Neural-Inspired Advances in Integral Cryptanalysis

Liu Zhang[2,3], Yiran Yao[3], Danping Shi[1✉], Dongchen Chai[2], Jian Guo[3], and Zilong Wang[2 ✉]

[1] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences shidanping@iie.ac.cn
[2] School of Cyber Engineering, Xidian University, Xi'an, China
liu.zhang@ntu.edu.sg,chaidc@foxmail.com,zlwang@xidian.edu.cn
[3] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore {yiran005@e,guojian@}.ntu.edu.sg

Abstract. The study by Gohr *et al.* at CRYPTO 2019 and sunsequent related works have shown that neural networks can uncover previously unused features, offering novel insights into cryptanalysis. Motivated by these findings, we employ neural networks to learn features specifically related to integral properties and integrate the corresponding insights into optimized search frameworks. These findings validate the framework of using neural networks for feature exploration, providing researchers with novel insights that advance established cryptanalysis methods.

Neural networks have inspired the development of more precise integral search models. By comparing the integral distinguishers obtained via neural networks with those identified by classical methods, we observe that existing automated search models often fail to find optimal distinguishers. To address this issue, we develop a meet-in-the-middle search framework that balances model accuracy and computational efficiency. As a result, we reduce the number of active plaintext bits required for an 11-round integral distinguisher on `SKINNY-64-64`, and further identify a 12-round key-dependent integral distinguisher—achieving one additional round over the previous best-known result.

The integral distinguishers discovered by neural networks enable key-recovery attacks on more rounds. We identify a 7-round key-independent integral distinguisher from neural networks with even only one active plaintext cell, which is based on linear combinations of bits. This distinguisher enables a 15-round key-recovery attack on `SKINNY-n-n` through a strategy with 3 rounds of forward decryption and 5 rounds of backward encryption, improving upon the previous record by one round. The same distinguisher also enhances attacks on `SKINNY-n-2n` and `SKINNY-n-3n`. Additionally, we discover an 8-round key-dependent integral distinguisher using neural network that further reduces the time complexity of key-recovery attacks against `SKINNY`.

Keywords: Neural Network · Feature Explorer · Integral Property · Limited Data · `SKINNY`.

1 Introduction

Integral cryptanalysis, first introduced by Knudsen in [13], analyzes the algebraic structure of a block cipher by computing the sum of ciphertexts derived from a set of plaintexts that span a linear subspace. It tracks how integral properties—`ALL`, `BALANCE`, `CONSTANT`, and `UNKNOWN`—propagate through the internal state words via the cipher's round transformations. The division property, originally proposed in [16], provides a more general and precise framework for identifying integral distinguishers. For a set of texts $\mathbb{X} \subseteq \mathbb{F}_2^n$, the division property is described using a subset of indicator vectors $u \in \mathbb{F}_2^n$, which is divided into two categories: the 0-subset and the *unknown*-subset. To further refine the granularity of integral analysis, the three-subset bit-based division property was introduced in [18]. In this approach, the set of u is partitioned into three subsets: the 0-subset, the *unknown*-subset, and the 1-subset. Also in 2016, Boura and Canteaut introduced the notion of parity sets [6], offering an alternative view of the division property by connecting it with the algebraic normal form (ANF) of the cipher's components. Lambin *et al.* extended the analysis by considering ciphers with linearly equivalent S-boxes and successfully identified high-round integral distinguishers for the `RECTANGLE` [14]. More recently, in FSE 2023, Beyne and Verbauwhede introduced the notion of generalized integral properties, which not only consider mapping-based integral properties on ciphertext sets, but also take into account transformations applied to the plaintext inputs [5]. The above discussions focus on deterministic integral properties. In contrast, probabilistic integral properties, which correspond to the cube tester, were introduced in [1].

