Improved Related-Key Differential Neural Distinguishers for SPN Block Ciphers

Chuchu Ge[1] and Qichun Wang[1*]

School of Computer and Electronic Information

School of Artificial Intelligence, Nanjing Normal University, Nanjing, China

242202048@njnu.edu.cn, qcwang@fudan.edu.cn

March 17, 2026

Abstract

Related-key differential neural distinguishers have recently attracted increasing attention in block-cipher cryptanalysis, yet their construction still relies heavily on cipher-specific manual design. In this paper, we study the systematic construction of related-key differential neural distinguishers for lightweight substitution–permutation network (SPN) block ciphers and propose a unified framework covering difference selection, dataset construction, network architecture, and training and evaluation. Within this framework, we develop a feature-enhancement method that exploits the invertibility of SPN components to derive representations more informative about the final-round internal state, and a sample-enhancement method that reuses each plaintext pair across related keys to derive multiple ciphertext-pair relations, thereby enriching each sample without increasing the plaintext budget. We validate the proposed framework on `SKINNY-64/64` and `PRESENT-64/80`. Experimental results demonstrate that the proposed method can effectively construct multi-round related-key differential neural distinguishers, with accuracy improving consistently as the number of plaintext pairs per sample increases. In particular, for `SKINNY-64/64`, the single-pair setting achieves classification accuracies of 100.0%, 68.2%, and 59.3% for 7, 8, and 9 rounds, respectively, providing, to the best of our knowledge, the first experimental results on related-key differential neural distinguishers for this cipher. For `PRESENT-64/80`, under the four-pair setting, the proposed method achieves competitive distinguishing performance up to 9 rounds, with accuracies of 95.6%, 72.0%, and 53.7% for 7, 8, and 9 rounds, respectively.

*Corresponding author.

1 Introduction

In recent years, wireless sensor networks, radio-frequency identification (RFID) systems, and Internet of Things (IoT) devices have been widely deployed in various application scenarios, leading to the widespread use of resource-constrained devices in practical environments. Because such devices are typically subject to strict limitations in computational capability, memory, and energy consumption, lightweight cryptographic primitives have become an important solution for protecting their data security. Among these primitives, lightweight block ciphers play a particularly important role. In the design of lightweight block ciphers, the Substitution–Permutation Network (SPN) structure is widely adopted because of its simple design, efficient implementation, and suitability for hardware-constrained environments, following the seminal work of Shannon [1]. Representative lightweight SPN block ciphers include `SKINNY` [2] and `PRESENT` [3]. Therefore, the systematic security analysis of lightweight SPN block ciphers is of both theoretical and practical importance.

In the security evaluation of block ciphers, differential cryptanalysis [5], linear cryptanalysis [6], integral cryptanalysis [7], zero-correlation linear cryptanalysis [8], and related methods have been extensively studied. Among them, differential cryptanalysis is one of the most classical and widely used techniques for evaluating block-cipher security [5]. Its core task is typically to identify differential characteristics or differentials with relatively high probability. Early studies on this task relied mainly on manual derivation, which was costly and inefficient. With the development of cryptanalytic techniques, researchers have proposed a variety of automated search tools, including branch-and-bound methods [9], mixed-integer linear programming (MILP) [10], constraint programming (CP) [11, 12], and SAT-based methods [13], thereby significantly improving the efficiency of searching for high-probability differentials. In this sense, relatively mature frameworks for automated analysis have already been established in traditional cryptanalysis.

More recently, the rapid development of deep learning has motivated researchers to explore its application to block-cipher cryptanalysis. At CRYPTO 2019, Gohr first systematically demonstrated the potential of neural distinguishers in block-cipher cryptanalysis, showing that neural networks can directly learn statistical features from ciphertext pairs and thereby distinguish the distribution induced by a real cipher from that of random data [14]. Since then, neural distinguishers have been extended to a wider range of ciphers and attack models, including the single-key setting [15, 16, 17] and the related-key setting [18, 19]. Existing studies have improved neural distinguishers from multiple aspects, such as network architecture design [15, 20, 21], input data representation and dataset construction [22, 23], multi-ciphertext-pair formats [24], and difference-screening strategies [16]. These studies indicate that neural methods can exhibit strong distinguishing capability in certain round settings and have become an active research direction in neural cryptanalysis [25, 26, 27].

Despite these advances, existing studies on neural distinguishers still focus mainly on specialized optimizations for individual ciphers, and many of them

concentrate on the single-key setting [14, 15, 28]. For related-key differential neural distinguishers, especially for lightweight SPN block ciphers as a class of structured algorithms, a unified and systematic construction framework is still lacking [18, 19]. Compared with the relatively mature automated methodologies developed in traditional differential and linear cryptanalysis, the construction of related-key differential neural distinguishers still relies heavily on human expertise. More specifically, this task involves at least the following difficulties. First, in the related-key setting, one must simultaneously select a plaintext difference and a key difference. The candidate space has size 2^{n+k}, which makes direct exhaustive search infeasible, while an unsuitable difference pair may render the subsequent training process ineffective. Second, the round structures of different ciphers vary significantly, and extracting data representations that are more suitable for neural learning often depends on cipher-specific manual design. Third, existing studies on neural distinguishers still focus mainly on cipher-specific optimizations. As a result, although good performance can be achieved for individual ciphers, the resulting distinguishers often lack transferability and generality. In other words, although neural cryptanalysis has already shown strong potential, there is still a clear gap before it can develop into a broadly applicable auxiliary tool for cryptanalysis, comparable to MILP-based or SAT-based methods.

To reduce this excessive dependence on human expertise, this paper studies the systematic construction of related-key differential neural distinguishers for lightweight SPN block ciphers. To this end, we establish a unified framework consisting of four components: difference selection, data representation, network architecture, and training and evaluation. First, based on the bias score and the weighted bias score, we evaluate candidate difference pairs consisting of a plaintext difference and a key difference $(\Delta P, \Delta K)$ and design a difference-selection strategy for the related-key setting by combining low-Hamming-weight preference, structural constraints, and progressive expansion. Second, by exploiting the invertibility of the linear and nonlinear layers in SPN ciphers, we propose a feature-enhancement method that traces ciphertext pairs back as close as possible to the internal state before the final round, thereby exposing more discriminative information in the input features. In addition, by encrypting the same plaintext pair under both keys in a related-key pair and reorganizing the resulting ciphertext pairs, we construct multiple ciphertext pairs with different pairing relationships , thereby increasing the amount of information contained in each training sample without increasing the plaintext budget. Finally, on the model side, we adopt SE-ResNet as a unified backbone and combine automated hyperparameter search with a cyclic learning-rate strategy, resulting in a more general training and evaluation pipeline.

Using this framework, we conduct experiments on two representative lightweight SPN block ciphers, namely `SKINNY-64/64` and `PRESENT-64/80`. The experimental results show that the proposed method can effectively construct multi-round related-key differential neural distinguishers. The overall results are summarized in Table 1. They further show that the distinguishing performance improves steadily as the number of plaintext pairs per sample, denoted by m, increases. In particular, to the best of our knowledge, no related-key differential neural

distinguisher results for SKINNY-64/64 have been reported previously in the literature, and this paper provides the first such experimental results. Under the single-pair setting ($m = 1$), the classification accuracies of the distinguishers for 7, 8, and 9 rounds reach 100.0%, 68.2%, and 59.3%, respectively, all of which are significantly higher than the 50.0% random-guessing baseline. For PRESENT-64/80, we observe that high-quality difference pairs exhibit a pronounced alignment property between the plaintext difference and the key difference, and we obtain competitive experimental results for 7 to 9 rounds, outperforming prior work in multiple settings. These results indicate that related-key differential neural distinguishers for lightweight SPN ciphers can be constructed in a systematic manner while achieving stable and competitive performance with substantially reduced manual tuning.

Contributions. The main contributions of this paper are summarized as follows:

1. We propose a unified framework for constructing related-key differential neural distinguishers for lightweight SPN block ciphers. The framework covers the key stages of difference selection, dataset construction, feature enhancement, sample enhancement, network architecture, and model training and evaluation.

2. We develop a difference-selection strategy for the related-key setting. It uses the bias score and weighted bias score as evaluation criteria, and combines low-Hamming-weight preference, structural constraints, and progressive expansion to effectively reduce the candidate difference space.

3. We propose feature and sample enhancement methods tailored to SPN ciphers. These methods enrich the learnable statistical information contained in each training sample without increasing the plaintext budget, thereby improving the training effectiveness of neural distinguishers.

4. We validate the effectiveness of the proposed framework on SKINNY-64/64 and PRESENT-64/80. In particular, we provide the first experimental results on related-key differential neural distinguishers for SKINNY-64/64, and achieve competitive distinguishing accuracy on PRESENT-64/80, outperforming prior work in multiple settings.

Organization. The remainder of this paper is organized as follows. In Section 2, we introduce the necessary preliminaries, including SPN block ciphers, the basic structures of SKINNY and PRESENT, related-key differential cryptanalysis, and the basic concepts of neural distinguishers. In Section 3, we present the proposed unified framework, including the difference-selection strategy, dataset-construction method, network architecture, and training and evaluation setup. In Section 4, we apply the framework to SKINNY and PRESENT and report the corresponding difference-selection procedures and experimental results. Finally,

Table 1: Summary of the neural distinguishers for SKINNY and PRESENT

Cipher	Round	Source	Accuracy		
			$m = 1$	$m = 2$	$m = 4$
PRESENT (64/80)	7	[22]	0.549	0.550	0.585
		[35]	–	0.574	0.609
		[32]	0.563	0.593	0.635
		[19]	0.719	0.804	0.896
		This work	**0.788**	**0.881**	**0.956**
	8	[35]	–	0.514	0.520
		[16]	0.509	–	–
		[19]	0.570	0.607	0.658
		This work	**0.601**	**0.651**	**0.720**
	9	[19]	–	0.502	0.529
		This work	**0.515**	**0.524**	**0.537**
SKINNY (64/64)	7	This work	**1.000**	**1.000**	**1.000**
	8	This work	0.682	0.766	**0.867**
	9	This work	0.593	0.668	**0.778**

(1) m denotes the number of plaintext pairs contained in each sample.
(2) "–" indicates that the corresponding result was not reported.

in Section 5, we conclude the paper and discuss possible directions for future research.

