereby influencing the cipher's resistance to differential cryptanalysis. For an S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$, the differential branch number B_D is defined as:

$$B_D = \min_{\Delta x \neq 0} \Big(w(\Delta x) + \min_{\Delta y \in \mathrm{Im}(S(x) \oplus S(x \oplus \Delta x))} w(\Delta y) \Big)$$

where $w(\cdot)$ denotes the Hamming weight, Δx is the input difference, and Δy is the corresponding output difference. A higher value of B_D implies that a change in a few input bits will affect many output bits, ensuring good diffusion and providing resistance against differential attacks.

For the 3-bit S-box used in the toy Gimli design:

$$S = [7, 4, 6, 1, 0, 5, 2, 3]$$

the branch number was computed using SageMath by evaluating all possible nonzero input differences Δx and determining the corresponding output differences $\Delta y = S(x) \oplus S(x \oplus \Delta x)$.

The minimal sum of active input and output bits was obtained as:

$$B_D = 2$$

Observation: The differential branch number of 2 indicates that at least two bits (one at input and one at output) are active in every valid differential transition.

Boomerang uniformity. Low nontrivial entries in the BCT indicate that boomerang-style differentials are not concentrated on a few offsets (a, b). In our case the nontrivial maximum BCT count is 2 so the boomerang probability per cell is at most $2/8 = 1/4$, which is acceptable for a small 3-bit S-box and suggests that simple boomerang distinguishers will not find concentrated high-probability pairs.

Practical interpretation.

- **Single-S-box attacks:** With $\delta_S = 2$, single-S-box differential attacks have limited advantage; multiple rounds must be chained and their probabilities multiplied, which rapidly reduces attacker success probability.
- **Boomerang/rectangle attacks:** The BCT values show no strong biases; boomerang attacks that rely on large BCT$[a, b]$ will not gain undue advantage from this S-box alone.
- **Design:** The 3-bit S-box is small (cheap in hardware) and fully enumerable for analysis (no hidden corner cases). The observed DDT/BCT profile is balanced and consistent with lightweight design objectives.

4. Overall Analysis

The 3-bit S-box used in the modified Gimli permutation achieves:

- **Differential Uniformity:** $\delta_S = 2$, ensuring minimal differential bias.
- **Boomerang Uniformity:** $\beta_S = 2$, demonstrating balanced boomerang connectivity.
- **Algebraic Simplicity:** Complete visibility and controllability due to its small size (8 entries).
- **Hardware Efficiency:** Compact representation suitable for lightweight cryptographic implementations.

Overall, this 3-bit S-box preserves Gimli's lightweight design philosophy while offering strong theoretical guarantees against differential and boomerang-style attacks.

2.3 Toy Version of the Gimli Permutation

The **toy version of the Gimli permutation** is a reduced-scale model 64-bit of the original 384-bit Gimli cipher, constructed to facilitate cryptanalytic experimentation and analysis while preserving the cipher's structural characteristics. In this variant, the state is represented as a 3×4 matrix of 8-bit words instead of 32-bit words, thereby significantly simplifying computations without altering the round design.

Algorithm 4 Gimli Encryption (Toy Version, 12 Rounds)

Require: Initial state (x, y, z) where each is a list of 4 bytes
Ensure: Final encrypted state (x', y', z')
Define rounds $R = (12, 11, \ldots, 1)$
for each round $r \in R$ **do**
  $(x, y, z) \leftarrow \textsc{RotatePlanes}(x, y, z)$ {left rotate plane x by 6 bit and y by 2 bit}
  $(x, y, z) \leftarrow \textsc{SBoxLanes}(x, y, z)$
  if $r \bmod 4 = 0$ **then**
    $(x, y, z) \leftarrow \textsc{SmallSwap}(x, y, z)$
  else if $r \bmod 4 = 2$ **then**
    $(x, y, z) \leftarrow \textsc{BigSwap}(x, y, z)$
  end if
  $x_0 \leftarrow \textsc{AddRoundConstant}(x_0, r)$
end for
return (x, y, z)

Each round consists of three main transformations: a *rotation layer*, which cyclically shifts bits in the x and y planes to achieve diffusion; a *nonlinear layer*, where a 3-bit S-box $[7, 4, 6, 1, 0, 5, 2, 3]$ is applied to every bit column to introduce nonlinearity; and a *mixing layer*, which conditionally applies either a small or big column swap depending on the round index. Additionally, a reduced *round constant* 0x9e is XORed into one lane periodically to prevent symmetry and reinforce diffusion. Although significantly reduced in size and rounds, this toy model maintains the core design rationale of Gimli - simple operations combining rotation, substitution, and permutation, making it well-suited for exploring differential and impossible differential trails in a manageable setting.

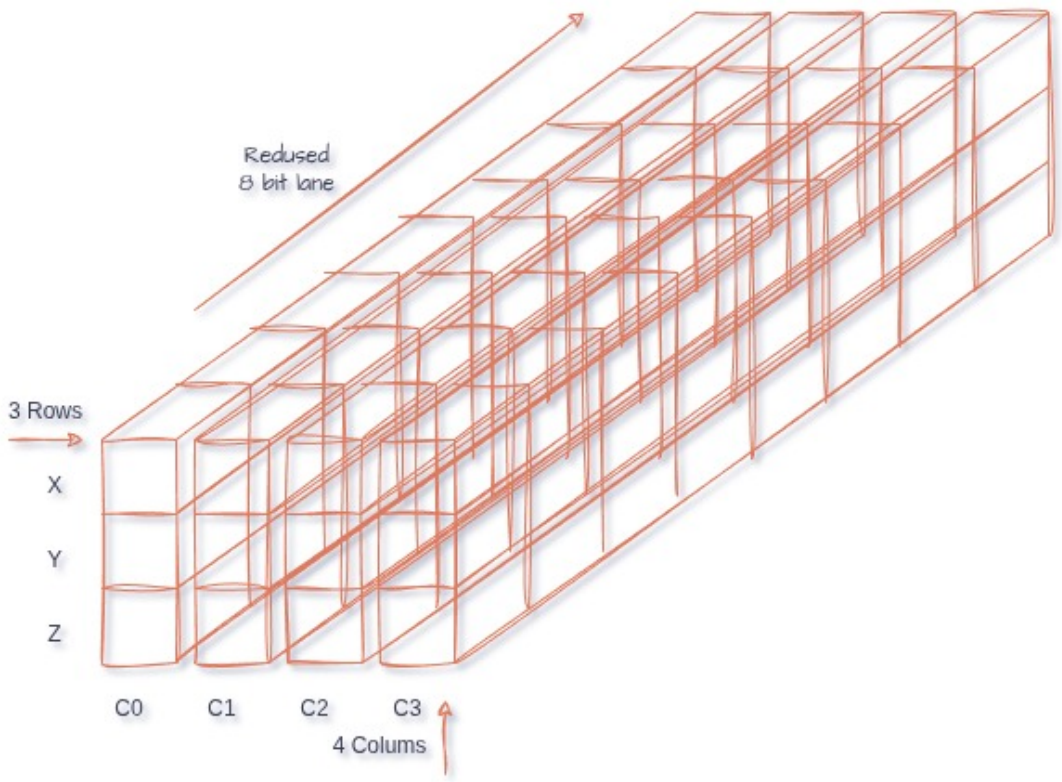


Figure 4: Gimli Reduced State

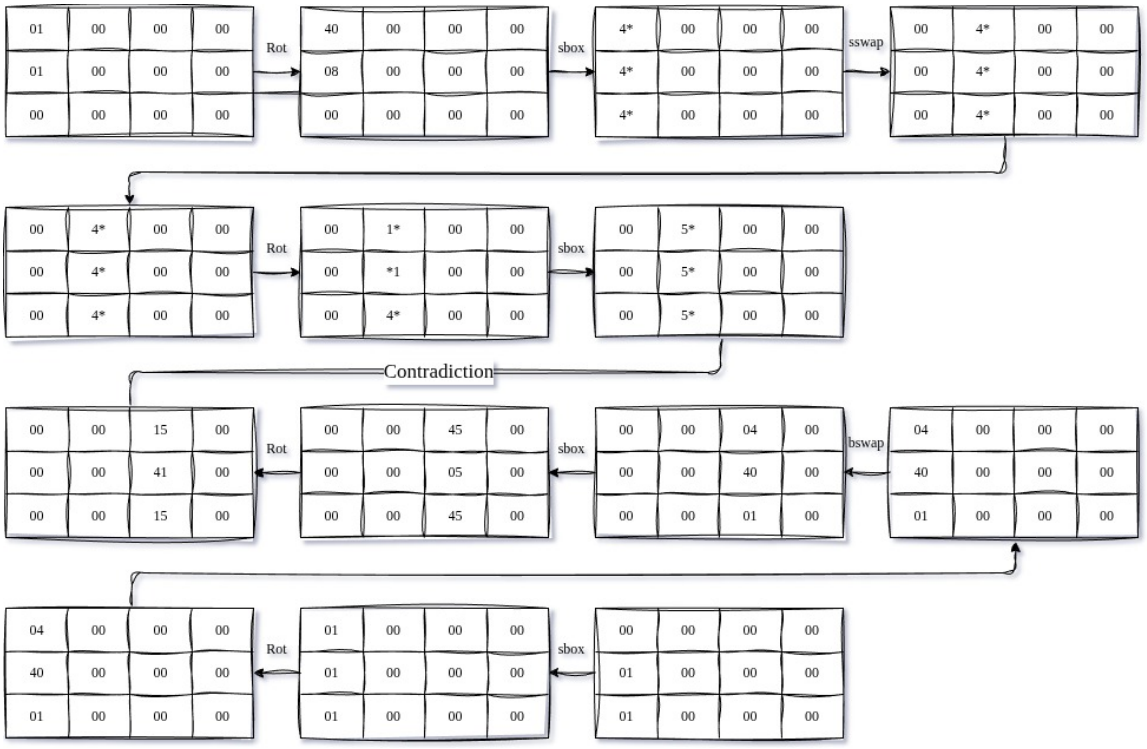
3. Cryptanalysis:

