Volume 1, Issue I, October 2025
Pages 41- 66

Differential Cryptanalysis on Lightweight Block Cipher SPECK: Classical and Neural Network Based Approaches

Ahel Singha
National Institute of Technology Jamshedpur
Email: singhaahel545@gmail.com

Abstract

With the proliferation of connected devices in IoT environments, there is an urgent need for encryption algorithms that are both secure and computationally lean. The SPECK block cipher, an ARX-based design introduced by the NSA, fits this need well and has consequently become a focal point for cryptanalytic research. This thesis undertakes a systematic study of how differential cryptanalysis – in both its classical and machine-learning-assisted forms – can be applied to round-reduced SPECK.

The work unfolds across three interconnected themes. The first is a detailed exposition of the SPECK cipher family itself: its parametric structure, the ARX round function, and the key schedule. The second theme covers classical differential cryptanalysis as applied to ARX ciphers, with particular focus on the partial Difference Distribution Table (pDDT) and the Nested Monte Carlo Search (NMCS) algorithm that together enable practical trail-finding at scale.

The third and central theme examines how deep learning has transformed this field. Starting from Gohr's 2019 CRYPTO result – where a residual neural network outperformed classical distinguishers on SPECK-32/64 – the thesis traces subsequent advances: attention-augmented architectures, Inception-module-based networks, enriched input representations that exploit partial decryption, and batch-based inference strategies. The interpretability of these models is also explored through the Masked Output Distribution Table (M-ODT) framework.

Taken together, the evidence shows that neural methods can uncover statistical regularities in cipher outputs that classical analysis misses entirely, while the full-round variants of SPECK maintain their designed security margins.

1 Introduction

1.1 Motivation and Background

The ongoing expansion of the Internet of Things has brought with it a class of devices – sensors, RFID tags, embedded controllers – that communicate wirelessly yet operate under severe constraints of memory, power, and processing speed. Conventional encryption standards such as AES, though cryptographically robust, demand computational resources that many such devices simply do not possess [2]. This gap gave rise to the discipline of *lightweight cryptography*, whose central goal is to achieve strong security guarantees with minimal overhead.

The SPECK cipher family, introduced publicly by the NSA in 2013 [1], stands out within this discipline. Built entirely from modular addition, bitwise rotation, and XOR – collectively called the ARX paradigm – it achieves high throughput on commodity hardware while remaining accessible

enough for embedded microcontrollers. Its algebraic simplicity, however, also makes it a particularly attractive subject for cryptanalysis, since the behaviour of each operation under difference propagation can be characterised precisely.

The dominant tool for studying such ciphers is *differential cryptanalysis*, a chosen-plaintext method due to Biham and Shamir [3] that traces statistical patterns as input differences evolve through cipher rounds. More recently, this classical framework has been extended by machine learning: Gohr's 2019 result [10] showed that a deep residual network trained as a binary classifier could distinguish round-reduced SPECK output from uniformly random data more reliably than the best hand-crafted differential distinguisher. This intersection of cryptography and AI forms the central motivation of the present work.

This thesis provides a self-contained study of (1) the SPECK cipher family, (2) classical differential cryptanalysis and heuristic trail search, and (3) neural-network-based distinguishers and their improvements.

1.2 Objectives of the Thesis

1. Provide a complete mathematical description of the SPECK family (round function, key schedule, all variants).
2. Develop differential cryptanalysis theory: DDT, pDDT, and differential trails for ARX ciphers.
3. Explain Nested Monte Carlo Search (NMCS) for finding high-probability differential trails in SPECK.
4. Survey neural differential distinguishers (Gohr, Chen, Deng, Yue & Wu, Benamira).
5. Compare classical and neural approaches in accuracy, complexity, and security implications.

2 The SPECK Block Cipher

2.1 Overview and Design Philosophy

The SPECK family was developed by a research team at the NSA and released in 2013 [1]. The design brief balanced four priorities simultaneously:

- **Security:** A comfortable safety margin against all published attack techniques.

- **Flexibility:** A range of block and key sizes to suit applications from RFID to server-side encryption.
- **Efficiency:** Top-tier throughput in software, particularly on constrained 8-bit and 16-bit processors.
- **Simplicity:** A small, auditable codebase that is easy to implement correctly.

SPECK falls within the family of *ARX ciphers* – designs that confine themselves to modular Addition ($\boxplus$), bitwise Rotation ($\lll$, $\ggg$), and XOR ($\oplus$). Because all three of these operations are executed natively by virtually every processor instruction set, SPECK software implementations are both fast and compact, with no dependency on precomputed lookup tables.

Unlike its sibling SIMON, which was tuned for hardware gate-count minimisation using AND operations, SPECK deliberately prioritises software performance. Modular addition is SPECK's sole source of nonlinearity; it offers a richer mixing effect than bitwise AND, which contributes to a stronger per-round diffusion.

2.2 The ARX Paradigm

SPECK uses only three operations: modular Addition ($\boxplus$), bitwise Rotation ($\lll/\ggg$), and XOR ($\oplus$). Modular addition $x \boxplus y = (x + y) \bmod 2^n$ is the sole *nonlinear* operation, producing non-trivial XOR differential distributions. Rotations and XOR are linear over $\mathbb{F}_2^n$. Security arises from the repeated composition of these operations across multiple rounds.

2.3 SPECK Parameters and Variants

SPECK is a *parameterised family* rather than a single algorithm. Each member is identified by its block size $2n$ and key size mn, written as $\text{SPECK}_{2n/mn}$. The number of rounds T and rotation constants (α, β) are set individually for each variant. Table 1 lists the complete parameter set across all ten standardised instances.

Table 1: SPECK family parameters [1, 2]

Variant	**Block** ($2n$)	**Word** (n)	**Key** (mn)	**Key Words** (m)	**Rounds** (T)	(α, β)
SPECK32/64	32	16	64	4	22	(7,2)
SPECK48/72	48	24	72	3	22	(8,3)
SPECK48/96	48	24	96	4	23	(8,3)
SPECK64/96	64	32	96	3	26	(8,3)
SPECK64/128	64	32	128	4	27	(8,3)
SPECK96/96	96	48	96	2	28	(8,3)
SPECK96/144	96	48	144	3	29	(8,3)
SPECK128/128	128	64	128	2	32	(8,3)
SPECK128/192	128	64	192	3	33	(8,3)
SPECK128/256	128	64	256	4	34	(8,3)

Two choices of rotation constants appear across the family. SPECK32/64 uses $\alpha = 7$ and $\beta = 2$, whereas all larger variants adopt $\alpha = 8$ and $\beta = 3$. The value $\alpha = 8$ is particularly attractive for 8-bit microcontrollers: rotating a byte-aligned word by 8 positions is equivalent to a register renaming operation, incurring zero clock cycles. Both constants were also optimised for minimal area in bit-serial ASIC realisations [2].