One primary challenge in discovering integral distinguishers lies in effectively modeling the propagation of the division property through the round functions of a cipher. The propagation was initially evaluated using a breadth-first search algorithm in [16,18,17], but this approach becomes computationally infeasible for block ciphers with large state sizes. To improve efficiency, Xiang *et al.* introduced the concept of division trails and proposed a MILP-based automatic search method [20]. Sun *et al.* developed an alternative framework based on Boolean Satisfiability Problem (SAT) to analyze ARX ciphers [15]. Wang *et al.* further refined the MILP-based approach by accurately modeling the three-subset division property and incorporating pruning techniques and fast propagation methods [19]. Hu *et al.* introduced the monomial prediction technique, which accelerates the enumeration of monomial trials and enables the recovery of more superpolies [12]. Lambin *et al.* extended ciphers with linear mappings and successfully identified high-round integral distinguishers based on linear combinations of bits for `RECTANGLE` [14]. Beyne and Verbauwhede proposed the use of algebraic transition matrices to search for generalized integral properties and applied this approach to `PRESENT` [5].

Artificial intelligence-based approaches to symmetric cryptanalysis, as demonstrated in works published at CRYPTO 2019 [9], EUROCRYPT 2021 [4], and ASIACRYPT 2023 [2], have shown that neural networks are capable of extracting additional non-random features from limited amounts of data. Consequently, a promising direction is to treat neural networks as auxiliary tools for classical

cryptanalysis. Unlike classical automated methods that require significant manual effort to design and tune models, neural networks can learn novel cryptographic features directly from ciphertext datasets generated by the cipher itself. A beneficial framework is to discover novel insights from neural networks and use them to advance classical cryptanalysis methods. In this work, we prove the effectness of the framework by focusing on the integral properties inherent in ciphertext sets under a given plaintext structure, and improve existing integral results with what we discovered with neural networks.

- **Improving Automated Search Models Inspired by Neural Networks.** We adopt the concept of parity sets to preprocess ciphertext multisets, ensuring that only properties relevant to integral analysis are retained. This allows the neural network to focus exclusively on learning features associated with the integral property. By comparing the integral distinguishers discovered by neural networks with those identified using classical methods under the same number of active plaintext bits, we observe that existing automated search models often fail to produce optimal distinguishers. To address this limitation, we propose a meet-in-the-middle strategy that combines the bit-based division property over two subsets with the monomial prediction technique—effectively mitigating the inaccuracy of the former and the computational overhead of the latter. As a result, our improved automated search model successfully identifies an 11-round key-independent integral distinguisher with fewer active bits for both `SKINNY-64-64` and `SKINNY-128-128`, as well as a 12-round key-dependent integral distinguisher for `SKINNY-64-64`, as shown in Table 1.
- **Enhanced Key-Recovery Attacks Based on Integral Distinguishers.** Beyond the pursuit of longer-round distinguishers, we identify short-round distinguishers that significantly improve key-recovery attacks. In particular, we utilize a 7-round key-independent integral distinguisher to mount a 15-round key-recovery attack on `SKINNY-n-n`, by performing 3 rounds of forward decryption and 5 rounds of backward encryption—improving the previous best result by one round. Since the distinguisher is key-independent, it can also be applied to `SKINNY-n-2n` and `SKINNY-n-3n`. Furthermore, we employ an 8-round key-dependent integral distinguisher to reduce the complexity of key-recovery attacks on `SKINNY`. However, this attack is only effective for a subset of keys and does not apply to the full key space, as summarized in Table 2. We have made our source code publicly available at `https://anonymous.4open.science/r/skinny-automatic-and-AI-analysis-21BA/`..
- **Neural Networks as Feature Explorers for Cryptanalysis.** Unlike classical cryptanalysis methods, which typically rely on precisely characterizing specific non-random properties, neural networks excel at learning features directly from the data distribution. This capability enables them to discover less well-defined features compared with classical methods. By comparing the results of neural networks and automated search models under identical data constraints and analyzing the internal behavior of the trained models,

we are able to uncover feature properties previously ignored by conventional analysis. These insights can, in turn, guide the design of more effective and delicate classical cryptanalytic techniques. While neural networks are inherently limited in their ability to construct long-round distinguishers due to data complexity constraints, they often outperform classical methods in the discovery of short-round distinguishers. Notably, the distinguishers used in our key-recovery attacks are directly discovered through neural networks. These findings underscore the unique and complementary role of neural networks as auxiliary tools in cryptanalysis.