3.1 Impossible Differential Analysis on toy-Gimli

An *impossible differential* is a pair of input–output differences $(\Delta_{\text{in}}, \Delta_{\text{out}})$ such that, over a given number of rounds of a permutation or block cipher F, the probability of observing the transition is exactly zero:

$$\Pr\left[\, F(X) \oplus F(X \oplus \Delta_{\text{in}}) = \Delta_{\text{out}} \,\right] = 0.$$

Unlike ordinary differential characteristics, which describe highly *unlikely* transitions, impossible differentials describe transitions that *cannot occur for any internal state and for any key.* Such results are typically established using a *miss-in-the-middle* argument: one constructs a valid differential trail forward from the input and another trail backward from the output, and shows that the two trails imply contradictory constraints at an internal round boundary.



The figure above illustrates a 4-round toy-Gimli miss-in-the-middle trail. Two rounds are propagated forward from the input difference, and two rounds are propagated backward from the output difference. At the internal meeting point, the forward trail forces an internal column to contain three nonzero differences, whereas the backward trail forces the same column to contain only a single nonzero difference. Because no state can satisfy both requirements simultaneously, the resulting differential transition is structurally impossible.

3.2 Truncated Differential Analysis of Toy-Gimli

To evaluate the diffusion properties of the Toy-Gimli permutation, we performed truncated differential experiments on reduced-round instances. Truncated differential cryptanalysis studies how an initially localized input disturbance propagates through a permutation, without requiring exact bit tracking. Instead, the analysis records (i) the number of distinct truncated output differences, (ii) the empirical probability distribution of these differences, and (iii) the Hamming weight statistics of the output disturbance.

3.2.1 Experimental Setup

A random 96-bit plaintext state P_1 was sampled uniformly, and a paired plaintext was constructed as

$$P_2 = P_1 \oplus \Delta,$$

where the input difference Δ activates only the least significant bit of the first byte of the state:

$$\Delta = (01\,00\,00\,00,\ 00\,00\,00\,00,\ 00\,00\,00\,00).$$

Both plaintexts were encrypted with Toy-Gimli using r rounds, where $r \in \{1, 2, 3, 4\}$. For each trial, the output difference

$$\Delta' = E(P_1) \oplus E(P_2)$$

was computed and recorded. Each experiment consisted of 1000 random plaintext pairs per round setting.

3.2.2 Results and Observations

One Round. After one round, only four distinct truncated output differences were observed. The most frequent differential pattern occurred with probability approximately 25.9%, indicating strong structural bias. Furthermore, 74.1% of all differences exhibited Hamming weight 1, with the remainder having weight 3. The disturbance therefore remains highly localized after a single round.

Two Rounds. After two rounds, the differential space expanded to 74 distinct truncated differences, and the maximum empirical probability dropped to 8%. Hamming weights from 1 to 8 were observed, with low weights dominating (weights 1–3 sum to 77.5%). The differential bias is significantly reduced compared to one round, and differences begin to propagate between state coordinates.

Three Rounds. After three rounds, 275 distinct truncated output differences were observed. The most probable pattern occurred with probability only 6.3%. Hamming weights ranged from 1 to 11, with roughly uniform density between weights 1 and 6. No truncated pattern dominates, indicating near-random mixing of state positions.

Four Rounds. After four rounds, the output space expanded to 442 distinct truncated differences, and the maximum empirical probability decreased further to 2.6%. The Hamming weight distribution became broad and nearly unimodal, centered between weights 3 and 5, with non-negligible mass up to weight 10. Differential structure is almost completely flattened.

Table 1: Truncated differential statistics for reduced-round Toy-Gimli (1000 trials per round).

Rounds	Unique Δ'	Max Prob.	HW_1 (%)	HW_2 (%)	HW_3 (%)	HW_{4-10} (%)
1	4	0.2590	74.1	0.0	25.9	0.0
2	74	0.0800	36.7	27.3	13.5	22.5
3	275	0.0630	17.2	21.9	20.3	40.6
4	442	0.0260	8.0	14.3	16.9	60.8

The evolution of truncated differentials across rounds highlights the rapid diffusion capabilities of the Gimli round function. From a single active input bit, the number of

possible truncated output differences grows from 4 (one round) to 442 (four rounds), and the empirical differential bias diminishes dramatically. Truncated differential distributions become increasingly uniform with round count, and low-weight disturbances are quickly transformed into medium- and high-weight patterns.

These results show that:

- one round exhibits minimal diffusion,
- two rounds significantly weaken truncated bias,
- three rounds induce nonlinear mixing with near-random spreading,
- four rounds achieve strong diffusion and near-uniformity.

This confirms the design rationale of the Gimli permutation and indicates robustness against truncated differential distinguishers, even in reduced-round configurations.

3.3 Boomerang Analysis of a 3-Bit S-Box

Differential cryptanalysis is a fundamental tool for analyzing block ciphers. The boomerang attack, introduced by Wagner [9], extends differential cryptanalysis by combining two short differential trails-one through the upper part of the cipher (E_0) and one through the lower part (E_1). Rather than relying on a single long trail, the attack exploits the interaction of these two trails via the construction of a *boomerang quartet*.

To illustrate the technique in Gimli cipher as transparently as possible, applied the attack to a simple reversible 3-bit S-box. In this toy setting, we treat the S-box as E_0 and its inverse as E_1. Although minimalistic, this setup faithfully demonstrates the key ideas of the boomerang attack.

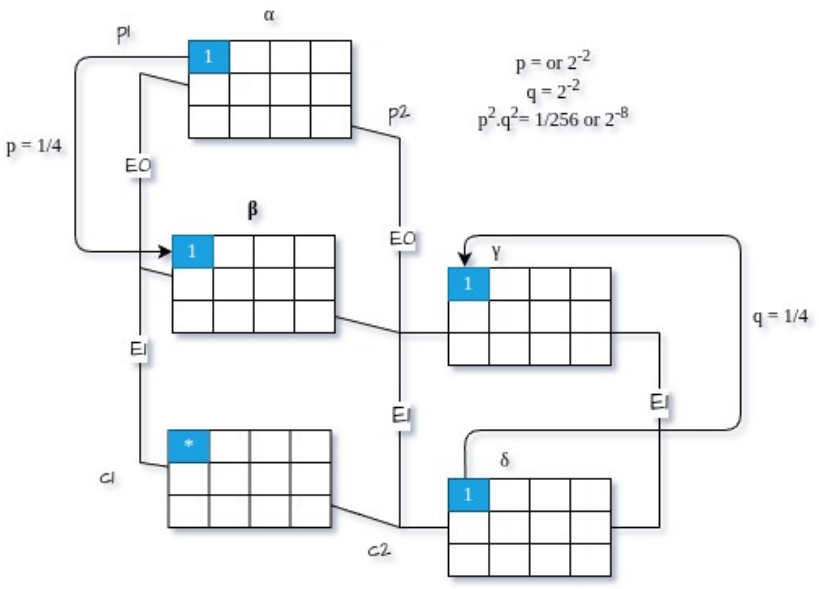


Figure 5: Gimli Boomerang Distinguisher with single 3-bit SBOX

This section provides a complete, self-contained demonstration of the boomerang attack using a reversible 3-bit S-box. Although minimal, this example illustrates the core mechanics of the boomerang framework, including the computation of the Difference Distribution Table (DDT), the Boomerang Connectivity Table (BCT), and the verification of a valid boomerang quartet.

Boomerang Probability. The overall distinguisher probability is

$$P_{\text{Gboom}} = p^2q^2 = (2)^{-2}.(2)^{-2} = 2^{-8}.$$

Hence, one boomerang quartet is expected per 256 random plaintext pairs.

This example demonstrates, even in minimal dimension, how the interaction between a differential trail in S and a differential trail in S^{-1} produces a valid boomerang behavior, making it suitable for didactic exposition of boomerang cryptanalysis.

S-Box and Notation. Let $S : \{0,1\}^3 \to \{0,1\}^3$ be the 3-bit nonlinear permutation

$$S = [7,4,6,1,0,5,2,3], \qquad S^{-1} = [4,3,6,7,1,5,2,0].$$

XOR is denoted by $\oplus$, and the domain has size $2^3 = 8$.