2 Preliminaries

In this section, we first summarize the main notations used throughout the paper. We then introduce the general structure of SPN block ciphers and briefly describe PRESENT and SKINNY. Finally, we present the basic concepts of related-key differential cryptanalysis and related-key differential neural distinguishers.

2.1 Notations

Table 2 summarizes the main notations used throughout this paper.

2.2 SPN Block Ciphers

A substitution–permutation network (SPN) is a block-cipher structure built from repeated applications of nonlinear and linear transformations, together achieving

Table 2: Main notations used in this paper.

Symbol	Definition
n	Block size of the cipher, in bits.
k	Master-key size of the cipher, in bits.
R	Total number of encryption rounds.
r	Target number of rounds under analysis.
P, P'	Plaintext pair.
C, C'	Ciphertext pair.
K, K'	Related master-key pair, where $K' = K \oplus \Delta K$.
$E_K(\cdot)$	Encryption function under master key K.
X_i, X'_i	Internal states at the beginning of Round i under keys K and K', respectively.
k_i	Round subkey used in Round i.
$\mathcal{R}_i$	Round function of Round i.
ΔP	Plaintext difference, defined by $\Delta P = P \oplus P'$.
ΔC	Ciphertext difference, defined by $\Delta C = C \oplus C'$.
ΔK	Master-key difference, defined by $\Delta K = K \oplus K'$.
ΔX_i	State difference at the beginning of Round i, defined by $\Delta X_i = X_i \oplus X'_i$.
$\mathrm{HW}(\cdot)$	Hamming weight of a binary vector.
$S(\cdot)$	Parallel nonlinear substitution layer.
$s(\cdot)$	Nonlinear mapping of one S-box.
$L(\cdot)$	Linear diffusion layer or permutation layer.
w	Input/output bit-size of one S-box.
u	Number of parallel S-boxes in one substitution layer, where $u = n/w$.
t	Number of sampled plaintext-key pairs used to estimate the bias score.
$\tilde{b}_t^r(\Delta P, \Delta K)$	Estimated bias score of $(\Delta P, \Delta K)$ for r rounds using t samples.
$\widetilde{W}_R^t(\Delta P, \Delta K)$	Weighted bias score accumulated from Round 1 to Round R.
$f(C, C')$	Feature-enhancement mapping applied to a ciphertext pair.
C_0, C_1, C_2, C_3	Four ciphertexts obtained by encrypting one plaintext pair under a related-key pair.
m	Number of plaintext pairs contained in one training sample.
N	Total number of samples in a dataset.
x_i	Input feature vector of the i-th sample.
y_i	Label of the i-th sample, where $y_i \in \{0, 1\}$.
$\mathcal{D}$	Dataset, written as $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$.

the confusion and diffusion principles introduced by Shannon [1]. The SPN block ciphers considered in this paper operate by iteratively updating an n-bit internal state.

In general, each round function typically consists of the following three basic layers, although their execution order may vary from one cipher to another:

- **Substitution layer (S):** The n-bit state is partitioned into u parallel w-bit words, and a nonlinear S-box $s : \mathbb{F}_2^w \to \mathbb{F}_2^w$ is applied independently to each word. This layer is the main source of confusion.
- **Linear layer (L):** An invertible linear transformation, such as a bit permutation, a cell shift, or an MDS-style matrix multiplication, is applied to spread local differences across the state and provide diffusion.
- **Key-addition layer (AK):** A round subkey $k_i \in \{0,1\}^n$ is mixed into the state through a bitwise XOR operation.

Although the exact ordering of these layers depends on the concrete cipher, the state update in round i can be uniformly formulated as:

$$X_{i+1} = \mathcal{R}_i(X_i, k_i) \tag{1}$$

where $\mathcal{R}_i$ denotes the round function, X_i is the internal state at the beginning of round i, and k_i is the corresponding round subkey. In particular, X_0 denotes the input plaintext P. After R rounds of iteration, the final ciphertext is denoted by C.

2.3 Two SPN Block Ciphers

We consider two representative SPN block ciphers in this work.

2.3.1 Brief Description of SKINNY

SKINNY is a family of lightweight tweakable block ciphers proposed by Beierle *et al.* at CRYPTO 2016 [2]. Inspired by the AES design, SKINNY follows the classical SPN framework while being optimized for both hardware and software efficiency. A SKINNY instance is denoted by SKINNY-n/t, where $n \in \{64, 128\}$ represents the block size in bits and the second parameter denotes the tweakey size. In this work, we focus on the SKINNY-64/64 variant, i.e., the version with a 64-bit block size and a 64-bit tweakey.

The internal state of SKINNY is organized as a 4×4 matrix of cells, where each cell has size $s = n/16$ bits (e.g., $s = 4$ when $n = 64$). Throughout this paper, for the later differential-path analysis, we index the 16 state cells from Cell 0 to Cell 15 in row-major order. The round function $\mathcal{R}_i$ of SKINNY consists of the following five operations applied sequentially:

- **SubCells (SC):** an s-bit S-box is independently applied to each state cell to provide nonlinearity;

- **AddConstants (AC):** a round constant is XORed into designated state positions to break structural symmetry;
- **AddRoundTweakey (ART):** the round tweakey is extracted from the tweakey state and XORed only into the first two rows of the state;
- **ShiftRows (SR):** each row j, where $j \in \{0, 1, 2, 3\}$, is cyclically shifted to the right by j positions;
- **MixColumns (MC):** each column is transformed by a lightweight binary diffusion matrix to provide linear diffusion.

The round function of SKINNY is illustrated in Figure 1. The S-box mapping used in SKINNY is listed in Table 3. Since the ART layer affects only the first two rows of the state, this asymmetric round-key injection pattern has an important impact on differential propagation.

In addition to the round function, the tweakey state is updated across rounds through a fixed cell permutation, which determines how key differences propagate into subsequent round tweakeys. For larger tweakey variants, additional LFSR-based updates are also applied. This structured tweakey schedule provides important structural background for the related-key differential analysis developed later in this paper.

The total number of rounds R depends on both the block size and the tweakey size, as summarized in Table 4. Owing to its compact SPN structure, partial round-key injection, and highly regular tweakey schedule, SKINNY serves as a suitable benchmark for studying neural distinguishers in the related-key setting.

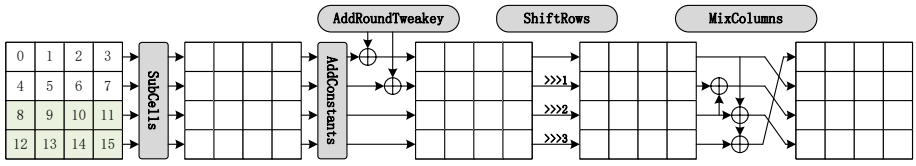


Figure 1: The round function of the SKINNY block cipher.

Table 3: The 4-bit S-box of SKINNY in hexadecimal form.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	6	9	0	1	A	2	B	3	8	5	D	4	E	7	F

2.3.2 Brief Description of PRESENT

PRESENT is a prominent instance of the SPN architecture introduced in Section 2.2. It is an ultra-lightweight block cipher proposed by Bogdanov *et al.* [3]. It is specifically optimized for hardware-constrained environments and operates on a

Table 4: Parameters of different SKINNY variants.

Variant	Block size n	Tweakey size t	Rounds
SKINNY-64/64	64	64	32
SKINNY-64/128	64	128	36
SKINNY-64/192	64	192	40
SKINNY-128/128	128	128	40
SKINNY-128/256	128	256	48
SKINNY-128/384	128	384	56

64-bit state ($n = 64$) with two supported key lengths, namely 80 and 128 bits. The encryption process consists of 31 iterations of a round function, followed by a final key addition. Let X_i denote the internal state at the beginning of Round i. Each round $\mathcal{R}_i$ comprises the following three layers:

- **AddRoundKey (AK):** The 64-bit internal state $X_i = x_{63} \dots x_0$ is combined with the round subkey k_i through a bitwise XOR operation:

$$Y_i = X_i \oplus k_i. \tag{2}$$

- **sBoxLayer (S):** This layer serves as the nonlinear substitution layer. The state Y_i is partitioned into 16 4-bit nibbles, each of which is updated in parallel by a single 4-bit S-box $s : \mathbb{F}_2^4 \to \mathbb{F}_2^4$. The specific mapping of the S-box is listed in Table 5.

- **pLayer (P):** This layer is a bit-level permutation that provides linear diffusion. Each bit at position i is moved to a new position $P(i)$ defined as

$$P(i) = \begin{cases} 16i \pmod{63}, & 0 \le i < 63, \\ 63, & i = 63. \end{cases} \tag{3}$$

Each round of PRESENT applies the operations in the order AK $\to$ S $\to$ P; equivalently, the round transformation can be written as $P \circ S \circ AK$. After 31 rounds, a final post-whitening key k_{32} is applied to produce the ciphertext:

$$C = X_{32} \oplus k_{32}. \tag{4}$$

Since the final step of PRESENT is a key addition and the permutation layer is publicly invertible, these structural properties are particularly relevant to the feature-enhancement procedure introduced later in this paper. The overall encryption process is illustrated in Figure 2.