Table 1. The integral distinguisher against SKINNY in single tweakey setting

Cipher	Round	Type	Data	# Balanced Bits	Reference
SKINNY-64-64	11	Key-Independent + Linear Combination of Bits	2^{63}	11	[8]
	11	Key-Independent + Linear Combination of Bits	2^{60}	16	Sect.5.2
	11	Key-Dependent + Nonlinear Combination of Bits	2^{60}	2	Sect.6.3
	12	Key-Dependent + Linear Combination of Bits	2^{60}	3	Sect.6.2
SKINNY-128-128	11	Key-Independent + Linear Combination of Bits	2^{60}	48	Sect.5.2

The structure of the paper is as follows. Section 2 introduces the necessary background used throughout the paper. Section 3 presents the overall motivation and core ideas. In Section 4, we use neural networks to construct short-round integral distinguishers. Section 5 introduces a more precise model to better characterize the learned features. Section 6 proposes key-dependent integral distinguishers. In Section 7, we improve the key recovery attack based on the proposed techniques. Finally, Section 8 concludes the paper.

2 Preliminaries

2.1 Brief Description of SKINNY

SKINNY is a family of tweakable block ciphers [3]. SKINNY-n has a n-bit block size. We define bit numbering from left to right. Let c denote the size of a cell in SKINNY. The state is represented as a 4×4 array, where each cell has a size

Table 2. The integral attack against SKINNY in single tweakey setting

Cipher	Round	Configure	Time	Data	Key Space	Reference
SKINNY-64-64	14	4+6+4	$2^{48.087}$	2^{48}	2^{64}	[3]
	15	3+7+5	$2^{61.976}$	2^{48}	2^{64}	Sect.7.1
	15	3+7+5	$2^{53.926}$	2^{48}	2^{62}	Sect.7.2
SKINNY-64-128	17	3+7+7	$2^{126.421}$	2^{48}	2^{128}	Sect.7.1
	17	3+7+7	$2^{118.388}$	2^{48}	2^{126}	Sect.7.2
SKINNY-64-192	19	3+7+9	$2^{190.695}$	2^{48}	2^{192}	Sect.7.1
	19	3+7+9	$2^{182.671}$	2^{48}	2^{190}	Sect.7.2
SKINNY-128-128	14	4+6+4	$2^{96.006}$	2^{96}	2^{128}	[3]
	15	3+7+5	$2^{125.976}$	2^{96}	2^{128}	Sect.7.1
	15	3+7+5	$2^{109.926}$	2^{96}	2^{126}	Sect.7.2
SKINNY-128-256	17	3+7+7	$2^{254.421}$	2^{96}	2^{256}	Sect.7.1
	17	3+7+7	$2^{238.388}$	2^{96}	2^{254}	Sect.7.2
SKINNY-128-384	19	3+7+9	$2^{382.695}$	2^{96}	2^{384}	Sect.7.1
	19	3+7+9	$2^{366.671}$	2^{96}	2^{382}	Sect.7.2

of c bits. The input state of rth round is denoted by

$$\mathbf{S}_r = \begin{pmatrix} \mathbf{S}_r[0] & \mathbf{S}_r[1] & \mathbf{S}_r[2] & \mathbf{S}_r[3] \\ \mathbf{S}_r[4] & \mathbf{S}_r[5] & \mathbf{S}_r[6] & \mathbf{S}_r[7] \\ \mathbf{S}_r[8] & \mathbf{S}_r[9] & \mathbf{S}_r[10] & \mathbf{S}_r[11] \\ \mathbf{S}_r[12] & \mathbf{S}_r[13] & \mathbf{S}_r[14] & \mathbf{S}_r[15] \end{pmatrix}.$$

For each block size n, SKINNY-n supports three tweakey sizes: $t = n$, $t = 2n$, and $t = 3n$. SKINNY-n-t denotes the version with block size n and tweakey size t. The t-bit tweakey is arranged as a set of t/n 4×4 arrays, denoted by TK_z, where $z \in \{1, \ldots, t/n\}$. Each tweakey undergoes a permutation at the beginning of each round. Additionally, every cell in the first and second rows of TK_2 and TK_3 is individually updated using a linear feedback shift register (LFSR), with the details of the LFSR described in [3]. The round function consists of five operations: SubCells (SB), AddConstants (AC), AddRoundTweakey (AK), ShiftRows (SR), and MixColumns (MC), as illustrated in Fig. 1.