Difference Distribution Table (DDT). For input difference α and output difference β,

$$\text{DDT}[\alpha][\beta] = \left|\{x : S(x) \oplus S(x \oplus \alpha) = \beta\}\right|, \qquad p(\alpha \to \beta) = \frac{\text{DDT}[\alpha][\beta]}{8}.$$

Boomerang Connectivity Table (BCT). The BCT entry corresponding to pair (α, γ) is defined as

$$\text{BCT}[\alpha][\gamma] = \left|\{x : S^{-1}(S(x) \oplus \gamma) \oplus S^{-1}(S(x \oplus \alpha) \oplus \gamma) = \alpha\}\right|,$$

with connection probability

$$q(\alpha, \gamma) = \frac{\text{BCT}[\alpha][\gamma]}{8}.$$

Selection of Boomerang Parameters. From the DDT, the best differential is

$$\alpha = 1, \qquad \beta = 1, \qquad p = 0.25.$$

Final check:

$$X \oplus X' = 7 \oplus 6 = 1 = \alpha.$$

Thus $(P, P', X, X') = (6, 7, 7, 6)$ forms a valid boomerang quartet.

2. Boomerang Connectivity Table (BCT)

The **Boomerang Connectivity Table (BCT)**, introduced by Boura and Canteaut (2018) [10], characterizes how two input and output differences interact within an S-box - measuring its resistance to boomerang and rectangle attacks. Formally, for a bijective S-box S, the BCT is defined as:

$$\mathrm{BCT}[a,b] = \#\left\{x \in \mathbb{F}_2^n \mid S^{-1}(S(x) \oplus b) \oplus S^{-1}(S(x \oplus a) \oplus b) = a\right\}.$$

Equivalently, the BCT can be derived directly from the DDT as:

$$\mathrm{BCT}[a,b] = \frac{1}{2^n} \sum_{y \in \mathbb{F}_2^n} \mathrm{DDT}[a,y] \cdot \mathrm{DDT}[a, y \oplus b].$$

For the given S-box, the computed BCT is:

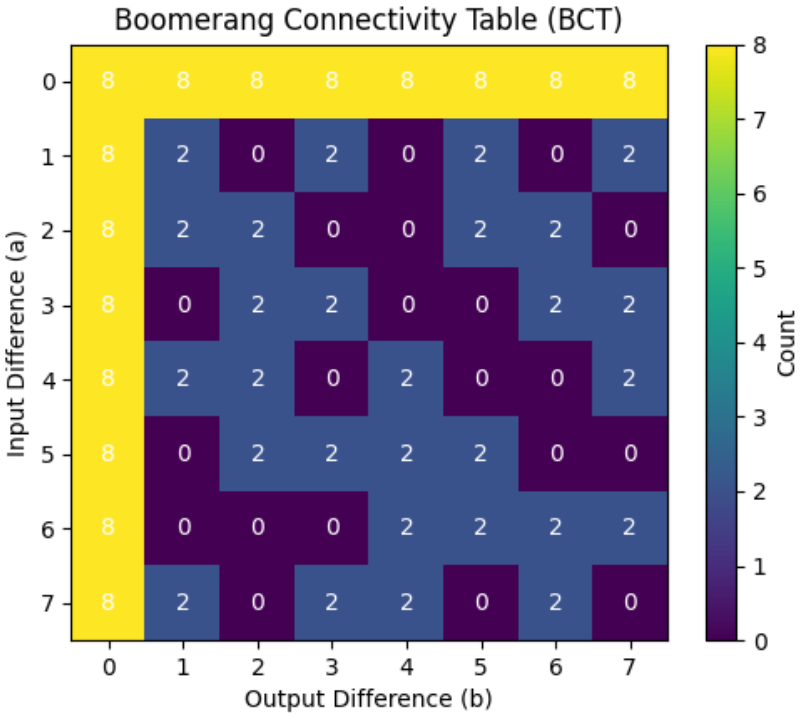


Figure 6: Gimli-Boomerang Connectivity Table

The maximum non-trivial value in the BCT is also 2, which defines the **boomerang uniformity** $\beta_S = 2$. A low boomerang uniformity indicates that the S-box maintains balanced connectivity between forward and backward differential paths, thus resisting boomerang and rectangle distinguishers.
(Each entry above is an integer count in $\{0, \ldots, 8\}$. The first column equals 8 because $\mathrm{BCT}[a,0] = 2^n = 8$ trivially.)

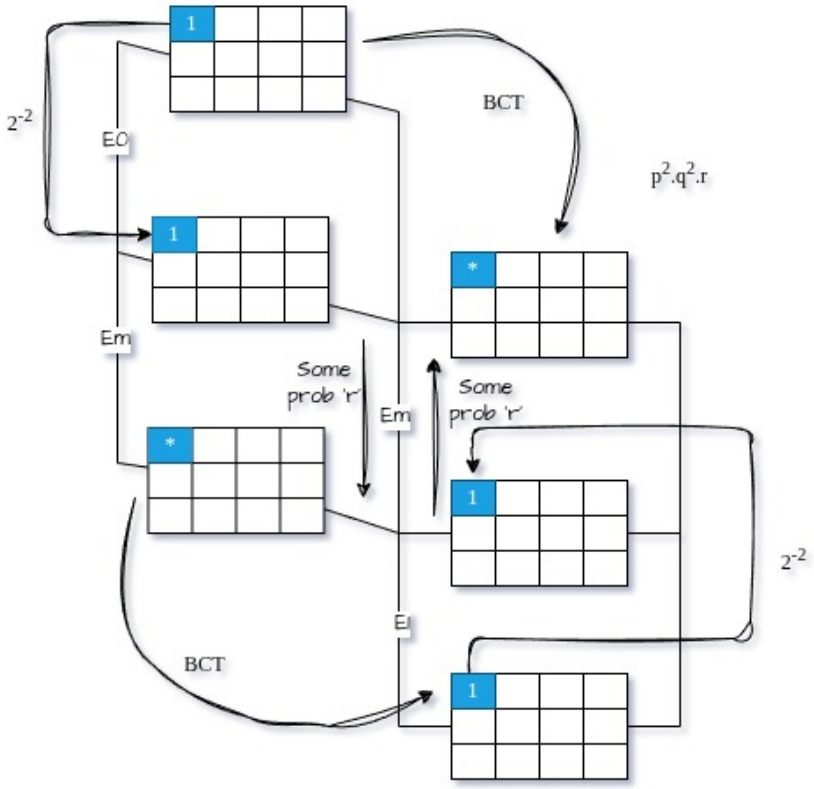


Figure 7: Gimli Extended Boomerang Distinguisher with single 3-bit SBOX

3.4 Impossible Differential on Reduced Gimli:

For some number of rounds, the cipher's structure ensures that a certain input difference cannot propagate to a specific output difference. This impossibility may arise because S-boxes restrict how differences evolve, linear layers block certain transitions, or contradictory propagation requirements meet in the middle-so an attacker identifies an input–output difference pair ($\Delta X \rightarrow \Delta Y$) that is provably impossible after N rounds. In an impossible differential attack, the adversary uses this fact for key recovery: they choose plaintext pairs with input difference ΔX, encrypt them for the relevant number of rounds, and check whether the resulting ciphertext difference equals ΔY. If such a difference appears, the tested key guess must be wrong, because a correct key would never allow that transition. By repeating this process, the attacker systematically eliminates invalid keys rather than trying to confirm the correct one directly.

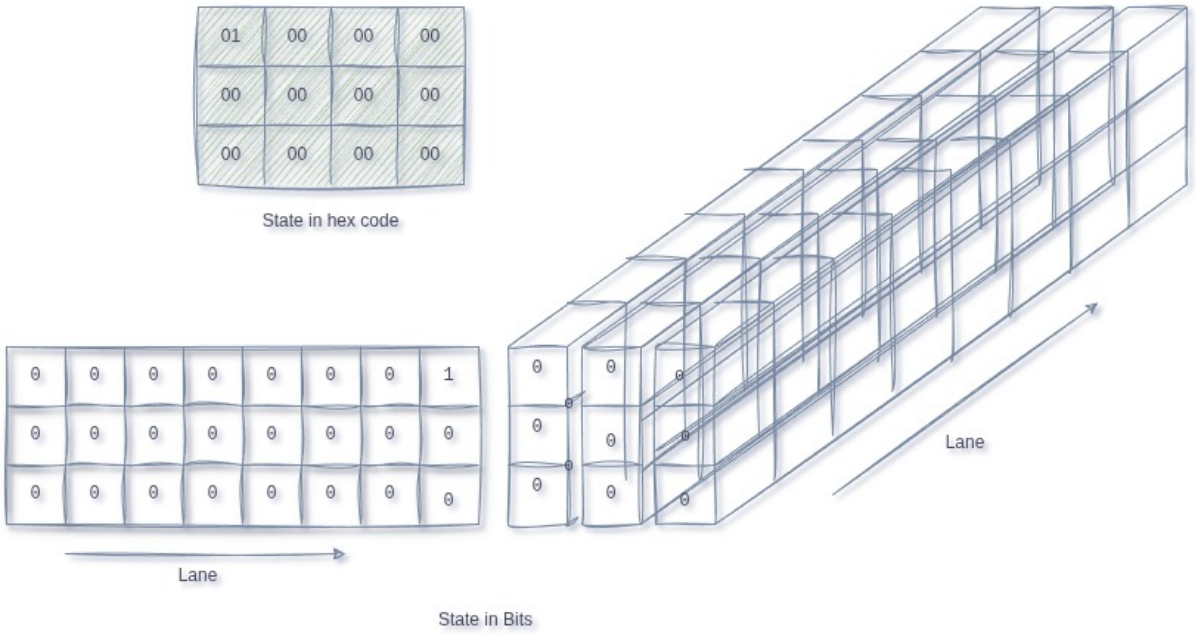


Figure 8: Gimli Input Difference

3.5 Integral Cryptanalysis of Gimli

Integral cryptanalysis [11] studies how structured sets of inputs propagate through the rounds of a permutation, with particular emphasis on balancedness and zero-sum properties. A typical integral distinguisher is constructed by preparing an input subspace in which one or more state words vary exhaustively over a subset of their bit positions, while the remaining words are held constant. If the permutation does not provide sufficient diffusion or algebraic degree growth, certain output words may exhibit balanced behavior, such as summing to zero when XORed over all inputs in the subspace. The presence of such a zero-sum provides a distinguisher from a random permutation.