2.4 The SPECK Round Function

Each encryption round of SPECK operates on a two-word state (x, y), each word being n bits wide. The structure loosely resembles a generalised Feistel network in that one branch is mixed into the other, though the operations used are purely ARX rather than S-box-based.

Definition 2.1 (SPECK Round Function) *Given an n-bit round key k_i, the i-th round of SPECK transforms the state (x, y) as follows:*

$$x \leftarrow (x \ggg \alpha) \boxplus y \tag{2.1}$$

$$x \leftarrow x \oplus k_i \tag{2.2}$$

$$y \leftarrow y \lll \beta \tag{2.3}$$

$$y \leftarrow y \oplus x \tag{2.4}$$

More compactly, the round function $R_{k_i}(x, y)$ maps:

$$(x, y) \mapsto ((x \ggg \alpha) \boxplus y) \oplus k_i,\ y \lll \beta \oplus ((x \ggg \alpha) \boxplus y) \oplus k_i) \tag{2.5}$$

The round function can be visualised as shown in Figure 1. The full encryption of a plaintext (p_l, p_r)

under key K consists of iterating this round function T times, initialising with $(x_0, y_0) = (p_l, p_r)$:

$$(x_{i+1}, y_{i+1}) = R_{k_i}(x_i, y_i), \quad i = 0, 1, \ldots, T-1 \tag{2.6}$$

$$C = (x_T, y_T) \tag{2.7}$$

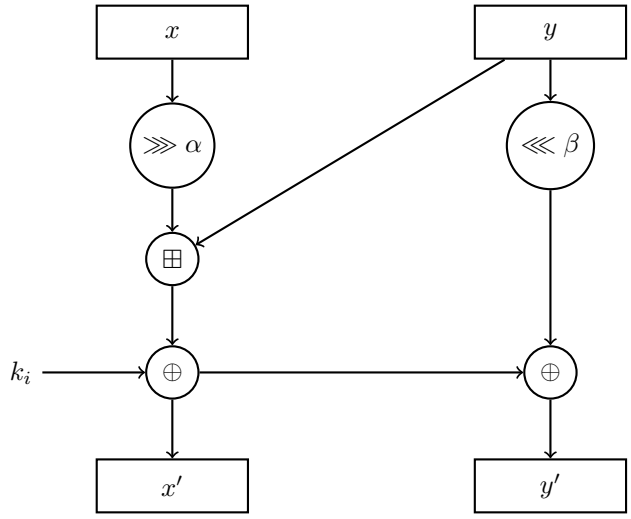

Figure 1: The SPECK round function. The left branch undergoes right rotation by α, modular addition with the right branch, and XOR with the round key k_i. The right branch undergoes left rotation by β followed by XOR with the updated left branch.

2.5 The SPECK Key Schedule

An appealing feature of SPECK is that its key schedule is built from the same round function as the encryption itself, keeping the implementation footprint minimal. Starting from a master key expressed as m words $(k_{m-1}, \ldots, k_1, k_0)$, successive round keys are produced by the iterative relations:

$$\ell_{i+m-1} = (k_i \boxplus (\ell_i \ggg \alpha)) \oplus i \tag{2.8}$$

$$k_{i+1} = (k_i \lll \beta) \oplus \ell_{i+m-1} \tag{2.9}$$

for $i = 0, 1, \ldots, T-2$, with initial words $\ell_0, \ldots, \ell_{m-2}$ set to $k_1, \ldots, k_{m-1}$. Incorporating the loop counter i as an explicit additive term in equation (2.8) is a deliberate measure against slide attacks, which exploit periodic or weakly structured key schedules.

2.6 Decryption

Because every step of the round function is reversible, SPECK decryption is obtained by inverting the sequence of operations. Given an output pair (x', y'), the corresponding inputs are recovered as:

$$y = (x' \oplus y') \ggg \beta \tag{2.10}$$

$$x = ((x' \oplus k_i) \ominus y) \lll \alpha \tag{2.11}$$

where $\ominus$ denotes modular subtraction. Decryption thus requires modular subtraction in place of addition and reverses the direction of rotation; this slight asymmetry is the price paid for SPECK's stronger nonlinearity compared with SIMON's AND-based construction.

2.7 Performance and Security Margin

In practice, SPECK delivers impressive throughput across a wide range of platforms. Benchmarks on an Intel Haswell CPU show SPECK128/256 encrypting at roughly 1.36 cycles per byte, within striking distance of hardware-accelerated AES-NI at 1.02 cycles per byte [2]. On 8-bit and 16-bit embedded processors, SPECK and SIMON collectively outperform 13 other lightweight candidates in authentication-protocol benchmarks [2].

Table 2 summarises the security margin of SPECK variants in terms of the best published reduced-round attacks relative to full rounds.

Table 2: Security margin of selected SPECK variants (as of 2023)

Variant	Total Rounds	Best Attack Rounds	Coverage	Attack Type
SPECK32/64	22	11	50.0%	Key recovery (neural) [10]
SPECK64/128	27	19	70.3%	Differential [2]
SPECK128/128	32	20	62.5%	Differential

3 Differential Cryptanalysis

3.1 Introduction to Differential Cryptanalysis

Originally developed by Biham and Shamir as an attack against DES in 1990 [3], differential cryptanalysis has since become the foundational benchmark against which every new block cipher is evaluated. Its core insight is deceptively simple: track how a chosen difference between two plaintext inputs survives and transforms as it passes through successive encryption rounds. When the cipher is not a perfect pseudo-random permutation, certain input–output difference pairs occur far more

often than chance would predict. An attacker who collects enough ciphertext pairs can exploit this statistical bias to narrow down the secret key.

For ARX ciphers such as SPECK, modular addition is the only component that introduces nonlinearity, and it is also the only component that resists simple bit-level analysis. This makes differential cryptanalysis not only applicable but especially illuminating: understanding how addition handles carry propagation under XOR differences is the crux of the analysis.