SB substitutes each cell with a c-bit S-box. For SKINNY-64-64 (resp. SKINNY-128-128), c=4 (resp. 8). AC updates the state by XORing it with round constants. AK updates the state by XORing the first two rows of the state with the t/n tweakey arrays. SR rotates the i-th row to the right by i cells. MC multiplies

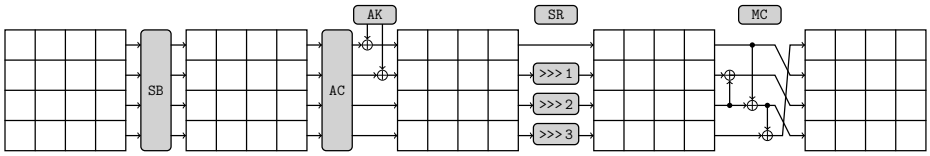

Fig. 1. The round function of SKINNY

each column of the state by a binary matrix:

$$\begin{pmatrix} 1\,0\,1\,1 \\ 1\,0\,0\,0 \\ 0\,1\,1\,0 \\ 1\,0\,1\,0 \end{pmatrix}.$$

2.2 Integral Property

Notation Let $\mathbb{F}_2$ denote the finite field $\{0,1\}$, and let $a = (a_0, a_1, \ldots, a_{m-1}) \in \mathbb{F}_2^m$ represent an m-bit vector, where $a[i]$ denotes the i-th bit of a. The Hamming weight $wt(a)$ is defined as $wt(a) = \sum_{i=0}^{m-1} a[i]$. For any $v \in \mathbb{F}_2^m$ and $v' \in \mathbb{F}_2^m$, we denote $v \succeq v'$ if $v_i \geq v'_i$ holds for all $i = 0, 1, \ldots, m-1$.

The integral property utilises a set of chosen plaintexts in which certain bits take all possible values while the remaining bits are fixed to a constant. The corresponding ciphertexts are then computed using an encryption oracle. If the XOR of all ciphertexts in the set always results in 0, the cypher is said to possess an integral distinguisher. The division property, originally proposed in [16], provides a more precise and generalized method for identifying integral distinguishers.

Bit Product Functions π_u [16]. Let $\pi_u : \mathbb{F}_2^m \to \mathbb{F}_2$ be a function for any $u \in \mathbb{F}_2^m$. Let $x \in \mathbb{F}_2^m$ be an input of π_u, and $\pi_u(x)$ is the AND of $x[i]$ satisfying $u[i] = 1$, namely, it is defined as

$$\pi_u(x) := \prod_{i=0}^{m-1} x[i]^{u[i]}.$$

Definition 1 (Bit-based Division Property using Two Subsets [18]). *A set $\mathbb{X} \in \mathbb{F}_2^m$ has division property $D_{\mathbb{K}}^m$, where $\mathbb{K} \in \mathbb{F}_2^m$ is a set, if for all $u \in \mathbb{F}_2^m$, we have*

$$\bigoplus_{x \in \mathbb{X}} \pi_u(x) = \begin{cases} unknown & \text{if there is } k \in \mathbb{K} \text{ s.t. } u \succeq k, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 2 (Algebraic Normal Form). *The Algebraic Normal Form (ANF) of a Boolean function $f : \mathbb{F}_2^m \to \mathbb{F}_2$ can be expressed as*

$$f(x) = f(x_0, \cdots, x_{m-1}) = \bigoplus_{u \in \mathbb{F}_2^m} a_u \pi_u(x),$$

where $a_u \in \mathbb{F}_2$ *and* $\pi_u(x)$ *is called a monomial. If the coefficient of* $\pi_u(x)$ *in* f *is* 1*, we say* $\pi_u(x)$ *is contained in* f*, denoted by* $\pi_u(x) \to f$*.*