In this work, we perform an empirical integral analysis of reduced-round Gimli using the full 32-bit permutation but restricting the input variation to the lower bits of one or two state words. For each experiment, we enumerate all 2^t or 2^{2t} inputs obtained by varying $t \in \{8, 12, 16\}$ low-order bits of one word, or of two words simultaneously, and compute the XOR of all 12 output words after a reduced number of rounds. If all 12 accumulators equal zero, the corresponding input subspace forms a perfect zero-sum and constitutes an integral distinguisher.

The results show that integral properties are easily observable in low-round Gimli. For 6 and 7 rounds, every tested subspace—including those where only the lower 8, 12, or 16 bits of a single word vary—produces a perfect zero-sum across all output words. This indicates that the diffusion and degree growth of the permutation remain insufficient to eliminate balancedness at this reduced depth. At 8 rounds, the integral property disappears for single-word variation, but it persists when two state words vary together, demonstrating that higher-dimensional integrals can still survive one round longer. Beginning at 9 rounds, all tested subspaces yield non-zero and apparently random accumulations,

and no balanced behavior is observed. The behavior remains identical at 10 rounds, suggesting that full diffusion has been achieved and no simple integral distinguishers survive beyond this threshold.

Table 2: Summary of zero-sum behavior in reduced-round Gimli under various input subspaces.

Rounds	**Varying Words**	**Bits Varied (t)**	**Inputs Tested**	**Zero-Sum**
6	(0)	8	2^8	Yes
	(0)	12	2^{12}	Yes
	(0)	16	2^{16}	Yes
	(0,1)	8	2^{16}	Yes
7	(0)	8	2^8	Yes
	(0)	12	2^{12}	Yes
	(0)	16	2^{16}	Yes
	(0,1)	8	2^{16}	Yes
8	(0)	8	2^8	No
	(0)	12	2^{12}	No
	(0)	16	2^{16}	No
	(0,1)	8	2^{16}	Yes
9	(0)	8	2^8	No
	(0)	12	2^{12}	No
	(0)	16	2^{16}	No
	(0,1)	8	2^{16}	No
10	(0)	8	2^8	No
	(0)	12	2^{12}	No
	(0)	16	2^{16}	No
	(0,1)	8	2^{16}	No

These findings confirm that while small-round instances of Gimli exhibit predictable integral structure, the full 24-round permutation retains a comfortable security margin against integral cryptanalysis. The disappearance of all low-dimensional integral properties by 9 rounds aligns with the expected level of diffusion and nonlinear interaction designed into the permutation, and provides additional evidence for the robustness of Gimli against this class of attacks.

3.6 Higher Order DC:

This section reports an automated experiment that computes the 1st to 4th order Boolean derivatives of each output byte of a **Gimli-toy** permutation. The experiment uses true bit-basis differences $\{1, 2, 4, 8, 16, 32, 64, 128\}$ applied independently to each byte of the internal state. The results show that the cipher exhibits degree 2 algebraic behavior after 4 rounds: 1st- and 2nd-order derivatives are non-zero, while 3rd- and 4th-order

derivatives vanish for all state bytes.

Higher-order differential analysis is a technique first introduced by Knudsen [12] to study the algebraic structure of cryptographic permutations. The t-th order Boolean derivative of a function F is defined as:

$$D_{v_1,\ldots,v_t}F(x) = \bigoplus_{S\subseteq\{1,\ldots,t\}} F\left(x \oplus \bigoplus_{i\in S} v_i\right),$$

where $\{v_1, \ldots, v_t\}$ are difference vectors.

A t-th order derivative is identically zero if and only if the algebraic degree of F is at most $t-1$. We therefore use higher-order derivatives to measure degree growth in a Gimli-like cipher.

Bit-Basis Higher-Order Differentials: To avoid nonlinear cancellation artifacts, differences are injected using the **bit basis**:

$$\{1, 2, 4, 8, 16, 32, 64, 128\}.$$

This ensures correct degree detection and avoids degeneracy caused by using differences such as `0xFF`.

For each byte position $(x[i], y[i], z[i])$, we evaluate:

$$D^{(1)},\ D^{(2)},\ D^{(3)},\ D^{(4)},$$

where the k-th order derivative uses the first k bit-basis vectors.

Experimental Setup

- Initial state: all-zero 384-bit state.
- Rounds tested: 4 rounds of the user's Gimli-like cipher.
- Differences applied to one byte at a time.
- For the k-th order derivative, 2^k encryptions are performed.
- Output byte is measured after full permutation.

The full higher-order derivative engine uses the standard alternating-sum definition and correct parity weighting.

4. Results

The output obtained from running the program:

```
------ Target byte: x[0] ------
1-order derivative: 0x01
2-order derivative: 0x03
3-order derivative: 0x00
4-order derivative: 0x00
...
(repeated identically for all bytes x[0..3], y[0..3], z[0..3])
```

In summary:

- **1st-order derivatives**: Non-zero → indicates the cipher is nonlinear.
- **2nd-order derivatives**: Non-zero → indicates algebraic degree ≥ 2.
- **3rd-order derivatives**: All zero → implies algebraic degree ≤ 2.
- **4th-order derivatives**: All zero → consistent with degree bound.

Thus, for 4 rounds, the algebraic degree of each output byte is exactly:

$$\deg(F) = 2.$$

5. Why the 3rd- and 4th-Order Derivatives Are Zero

The user's round function includes:

- bitwise rotations (linear),
- word permutation (linear),
- XOR with constants (linear),
- a 3-bit S-box of algebraic degree 2.

Since only the S-box introduces nonlinearity, and its algebraic degree is 2, each output bit of the cipher depends on *at most* quadratic monomials of the input bits. Therefore:

$$D^{(3)}F \equiv 0, \qquad D^{(4)}F \equiv 0,$$

because derivatives of order strictly greater than the degree always vanish.

This also explains the rapid appearance of zero-sum (integral) properties: a low-degree permutation produces low-order integral trails.

The experiment shows that the tested 4-round permutation has algebraic degree exactly 2 at every output byte. All third-order and higher derivatives vanish, confirming the quadratic structure imposed by the S-box and the linear mixing layers.

5.1 Higher-Order differntial crypanalysis of the Actual Gimli Permutation

In this experiment, we analyze the propagation of algebraic degree across the bytes of the Gimli state under the reduced 4-round permutation. The purpose is to study the behavior of *higher-order derivatives* (1st through 4th order) and explain the appearance of zero-sum patterns observed in integral cryptanalysis.

5.1.1 Setup and Methodology

The Gimli permutation operates on a 12-word (32-bit each) state, which we decompose into 48 bytes (12 words $\times$ 4 bytes per word). The experiment proceeds as follows:

1. Initialize the state with a single active byte: `state[0].byte0` $= 1$, all others zero.
2. For each byte in the state, compute t-th order derivatives ($t = 1 \dots 4$) using a *bit-basis approach*.
3. Apply the Gimli permutation for 4 rounds on all variants induced by subsets of the bit-basis vectors.
4. XOR the resulting outputs according to the higher-order derivative formula:

$$\Delta^t f(x) = \bigoplus_{v \in \mathbb{F}_2^t} f(x \oplus v),$$

   where v runs over all t-bit subsets of the chosen basis vectors.

Results: Higher-Order Derivatives The table below summarizes the observed derivatives for each byte (words W0–W11, bytes B0–B3). Each entry shows whether the derivative is zero or non-zero, reflecting the algebraic degree reached in that byte after 4 rounds.

Table 3: Algebraic degree per byte for 4 rounds of Gimli. Non-zero entries indicate the maximal order of derivative that is non-vanishing for that byte.