2.4 Related-Key Differential Cryptanalysis

Differential cryptanalysis, introduced by Biham and Shamir [5], is a fundamental statistical attack that studies how plaintext differences propagate through a block

Table 5: The 4-bit S-box of PRESENT in hexadecimal form.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

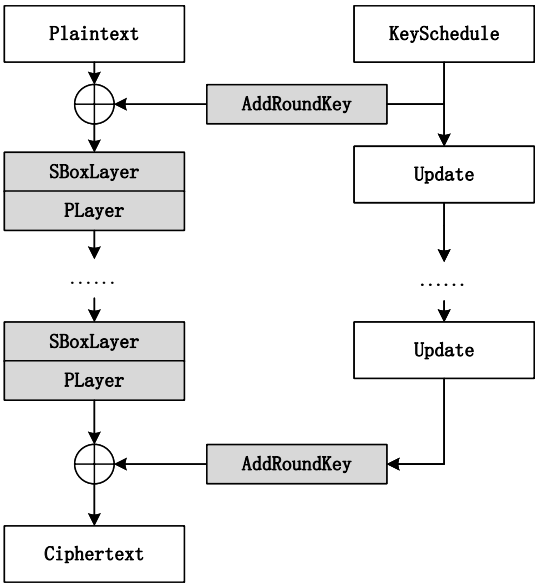


Figure 2: The encryption process of PRESENT.

cipher. In 1994, Biham [30] further proposed related-key cryptanalysis, which evaluates the security of a cipher under two keys with a known mathematical relation. Related-key differential cryptanalysis combines these two ideas by analyzing differential propagation when both the plaintext and the key are assigned specific differences.

Definition 1 (Differences)**.** *For a block cipher, given a plaintext pair* (P, P')*, a ciphertext pair* (C, C')*, and a key pair* (K, K')*, the corresponding plaintext, ciphertext, and key differences are defined as in [9]:*

$$\Delta P = P \oplus P', \qquad \Delta C = C \oplus C', \qquad \Delta K = K \oplus K'.$$

Definition 2 (Related-key differential characteristic)**.** *Given a plaintext pair* (P, P') *and a key pair* (K, K') *with differences* ΔP *and* ΔK*, let* (X_i, X_i') *denote the pair of intermediate states at the beginning of Round* i *under keys* K *and* K'*, respectively. Then an* R*-round related-key differential characteristic, as in [29], is represented by the sequence*

$$(\Delta P, \Delta X_1, \ldots, \Delta X_R),$$

where $\Delta X_i = X_i \oplus X_i'$ *for* $1 \leq i \leq R$.

Definition 3 (Related-key differential probability)**.** *For a block cipher with block size* n *and key size* k*, the related-key differential probability associated with plaintext difference* ΔP*, key difference* ΔK*, and ciphertext difference* ΔC *is defined as in [29]:*

$$DP(\Delta P, \Delta K, \Delta C) = \frac{\#\left\{(x,k) \in \mathbb{F}_2^n \times \mathbb{F}_2^k : E_{k\oplus\Delta K}(x \oplus \Delta P) \oplus E_k(x) = \Delta C\right\}}{2^{n+k}}, \quad (5)$$

where $x \in \mathbb{F}_2^n$ *and* $k \in \mathbb{F}_2^k$.

Definition 4 (Hamming weight)**.** *For a binary vector* $X \in \mathbb{F}_2^n$*, its Hamming weight is defined as the number of nonzero bits in its binary representation, namely*

$$\mathrm{HW}(X) = \sum_{i=0}^{n-1} X_i, \quad (6)$$

where X_i *denotes the* i*-th bit of* X.

2.5 Related-Key Differential Neural Distinguishers

A significant breakthrough in applying deep learning to differential cryptanalysis occurred at CRYPTO 2019. Gohr [14] pioneered an effective framework that utilizes ciphertext pairs to train neural networks to distinguish real cryptographic features from random data. Following this work, Wang *et al.* [31] extended the single-key neural distinguisher to the related-key model. Since then, related-key neural distinguishers have been widely used to evaluate the security of various widely used block ciphers, including algorithms such as Simon, Simeck [18, 33], and PRESENT [34].

2.5.1 Difference Evaluation

In the related-key setting, selecting an appropriate plaintext difference ΔP and master-key difference ΔK is crucial for constructing neural distinguishers. This is because the choice of $(\Delta P, \Delta K)$ directly affects the statistical characteristics present in the generated ciphertext pairs. Bellini *et al.* [16] demonstrated that the weighted bias score of ciphertext differences can approximately indicate the suitability of an input-difference pair for constructing differential neural distinguishers. Therefore, we adopt both the bias score and the weighted bias score as the primary evaluation metrics. The related definitions are provided below.

Definition 5 (Exact bias score)**.** *For an* r*-round block cipher* $E : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$*, let* $(P, P\oplus\Delta P)$ *be a plaintext pair with difference* $\Delta P \in \mathbb{F}_2^n$*, and let* $(K, K\oplus\Delta K)$ *be a key pair with difference* $\Delta K \in \mathbb{F}_2^k$*. The* r*-round exact bias score* $b^r(\Delta P, \Delta K)$ *is defined as follows [32]:*

$$b^r(\Delta P, \Delta K) = \frac{1}{n}\sum_{j=1}^{n}\left|\frac{1}{2} - \frac{1}{2^{n+k}}\sum_{P\in\mathbb{F}_2^n}\sum_{K\in\mathbb{F}_2^k}\big(E_K(P)\oplus E_{K\oplus\Delta K}(P\oplus\Delta P)\big)_j\right|. \quad (7)$$

The exact bias score cannot be computed for practical ciphers, since it requires exhaustive enumeration over all plaintexts and keys. In practice, we therefore use an approximation based on a limited number of samples.

Definition 6 (Bias score)**.** *For an r-round block cipher $E : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$, let (P_i, K_i), for $1 \le i \le t$, denote t sampled plaintext-key pairs. The r-round bias score of plaintext difference ΔP and key difference ΔK, computed from t samples and denoted by $\tilde{b}_t^r(\Delta P, \Delta K)$, is defined as follows [32]:*

$$\tilde{b}_t^r(\Delta P, \Delta K) = \frac{1}{n}\sum_{j=1}^{n}\left|\frac{1}{2} - \frac{1}{t}\sum_{i=1}^{t}\big(E_{K_i}(P_i)\oplus E_{K_i\oplus\Delta K}(P_i\oplus\Delta P)\big)_j\right|. \quad (8)$$

Definition 7 (Weighted bias score)**.** *The R-round weighted bias score of plaintext difference ΔP and key difference ΔK, denoted by $\widetilde{W}_R^t(\Delta P, \Delta K)$, is defined as follows [33]:*

$$\widetilde{W}_R^t(\Delta P, \Delta K) = \sum_{r=1}^{R} r \cdot \tilde{b}_t^r(\Delta P, \Delta K). \quad (9)$$

This cumulative metric prioritizes differences that exhibit significant biases in higher rounds and thus provides a useful indicator of the round range that a neural distinguisher may effectively handle.

2.5.2 The Distinguishing Pipeline

The construction of a related-key neural distinguisher can be summarized as a four-stage pipeline, connecting classical difference analysis with neural-based statistical learning.

Stage 1: Difference Evaluation. The analyst first explores the related-key difference space and evaluates candidate difference pairs $(\Delta P, \Delta K)$. As defined previously, the bias score and the weighted bias score can be used as heuristic criteria for identifying difference pairs with strong statistical potential over multiple rounds. This step provides the basis for selecting suitable input differences for subsequent sample generation and neural training.

Stage 2: Dataset Construction. After fixing a candidate difference pair $(\Delta P, \Delta K)$, a labeled dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ is constructed. Each sample $x^{(i)}$ is derived from ciphertext data generated under the selected plaintext and key differences, while the label $y^{(i)} \in \{0, 1\}$ indicates whether the sample follows the target differential structure. Positive samples are generated according to the

chosen difference pair, whereas negative samples are generated using random or non-targeted difference pairs.

Stage 3: Neural Training. The dataset is used to train a neural distinguisher. The network takes $x^{(i)}$ as input and outputs a prediction score $s^{(i)}$, which is mapped to a binary decision via a threshold τ:

$$\hat{y}^{(i)} = \begin{cases} 1, & s^{(i)} > \tau, \\ 0, & s^{(i)} \leq \tau. \end{cases} \tag{10}$$

During training, the network parameters are optimized so that the distinguisher can learn statistical features that distinguish the target differential distribution from random distributions.

Stage 4: Performance Evaluation. Finally, the effectiveness of the distinguisher is evaluated on an independent validation set. The primary criterion is the classification accuracy, which measures the proportion of correctly classified samples. If the accuracy achieved is consistently higher than the random-guessing baseline of 50%, then the resulting model can be regarded as a valid neural distinguisher.

3 A Generalized Framework for Related-Key Differential Neural Distinguishers on SPN Ciphers

The construction of related-key differential neural distinguishers involves four main stages: difference selection, dataset construction, network architecture design, and training and evaluation. In this section, we present a unified framework for related-key differential neural distinguishers.

3.1 Difference Selection Strategy

To construct an effective related-key differential neural distinguisher, the selected pair of plaintext difference and key difference, denoted by $(\Delta P, \Delta K)$, should yield a sufficiently high weighted bias score after propagation through multiple encryption rounds. A larger weighted bias score indicates that the ciphertext samples generated by this difference pair retain more noticeable statistical features, thereby making the target distribution easier to distinguish from random data during training.

Recent work by Bellini *et al.* [16] has shown that the weighted bias score is an effective criterion for evaluating the suitability of candidate differences. However, identifying a high-quality difference pair in the related-key setting remains computationally challenging. For a block cipher with block size n and key size k, the search space of difference pairs $(\Delta P, \Delta K)$ has size 2^{n+k}. Exhaustively training and evaluating neural distinguishers for all candidates is therefore infeasible. Even evaluating only a very small fraction of this space may

already require an impractical amount of computation, since training a single neural distinguisher typically takes several hours.

Fortunately, high-scoring difference pairs are usually extremely sparse in the full search space. Moreover, cryptographic experience suggests that differences capable of retaining noticeable statistical features through multiple rounds often originate from very low-Hamming-weight patterns. Based on this observation, we adopt a layered search strategy. We begin with the lowest-Hamming-weight candidates and compute their weighted bias scores exhaustively, using the score as an efficient screening criterion to identify promising difference pairs. We then analyze the structural regularities exhibited by these high-scoring candidates and use them to guide the exploration of higher-Hamming-weight spaces.