3.2 Fundamental Definitions

Definition 3.1 (Difference) *For a block cipher operating on n-bit blocks, the XOR difference between two n-bit values x and x' is defined as:*

$$\Delta x = x \oplus x'$$

Definition 3.2 (Differential) *A differential for a function $f : \{0,1\}^n \to \{0,1\}^m$ is a pair $(\Delta x, \Delta y)$ where Δx is an input difference and Δy is a corresponding output difference.*

Definition 3.3 (Differential Probability) *The differential probability (DP) of a differential $(\Delta x \to \Delta y)$ through a function f is:*

$$DP(\Delta x \to \Delta y) = \frac{\#\{x \in \{0,1\}^n : f(x) \oplus f(x \oplus \Delta x) = \Delta y\}}{2^n} \tag{3.1}$$

Definition 3.4 (Difference Distribution Table (DDT)) *For a function $f : \{0,1\}^n \to \{0,1\}^m$, the Difference Distribution Table (DDT) is a $2^n \times 2^m$ table where the entry $DDT[\Delta x][\Delta y]$ counts the number of inputs x such that $f(x \oplus \Delta x) \oplus f(x) = \Delta y$:*

$$DDT[\Delta x][\Delta y] = \#\{x \in \{0,1\}^n : f(x) \oplus f(x \oplus \Delta x) = \Delta y\} \tag{3.2}$$

Hence, $DP(\Delta x \to \Delta y) = DDT[\Delta x][\Delta y]/2^n$.

For S-box-based ciphers with small lookup tables (4-bit or 8-bit), a full DDT can be computed in time $O(2^{2n})$, which is perfectly tractable. The situation differs sharply for ARX ciphers: here the fundamental nonlinear operation is 32-bit or 64-bit modular addition, and constructing an exhaustive DDT for a 32-bit operation would require storing 2^{64} entries – a task far beyond present computational reach.

3.3 Differential Properties of Modular Addition

Within SPECK, the function $f(x, y) = x \boxplus y$ over n-bit words is the sole nonlinear building block. Understanding how XOR differences propagate through it is therefore essential for any differential attack. Lipmaa and Moriai provided an exact closed-form answer in [4].

Theorem 3.1 (Differential Probability of Modular Addition [4]) *Let $a, b, c \in \{0,1\}^n$ represent input differences and output difference for the modular addition $f(x, y) = x \boxplus y \bmod 2^n$. The differential $(a, b \to c)$ is valid (i.e., has non-zero probability) if and only if:*

$$eq(a \lll 1, b \lll 1, c \lll 1) \wedge ((a \oplus b \oplus c \oplus (b \lll 1)) = 0) \tag{3.3}$$

where $eq(p, q, r) := (\neg p \oplus q) \wedge (\neg p \oplus r)$ tests bitwise equality of p, q, r at each position.

For a valid differential $(a, b \to c)$, the XOR differential probability is:

$$xdp^+(a, b \to c) = 2^{-w(a,b\to c)} \tag{3.4}$$

where the weight $w(a, b \to c) = h^*(\neg eq(a, b, c))$, and $h^*(x)$ denotes the number of non-zero bits of x, not counting the most significant bit.

This result is computationally powerful: rather than enumerating all 2^{2n} input pairs, one can determine the validity and probability of any modular addition differential in $O(n)$ time using simple bitwise operations. This efficiency is what makes large-scale automated trail search in ARX ciphers feasible.

Example 3.1 (Modular Addition Differential) *Take 4-bit words ($n = 4$). Set input differences $a = 0100_2$ and $b = 0000_2$, with candidate output difference $c = 0100_2$. Computing the weight gives $w = 0$, so $xdp^+(4, 0 \to 4) = 2^0 = 1$: the transition is* deterministic. *Adding zero to any value leaves it unchanged, so the XOR difference is preserved perfectly.*

3.4 Differential Trails in Block Ciphers

Definition 3.5 (Differential Trail) *A differential trail (or characteristic) for an r-round block cipher is a sequence of differences $(\Delta_0, \Delta_1, \ldots, \Delta_r)$ where Δ_0 is the plaintext difference, Δ_i is the difference entering round $i + 1$, and Δ_r is the ciphertext difference.*

The probability of a differential trail, under the hypothesis that consecutive round differences are independent (Markov assumption), is:

$$DP^r(\Delta_0 \to \Delta_r) \approx \prod_{i=0}^{r-1} DP_i(\Delta_i \to \Delta_{i+1}) \tag{3.5}$$

where DP_i is the differential probability through round i.

Definition 3.6 (Best Differential Trail) *The best r-round differential trail is the one maximising the product of per-round probabilities: Finding this optimum is equivalent to a shortest-path problem in a weighted graph with exponentially many nodes.*

Definition 3.7 (Differential Attack) *A differential attack on a block cipher uses a high-probability differential trail for r rounds to recover information about the key used in the remaining rounds. Specifically, if we have an (r)-round differential trail $(\Delta_{in} \to \Delta_{out})$ with probability p, we can attack $(r + k)$ rounds by:*

1. *Collecting $N \approx p^{-1}$ plaintext pairs (P, P') with $P \oplus P' = \Delta_{in}$.*
2. *For each candidate last-round subkey k, partially decrypting the ciphertexts (C, C') and checking whether the decrypted difference equals Δ_{out}.*
3. *The correct subkey will satisfy the condition with probability p, while wrong keys satisfy it with probability 2^{-n}.*

3.5 Differential Cryptanalysis of SPECK: Classical Approach

Applying classical differential cryptanalysis to SPECK immediately runs into a storage problem. For SPECK32/64, words are 16 bits wide, so a full DDT for the 16-bit modular addition would contain $2^{16} \times 2^{16} = 2^{32}$ entries. Holding this table in memory requires roughly 4 GB – which is impractical within an automated search procedure.

Biryukov and Velichkov addressed this by introducing the *Partial Difference Distribution Table* (pDDT) [5]: rather than storing all differentials, one retains only those above a chosen probability threshold.

Definition 3.8 (Partial DDT (pDDT)) *For a threshold p_{thres}, the pDDT is the set of all modular-addition differentials meeting that probability:*

$$pDDT = \{(a, b, c) : xdp^{+}(a, b \to c) \geq p_{thres}\} \tag{3.6}$$

The construction is made efficient by a key monotonicity property: truncating a differential to fewer bits never decreases its probability. A backtracking algorithm therefore prunes entire subtrees early, producing the pDDT without ever evaluating the low-probability entries.

Table 3 documents how sharply the table size and build time grow as the threshold is relaxed [9].