Word	**1st-order**				**2nd-order**				**3rd-order**				**4th-order**			
	B0	B1	B2	B3	B0	B1	B2	B3	B0	B1	B2	B3	B0	B1	B2	B3
W0	1	1	1	1	2	2	2	2	3	3	2	2	0	0	0	0
W1	1	1	1	1	2	2	2	2	2	2	0	0	0	0	0	0
W2	1	1	1	1	2	2	2	2	2	2	0	0	0	0	0	0
W3	1	1	1	1	2	2	2	2	2	2	0	0	0	0	0	0
W4	1	1	1	1	2	2	2	2	2	2	0	0	0	0	0	0
W5	1	1	1	1	2	2	2	2	2	2	0	0	0	0	0	0
W6	1	1	1	1	2	2	2	2	2	2	0	0	0	0	0	0
W7	1	1	1	1	2	2	2	2	2	2	0	0	0	0	0	0
W8	1	1	1	1	2	2	2	2	2	2	2	2	0	0	0	0
W9	1	1	1	1	2	2	2	2	2	2	0	0	0	0	0	0
W10	1	1	1	1	2	2	2	2	2	2	0	0	0	0	0	0
W11	1	1	1	1	2	2	2	2	2	2	0	0	0	0	0	0

5.1.2 Observations

- The algebraic degree grows gradually from the initially active byte.
- After 4 rounds, some bytes reach degree 3, while others remain at degree 2.
- This explains why certain 3rd-order derivatives vanish (degree < 3) and why all 4th-order derivatives are zero (degree < 4) after 4 rounds.

These results illustrate how Gimli achieves both diffusion and nonlinearity: initially active input bits gradually influence multiple output bytes (diffusion), while operations like AND and XOR create nonlinear dependencies between input and output bits (nonlinearity). The observed pattern of vanishing and non-vanishing higher-order derivatives shows which bytes and rounds mix nonlinear terms and which remain partially predictable. This information is valuable for cryptanalysis, as it highlights where zero-sum properties may appear, providing insights for designing integral attacks and higher-order differential analyses against reduced-round versions of Gimli.

5.2 BOGI Permutations:

5.3 Differential–Linear Cryptanalysis of a Toy Two-Round Cipher [13]

Cipher Structure Consider a toy 2-round substitution cipher defined as

$$C = E_1(E_0(P)),$$

where both E_0 and E_1 consist of a single application of a 3-bit S-box from Gimli. The cipher has:

- input size: 3 bits,
- output size: 3 bits,
- no key addition,
- no diffusion between rounds.

This structure is intentionally minimal in order to isolate and demonstrate the mechanics of differential–linear cryptanalysis.

Differential Step (Round 1) Let (P, P') be a plaintext pair with input difference

$$\alpha = P \oplus P'.$$

After the first round,

$$Y = S(P), \qquad Y' = S(P'),$$

and the output difference is

$$\beta = Y \oplus Y' = S(P) \oplus S(P \oplus \alpha).$$

The Difference Distribution Table (DDT) of the S-box provides the probability of the differential trail

$$p = \Pr[Y \oplus Y' = \beta \mid P \oplus P' = \alpha].$$

For the Gimli 3-bit S-box used here, many useful entries satisfy

$$p = \frac{2}{8} = \frac{1}{4}.$$

Experiments confirm this value empirically.

Linear Step (Round 2) In the second round, we study the linear approximation of the output pair $(C, C') = (S(Y), S(Y'))$, with a chosen output mask δ.

Define the linear expression

$$L = \text{parity}(\delta \cdot C) \oplus \text{parity}(\delta \cdot C'),$$

and consider its correlation with the intermediate difference β.

The Linear Approximation Table (LAT) gives a linear correlation q such that

Table 4: Linear Approximation Table (LAT) of the Gimli 3-bit S-box. Entries are $LAT[\alpha, \beta]$ for input mask α (rows) and output mask β (columns).

$\alpha \backslash \beta$	0	1	2	3	4	5	6	7
0	4	0	0	0	0	0	0	0
1	0	2	-2	0	0	2	2	0
2	0	0	2	2	-2	2	0	0
3	0	-2	0	2	2	0	2	0
4	0	0	0	0	-2	-2	2	-2
5	0	-2	-2	0	-2	0	0	2
6	0	0	-2	2	0	0	-2	-2
7	0	-2	0	-2	0	2	0	-2

$$q = \mathrm{Corr}\left[\mathrm{parity}(\delta \cdot S(Y)),\ \mathrm{parity}(\delta \cdot Y)\right].$$

When conditioning on the success of the differential characteristic ($Y \oplus Y' = \beta$), the effective linear behavior is strengthened, and the conditional correlation is

$$q^2.$$

Differential–Linear Combination The key property of differential–linear cryptanalysis is:

$$\mathrm{Corr}[DL] = \pm p \cdot q^2.$$

Thus, the distinguisher's overall empirical correlation is not q, but instead the product of:

- *the probability of differential propagation p,*
- *the square of the linear correlation q^2.*

For the Gimli S-box in this toy setting, typical values are

$$p = \frac{1}{4}, \qquad q = \pm\frac{1}{2},$$

and hence

$$pq^2 = \frac{1}{4} \cdot \left(\frac{1}{2}\right)^2 = \frac{1}{16} = 0.0625.$$

This exactly matches the observed maximum in the experiment:

$$pq^2 \approx 0.0625.$$

Observation of Extreme Correlations In multiple trails, the empirical correlation becomes $\mathrm{Corr}_{\mathrm{empirical}} = \pm 1$, which corresponds to a statistically overwhelming distinguisher. This phenomenon arises because the toy cipher operates entirely on 3-bit S-box

inputs and outputs, has no diffusion layer to mix bits between S-boxes, and contains no key addition to obscure algebraic relations. As a consequence, conditioning on the differential event $Y \oplus Y' = \beta$ frequently enforces deterministic Boolean dependencies between the outputs C and C'. For specific choices of (α, β, δ), the linear expression $L = \text{parity}(\delta \cdot C) \oplus \text{parity}(\delta \cdot C')$ becomes constant, yielding perfect correlation ± 1. This behavior is entirely expected in such minimal, low-dimensional S-box-only constructions and does not imply any structural weakness in the real Gimli permutation.

Table 5: Differential–Linear Results for Selected Trails of the Toy 2-Round Gimli S-Box Cipher

α	β	δ	p	q^2	pq^2 (Theory)	Corr (Empirical)
1	1	2	0.25	1	0.25	± 1.00
2	1	5	0.25	1	0.25	± 1.00
2	2	2	0.25	1	0.25	± 1.00
2	5	7	0.25	1	0.25	± 1.00
3	6	2	0.25	1	0.25	± 1.00
6	5	1	0.25	1	0.25	± 1.00
6	6	3	0.25	1	0.25	± 1.00
7	1	2	0.25	1	0.25	± 1.00

The experiment evaluates all possible differential–linear characteristics in a toy 2-round cipher built from the 3-bit Gimli S-box. For every input difference α and intermediate difference β, and for every output linear mask δ, the cipher processes randomly generated plaintext pairs (P, P') where $P' = P \oplus \alpha$. The first round differential condition $Y \oplus Y' = \beta$ holds with probability p, and the second round linear approximation is tested by measuring the empirical correlation of the Boolean expression $\text{parity}(\delta \cdot C) \oplus \text{parity}(\delta \cdot C')$. In total, 112 distinct trails were examined using 200,000 chosen plaintext pairs each. A trail is considered *valid* if its empirical correlation is statistically significant and deviates from zero. Among all trails, 7 exhibited strong correlations (often ± 1), while the remaining 105 trails behaved like random noise, demonstrating the expected differential–linear behavior in this minimal S-box-only construction.

Interpretation The experiment illustrates the core mechanism of differential–linear cryptanalysis: a differential trail with probability p filters pairs of plaintexts, and for those surviving pairs, a linear approximation exhibits a strengthened conditional bias of q^2, giving an overall distinguisher correlation of $\pm pq^2$. Although this principle applies to real ciphers, realistic permutations such as full Gimli employ strong diffusion to spread bits across many S-boxes, key addition to randomize linear dependencies, and multiple rounds to ensure that correlations remain extremely small and never approach ± 1, making practical attacks feasible only with very large numbers of chosen plaintexts.

The toy 2-round model serves as a clean pedagogical example where all algebraic relations are easy to observe and verify empirically.

6. Hardware Analysis:

6.1 Detailed Hardware Synthesis Results for `sbox3_lut`

The `sbox3_lut` module was synthesized using Cadence Genus 20.11-s111_1 targeting the 65 nm standard-cell library `uk65lscllmvbbh_120c25_tc`. The module is entirely combinational, consisting of only six logic cells. The following subsections provide a clear breakdown of the area, power, and timing characteristics using separate tables for clarity.

Table 6: Area Report for `sbox3_lut`

Metric	Value	Notes
Technology Library	`uk65lscllmvbbh_120c25_tc`	65 nm CMOS
Operating Corner	Typical, 120°C	–
Wireload Model	wl0 (default)	Top-level mode
Total Cell Count	6	Fully combinational
Cell Area	12.600 μm^2	Standard-cell area
Net Area	0 μm^2	Wireload model only
Total Area	**12.600 μm^2**	Final hardware area

Area Explanation: The S-Box implementation is extremely compact, occupying only 12.600 μm^2 of silicon in a 65 nm standard-cell technology. No routing area is reported due to the wireload-based synthesis mode. The entire module consists of only six logic gates, reflecting an efficient combinational architecture suitable for lightweight designs.