More specifically, for low-weight candidates, we perform exhaustive evaluation whenever the search space remains manageable. For higher-weight candidates, instead of traversing the entire space without restriction, we constrain the search to structurally meaningful subspaces suggested by the low-weight results, and further use greedy expansion to control the computational cost. In this way, the search process is not a uniform exhaustive scan over all low-weight and high-weight candidates, but a progressively refined procedure: low-weight exhaustive search is used to discover effective structural patterns, and higher-weight search is then restricted to candidates consistent with those patterns. The procedure is summarized in Algorithm 1. Here, "patterns" refers to the structural regularities observed from the top-ranked low-weight candidates, such as preferred active-bit positions, alignment relations between plaintext and key differences, and other cipher-specific constraints used to restrict the high-weight search space.

This strategy enables us to efficiently identify the difference pairs most promising for constructing effective related-key neural distinguishers, while keeping the overall search cost within a practical range. Because the weighted bias score is only a heuristic indicator and does not always perfectly correlate with the final neural-distinguishing accuracy, we use it as a screening criterion rather than the sole selection rule. In practice, after ranking candidates by $\widetilde{W}_R^t(\Delta P, \Delta K)$, we construct preliminary neural distinguishers for the top 10 candidates and select the pair achieving the highest validation accuracy as the final input difference pair.

3.2 Dataset Construction

3.2.1 Feature Enhancement

In neural distinguisher training, the informativeness of the input features directly affects convergence and distinguishing accuracy. Benamira *et al.* [25] showed that the discriminative information exploited by neural networks comes not only from ciphertext differences, but also from internal-state differences closer to the final rounds (e.g., the penultimate round). Meanwhile, as the number of rounds increases, the bias observable from ciphertexts gradually weakens, leading to a decrease in distinguishing accuracy.

Motivated by this observation, we propose a feature-enhancement strategy

Algorithm 1 Procedure of Difference Selection Strategy

Require: Target round number R, sample size t, low-weight threshold w_{low}, maximum total weight $w_{\max}$, population size L

Ensure: A ranked candidate set of promising difference pairs $(\Delta P, \Delta K)$

initial_candidates $\leftarrow \emptyset$
for $w \leftarrow 1$ to w_{low} **do**
  for each $(\Delta P, \Delta K)$ such that $\mathrm{HW}(\Delta P) + \mathrm{HW}(\Delta K) = w$ **do**
    $s \leftarrow \widetilde{W}_R^t(\Delta P, \Delta K)$
    Add $(\Delta P, \Delta K, s)$ to initial_candidates
  end for
end for
Sort initial_candidates by score s in descending order
patterns $\leftarrow$ structural patterns extracted from the top-ranked candidates in initial_candidates
current_population $\leftarrow$ the first L triples in initial_candidates
for $w \leftarrow w_{\text{low}} + 1$ to $w_{\max}$ **do**
  candidates $\leftarrow \emptyset$
  for each $(\Delta P, \Delta K, s)$ in current_population **do**
    expanded_set $\leftarrow$ all candidates obtained by activating one previously inactive bit in either ΔP or ΔK
    subject to $\mathrm{HW}(\Delta P') + \mathrm{HW}(\Delta K') = w$
    Remove candidates in expanded_set that are inconsistent with patterns
    Remove duplicate pairs $(\Delta P', \Delta K')$ from expanded_set
    for each $(\Delta P', \Delta K')$ in expanded_set **do**
      $s' \leftarrow \widetilde{W}_R^t(\Delta P', \Delta K')$
      Add $(\Delta P', \Delta K', s')$ to candidates
    end for
  end for
  Remove duplicate triples in candidates by keeping the highest-scoring one for each distinct $(\Delta P, \Delta K)$
  Sort candidates by score in descending order
  current_population $\leftarrow$ the first L triples in candidates
end for
return current_population

for SPN block ciphers. The main idea is to construct training representations as close as possible to the final-round input state, even when the round key is unknown, so as to expose clearer differential structure to the network.

An SPN round typically involves a nonlinear substitution layer, a linear diffusion layer, and an `AddRoundKey` operation. Since the substitution and linear diffusion layers are public and invertible, we can apply the corresponding inverse transformations (e.g., `InvShiftRows` and `InvMixColumns`) to partially backtrack ciphertext pairs and obtain intermediate states closer to the final rounds, as long

as no unknown round subkey must be crossed. Given a ciphertext pair (C, C'), its difference is denoted by $\Delta C = C \oplus C'$.

Depending on the position of the last-round `AddRoundKey`, feature enhancement can be divided into two cases:

(1) `AddRoundKey` **is not the last step.** We first apply the inverse public round transformations to (C, C') and backtrack to the closest state position that remains computable without knowledge of the round subkey. If `AddRoundKey` affects only part of the state bits, we continue inverse substitution on the unaffected bits to recover earlier state information; if the subkey affects all state bits, we stop backtracking. The resulting enhanced states are denoted by $(\bar{C}, \bar{C}')$, and the enhanced difference is $\Delta\bar{C} = \bar{C} \oplus \bar{C}'$.

(2) `AddRoundKey` **is the last step.** In this case, the absolute states cannot be backtracked beyond the final key addition, so we keep $(\bar{C}, \bar{C}') = (C, C')$. However, since XOR key addition cancels out in the ciphertext difference, we can still apply inverse transformations to ΔC and obtain the enhanced differential feature $\Delta\bar{C}$. Thus, in this case, the enhancement is achieved at the differential level rather than at the absolute-state level.

Therefore, for a ciphertext pair (C, C'), we denote by $f(\cdot)$ the feature-enhancement mapping defined above, i.e., $f(C, C') = (\bar{C}, \bar{C}', \Delta\bar{C})$.

3.2.2 Sample Enhancement

Existing studies have shown that providing more ciphertext information in each training sample often improves the accuracy of neural distinguishers. However, under a limited plaintext budget, simply increasing the number of independent plaintext pairs does not always lead to satisfactory performance, and may still fail to provide sufficient distinguishing performance. Therefore, our goal is to increase the amount of learnable information carried by each training sample without increasing the number of plaintexts.

To this end, we adopt a sample-enhancement method for related-key differential neural distinguishers. By reusing the same plaintext pair under a related-key pair and performing cross-encryption, multiple ciphertext relations can be constructed from the same underlying plaintext data, thereby increasing the amount of useful information available to the network.

Specifically, given a plaintext pair (P, P') with $P \neq P'$ and a related-key pair (K, K') satisfying $K' = K \oplus \Delta K$, we encrypt the same plaintext pair under both keys to obtain four ciphertexts: $C_0 = E_K(P)$, $C_1 = E_K(P')$, $C_2 = E_{K'}(P)$, and $C_3 = E_{K'}(P')$. From one plaintext pair, we therefore obtain two single-key ciphertext pairs, (C_0, C_1) and (C_2, C_3), and four related-key ciphertext pairs, (C_0, C_2), (C_1, C_3), (C_0, C_3), and (C_1, C_2). Hence, under the same plaintext budget, one plaintext pair gives rise to six ciphertext-pair relations derived from the same ciphertext quadruple.

To encode these six ciphertext pairs in a unified representation, we reuse the feature-enhancement mapping $f(\cdot)$ defined in Section 3.2.1. Applying $f(\cdot)$ to the

six ciphertext pairs and concatenating the outputs in a fixed order yields the enhanced representation of one plaintext pair: $f(C_0, C_1) \| f(C_2, C_3) \| f(C_0, C_2) \| f(C_1, C_3) \| f(C_0, C_3) \| f(C_1, C_2)$. Thus, one plaintext pair produces one ciphertext quadruple (C_0, C_1, C_2, C_3) and, after feature enhancement, one concatenated feature vector.

3.2.3 Dataset Generation Workflow

To construct the dataset for related-key differential neural distinguishers, we fix the input difference ΔP and the related-key difference ΔK. Let m denote the number of plaintext pairs contained in each sample, N the total number of samples, and r the number of encryption rounds. The dataset is generated as follows:

(1) **Label generation.** Randomly generate a label vector $\mathbf{Y} = (y_1, \ldots, y_N)$, where $y_i = 1$ denotes a positive sample and $y_i = 0$ denotes a negative sample.

(2) **Key generation.** For each sample, randomly generate one key K, and derive the related key as $K' = K \oplus \Delta K$. The m plaintext pairs within the same sample share the same related-key pair (K, K').

(3) **Plaintext-pair generation.** For each sample, generate m random plaintexts $P_1, \ldots, P_m$. If $y_i = 1$, set $P'_j = P_j \oplus \Delta P$ for each $j \in \{1, \ldots, m\}$ to preserve the target differential structure. If $y_i = 0$, sample each P'_j independently at random to break the target differential structure.

(4) **Encryption.** For each plaintext pair (P_j, P'_j), perform r-round encryption under both K and K' to obtain $C_0 = E_K(P_j)$, $C_1 = E_K(P'_j)$, $C_2 = E_{K'}(P_j)$, and $C_3 = E_{K'}(P'_j)$. Thus, each plaintext pair corresponds to one ciphertext quadruple (C_0, C_1, C_2, C_3).

(5) **Feature-vector construction.** From each ciphertext quadruple, construct the six ciphertext pairs (C_0, C_1), (C_2, C_3), (C_0, C_2), (C_1, C_3), (C_0, C_3), and (C_1, C_2). Apply the feature-enhancement mapping $f(\cdot)$ to each pair and concatenate the results in the fixed order defined above to obtain one plaintext-pair feature vector.

(6) **Dataset assembly.** Concatenate the m plaintext-pair feature vectors belonging to the same sample along the feature dimension to obtain the final sample input x_i, and pair it with the corresponding label y_i. The final dataset is denoted by $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$.