Table 3: pDDT statistics for 32-bit modular addition at different thresholds [9]

Threshold Probability	Entries in pDDT	Computation Time
2^{-3} (0.125)	3,951,388	1.23 min
$2^{-3.8}$ (0.07)	3,951,388	2.29 min
2^{-4} (0.06)	167,065,948	44.36 min
$2^{-6.6}$ (0.01)	$>$ 72,589,325,174	$>$ 29 days

3.6 Differential Trails for SPECK Variants

Using automated search methods (detailed in Chapter 4), the best known differential trails for all SPECK variants have been computed. Table 4 lists representative results.

Table 4: Best known differential trails for SPECK variants [9, 6]

Variant	Rounds	Weight ($-\log_2 p$)	Method
SPECK32/64	8	30	NMCS + pDDT [9]
SPECK32/64	9	31	Start-from-middle
SPECK48/72	9	47	NMCS + pDDT
SPECK48/72	10	43	Start-from-middle
SPECK64/96	11	63	NMCS + pDDT
SPECK64/96	12	63	Start-from-middle
SPECK96/96	10	92	NMCS + pDDT
SPECK96/96	13	89	Start-from-middle
SPECK128/128	11	125	NMCS + pDDT
SPECK128/128	15	127	Start-from-middle

The strongest 8-round trail found for SPECK32/64 carries probability 2^{-30}. When plugged into Dinur's key enumeration framework [8], this trail enables a 12-round attack that requests $2 \cdot 2^{30}$ plaintext pairs at a time cost of 2^{63} cipher evaluations.

Of particular cryptanalytic importance is the 9-round trail anchored at input difference $(0x0040, 0x0000)$ [7]. After a single round, this difference evolves deterministically to $(0x8000, 0x8000)$: a right rotation by 7 maps $0x0040 = 0000\ 0000\ 0100\ 0000_2$ exactly to $0x8000 = 1000\ 0000\ 0000\ 0000_2$, and since the right word is zero, the modular addition introduces no carry at all. The carry-propagation behaviour over subsequent rounds creates a distinctive bias in the output difference that can be statistically detected – and, as we shall see in Chapter 5, this same starting difference is the one exploited by Gohr's neural distinguisher.

4 Heuristic Search for Differential Trails in ARX Ciphers

4.1 The Trail Search Problem

Identifying an optimal differential trail for a fixed number of SPECK rounds is, in essence, a large-scale combinatorial optimisation problem. At every round, a pair of n-bit input differences enters the modular addition, and one of several valid output differences must be selected; the probability of each choice is given by Theorem 3.1. The goal is to assemble a sequence of such choices – one per round – so that their combined probability is as high as possible.

Formally, for SPECK, each round takes a pair of n-bit differences $(\Delta x_i, \Delta y_i)$ and maps them to $(\Delta x_{i+1}, \Delta y_{i+1})$ through:

$$\Delta a_i = \Delta x_i \ggg \alpha \tag{4.1}$$

$$\Delta x_{i+1} = (\Delta a_i \boxplus \Delta y_i)^* \oplus 0 = f(\Delta a_i, \Delta y_i) \quad \text{with probability } p_i \tag{4.2}$$

$$\Delta y_{i+1} = (\Delta y_i \lll \beta) \oplus \Delta x_{i+1} \tag{4.3}$$

where $(\Delta a_i \boxplus \Delta y_i)^*$ denotes a valid output difference for the modular addition differential $(\Delta a_i, \Delta y_i \to \cdot)$, and $p_i = \mathrm{xdp}^+(\Delta a_i, \Delta y_i \to \Delta x_{i+1})$.

The search space expands exponentially: each modular addition node offers $O(2^n)$ candidate output differences, so a naive enumeration over r rounds visits up to $O(2^{rn})$ states. For SPECK32/64 with $n = 16$ and $r = 10$, this gives 2^{160} candidates – entirely beyond exhaustive search.

4.2 Nested Monte Carlo Search for ARX Ciphers

To escape the intractability of exhaustive search, Dwivedi et al. [9] adapted the Nested Monte Carlo Search (NMCS) algorithm [23] – a recursive heuristic originally developed for game-playing – to the problem of differential trail search in ARX ciphers.

4.2.1 The NMCS Algorithm

NMCS structures the search as a tree where each node corresponds to a partial difference assignment for the cipher state. At each decision point, the algorithm:

1. Evaluates every available move by running a complete random rollout to the end of the trail.
2. Commits to the move that produced the lowest-weight rollout.
3. Applies this same logic recursively at every depth level, with higher levels guiding lower-level

searches.

At level 0 the algorithm is a pure random walk; at level 1 each step is greedy with respect to single rollouts; at higher levels, the recursive guidance steers exploration towards more promising regions of the search space.

4.2.2 Adaptation to Differential Trail Search

For ARX differential trail search, at each round NMCS samples a valid output difference from the pDDT for the current modular addition input, accumulates the weight $(-\log_2 p_i)$, and propagates the state differences forward. The best-weight path found across all playouts is selected. The pDDT lookup ensures only high-probability transitions are considered.

4.2.3 Start-from-Middle Technique

Finding long trails forward-only is difficult as probability drops rapidly with rounds. The *start-from-middle* approach runs NMCS forward from the midpoint and independently backward, then concatenates the two halves. This yields longer trails with better probabilities.

4.3 Differential Attack Complexity on SPECK

Once a high-probability trail has been found, the next step is to translate it into a concrete key-recovery attack. Dinur's enumeration method [8] provides a systematic way to extend an r-round trail into a full $(r+m)$-round attack, where m is the number of key words. The procedure is:

1. Request $N = 2 \cdot p^{-1}$ encryptions of plaintext pairs $(P, P \oplus \Delta_{\text{in}})$.
2. For each candidate last-m-round subkey, partially decrypt the last m rounds and check if the decrypted difference equals Δ_{out}.
3. The correct key satisfies the condition with probability p, while wrong keys satisfy it with probability 2^{-2n}.

Applying this to SPECK32/64 with the best 8-round trail (probability 2^{-30}) and $m = 2$ extra rounds:

$$\begin{aligned} \text{Data complexity:} &\quad 2 \cdot 2^{30} = 2^{31} \text{ chosen plaintexts} \\ \text{Time complexity:} &\quad 2^{63} \text{ encryptions} \\ \text{Memory:} &\quad 2^{22} \text{ units} \end{aligned}$$

Table 5 collects the attack parameters across all SPECK variants [9].