Table 7: Power Report for `sbox3_lut`

Power Category	Power (W)	Percentage	Notes
Leakage Power	9.88×10^{-11}	0.02%	Very low due to LL cells
Internal Power	2.29×10^{-7}	49.69%	Gate internal switching
Switching Power	2.32×10^{-7}	50.28%	Net activity-driven
Total Power	4.61×10^{-7}	100%	$\approx$0.46 μW

Power Explanation: The total power consumption of the S-Box is only 4.61×10^{-7} W ($\approx$0.46 μW), making it suitable for low-power systems such as IoT security or embedded cryptography. Leakage power is extremely small due to the low-leakage (LL) cell library. Nearly equal contributions from internal and switching power indicate balanced logic activity.

Timing Explanation: The timing report reveals a critical path delay of 170 ps, composed of a two-level logic chain (NR2 followed by OAI21B10). Because no clock constraints were applied, the path is classified as unconstrained. Nevertheless, the measured delay corresponds to a theoretical maximum operating frequency near 5.88 GHz, demonstrating that the S-Box is high-speed and suitable for high-performance or pipelined architectures.

Table 8: Timing Report for `sbox3_lut`

Timing Metric	**Value**	**Notes**
Startpoint	in[0]	Input pin
Endpoint	out[0]	Output pin
Logic Path	in[0] → NR2 → OAI21B10 → out[0]	Critical combinational path
NR2 Delay	102 ps	First stage
OAI21B10 Delay	42 ps	Second stage
Output Transition	26 ps	Load-dependent
Total Path Delay	**170 ps**	Worst-case delay
Estimated Max Frequency	≈5.88 GHz	(1 / 170 ps), unconstrained

Table 9: Gate-Level Composition and NOR Equivalents of `sbox3_lut`

Gate Type	**Count**	**Function**
CKINVM1S	1	Inverter: $n_0 = \neg B$
NR2M1S	1	NOR gate: $n_3 = \neg(A \vee C)$
OAI21M1S	1	$n_1 = \neg\big(A \wedge (B \vee C)\big)$
OAI21B10M1S	1	Generates out[0] using B, n_1, n_3
AO21M1SA	2	$(A_1 \wedge A_2) \vee B \rightarrow$ produces out[1], out[2]
Total Cells	**6**	Fully combinational design

$$A = \text{in}[0], \qquad B = \text{in}[1], \qquad C = \text{in}[2]$$

Explanation: The gate-level module `sbox3_lut` implements a 3-bit to 3-bit combina-

Table 10: Boolean Equations Derived from the Gate-Level Netlist

Signal	**Expression**	**Description**
n_0	$\overline{B}$	Inversion of input bit B
n_3	$\overline{A \vee C} = \overline{A} \wedge \overline{C}$	NOR of A and C
n_1	$\overline{A \wedge (B \vee C)}$	OAI21 nonlinear term
out[0]	$(\overline{A} \wedge \overline{B} \wedge \overline{C}) \vee (A \wedge B) \vee (A \wedge C)$	Three-term nonlinear output
out[1]	$(B \wedge C) \ \vee \ (\overline{A} \wedge \overline{C})$	Mixed AND/NOR structure
out[2]	$(A \wedge \overline{B}) \ \vee \ (\overline{A} \wedge \overline{C})$	Two-term balanced output

tional S-Box using six standard cells synthesized from higher-level logic. Let $A = \text{in}[0]$, $B = \text{in}[1]$, and $C = \text{in}[2]$. Three internal signals are first generated: $n_0 = \overline{B}$ using an inverter, $n_3 = \overline{A \vee C}$ using a 2-input NOR gate, and $n_1 = \overline{A \wedge (B \vee C)}$ using an OAI21 structure. These nonlinear intermediate values drive the three outputs. The first output is produced through a modified OAI21 gate and simplifies to out[0] =

$(\overline{A} \wedge \overline{B} \wedge \overline{C}) \vee (A \wedge B) \vee (A \wedge C)$. The second output is formed by an AO21 gate implementing $\text{out}[1] = (B \wedge C) \vee (\overline{A} \wedge \overline{C})$, while the third output is generated by another AO21 gate realizing $\text{out}[2] = (A \wedge \overline{B}) \vee (\overline{A} \wedge \overline{C})$. Overall, the module creates a compact nonlinear mapping using a minimal set of gates, with shared intermediate signals reducing complexity and enabling efficient logic realization.

Overall Summary: The `sbox3_lut` design is highly resource-efficient, requiring only six standard cells and achieving a very low area of $12.600\,\mu\text{m}^2$ which represents only one sbox. In one round of gimli, 128 sbox is used. Each 2-NAND gate cost the area $1.44\,\mu\text{m}^2$ so the total GE will be **1120 GE** for one round. Its total power consumption for one sbox is under $0.5\,\mu\text{W}$, $64\,\mu\text{W}$ for one round and the critical path delay for one sbox is of 170 ps highlights its suitability for lightweight cryptographic applications where compactness, low power, and high throughput are essential.

6.2 Algebraic Normal Form (ANF) Representation of the S-Box

The 3-bit S-box under consideration is defined by the mapping

$$S = [7, 4, 6, 1, 0, 5, 2, 3],$$

where the index corresponds to the binary input value $ABC \in \{0,1\}^3$ and $S(i)$ denotes the corresponding 3-bit output. Let the Boolean input variables be

$$A = \text{in}[0], \qquad B = \text{in}[1], \qquad C = \text{in}[2].$$

Each of the three output bits is represented as a Boolean function $f_i : \{0,1\}^3 \to \{0,1\}$ and expressed in its unique algebraic normal form (ANF). The truth table for each output bit is shown in Table 11, where rows follow lexicographic input order (A, B, C).

Table 11: Truth tables of the three S-box output bits

Index	(A, B, C)	out_0	out_1	out_2
0	000	1	1	1
1	001	0	0	1
2	010	0	1	1
3	011	1	0	0
4	100	0	0	0
5	101	1	0	1
6	110	0	1	0
7	111	1	1	0

Using SageMath, the ANF of each output bit is computed as the unique polynomial representation over $\mathbb{F}_2$, where addition corresponds to XOR and multiplication corresponds to AND. The resulting algebraic expressions are:

$$\text{out}_0(A, B, C) = 1 \oplus A \oplus B \oplus C \oplus (BC),$$
$$\text{out}_1(A, B, C) = 1 \oplus A \oplus C \oplus (AC) \oplus (BC),$$
$$\text{out}_2(A, B, C) = 1 \oplus C \oplus (AB) \oplus (AC).$$

Table 12 summarizes the algebraic degree and balance properties of each Boolean output function. All three output bits have algebraic degree 2, since their highest-degree monomials are quadratic terms such as BC, AC, or AB. Furthermore, each output bit has Hamming weight 4 (i.e., 4 ones and 4 zeros), implying that the S-box is balanced on each coordinate output function.

Table 12: Algebraic properties of the S-box output ANFs

Output bit	Highest-degree monomials	Algebraic degree	Balanced?
out_0	BC	2	Yes
out_1	$AC,\ BC$	2	Yes
out_2	$AB,\ AC$	2	Yes

The obtained ANF representation offers a compact algebraic description of the S-box and is useful for analyzing algebraic complexity, degree-based resistance to algebraic attacks, and simplifying hardware-friendly implementations.

7. Automated Tools:

Differential cryptanalysis remains one of the most fundamental techniques for evaluating the security margin of symmetric primitives. For lightweight ciphers and permutation-based designs such as Gimli, understanding how differences propagate through the non-linear layer is essential for estimating the resistance of the construction to differential attacks. However, manually identifying the best differential characteristics becomes computationally infeasible as the state size increases or when the structure involves complex bit-level permutations and S-box interactions.

7.1 MILP Analysis of One Round of the Actual Gimli Cipher [7]

Mixed-Integer Linear Programming (MILP) model is created for one round of the *Gimli cipher*, a lightweight cryptographic permutation used in symmetric encryption. The key steps of the experiment were:

7.1.1 Model Creation

- Defined **binary variables** representing the bits of the 12 32-bit words in the Gimli state.
- Modeled the **SP-box** (non-linear layer) using XOR, AND, and OR constraints on the binary variables.

- Applied the **linear layer (permutations)** and **round constants** for round 24.
- Set the **objective function** to minimize the Hamming weight (number of 1s) of the output state.

7.1.2 Input Difference Injection

A **single active input bit** was fixed (bit 0 of word 0), and all other input bits were set to zero.

7.1.3 Optimization with Gurobi

The MILP was solved to find the **minimum output Hamming weight** resulting from this single-bit difference.

This experiment focuses on the differential cryptanalysis of the Gimli cipher, which studies how differences in the input propagate through the cipher. While a single-bit input difference can generate multiple output differences, the minimal output Hamming weight is of particular interest because it reveals potential weaknesses in diffusion. Mixed Integer Linear Programming (MILP) is employed since the operations in Gimli—XOR, AND, and rotations—can be modeled as linear constraints, enabling exact computation of minimal or maximal differential propagation rather than relying on approximate simulation. Conducting a one-round study provides insight into differential patterns that can later be extended to multi-round analysis.