3.3 Network Architecture

To extract both local correlations and global statistical patterns from high-dimensional multi-pair inputs, we employ an SE-ResNet as the backbone of our neural distinguisher. Residual connections alleviate gradient degradation

in deep networks, while the squeeze-and-excitation (SE) mechanism performs channel-wise recalibration to emphasize more informative features. The overall architecture is illustrated schematically in Figure 3 and consists of the following components:

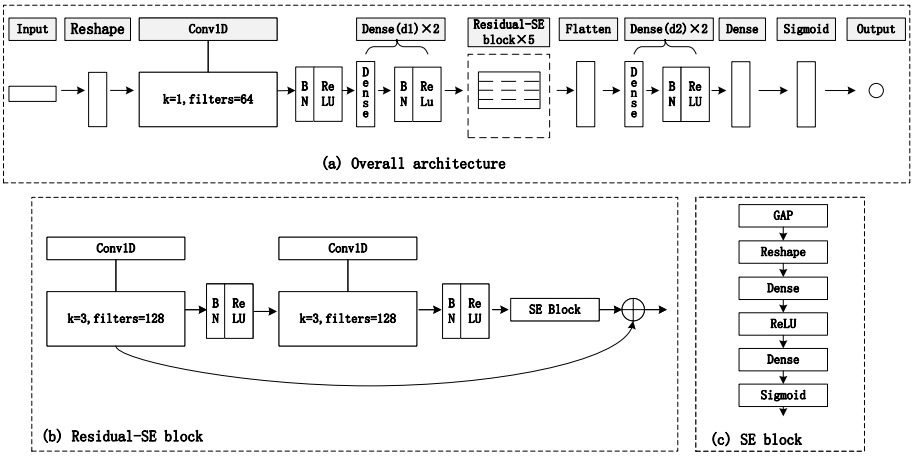


Figure 3: Neural network structure.

- **Input Layer:** For 64-bit block ciphers such as `SKINNY` and `PRESENT`, each sample containing m plaintext pairs is represented as a binary vector of length $1152 \times m$.
- **Reshape Layer:** This layer reshapes the one-dimensional input vector into a tensor of shape $(6m, 192)$, where each plaintext pair contributes six ciphertext-pair relations and each relation is encoded as a 192-bit enhanced feature vector.
- **Conv-1:** An initial Conv1D layer with 64 filters and kernel size 1 is used for feature projection, followed by batch normalization and a ReLU activation function.
- **Dense (d_1) $\times$ 2:** Two dense layers of width d_1 are applied before the residual backbone to further transform and mix the channel-wise features. Each layer is followed by batch normalization and a ReLU activation function.
- **SE-ResNet $\times$ 5:** The main body of the network consists of five residual blocks. Each block contains two Conv1D layers with 128 filters and kernel size 3, each followed by batch normalization and a ReLU activation function. An SE block is then applied to generate channel attention weights through global average pooling and two dense layers. The recalibrated features are multiplied with the convolutional feature maps and merged with the shortcut branch through element-wise addition.

- **Flatten:** This layer flattens the multi-dimensional output of the residual backbone into a one-dimensional feature vector.
- **Dense (d_2) $\times$ 2:** After flattening, the feature vector is passed through a dropout layer and then through two dense layers of width d_2, each followed by batch normalization and a ReLU activation function.
- **Output:** The final layer is a single-neuron classifier with sigmoid activation, which outputs the predicted probability that the input sample belongs to the positive class.

Here, the positive class denotes samples that follow the target differential structure under the real cipher.

3.4 Training and Evaluation

Once the overall neural network architecture is determined, the key issue shifts to hyperparameter selection. Reasonable hyperparameter settings are crucial for improving the convergence speed, classification accuracy, and generalization ability of neural distinguishers. In particular, for deep networks, parameters such as the regularization strength, dropout rate, network depth, convolution kernel size, and number of channels can all significantly affect model performance. By properly setting the L2 regularization coefficient and dropout rate, one can effectively control network complexity, reduce the risk of overfitting, and improve the stability of the model on the validation set.

For hyperparameter selection, we follow the baseline settings reported by Gohr [14] and first define the initial search ranges of the main hyperparameters accordingly. On this basis, we further optimize the parameter space using automated hyperparameter search. Specifically, we employ the Bayesian-optimization-based tool Hyperopt to jointly search over the network depth, convolution kernel size, number of convolution channels, width of the fully connected layers, regularization coefficient, dropout rate, and learning-rate, so as to obtain improved network configurations under an acceptable computational budget. The final hyperparameter settings are listed in Table 6.

For training, we adopt a cyclic learning-rate strategy to improve optimization efficiency. Let n denote the cycle length, and let l_i denote the learning rate at epoch i. The update rule is $l_i = \alpha + \frac{(n-i) \bmod (n+1)}{n} \cdot (\beta - \alpha)$, where α denotes the minimum learning rate and β denotes the maximum learning rate. In our experiments, we typically set $\alpha = 10^{-4}$ and $\beta = 3 \times 10^{-3}$, while the cycle length is adjusted according to the total number of training epochs. This strategy periodically varies the learning rate between α and β, which helps the optimizer escape poor local regions while maintaining stable convergence.

For the optimizer and loss function, we use Adam and MSE, respectively. The former provides good convergence efficiency and robustness, while the latter is used to optimize the agreement between the model output and the target label.

Table 6: Considered hyper-parameters for SKINNY and PRESENT.

Hyper-parameter	**Selection Range**	**Best**
Optimizer	Adam, RectifiedAdam	Adam
LR Schedule	constant, cyclic	cyclic
Loss Function	Binary Cross-entropy, MSE	MSE
Activation Function	ReLU, Leaky ReLU	ReLU
Batch Size	10^3 to $2 \cdot 10^4$	$2 \cdot 10^4$
Epochs	60 to 200	120
Number of Filters	32, 64, 128	64
Residual Depth	3, 5, 7	5
Kernel Size k_1	1, 3	1
Kernel Size k_2, k_3	3, 5, 7	3
Batch Normalization	true, false	true
Regularization	L1, L2, L1 & L2	L2
Regularization Factor	10^{-6} to 10^{-4}	10^{-5}
Dropout Rate	0.1 to 0.5	0.2
SE Ratio	8, 16, 32	16
Dense Width (d_1, d_2)	64, 128, 256	64
Maximum Learning Rate β	10^{-3} to 3×10^{-3}	3×10^{-3}
Minimum Learning Rate α	10^{-4} to 5×10^{-4}	10^{-4}
Cycle Parameter n	10, 12, 15, 29	29

For performance evaluation, we use accuracy as the primary metric for measuring the capability of a neural distinguisher. For a binary classification task, if the model accuracy remains consistently above the 50% random-guessing baseline, this suggests that the model has learned statistical patterns beyond random guessing and therefore possesses distinguishing capability. Hence, an accuracy above 50% serves as a basic indicator of a valid neural distinguisher. At the same time, we also monitor validation-set performance during training to ensure that the resulting distinguisher achieves not only high accuracy but also good generalization ability.

4 Related-key Differential Neural Distinguishers for Round-reduced SKINNY and PRESENT

In this section, based on the framework proposed in the preceding section, we construct related-key differential neural distinguishers for two classical block ciphers, SKINNY and PRESENT.

Table 7: Top-ranked related-key difference pairs for SKINNY-64. The weighted bias score is denoted by $\widetilde{W}_R^t(\Delta P, \Delta K)$.

Plaintext difference ΔP	Key difference ΔK	$\widetilde{W}_R^t(\Delta P, \Delta K)$
0x8000	0x200	4.9921
0x8	0x20000000	4.9689
0x80	0x2	4.8668
0x400	0x100	4.8105
0x4	0x10000000	4.8050
0x800	0x2000000	4.7605
0x40	0x1	4.6990
0x2	0x80000000	4.6257
0x2000	0x800	4.6160
0x1000	0x400	4.6106
0x400	0x1000000	4.6085

4.1 Difference Selection for SKINNY

The differences with Hamming weights of 1 and 2. For the related-key setting of 7-round SKINNY-64, we first exhaustively evaluated all candidate difference pairs satisfying $\mathrm{HW}(\Delta P) = 1$ and $\mathrm{HW}(\Delta K) = 1$, yielding a total of $64 \times 64 = 4096$ candidates. The results show that only 11 candidate pairs achieve weighted bias scores above 4.60, with scores ranging from 4.61 to 4.99, while the vast majority of the remaining candidates score below 4.50. The best candidate pair identified under our search strategy is $(\Delta P, \Delta K) = (\texttt{0x8000}, \texttt{0x200})$, which attains a weighted bias score of 4.99. The corresponding 11 high-scoring candidates are summarized in Table 7. Further examination shows that the other 10 high-scoring pairs do not correspond to independent structural patterns, but rather equivalent instances of the selected pair under the combined effects of linear-layer propagation and key-schedule permutation. For example, the second-ranked candidate pair, $(\texttt{0x8}, \texttt{0x20000000})$, achieves a score of 4.97. It preserves the same alignment pattern as the selected pair, with the only difference being that the active bits are shifted to another column. Therefore, these high-scoring pairs essentially belong to the same structural pattern.

Based on this observation, we further evaluated all candidate pairs satisfying $\mathrm{HW}(\Delta P) = 1, \mathrm{HW}(\Delta K) = 2$ and $\mathrm{HW}(\Delta P) = 2, \mathrm{HW}(\Delta K) = 1$, for a total of $2 \times 64 \times 2016 = 258048$ candidates. The best pair found in this stage is $(\Delta P, \Delta K) = (\texttt{0x9}, \texttt{0x20000000})$, with a weighted bias score of 4.98, which is still slightly lower than the best score of 4.99 obtained in the previous stage. Further examination shows that the top-ranked candidates in this stage still preserve an alignment pattern similar to that of the selected pair. For instance, the second-ranked candidate, $(\texttt{0x9000}, \texttt{0x200})$, has exactly the same key difference as the

selected pair. Meanwhile, $\Delta P =$ `0x9000` can be written as `0x8000` $\oplus$ `0x1000`, meaning that one additional bit is activated within the same nibble (Cell 12) as the selected plaintext difference. This suggests that such high-scoring HW $= 2$ pairs are better understood as local variants of the selected pair, rather than as genuinely new advantageous structures.