Table 5: Differential attacks on SPECK variants using NMCS-found trails [9]

Variant	Rounds/Total	Time	Data	Memory	Trail Rounds
SPECK32/64	12/22	2^{63}	2^{31}	2^{22}	8
SPECK48/72	12/22	2^{72}	2^{48}	2^{22}	9
SPECK48/96	13/23	2^{96}	2^{48}	2^{22}	9
SPECK64/96	14/26	2^{96}	2^{64}	2^{22}	11
SPECK64/128	15/27	2^{128}	2^{64}	2^{22}	11
SPECK96/96	12/28	2^{93}	2^{93}	2^{22}	10
SPECK96/144	13/29	2^{141}	2^{93}	2^{22}	10
SPECK128/128	13/32	2^{126}	2^{126}	2^{22}	11
SPECK128/192	14/33	2^{190}	2^{126}	2^{22}	11
SPECK128/256	15/34	2^{254}	2^{126}	2^{22}	11

5 Neural Network Based Differential Cryptanalysis of SPECK

5.1 Introduction and Motivation

Conventional differential cryptanalysis places heavy demands on the analyst: one must choose a good starting difference, trace that difference through every round of the cipher with care, and design a tailored key-guessing procedure. Furthermore, the approach discards a substantial portion of the available information by representing ciphertext pairs purely through their XOR difference, ignoring the actual values of both ciphertexts.

A fundamentally different strategy was demonstrated by Gohr at CRYPTO 2019 [10]: train a deep residual neural network as a binary classifier to tell apart genuine SPECK outputs from uniformly random data, using raw ciphertext pair values as input. The outcome was startling – the trained network achieved higher classification accuracy than the mathematically optimal DDT-based distinguisher for the same round counts. Three aspects of this result made it especially significant:

1. No prior knowledge of SPECK's algebraic structure was encoded into the classifier.
2. The learned model could be incorporated directly as a scoring function within a key-recovery framework.
3. The classifier appeared to pick up on statistical patterns that fell outside the scope of classical differential theory.

This chapter reviews the neural cryptanalysis literature for SPECK, beginning with Gohr's construction and moving through a sequence of architectural refinements that have continued to push

accuracy higher.

5.2 The Differential Distinguisher Framework

Definition 5.1 (Neural Differential Distinguisher) *A neural differential distinguisher for an r-round block cipher E with input difference Δ_{in} is a parametric classifier $f_\theta : \{0,1\}^{2n} \to [0,1]$ trained on two classes of ciphertext pairs:*

- ***Real pairs (label 1):*** *$(C, C') = (E_K(P), E_K(P \oplus \Delta_{in}))$ for uniformly random (K, P).*
- ***Random pairs (label 0):*** *(C, C') drawn uniformly at random from $\{0,1\}^{2n}$.*

The distinguisher outputs a confidence score $s = f_\theta(C, C') \in [0,1]$, assigning the pair to the "real" class when $s \geq 0.5$.

A classifier that achieves 50% accuracy is indistinguishable from a fair coin toss; every percentage point above 50% reflects a measurable statistical regularity in the cipher's output.

Definition 5.2 (True Positive Rate (TPR) and True Negative Rate (TNR))

$$TPR = P(\textit{predict real} \mid \textit{actually real})$$
$$TNR = P(\textit{predict random} \mid \textit{actually random})$$

Overall accuracy $= (TPR + TNR)/2$ for balanced datasets.

5.3 Gohr's Neural Distinguisher (CRYPTO 2019)

5.3.1 Input Representation

Gohr's experimental setup targets SPECK32/64, where each ciphertext consists of two 16-bit words. A ciphertext pair is therefore described by four 16-bit values (C_l, C_r, C'_l, C'_r), which are concatenated to form a 64-bit input vector and then arranged as a 4×16 binary matrix – one row per word.

The chosen input difference $\Delta_{\text{in}} = (0x0040, 0x0000)$ undergoes a deterministic transition in round

1:

$$\begin{aligned} 0x0040 \ggg 7 &= 0x8000 \quad \text{(deterministic rotation)} \\ 0x8000 \boxplus 0x0000 &= 0x8000 \quad \text{(no carry)} \\ 0x0000 \lll 2 &= 0x0000 \\ 0x0000 \oplus 0x8000 &= 0x8000 \end{aligned}$$

Because the round-1 output difference $(0x8000, 0x8000)$ is achieved with probability 1, the Hamming weight of the intermediate state is fully determined, giving the network a maximally informative starting condition.

5.3.2 Network Architecture

The distinguisher is built on a deep residual network (ResNet) [15] with three main stages:

Initial convolution: A single 1D convolutional layer (32 filters, kernel size 1) with batch normalisation and ReLU maps the 4×16 input to a 32×16 feature tensor.

Residual tower: Ten consecutive residual blocks, each containing two 1D convolutions (kernel size 3, zero-padding, 32 filters each) with batch normalisation and ReLU, connected by a skip connection:

$$\text{output} = F(x) + x \tag{5.1}$$

where F is the two-layer convolution pathway.

Classification head: After global flattening to 512 dimensions, three fully connected layers $(512 \to 64 \to 64 \to 1)$ with batch normalisation, ReLU, and a sigmoid output produce the binary classification score.

The model totals roughly 100,000 parameters. It was trained for 200 epochs on 10 million ciphertext pairs using the Adam optimiser with a cyclic learning rate schedule.

5.3.3 Results

Table 6 reports the accuracy, TPR, and TNR of Gohr's distinguisher against the classical DDT baseline across rounds 5 through 8 [10].

Table 6: Gohr's neural vs classical distinguisher accuracy for SPECK32/64 [10]

Rounds	Classical (DDT)			Neural (ResNet)		
	Acc.	**TPR**	**TNR**	**Acc.**	**TPR**	**TNR**
5	0.911	0.877	0.947	0.929	0.904	0.954
6	0.758	0.680	0.837	0.788	0.724	0.853
7	0.591	0.543	0.640	0.616	0.533	0.699
8	0.512	0.496	0.527	0.514	0.519	0.508

For rounds 5, 6, and 7 the neural classifier consistently edges out the DDT baseline. The advantage widens at 6 and 7 rounds, where the neural network is evidently leveraging structural information in the ciphertext values that the classical difference-only approach discards.