7.1.4 MILP Solver Output

- Number of binary variables: 2753
- Number of constraints: 5121
- Optimal solution found: Objective value = 20 (minimum Hamming weight of output)

7.1.5 Output State

```
s_out[ 0]: 0x9e377918
s_out[ 4]: 0x03000000
s_out[ 8]: 0x01000000
```

The interpretation shows that only three words have non-zero bits while the remaining words are zero, with a Hamming weight of 20 indicating that, out of 384 output bits (12 words $\times$ 32 bits), 20 bits are 1. This output illustrates how the input difference propagates through the SP-box and linear layer in a single round.

The significance of this experiment lies in several aspects: for security analysis, low Hamming weight propagation patterns are important in differential attacks as they indicate potential high-probability differential characteristics; for cipher design verification, it

confirms that the non-linear SP-box and permutation layer provide good diffusion, with a single-bit input affecting 20 bits after one round, demonstrating moderate diffusion; and as a foundation for multi-round analysis, the one-round model can be extended to multiple rounds to identify minimal differential trails across the full Gimli permutation.

Table 13: Key results from one-round MILP analysis of Gimli

Aspect	**Observation**
Input difference	Single bit active (word 0, bit 0)
Round	24
MILP variables	2753 binary variables
MILP constraints	5121
Optimal output Hamming weight	20
Non-zero output words	s_out[0], s_out[4], s_out[8]
Security implication	Shows diffusion, minimal-weight differential pattern

Summary of Key Findings The experiment successfully models one round of Gimli using MILP, demonstrating how a single-bit input difference propagates, with a minimum Hamming weight of 20 providing insight into the cipher's diffusion properties. This study serves as a critical first step in differential cryptanalysis and overall cipher evaluation.

8. Browny Point

8.1 Minimum Active sbox using MILP(Mixed Integer linear Programming) on 3bit sbox version Gimli:

Mixed Integer Linear Programming (MILP) has emerged as a powerful tool for automatically determining optimal differential trails. MILP-based modeling converts differential propagation rules into linear constraints over binary variables, enabling the solver to compute characteristics with minimum activity, minimum number of active S-boxes, or minimum differential probability.

The experiment conducted in this work uses MILP to identify the lowest-weight differential trail across the nonlinear transformation of Gimli, specifically over six rounds. This analysis is critical for determining whether low-weight trails exist that could potentially lead to distinguishers or key-recovery attacks.

8.1.1 Security Evaluation Against Differential Attacks

Importance of Differential Analysis Differential cryptanalysis is a fundamental tool for evaluating the robustness of symmetric primitives. In particular, understanding how few nonlinear components can remain active over several rounds directly impacts the minimum differential probability. If a cipher permits long trails with very few active

nonlinear operations, an adversary may exploit them to construct efficient distinguishers or attacks. This motivates the search for the smallest possible number of active nonlinear components across a sequence of rounds using MILP-based optimization.

Detection of Weak or Trivial Differential Trails Block ciphers and permutations may unintentionally admit *weak* differential trails, in which differences propagate without sufficiently activating nonlinear components. Identifying such behavior is essential to:

- ensure that no exploitable propagation patterns exist,
- diagnose structural weaknesses caused by bitwise rotations or index interactions,
- guide designers toward improved diffusion and nonlinearity.

Finding such trails, especially in Boolean-based designs like Gimli, provides crucial insight into the cipher's internal diffusion structure.

Validation of Modeling for Boolean Nonlinearities Unlike classical S-box-based primitives, Gimli's nonlinear layer consists solely of Boolean XOR and AND operations. Accurately modeling these operations in an MILP framework requires:

- exact constraints for XOR behavior,
- correct inequalities for AND-based differential propagation,
- precise handling of bit-index rotations.

The experiment confirms that the MILP model captures these operations correctly and yields deterministic, valid differential trails consistent with the Boolean specification.

8.1.2 Modeling the Nonlinear Layer Using MILP

State Structure Gimli operates on three parallel 32-bit branches:

$$X, \qquad Y, \qquad Z.$$

At each round, the nonlinear step applies Boolean operations combined with rotations to all bit positions.

Nonlinear Transformation Per Bit For each bit position i, the nonlinear update is defined as:

$$\begin{aligned} z'_i &= (z \lll 1)_i \ \oplus \ (y_i \wedge x_i), \\ x'_i &= (x \lll 1)_i \ \oplus \ ((y \lll 2)_i \wedge (z \lll 2)_i), \\ y'_i &= (y \lll 1)_i \ \oplus \ ((x \lll 3)_i \wedge (z \lll 3)_i). \end{aligned}$$

These Boolean equations are transformed into MILP constraints using:

- four linear inequalities per XOR gate,
- two inequalities per AND gate,
- rotation constraints that map bit indices accordingly.

Activity Variable To quantify the cost of differential propagation, each bit position is assigned an activity variable:

$$a_i(r) = 1 \quad \text{if any of } \{x_i, y_i, z_i, x'_i, y'_i, z'_i\} = 1.$$

The total objective minimized by the solver is:

$$\min \sum_{r=0}^{R-1} \sum_{i=0}^{31} a_i(r).$$

Boundary Condition To ensure a single-bit input difference, the initial condition enforces:

$$\sum_i (X_0[i] + Y_0[i] + Z_0[i]) = 1.$$

8.1.3 Experimental Setup

- **Number of rounds:** 6
- **Objective:** Minimize total activity over all rounds
- **Variables:** 1440 binary variables
- **Constraints:** 4609 linear constraints
- **Solver:** Gurobi 12.0.3 (default parameters)

During presolve, Gurobi eliminated all constraints, indicating that XOR and rotation rules leave the system fully determined once boundary conditions are fixed.

8.1.4 Results

Optimization Outcome The optimal solution has:

$$\text{Objective value} = 12, \qquad \text{Activity per round} = 2.$$

Table 14: Differential Trail for 6 Rounds of the Gimli Nonlinear Layer (MILP Result)

Round	**X (32-bit state)**	**Y**	**Z**	**Active Bit in X**	**Activity**
R0	00..0100000	00..0000000	00..0000000	5	2
R1	00..0010000	00..0000000	00..0000000	4	2
R2	00..0001000	00..0000000	00..0000000	3	2
R3	00..0000100	00..0000000	00..0000000	2	2
R4	00..0000010	00..0000000	00..0000000	1	2
R5	00..0000001	00..0000000	00..0000000	0	2
R6	10..0000000	00..0000000	00..0000000	31	–

The solver verifies that the nonlinear AND terms never activate and that the differential propagates in a purely linearized manner.

Differential Trail Obtained The optimal trail satisfies:

$$Y_r = 0, \qquad Z_r = 0 \quad \forall r.$$

Exactly one bit remains active in X, rotating left each round:

$$X_0 = 2^5 \to X_1 = 2^4 \to X_2 = 2^3 \to \cdots \to X_6 = 2^{31}.$$

Since $y = z = 0$ during all rounds, all AND terms evaluate to zero:

$$(y \wedge x) = 0, \qquad (y \lll 2 \wedge z \lll 2) = 0, \qquad (x \lll 3 \wedge z \lll 3) = 0.$$

Thus,

$$x' = x \lll 1, \qquad y' = 0, \qquad z' = 0.$$

This experiment demonstrates the effectiveness of MILP modeling for analyzing differential propagation in modern lightweight cryptographic designs. The results show that for the first six iterations of the nonlinear operation in Gimli, an extremely low-weight trail exists, consisting of a single active bit traversing through adjacent positions via deterministic rotations. The derived trail serves as a lower bound for differential resistance within the isolated nonlinear layer.

8.2 Boolean Satisfiability Encoding of a Single Gimli Round [8]

In this section we describe the conversion of one round of the Gimli permutation into a propositional Boolean satisfiability problem (SAT). The objective is to represent the nonlinear SP-box, the linear layer (swapping operations), and the round-constant injection as a system of CNF clauses. This enables differential trail computation or structural

analysis using modern SAT solvers.

State Representation

The internal state of Gimli consists of 12 words of 32 bits, arranged as a 3×4 matrix. We index the words in row-major order as

$$(X_0, X_1, X_2, X_3,\ Y_0, Y_1, Y_2, Y_3,\ Z_0, Z_1, Z_2, Z_3),$$

where each word is expanded into 32 Boolean variables. The SAT instance therefore contains 384 state bits for both the input and the output of a round.

Nonlinear SP-Box Encoding

The SP-box processes each column independently. Let (x, y, z) denote the three 32-bit words of a fixed column, after applying the column-dependent rotations

$$x = \mathrm{ROTL}_{32}(X_i, 24), \qquad y = \mathrm{ROTL}_{32}(Y_i, 9), \qquad z = Z_i.$$

For each bit position $j \in \{0, \ldots, 31\}$ the nonlinear update is

$$\begin{aligned} Z_i' &= x \oplus (z \ll 1) \oplus (y \wedge z \ll 2), \\ Y_i' &= y \oplus x \oplus (x \vee z \ll 1), \\ X_i' &= z \oplus y \oplus (x \wedge y \ll 3), \end{aligned}$$

where "$\ll k$" denotes bit-shifts with zero-fill.