The differences with a Hamming weight greater than 2. For related-key differences with higher Hamming weights, continuing a full-space exhaustive search would quickly become computationally prohibitive. We therefore further restricted the search space by incorporating the structural properties of `SKINNY`. Based on the HW $= 1$ and HW $= 2$ results above, we observe that the active bits in the discovered high-scoring pairs are mainly concentrated in bit positions 32 to 63 for both the plaintext difference and the key difference. This distribution has a clear structural explanation. When the plaintext difference is located in the last two rows of the state matrix, it bypasses the subkey injection in Round 1 and, after propagating through the linear layer, becomes aligned in Round 2 with the key difference permuted by the key schedule at a specific cell. Based on this observation, for candidate pairs satisfying HW$(\Delta P) > 2$ or HW$(\Delta K) > 2$, we restrict the search to the structured subspace in which all active bits lie in the last 32 bit positions. Under this constraint, we use the 11 high-scoring pairs identified in the HW $= 1$ stage as initial seeds and continue the search using a greedy expansion strategy that incrementally adds active bits. The results show that no candidate pair found in this structure-constrained high-weight search exceeds the weighted bias score of 4.99 achieved by the selected pair $(\Delta P, \Delta K) =$ (`0x8000`, `0x200`). Combining the low-Hamming-weight exhaustive search and the structure-constrained high-Hamming-weight search, we first retain the top 10 candidates ranked by weighted bias score. We then construct preliminary neural distinguishers for these candidates and observe that $(\Delta P, \Delta K) =$ (`0x8000`, `0x200`) yields the highest validation accuracy. Therefore, we select this pair as the final input difference pair for the related-key setting of `SKINNY`-64.

Structural features of `SKINNY`. To explain why this selected difference pair achieves the highest weighted bias score among the candidates identified in our search, we next analyze its propagation in the first few rounds of `SKINNY`-64. The analysis below shows that the plaintext-difference path and the key-difference path become aligned in Round 2 and cancel each other at Cell 3. In `SKINNY`-64, the internal state can be represented as a 4×4 cell matrix, while round keys are injected only into the first two rows, namely Cells 0 to 7. Taking the selected difference pair $(\Delta P, \Delta K) =$ (`0x8000`, `0x200`) as an example, its propagation and cancellation process in the first two rounds is as follows.

1) **Bypassing the initial key injection.** The plaintext difference $\Delta P =$ `0x8000` is initially located at **Cell 12** (the first cell of the fourth row). Since the first-round key injection only affects rows 0 and 1, ΔP is not

affected by the round key in Round 1 and therefore preserves its sparse active pattern.

2) **Linear-layer propagation in Round 1.** Within the linear layer of Round 1, the active difference at **Cell 12** is shifted three cells to the right by `ShiftRows`, moving it to **Cell 15**. It is then mapped by `MixColumns` to **Cell 3** at the beginning of the next round.

3) **Key-schedule alignment from Round 1 to Round 2.** At the same time, the master-key difference $\Delta K = \texttt{0x200}$, initially located at **Cell 13**, is permuted by `PermuteCells`. According to the `SKINNY` key schedule, the content of **Cell 13** is relocated to **Cell 3**, where it is injected as part of the second-round key.

4) **Cancellation in Round 2.** During the `AddRoundKey` step of Round 2, the plaintext-difference path and the key-difference path meet at **Cell 3**. These two differences are XORed together and cancel each other out, resetting the active cell to zero.

This alignment helps suppress the rapid diffusion of differences in the early rounds, allowing the state to return to a low-Hamming-weight configuration after the initial propagation. Here, ΔS_r denotes the propagated state difference in Round r, and ΔTK_r denotes the corresponding propagated round-key difference derived from ΔK. The round-by-round propagation details are summarized in Table 8.

Table 8: Round-by-round propagation of the selected related-key difference pair for `SKINNY`-64. Cell i denotes the i-th nibble position (0–15).

Round	**State Path** (ΔS_r from ΔP)	**Key Path** (ΔTK_r from ΔK)	**XOR Sum** ($\Delta S_r \oplus \Delta TK_r$)
1	Cell 12	0	Cell 12
2	**Cell 3**	**Cell 3**	**0**
3	0	0	0
4	0	Cell 7	Cell 7

4.2 Application to the `SKINNY` Block Cipher

After determining the optimal difference pair $(\Delta P, \Delta K) = (\texttt{0x8000}, \texttt{0x200})$, we instantiate the generalized dataset-construction workflow for `SKINNY`-64. We set the training set size to $N_{\text{train}} = 2 \times 10^7$ samples and the test set size to $N_{\text{test}} = 2 \times 10^6$ samples. Let each sample contain m plaintext pairs; then the total number of plaintext pairs is $N_{\text{train}} \times m$ for training and $N_{\text{test}} \times m$ for testing. Taking into account the round-function structure of `SKINNY`-64, we instantiate the feature-enhancement and sample-enhancement procedures as follows:

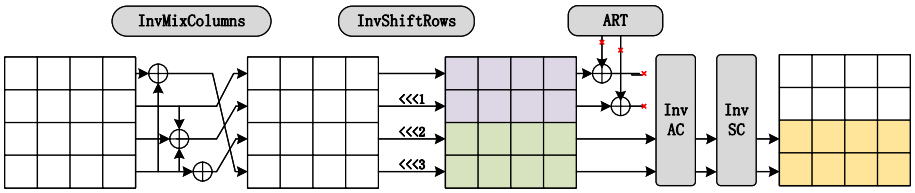


Figure 4: Ciphertext state change of the SKINNY block cipher.

1) Selective state backtracking for SKINNY. The backtracking process of the ciphertext state is illustrated in Figure 4. For any ciphertext pair (C, C'), we first apply `InvShiftRows` and `InvMixColumns` to invert the linear diffusion layer and undo the linear diffusion of the last round. We then perform selective inverse substitution. Since the subkey of SKINNY affects only the first and second rows of the state matrix, we continue to apply the inverse S-box to the third and fourth rows, which are unaffected by the unknown subkey, so as to recover internal-state information closer to the final rounds. For the first and second rows affected by the subkey, the backtracking process stops before `AddRoundKey`. This yields the enhanced state pair $(\bar{C}, \bar{C}')$ and the corresponding enhanced difference $\Delta\bar{C}$, i.e., the feature-enhancement mapping $f(C, C')$ defined in Section 3.2.1.

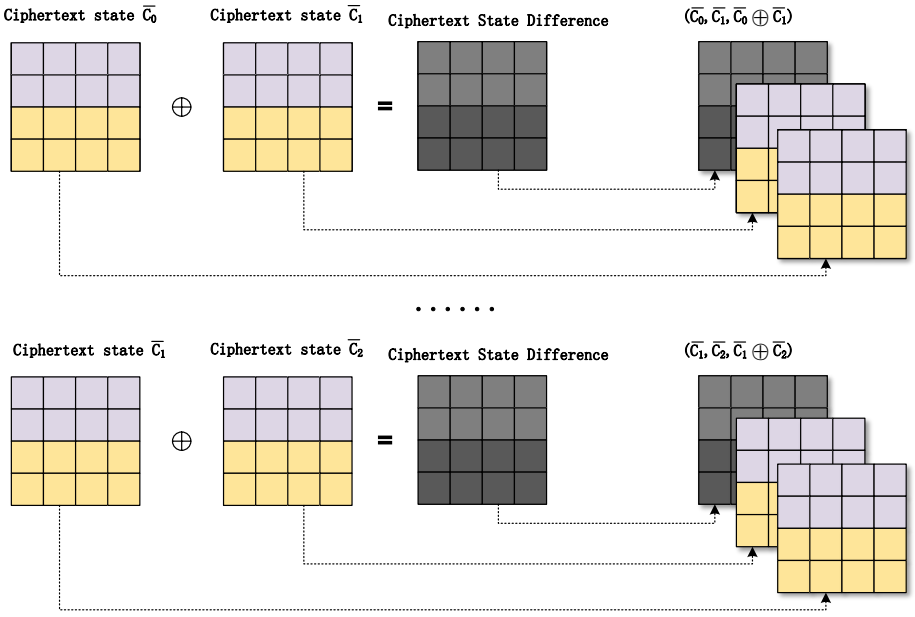


Figure 5: Data format composition for SKINNY-64/64.

2) Multi-pair input construction. To increase the amount of information extracted from a single plaintext pair, we perform cross-encryption on the same plaintext pair under the related-key pair, producing one ciphertext quadruple (C_0, C_1, C_2, C_3). From this quadruple, six ciphertext pairs are constructed: (C_0, C_1), (C_2, C_3), (C_0, C_2), (C_1, C_3), (C_0, C_3), and (C_1, C_2). The first two are single-key pairs, whereas the remaining four are related-key pairs. We apply the feature-enhancement mapping $f(\cdot)$ to each ciphertext pair and concatenate the outputs in a fixed order to obtain the feature vector associated with this plaintext pair. The data-format construction process is illustrated in Figure 5.

3) Single-sample input size. For SKINNY-64, the output length of the feature-enhancement mapping $f(C, C') = (\bar{C}, \bar{C}', \Delta\bar{C})$ is $64+64+64 = 192$ bits. Since each ciphertext quadruple yields six ciphertext pairs, the feature vector associated with one plaintext pair has length $6 \times 192 = 1152$ bits. When each sample contains m plaintext pairs, concatenating these feature vectors along the feature dimension yields the final network input of length $1152 \times m$ bits.

Under the above input representation, we conducted extensive training and testing on SKINNY across different numbers of rounds and values of m. The experimental results of the corresponding neural distinguishers are summarized in Table 9.

Table 9: Experimental results for SKINNY-64/64.