5.3.4 Key Recovery Attack

Gohr extended this to a full key-recovery attack against 11-round SPECK32/64 by using the neural distinguisher as a scoring oracle within a standard differential key-recovery loop:

1. Collect $2^{14.5}$ ciphertext pairs encrypted under the target key, each derived from a plaintext pair with difference $\Delta_{\text{in}} = 0x0040/0000$.
2. For every candidate assignment to the four last-round subkeys, peel off the final four rounds.
3. Feed the peeled-back pair to the 7-round neural distinguisher and record the score.
4. Rank candidates by score and select the highest-scoring one.

The resulting attack runs in time 2^{38}, a reduction of 2^8 compared with the prior best classical result [8] at 2^{46}.

5.3.5 Interpretability: The M-ODT Framework

Benamira et al. [11] showed that Gohr's network primarily exploits the differential distribution at the *penultimate* and *antepenultimate* rounds (not just round r), and uses absolute ciphertext values beyond the XOR difference. They proposed transforming input to four derived quantities:

$$\Delta L = C_l \oplus C'_l, \qquad \Delta V = (C_l \oplus C_r) \oplus (C'_l \oplus C'_r) \tag{5.2}$$

$$V_0 = C_l \oplus C_r, \qquad V_1 = C'_l \oplus C'_r \tag{5.3}$$

This is essentially what Gohr's first convolutional block computes. The *Masked ODT* (M-ODT) compresses the full output distribution table using binary masks extracted via attribution methods,

and a gradient-boosted classifier on the M-ODT features achieves accuracy within 0.6% of the full neural network [11]. This confirms the neural distinguisher is, in essence, an efficient compressed lookup of differential output distributions.

5.4 Improved Neural Distinguishers

Gohr's work sparked a productive line of research aimed at pushing the accuracy ceiling higher. This section reviews the three most consequential architectural refinements.

5.4.1 Multiple Ciphertext Pairs

Chen et al. [14] observed that using a single ciphertext pair per inference query wastes available information. By feeding P pairs generated from the same key and input difference as a combined sample, the network sees a richer statistical signal. With P pairs the input grows to a $P \times 96$ matrix; Table 7 records how classification accuracy scales with P.

Table 7: Accuracy vs. number of input pairs for SPECK32/64 [14]

Rounds	$P = 1$ **(Gohr)**	$P = 2$	$P = 4$	$P = 8$
5	92.9%	97.39%	99.14%	99.91%
6	78.8%	86.67%	93.58%	95.28%
7	61.6%	63.96%	68.47%	70.09%

5.4.2 Attention Mechanism (Deng et al., 2023)

Deng et al. [12] added a multi-head self-attention module (4 heads) after Gohr's residual tower. The attention operation $\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{d_k})V$ allows the model to focus on the most cryptographically biased bit positions. Results in Table 8 show consistent gains, most pronounced at higher rounds.

Table 8: Accuracy of attention-based distinguisher vs. baselines [12]

Rounds	P	Gohr	Chen	Deng (Attn)	Gain
5	1	92.9%	–	93.44%	+0.54%
5	8	–	99.91%	99.91%	+0.00%
6	1	78.8%	–	79.16%	+0.36%
6	8	–	95.28%	97.81%	+2.53%
7	1	61.6%	–	61.69%	+0.09%
7	8	–	70.09%	72.80%	+2.71%

The gains accumulate most quickly at lower round counts but remain meaningful even at 7 rounds, where every fraction of a percent in accuracy matters for a practical attack.

5.4.3 Inception Module-Based Architecture (Yue and Wu, 2023)

Yue and Wu [13] proposed two improvements: (1) an enhanced data format incorporating partial decryption information without requiring the round key, and (2) Inception-style multi-scale convolutions replacing standard residual blocks.

From a ciphertext pair (C_l, C_r, C'_l, C'_r), they recover key-independent approximations of the previous-round right branch: $C\_r = (C_l \oplus C_r) \ggg \beta$ and compute $d\_l = ((C_l \ominus C\_r) \oplus (C'_l \ominus C\_r')) \lll \alpha$. The 7-tuple input $(C\_r, C\_r', d\_l, C_l, C_r, C'_l, C'_r)$ incorporates one extra round of information.

The Inception module $\text{Inception}(x) = \text{Conv}_{1\times1}(x) \| \text{Conv}_{2\times2}(x) \| \text{Conv}_{4\times4}(x)$ simultaneously captures local and global patterns. Results are in Table 9.

Table 9: Accuracy of Yue and Wu's improved distinguisher for SPECK32/64 [13]

Rounds	Gohr	Hou [21]	Zhang [22]	Yue & Wu
6	78.50%	97.67%	99.92%	**99.97%**
7	59.10%	70.74%	89.63%	**97.13%**

Achieving 97.13% accuracy on 7-round SPECK32/64 is a qualitative leap: no classical distinguisher comes close to this figure for the same round count. Yue and Wu further exploit this to build a key-recovery attack on 8-round SPECK32/64 with a reported success rate of 92%.

5.5 Batch-Based Distinguishers

Benamira et al. [11] observed that inference on a single ciphertext pair makes poor use of the information available when multiple queries can be made under the same key. Presenting a batch of B pairs simultaneously allows the model to accumulate evidence before making a decision – analogously to a hypothesis test with B observations rather than one.

Two aggregation strategies were compared:

1. **Averaging:** Run the single-pair distinguisher independently on each element of the batch and take the median output.
2. **2D-CNN:** Train a two-dimensional convolutional network directly on the $B \times 64$ input matrix, so the model can capture correlations across pairs.

Table 10 shows how accuracy grows with batch size B for the simpler averaging strategy:

Table 10: Batch-based distinguisher accuracy for SPECK32/64 (averaging method) [11]

Rounds	$B = 1$	$B = 5$	$B = 10$	$B = 50$	$B = 100$
5	92.9%	99.8%	**100%**	–	–
6	78.6%	95.4%	99.0%	**100%**	–
7	61.2%	73.5%	80.8%	96.7%	**99.7%**

With $B = 100$, the 7-round distinguisher reaches 99.7% accuracy – a figure that would have been considered unattainable by classical methods for this round count. The scaling is consistent across all tested round counts, confirming that query aggregation is a highly effective and general technique.

6 Comparative Analysis and Implications

6.1 Classical vs Neural Approaches

With both families of distinguishers now fully described, this chapter draws a direct comparison and examines what the aggregate results mean for practitioners and cipher designers.

6.1.1 Accuracy and Round Coverage

Table 11 places every major method on a common scale of round count and accuracy for SPECK32/64.