Each Boolean operator is encoded in CNF using Tseitin variables. For two Boolean literals a and b, we use standard encodings:

$$c \leftrightarrow (a \wedge b) \quad \Rightarrow \quad (\neg c \vee a), (\neg c \vee b), (\neg a \vee \neg b \vee c),$$

$$c \leftrightarrow (a \vee b) \quad \Rightarrow \quad (\neg c \vee a \vee b), (\neg a \vee c), (\neg b \vee c),$$

and for XOR,

$$c \leftrightarrow (a \oplus b) \Rightarrow (\neg a \vee \neg b \vee \neg c), (\neg a \vee b \vee c), (a \vee \neg b \vee c), (a \vee b \vee \neg c).$$

Shift operations do not introduce additional constraints: each output bit is directly connected to a previous bit or to a constant 0.

Linear Layer and Swap Encoding

After the SP-box, Gimli performs a lightweight linear layer, controlled by the round index r:

$$r \bmod 4 = 2 \quad \Rightarrow \quad \begin{cases} X'_0 \leftrightarrow X'_2, \\ X'_1 \leftrightarrow X'_3, \end{cases} \qquad \text{(Big-swap)},$$

$$r \bmod 4 = 0 \quad \Rightarrow \quad \begin{cases} X'_0 \leftrightarrow X'_1, \\ X'_2 \leftrightarrow X'_3, \end{cases} \qquad \text{(Small-swap)},$$

while the remaining eight state words $(Y_0, Y_1, Y_2, Y_3, Z_0, Z_1, Z_2, Z_3)$ are unchanged. Each swap is encoded by simple equality constraints between the corresponding Boolean literals.

It is important to emphasize that the swaps only affect the first row of four words. No operation in the official specification redistributes entire columns across rows. Thus, the set of indices $\{4, 5, 6, 7, 8, 9, 10, 11\}$ remains invariant under both the small-swap and big-swap.

Round Constant Injection

Whenever $r \bmod 4 = 0$, a 32-bit round constant

$$\mathrm{RC}(r) = (0x9e377900 \oplus r)$$

is XORed into the lowest word X'_0. Each constant bit is implemented as either:

$$X'_0[j] = X'_0[j] \quad \text{(if RC bit is 0)}, \qquad X'_0[j] = \neg X'_0[j] \quad \text{(if RC bit is 1)},$$

where the latter requires one Tseitin negation variable and two clauses.

Model Completeness

The resulting SAT instance is a complete functional representation of one round of Gimli. It contains:

- 384 Boolean input variables,
- 384 Boolean output variables,
- a linear number of Tseitin variables for AND/OR/XOR,
- CNF clauses describing all nonlinear and linear transformations.

Since every operation is encoded with equisatisfiable Tseitin gadgets, any satisfying assignment directly corresponds to a valid Gimli state transition.

This SAT model allows automated reasoning over differential propagation, low-weight output search, and structural reachability. Unlike MILP, SAT does not require linearization of Boolean arithmetic and can explore arbitrary state constraints. Modern solvers (e.g., Kissat, MiniSat or CryptoMiniSat) evaluate such instances efficiently.

8.3 Collision Attack on a Toy Gimli-Based Hash

In this experiment, we demonstrate a practical collision attack on a toy sponge hash constructed from the reference toy Gimli permutation. The goal of this experiment is not to break the real Gimli or Gimli-Hash construction, but to illustrate the fundamental behaviour of sponge hashing under reduced-security parameters and to experimentally observe collisions using a simple birthday attack.

Toy Hash Construction. The internal state of the toy hash consists of the 12-byte state of the toy Gimli permutation, represented as three planes (x, y, z) of four bytes each. During the absorbing phase, up to three message bytes are XORed directly into the rate portion of the state:

$$x[0] \oplus = m_0, \qquad y[0] \oplus = m_1, \qquad z[0] \oplus = m_2,$$

where m_0, m_1, m_2 denote the first three bytes of the message block. The remaining state bytes remain untouched and are interpreted as capacity. After absorption, the state is updated through a reduced-round Gimli permutation with R rounds:

$$S' = \text{Gimli}(S, R).$$

In the squeezing phase, the hash output consists of a single byte, namely the least significant byte of the state component $x[0]$:

$$\text{Hash}(m) = x[0] \mod 256.$$

This deliberately small output size (8 bits) ensures that collisions are statistically expected, enabling a controlled and pedagogical collision demonstration without the need for explicit differential trail construction.

Attack Strategy. The collision attack follows a standard birthday search: random messages of length one to three bytes are generated, hashed, and stored in a lookup table indexed by the resulting 8-bit hash value. Whenever two distinct messages produce the same hash value, a collision has been found. Let the output size be $n = 8$ bits. By the birthday bound, the expected number of hash evaluations required to observe a collision is approximately

$$\sqrt{2^n} = 2^{n/2} = 2^4 = 16,$$

rendering collisions trivial to find even when all 12 permutation rounds are used. No structural cryptanalysis is required: the reduced output width alone guarantees the existence of immediate collisions.

Experimental Results. The collision attack was performed for round counts $R \in \{2, \ldots, 12\}$. For each value of R, two distinct messages $m_1 \neq m_2$ were found such that

$$\text{Hash}(m_1) = \text{Hash}(m_2).$$

In most cases, a collision was found after only a single hash evaluation. In a few instances, the number of evaluations was close to the statistical expectation (e.g., 17 trials for $R = 8$). Table 15 summarises the observations.

Rounds R	**Collision Attempts**	**Hash Value**
2	1	0x00
3	1	0x00
4	1	0xFF
5	1	0x00
6	1	0xFF
7	1	0x00
8	17	0x04
9	4	0xFC
10	1	0x00
11	1	0xFF
12	1	0x00

Table 15: Collision results for the toy Gimli hash using R rounds and an 8-bit output.

As an example, for $R = 12$, the messages

$$m_1 = \texttt{b003ed}, \qquad m_2 = \texttt{962482}$$

produce the same 8-bit hash output $0x00$. Similar collisions were found for all other round counts.

Discussion. This experiment confirms that the round depth of the permutation has negligible influence on collision resistance when the output size is as small as 8 bits. Even a full 12-round permutation offers no protection under these reduced parameters. The collisions observed here are *not* indicative of a weakness in the real Gimli or Gimli-Hash constructions. Real sponge-based hashes rely on a sufficiently large rate and output (typically at least 128 bits) to achieve collision resistance. The collisions observed are merely a consequence of the intentionally reduced configuration of the toy hash and serve only to illustrate fundamental sponge hashing behaviour from an educational perspective.

9. Conclusion

Gimli achieves a rare combination of simplicity, efficiency, and security. Its ARX-based design avoids lookup tables and large constants, making it well-suited for secure embedded systems and IoT platforms. Future research explores its differential and boomerang resistance using MILP and SAT-based cryptanalysis.

References

[1] D. J. Bernstein, S. Kölbl, S. Lucks, *et al.*, "Gimli: A cross-platform permutation," *Cryptology ePrint Archive, Report 2017/043*, 2017. [Online]. Available: `https://eprint.iacr.org/2017/043`.

[2] National Institute of Standards and Technology, *Lightweight cryptography*, `https://csrc.nist.gov/projects/lightweight-cryptography`, Accessed: 2025-12-06, 2019.

[3] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "The keccak reference," in *Submission to NIST (Round 3)*, 2011. [Online]. Available: `http://keccak.noekeon.org/Keccak-reference-3.0.pdf`.

[4] J. Daemen, S. Hoffert, M. Peeters, G. V. Assche, and R. V. Keer, "Xoodoo, a permutation for lightweight applications," in *Lightweight Cryptography Workshop 2018*, ser. NIST Interagency Report 8268, NIST, 2018, pp. 83–85. [Online]. Available: `https://doi.org/10.6028/NIST.IR.8268`.

[5] E. Biham and A. Shamir, "Differential cryptanalysis of des-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, 1991.

[6] M. Matsui, "Linear cryptanalysis method for des cipher," in *Advances in Cryptology EUROCRYPT'93*, Springer, 1993, pp. 386–397.

[7] N. Mouha, Q. Wang, D. Gu, and B. Preneel, "Differential and linear cryptanalysis using mixed-integer linear programming," in *Information Security and Cryptology*, Springer, 2011, pp. 57–76.

[8] F. Massacci and L. Marraro, "Using sat and clp(fd) for cryptographic test-data generation," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2000, pp. 30–40.

[9] D. Wagner, "The boomerang attack," in *Fast Software Encryption*, Springer, 1999, pp. 156–170.

[10] C. Boura and A. Canteaut, "Boomerang connectivity table," in *Fast Software Encryption*, Springer, 2018, pp. 29–56.

[11] L. R. Knudsen and D. Wagner, "Integral cryptanalysis," in *Fast Software Encryption*, Springer, 2002, pp. 112–127.

[12] L. R. Knudsen, "Truncated and higher order differentials," in *Fast Software Encryption*, Springer, 1994, pp. 196–211.

[13] S. K. Langford and M. E. Hellman, "Differential-linear cryptanalysis," in *Advances in Cryptology—CRYPTO'94*, Springer, 1994, pp. 17–25.

Acknowledgment

Acknowledgment: I would like to thank the faculty and peers at IIT Bhilai for their guidance and support during this work.