Cipher	Round	Type	m	TPR	TNR	Accuracy
SKINNY-64/64	7	RK	1	1.000	1.000	**1.000**
			2	1.000	1.000	**1.000**
			4	1.000	1.000	**1.000**
	8	RK	1	0.628	0.736	**0.682**
			2	0.719	0.813	**0.766**
			4	0.844	0.889	**0.867**
	9	RK	1	0.190	0.996	**0.593**
			2	0.343	0.993	**0.668**
			4	0.569	0.987	**0.778**

4.3 Difference Selection for PRESENT

Similar to the analysis of SKINNY, we further investigate the applicability of low-Hamming-weight input differences for constructing neural distinguishers for PRESENT. For the related-key setting of 7-round PRESENT-64/80, we first

Table 10: Top-ranked related-key difference pairs for PRESENT-64/80. The weighted bias score is denoted by $\widetilde{W}_R^t(\Delta P, \Delta K)$.

Plaintext difference ΔP	Key difference ΔK	$\widetilde{W}_R^t(\Delta P, \Delta K)$
0x10	0x100000	6.8132
0x8	0x80000	6.3798
0x80	0x800000	6.3453
0x8000	0x80000000	6.1042
0x40	0x400000	6.0971
0x800	0x8000000	6.0822
0x100	0x1000000	6.0480
0x1000	0x10000000	6.0192
0x10000	0x100000000	5.9516
0x4000	0x40000000	5.8583
0x400	0x4000000	5.8005
0x40000	0x400000000	5.6436
0x20	0x200000	5.6408
0x200	0x2000000	5.6325
0x2000	0x20000000	5.4737
0x20000	0x200000000	5.4164

exhaustively evaluated all candidate difference pairs satisfying $\mathrm{HW}(\Delta P) = 1$ and $\mathrm{HW}(\Delta K) = 1$, yielding a total of $64 \times 80 = 5120$ candidates. The results reveal a clear structural pattern among the high-scoring candidates: significantly larger weighted bias scores are obtained only when the plaintext and key differences are active at the same bit position, i.e., $\Delta P = e_i$ and $\Delta K = e_i$, where e_i denotes a bitstring with only the i-th bit set to 1, indexed from the most significant bit. In our symmetric setting, this means that only the first 64 key bits are aligned with the plaintext bit positions in the initial whitening step, while the remaining 16 key bits are fixed to zero. In the interval $i \in [45, 60]$, 16 candidate pairs satisfying this symmetric relation achieve weighted bias scores between 5.40 and 6.81. The highest scores are obtained at $i = 59$ and $i = 60$, reaching approximately 6.81 and 6.38, respectively, as shown in Table 10. By contrast, for $i \in [0, 44] \cup [61, 63]$, the remaining 48 symmetric candidates only achieve scores between 3.30 and 5.40. For all asymmetric single-bit candidates, i.e., $\Delta P = e_i$ and $\Delta K = e_j$ with $i \neq j$, the scores drop significantly and all remain below 4.00. Overall, among all single-bit configurations examined in our experiments, every candidate with a weighted bias score above 4.00 follows this symmetric pattern.

To further explore the low-weight neighborhood, we then systematically traversed all candidate pairs satisfying $\mathrm{HW}(\Delta P) = 1, \mathrm{HW}(\Delta K) = 2$ and $\mathrm{HW}(\Delta P) = 2, \mathrm{HW}(\Delta K) = 1$, for a total of $64 \times \binom{80}{2} + \binom{64}{2} \times 80 = 202240 +$

161280 = 363520 candidates. The results show that once this strict bitwise alignment between the plaintext and key differences is broken, the weighted bias scores no longer exceed the best score found in the single-bit stage, even when additional active bits are introduced. This observation further suggests that, in the related-key setting of PRESENT, strict symmetric alignment between the plaintext and key differences at the input is much more likely to produce strong statistical biases.

Motivated by this strong symmetric regularity observed in the first two stages, we impose a structural restriction when extending the search to candidate pairs satisfying $\mathrm{HW}(\Delta P) = 2$ and $\mathrm{HW}(\Delta K) = 2$. Instead of exhaustively searching the full space of $\binom{64}{2} \times \binom{80}{2}$, we restrict the search to the symmetric subspace suggested by the previously identified high-scoring candidates, namely, $\Delta P = e_i \oplus e_j$ and $\Delta K = e_i \oplus e_j$. Under this restriction, only $\binom{64}{2} = 2016$ candidates need to be evaluated. Among them, 14 candidates achieve weighted bias scores above 6.0, and these high-scoring candidates are again concentrated on indices $i, j \in [46, 60]$. The best score in this stage is 6.64, obtained by the candidate with active positions $(56, 57)$. However, this value is still lower than the best score of 6.81 obtained in the single-bit stage.

Taken together, the results of the three search stages show that effective related-key differences for PRESENT exhibit a strong symmetric structural property: significant statistical biases emerge only when the plaintext and key differences are aligned at the same bit positions. Therefore, we retain the top 10 candidates ranked by weighted bias score and construct preliminary neural distinguishers for them. Among these candidates, $\Delta P = e_{59}$ and $\Delta K = e_{59}$, corresponding to $\Delta P =$ 0x10 and $\Delta K =$ 0x100000, achieves the best validation accuracy and is thus selected as the final input difference pair in the related-key setting of PRESENT-64/80.

4.4 Application to the PRESENT Block Cipher

After determining the optimal difference pair $(\Delta P, \Delta K) =$ (0x10, 0x100000), we instantiate the generalized dataset-construction workflow for PRESENT. We set the training set size to $N_{\text{train}} = 2 \times 10^7$ samples and the test set size to $N_{\text{test}} = 2 \times 10^6$ samples. Let each sample contain m plaintext pairs; then the total number of plaintext pairs is $N_{\text{train}} \times m$ for training and $N_{\text{test}} \times m$ for testing. According to the round-function structure of PRESENT, we instantiate the feature-enhancement and sample-enhancement procedures as follows:

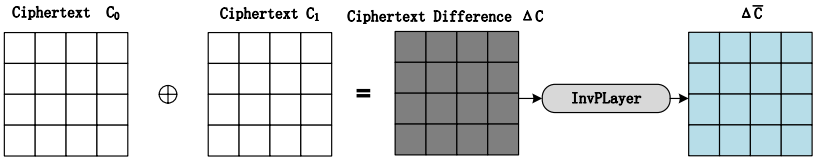


Figure 6: Ciphertext difference state change of the PRESENT block cipher.

1) **Difference-feature backtracking for `PRESENT`.** Unlike `SKINNY`, the last step of a `PRESENT` round is `AddRoundKey` applied to all 64 state bits. Under unknown last-round keys, the absolute states cannot be reliably backtracked; therefore, we keep the original ciphertext pair unchanged, i.e., $(\bar{C}, \bar{C}') = (C, C')$. However, since XOR with the same round key does not affect the pairwise difference, we can directly compute $\Delta C = C \oplus C'$. We then apply only the inverse permutation layer (`InvPLayer`) to ΔC to remove the last-round linear diffusion and obtain the enhanced differential feature $\Delta\bar{C}$, which is closer to the penultimate-round internal state. This gives the `PRESENT`-specific instantiation of the feature-enhancement mapping $f(C, C')$, as illustrated in Figure 6.

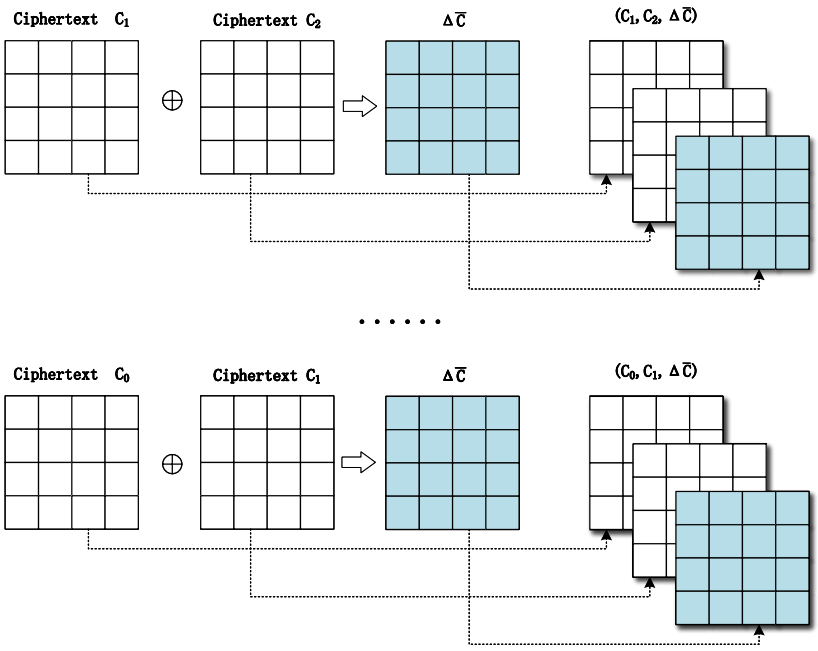


Figure 7: Data format composition for `PRESENT-64/80`.

2) **Multi-pair input construction.** To increase the amount of information extracted from a single plaintext pair, we encrypt the same plaintext pair under two related keys and obtain one ciphertext quadruple (C_0, C_1, C_2, C_3). From this quadruple, six ciphertext pairs are constructed: (C_0, C_1), (C_2, C_3), (C_0, C_2), (C_1, C_3), (C_0, C_3), and (C_1, C_2). The first two are single-key pairs, whereas the remaining four are related-key pairs. We apply the feature-enhancement mapping $f(\cdot)$ to each ciphertext pair and concatenate the outputs in a fixed order to obtain the feature vector associated with one plaintext pair, as shown in Figure 7.

3) **Single-sample input size.** For `PRESENT`, although the enhancement is

achieved at the differential level, we still use the unified representation $f(C, C') = (\bar{C}, \bar{C}', \Delta\bar{C})$. Its output length is therefore $64 + 64 + 64 = 192$ bits. Since each ciphertext quadruple yields six ciphertext pairs, the feature vector associated with one plaintext pair has length $6 \times 192 = 1152$ bits. When each sample contains m plaintext pairs, concatenating these feature vectors along the feature dimension yields the final network input of length $1152 \times m$ bits.