Table 11: Comprehensive comparison of SPECK32/64 distinguishers

Method	Rounds	Accuracy	Data	Reference
Classical DDT	5	91.1%	10^7 pairs	[10]
Classical DDT	6	75.8%	10^7 pairs	[10]
Classical DDT	7	59.1%	10^7 pairs	[10]
Gohr ResNet	5	92.9%	10^7 pairs	[10]
Gohr ResNet	6	78.8%	10^7 pairs	[10]
Gohr ResNet	7	61.6%	10^7 pairs	[10]
Chen (8 pairs)	6	95.3%	10^7 samp.	[14]
Chen (8 pairs)	7	70.1%	10^7 samp.	[14]
Deng (Attn, 8P)	6	97.8%	5×10^6	[12]
Deng (Attn, 8P)	7	72.8%	5×10^6	[12]
Yue & Wu	6	99.97%	10^6 samp.	[13]
Yue & Wu	7	97.13%	10^6 samp.	[13]
Benamira (B=100)	7	99.7%	10^7 pairs	[11]

6.1.2 What Neural Networks Learn

Benamira et al. [11] established that neural distinguishers exploit: (1) differential distributions at rounds $r-1$ and $r-2$, not just the final output; (2) absolute ciphertext values $V_0 = C_l \oplus C_r$ and $V_1 = C'_l \oplus C'_r$ beyond the XOR difference; and (3) truncated differentials—bit patterns holding with high probability for real pairs but not random ones. Taken together, the classifier is effectively performing automated *differential-linear* cryptanalysis [18], simultaneously exploiting differential biases over inner rounds and linear relationships between state components.

6.1.3 Computational Complexity

- **Classical DDT:** $O(|\text{DDT}|)$ precomputation, $O(1)$ per pair (table lookup).
- **Gohr ResNet:** $O(N \cdot n_{\text{train}})$ training; $O(n_{\text{params}}) \approx 2^{17}$ ops per pair online [11].
- **M-ODT:** Comparable precomputation; $O(|\mathcal{R}_M|)$ per pair online.

Neural approaches carry a substantial one-time training cost but deliver rapid online inference at a cost roughly comparable to a large DDT lookup.

6.2 Implications for SPECK Security

The emergence of reliable neural distinguishers has real consequences for how SPECK's security margins should be interpreted:

1. **Reduced-round SPECK32/64 is not safe:** Gohr's attack succeeds against 11 rounds – half the full 22-round cipher – and more recent distinguishers achieve near-perfect accuracy at 7 rounds. Any deployment that uses fewer than the full round count should be treated with caution.

2. **The distinguishing barrier has effectively moved:** With batch-based inference, 7-round SPECK32/64 is essentially indistinguishable from broken at 99.7% accuracy. Whether this enables efficient key recovery at higher round counts is a pressing open question.

3. **Full SPECK's margin appears intact:** No published attack yet covers more than approximately 70% of the designed round count for any variant. The original designers' target security margin [2] has not been undermined by current neural methods.

4. **IoT deployments warrant re-examination:** Devices that reduce SPECK's round count for performance gains may inadvertently enter the range where neural key recovery is practical.

5. **The principle of security by round count needs revisiting:** Historically, a cipher was considered safe if no attack reached beyond 70% of its rounds. Neural methods are narrowing this empirical threshold for small-block instances.

6.3 Comparison of Attack Complexities

Table 12 compares the key recovery attack complexities for SPECK32/64.

Table 12: Key recovery attack comparison for SPECK32/64

Rounds	Time	Data	Method	Reference
11	2^{46}	2^{22}	Classical Differential	[8]
11	2^{38}	$2^{14.5}$	Neural + Differential	[10]
12	2^{63}	2^{31}	Classical Differential	[9]
8	$\approx 2^{10}$	$\approx 2^{7}$	Neural Key Recovery	[13]

7 Conclusion and Future Directions

7.1 Summary of Contributions

This thesis has brought together classical and machine-learning-based perspectives on the differential security of SPECK, providing a unified treatment that bridges theoretical cryptanalysis and modern data-driven approaches. A brief synopsis of each chapter's contribution follows.

Chapter 2 laid out the complete mathematical specification of SPECK – its ARX round function, iterative key schedule, and the full table of parameterised variants – and examined the engineering rationale behind the rotation constant choices and the decision to prioritise software performance over hardware.

Chapter 3 built the theory of differential cryptanalysis from the ground up, establishing the DDT formalism, the modular-addition differential probability theorem of Lipmaa and Moriai, and the partial DDT construction that makes large-scale trail search tractable for word-sized operations.

Chapter 4 described how NMCS – a recursive heuristic search algorithm – was adapted to the trail-finding problem and how the start-from-middle technique substantially extends the reachable round count. Concrete attack complexities for all ten SPECK variants were compiled.

Chapter 5 surveyed the neural cryptanalysis literature systematically: Gohr's ResNet distinguisher, the multi-pair improvement of Chen et al., the attention-augmented model of Deng et al., the Inception-based architecture of Yue and Wu, the M-ODT interpretability analysis of Benamira et al., and batch-based inference.

Chapter 6 placed the two paradigms side by side, quantifying the accuracy gap and its implications for SPECK's practical security level.

7.2 Key Findings

The principal conclusions drawn from this investigation are:

1. Neural distinguishers surpass classical DDT-based methods at every round count tested for SPECK32/64, with the gap widening as more cipher rounds are applied.

2. The neural advantage is not magic: Benamira et al.'s interpretability work shows that these networks learn to look at penultimate- and antepenultimate-round differentials, exploiting the same data that would inform a differential-linear attack – but doing so automatically.

3. The Yue-Wu Inception architecture achieves 97.13% accuracy at 7 rounds, effectively making SPECK32/64 at that round count cryptanalytically transparent from a distinguishing

perspective.

4. Full-round SPECK retains its designed security margin: the most powerful current attacks account for no more than 70% of the full round count in any variant.

5. Batch inference, where multiple ciphertext pairs are aggregated before a decision is made, is an especially effective strategy – accuracy at 7 rounds reaches 99.7% with a batch of 100 pairs.

6. The M-ODT framework demonstrates that neural cryptanalysis models are interpretable, bridging the gap between black-box machine learning and principled cryptanalysis.

7.3 Future Directions

1. **Larger SPECK variants:** Extend neural cryptanalysis beyond SPECK32/64 to SPECK64/128 and SPECK128/256.

2. **Automated architecture search (NAS):** Replace heuristic ResNet/Inception choices with systematically optimised architectures.

3. **Other ARX ciphers:** Apply the methodology to SIMON, ChaCha20, and Threefish.

4. **Counter-measures and formal proofs:** Design lightweight ciphers resistant to neural cryptanalysis, and establish provable security bounds against neural distinguishers.

7.4 Concluding Remarks

The convergence of deep learning and classical cryptanalysis marks a genuinely new chapter in the history of cipher evaluation. Where manual analysis once required a cryptographer to enumerate every possible statistical bias by hand, a trained neural network can discover equivalent – or superior – discrimination criteria from data alone. For SPECK, this has translated into measurable improvements in distinguisher accuracy that no classical technique has been able to replicate.

Yet the full cipher family remains standing. Every known attack falls well short of the full round count, and the security margins that the NSA designers built into SPECK have held up against both classical and learning-based scrutiny. The interpretability work of Benamira et al. offers a reassuring insight: these powerful neural models are not inscrutable oracles; they can be analysed, understood, and ultimately connected back to the classical differential theory from which they emerged.

Looking ahead, the central question is not whether neural methods will improve further – they almost certainly will – but whether the analysis tools available to cipher designers will keep pace. The mathematical framework developed in this thesis provides one part of that toolkit, and the continued dialogue between machine learning and formal cryptanalysis will be essential for maintaining confidence in the lightweight ciphers that secure billions of constrained devices worldwide.

References

[1] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK families of lightweight block ciphers," *IACR Cryptology ePrint Archive*, Report 2013/404, 2013.

[2] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *Proceedings of the 52nd Annual Design Automation Conference (DAC '15)*, pp. 1–6. ACM, 2015.

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

[4] H. Lipmaa and S. Moriai, "Efficient algorithms for computing differential properties of addition," in *Fast Software Encryption – FSE 2001*, Lecture Notes in Computer Science, vol. 2355, pp. 336–350. Springer, 2002.

[5] A. Biryukov and V. Velichkov, "Automatic search for differential trails in ARX ciphers," in *Topics in Cryptology – CT-RSA 2014*, Lecture Notes in Computer Science, vol. 8366, pp. 227–250. Springer, 2014.

[6] A. Biryukov, V. Velichkov, and Y. Le Corre, "Automatic search for the best trails in ARX: Application to block cipher SPECK," in *Fast Software Encryption – FSE 2016*, Lecture Notes in Computer Science, vol. 9783, pp. 289–310. Springer, 2016.

[7] F. Abed, E. List, S. Lucks, and J. Wenzel, "Differential cryptanalysis of round-reduced Simon and Speck," in *Fast Software Encryption – FSE 2014*, Lecture Notes in Computer Science, vol. 8540, pp. 525–545. Springer, 2014.

[8] I. Dinur, "Improved differential cryptanalysis of round-reduced Speck," in *Selected Areas in Cryptography – SAC 2014*, Lecture Notes in Computer Science, vol. 8781, pp. 147–164. Springer, 2014.

[9] A. D. Dwivedi, P. Morawiecki, and G. Srivastava, "Differential cryptanalysis of round-reduced SPECK suitable for Internet of Things devices," *IEEE Access*, vol. 7, pp. 16476–16486, 2019.

[10] A. Gohr, "Improving attacks on round-reduced Speck32/64 using deep learning," in *Advances in Cryptology – CRYPTO 2019*, Lecture Notes in Computer Science, vol. 11693, pp. 150–179. Springer, 2019.

[11] A. Benamira, D. Gerault, T. Peyrin, and Q. Q. Tan, "A deeper look at machine learning-based cryptanalysis," in *Advances in Cryptology – EUROCRYPT 2021*, Lecture Notes in Computer Science, vol. 12696, pp. 805–835. Springer, 2021.

[12] H. Deng, X. Cao, and Y. Cheng, "Attention in differential cryptanalysis on lightweight block cipher SPECK," in *20th Annual International Conference on Privacy, Security and Trust (PST)*, 2023. DOI: 10.1109/PST58708.2023.10320201.

[13] X. Yue and W. Wu, "Improved neural differential distinguisher model for lightweight cipher Speck," *Applied Sciences*, vol. 13, no. 12, pp. 6994, 2023.

[14] Y. Chen, Y. Shen, H. Yu, and S. Yuan, "A new neural distinguisher considering features derived from multiple ciphertext pairs," *The Computer Journal*, 2022. DOI: 10.1093/comjnl/bxac004.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[18] S. K. Langford and M. E. Hellman, "Differential-linear cryptanalysis," in *Advances in Cryptology – CRYPTO '94*, Lecture Notes in Computer Science, vol. 839, pp. 17–25. Springer, 1994.

[19] L. Song, Z. Huang, and Q. Yang, "Automatic differential analysis of ARX block ciphers with application to SPECK and LEA," in *Information Security and Privacy – ACISP 2016*, Lecture Notes in Computer Science, vol. 9723, pp. 379–394. Springer, 2016.

[20] A. Biryukov, A. Roy, and V. Velichkov, "Differential analysis of block ciphers SIMON and SPECK," in *Fast Software Encryption – FSE 2014*, Lecture Notes in Computer Science, vol. 8540, pp. 546–570. Springer, 2014.

[21] Z. Hou, J. Ren, and S. Chen, "Improve neural distinguisher for cryptanalysis," IACR Cryptology ePrint Archive, Report 2021/1017, 2021.

[22] L. Zhang, Z. Wang, and B. Wang, "Improving differential-neural cryptanalysis with inception blocks," IACR Cryptology ePrint Archive, Report 2022/183, 2022.

[23] T. Cazenave, "Nested Monte-Carlo search," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 456–461, 2009.

[24] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An ultra-lightweight blockcipher," in *Cryptographic Hardware and Embedded Systems – CHES 2007*, Lecture Notes in Computer Science, vol. 4727, pp. 450–466. Springer, 2007.

[25] D. Dinu, Y. Le Corre, D. Khovratovich, L. Perrin, J. Großschädl, and A. Biryukov, "Triathlon of lightweight block ciphers for the Internet of Things," *Journal of Cryptographic Engineering*, pp. 1–20, 2015.

[26] K. Fu, M. Wang, Y. Guo, S. Sun, and L. Hu, "MILP-based automatic search algorithms for differential and linear trails for Speck," IACR Cryptology ePrint Archive, Report 2016/407, 2016.