Under the above input representation, we train and evaluate neural distinguishers for `PRESENT` with different numbers of rounds and different values of m. The corresponding results are summarized in Table 11.

Table 11: Experimental results for `PRESENT-64/80`.

Cipher	Round	Type	m	TPR	TNR	Accuracy
`PRESENT-64/80`	7	RK	1	0.714	0.862	**0.788**
			2	0.843	0.917	**0.881**
			4	0.945	0.967	**0.956**
	8	RK	1	0.491	0.711	**0.601**
			2	0.557	0.746	**0.651**
			4	0.653	0.788	**0.720**
	9	RK	1	0.429	0.600	**0.515**
			2	0.425	0.622	**0.524**
			4	0.447	0.627	**0.537**

5 Conclusion

This paper presented a unified framework for constructing related-key differential neural distinguishers for lightweight SPN block ciphers. The proposed framework integrates difference selection, feature enhancement, sample enhancement, network architecture, and training and evaluation into a systematic pipeline, thereby reducing the dependence on cipher-specific manual design in existing approaches. In particular, the framework combines bias-based difference screening with structure-guided search, exploits the invertibility of SPN components to derive features more informative about the final-round internal state, and reuses each plaintext pair across related keys to construct multiple ciphertext-pair relations without increasing the plaintext budget. Together with an SE-ResNet-based architecture, automated hyperparameter search, and a cyclic learning-rate schedule, these components provide a general training pipeline for related-key neural distinguishers on SPN ciphers.

Experimental results on `SKINNY-64/64` and `PRESENT-64/80` demonstrate the effectiveness of the proposed framework. For `SKINNY-64/64`, our method

achieves strong distinguishing performance over 7 to 9 rounds and, to the best of our knowledge, provides the first experimental results on related-key differential neural distinguishers for this cipher. For `PRESENT-64/80`, the results indicate that high-quality input differences exhibit a clear alignment property between plaintext and key differences, and that the resulting distinguishers achieve competitive accuracy across multiple round settings. For both ciphers, the distinguishing accuracy improves consistently as the number of plaintext pairs per sample increases, highlighting the effectiveness of the proposed sample-enhancement strategy.

Future work includes extending the framework to more lightweight SPN ciphers and other attack models, improving automated difference-search strategies for higher-round settings, and developing a deeper theoretical understanding of the statistical features learned by neural distinguishers and their relationship to classical cryptanalytic measures.

References

[1] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[2] C. Beierle, J. Jean, S. Kölbl, G. Leander, G. Peyrin, Y. Sasaki, S. M. Sim, and S. Todo, "The SKINNY family of block ciphers and its low-latency variant MANTIS," in *Advances in Cryptology – CRYPTO 2016*, Lecture Notes in Computer Science, vol. 9815, Springer, Berlin, Heidelberg, 2016, pp. 123–153.

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

[4] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, F. Regazzoni, and A. Poschmann, "Midori: A block cipher for low energy," in *Advances in Cryptology – ASIACRYPT 2015*, Lecture Notes in Computer Science, vol. 9453, Springer, Berlin, Heidelberg, 2015, pp. 411–436.

[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*, Lecture Notes in Computer Science, vol. 765, Springer, Berlin, Heidelberg, 1994, pp. 386–397.

[7] L. Knudsen and D. Wagner, "Integral cryptanalysis," in *Fast Software Encryption – FSE 2002*, Lecture Notes in Computer Science, vol. 2365, Springer, Berlin, Heidelberg, 2002, pp. 112–127.

[8] A. Bogdanov and V. Rijmen, "Linear hulls with correlation zero and linear cryptanalysis of block ciphers," *Designs, Codes and Cryptography*, vol. 70, no. 1, pp. 369–383, 2014.

[9] M. Matsui, "On correlation between the order of S-boxes and the strength of DES," in *Advances in Cryptology – EUROCRYPT '94*, Lecture Notes in Computer Science, vol. 950, Springer, Berlin, Heidelberg, 1995, pp. 366–375.

[10] N. Mouha, Q. Wang, G. Guo, and B. Preneel, "Differential and linear cryptanalysis using mixed-integer linear programming," in *Information Security and Cryptology – Inscrypt 2011*, Lecture Notes in Computer Science, vol. 7537, Springer, Berlin, Heidelberg, 2012, pp. 57–76.

[11] D. Gerault, M. Minier, and C. Solnon, "Constraint programming models for chosen key differential cryptanalysis," in *Principles and Practice of Constraint Programming – CP 2016*, Lecture Notes in Computer Science, vol. 9892, Springer, Cham, 2016, pp. 584–601.

[12] S. Sun, D. Gerault, P. Lafourcade, Q. Yang, Y. Todo, K. Qiao, and L. Hu, "Analysis of AES, SKINNY, and others with constraint programming," *IACR Transactions on Symmetric Cryptology*, vol. 2017, no. 1, pp. 281–306, 2017.

[13] F. Lafitte, "CryptoSMT: A tool for SAT-based cryptanalysis," *IET Information Security*, vol. 12, no. 6, pp. 463–474, 2018.

[14] 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, Springer, Cham, 2019, pp. 150–179.

[15] Z. Bao, J. Guo, M. Liu, L. Ma, and Y. Tu, "Enhancing differential-neural cryptanalysis," in *Advances in Cryptology – ASIACRYPT 2022*, Springer, Cham, 2022, pp. 318–347.

[16] E. Bellini, D. Gerault, A. Hambitzer, and M. Rossi, "A cipher-agnostic neural training pipeline with automated finding of good input differences," *IACR Transactions on Symmetric Cryptology*, vol. 2023, no. 3, pp. 184–212, 2023.

[17] J. Liu, M. Li, J. Ren, et al., "A highly efficient neural distinguisher framework for IoT-friendly lightweight SPN block ciphers," *IEICE Transactions on Information and Systems*, 2025.

[18] J. Lu, G. Liu, B. Sun, et al., "Improved (related-key) differential-based neural distinguishers for SIMON and SIMECK block ciphers," *The Computer Journal*, vol. 67, no. 2, pp. 537–547, 2024.

[19] R. T. Su, J. J. Ren, and S. Z. Chen, "Improved framework of related-key differential neural distinguisher and applications to the standard ciphers," *Cryptology ePrint Archive*, 2025.

[20] G. Wang, G. Wang, and S. Sun, "Investigating and enhancing the neural distinguisher for differential cryptanalysis," *IEICE Transactions on Information and Systems*, vol. 107, no. 8, pp. 1016–1028, 2024.

[21] S. Zhu, L. Li, Z. Hu, et al., "BCS: A neural distinguisher method based on differential propagation uncertainty of nonlinear components and network adaptability," *Physica Scripta*, vol. 100, no. 3, 035008, 2025.

[22] Y. Chen, Y. Shen, H. Yu, et al., "A new neural distinguisher considering features derived from multiple ciphertext pairs," *The Computer Journal*, vol. 66, no. 6, pp. 1419–1433, 2023.

[23] B. Seok and C. Lee, "A novel approach to construct a good dataset for differential-neural cryptanalysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, no. 1, pp. 246–262, 2024.

[24] Y. Hou, J. Liu, S. Han, et al., "Improving deep learning-based neural distinguisher with multiple ciphertext pairs for Speck and Simon," *Scientific Reports*, vol. 15, no. 1, 13696, 2025.

[25] A. Benamira, D. Gerault, T. Peyrin, and Q. Q. Tan, "A deeper look at machine learning-based cryptanalysis," in *Advances in Cryptology – EUROCRYPT 2021*, Springer, Cham, 2021, pp. 805–835.

[26] A. Baksi, "Machine learning-assisted differential distinguishers for lightweight ciphers," in *Classical and Physical Security of Symmetric Key Cryptographic Algorithms*, Springer, Singapore, 2022, pp. 141–162.

[27] A. Bose, D. Pal, and D. Roy Chowdhury, "Deep learning-based differential distinguishers for cryptographic sequences," in *Progress in Cryptology – INDOCRYPT 2024*, Springer Nature Switzerland, Cham, 2024, pp. 114–133.

[28] D. Gerault, A. Hambitzer, M. Huppert, et al., "SoK: 6 years of neural differential cryptanalysis," *Cryptology ePrint Archive*, 2024.

[29] G. Jakimoski and Y. Desmedt, "Related-key Differential Cryptanalysis of 192-bit Key AES Variants," in *International Workshop on Selected Areas in Cryptography*, Lecture Notes in Computer Science, vol. 3006, pp. 208–221, 2003, Springer, Cham.

[30] E. Biham, "New types of cryptanalytic attacks using related keys," *Journal of Cryptology*, vol. 7, no. 4, pp. 229–246, 1994.

[31] G. Wang, S. Sun, and H. Zhang, "Improved machine learning-assisted (related-key) differential distinguishers for lightweight ciphers," in *TrustCom 2021*, IEEE, 2021, pp. 164–171.

[32] A. Gohr, G. Leander, and P. Neumann, "An assessment of differential-neural distinguishers," *Cryptology ePrint Archive*, 2022.

[33] G. Wang, G. Wang, "Enhanced related-key differential neural distinguishers for SIMON and SIMECK block ciphers," *PeerJ Computer Science*, vol. 10, e2566, 2024.

[34] P. S. and G. Mishra, "Related-Key Neural Distinguisher for Round-Reduced PRESENT Cipher," in *Advances in Data-Driven Computing and Intelligent Systems, BITS Pilani, India LNCS 891*, Springer, 2024, pp. 393-405.

[35] L. Zhang, Z. Wang, and Y. Chen, "Improving the accuracy of differential-neural distinguisher for DES, Chaskey, and PRESENT," *IEICE Transactions on Information and Systems*, vol. 106, no. 7, pp. 1240–1243, 2023.