Definition 3 (Monomial Prediction [12]). *If there exists a monomial sequence satisfied*

$$\pi_{u^{(0)}}(x^{(0)}) \to \pi_{u^{(1)}}(x^{(1)}) \to \cdots \to \pi_{u^{(r)}}(x^{(r)}),$$

we call that there is a monomial trail connecting $\pi_{u^{(0)}}(x^{(0)})$ *and* $\pi_{u^{(r)}}(x^{(r)})$*, denoted by* $\pi_{u^{(0)}}(x^{(0)}) \rightsquigarrow \pi_{u^{(r)}}(x^{(r)})$*. If the number of monomial trails connecting* $\pi_{u^{(0)}}(x^{(0)})$ *and* $\pi_{u^{(r)}}(x^{(r)})$ *is odd, then we have* $\pi_{u^{(0)}}(x^{(0)}) \to \pi_{u^{(r)}}(x^{(r)})$*.*

Beyne and Verbauwhede[5] consider a general definition of integral properties that encompasses the original integral properties from [13], division properties[16,18] and the properties from the linearly equivalent S-boxes method of [14], which was further developed by Derbez and Fouque [8].

Definition 4 (Integral Property [5]). *Let* $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ *be a vectorial Boolean function. An integral property for* F *is a pair* $(\mathbb{X}, g)$ *with* $\mathbb{X} \in \mathbb{F}_2^n$ *and* $g : \mathbb{F}_2^m \to \mathbb{F}_2$*, and its evaluation is equal to*

$$\sum_{x \in \mathbb{X}} g(F(x)). \tag{1}$$

3 The Overall Motivation and Core Ideas

In cryptanalysis, distinguishers are constructed based on non-random features observed in plaintext-ciphertext pairs to distinguish ciphertexts from random values. Meanwhile, neural networks are data-driven models capable of classifying inputs by learning discriminative patterns from labeled data. This naturally motivates the idea of integrating neural networks into cryptanalytic methodologies. Recent studies, including CRYPTO 2019 [9], EUROCRYPT 2021 [4], and ASIACRYPT 2023 [2], have demonstrated that neural networks can uncover previously unexplored features, offering novel insights into modern cryptanalysis. Therefore, we propose employing neural networks as an auxiliary tool in integral cryptanalysis.

High-round distinguishers typically require significantly more data, which inherently limits the ability of neural networks to distinguish between ciphertexts and random values in such settings. Consequently, our objective is not to construct high-round distinguishers using neural networks, but rather to extract richer and more informative features from limited data.

Since the features learned by neural networks are entirely derived from data, and interpretability is essential in cryptanalysis, we introduce the concept of parity sets in Section 4.1 to constrain the type of features embedded in the training instances. By encoding input data as vectorial division sequences, we successfully obtain 7-round and 8-round integral distinguishers with low data complexity, as presented in Section 4.3.

In Section 5.1, we compare the distinguishers discovered by neural networks with those found by existing automated search models. This comparison reveals that the latter are not always precise: for a fixed number of active bits, automated approaches may fail to identify optimal distinguishers. Motivated by this observation, we enhance the automated search framework by incorporating the division property with two subsets and the monomial prediction technique. The resulting improved model not only reproduces the distinguishers found by neural networks but also yields longer distinguishers requiring fewer active bits than prior results, as detailed in Section 5.2.

Furthermore, in Section 6.1, we observe that neural networks are capable of learning both deterministic and probabilistic integral features. Our analysis shows that the probabilistic nature of some features stems from key-dependent monomials, where variations in key values influence the presence of integral properties. Leveraging this insight, we extend our search strategy and identify a 12-round key-dependent integral distinguisher for `SKINNY-64-64`, which, to the best of our knowledge, represents the longest known distinguisher for this cipher (see Section 6.2). Most importantly, guided by neural network, we progressively expand the feature space in our search model—from linear combinations of bits to nonlinear combinations—as described in Section 6.3.

While our improved model is designed to identify long-round distinguishers, we emphasize that longer distinguishers do not necessarily lead to better key-recovery attacks. In Section 7.1, we demonstrate a 15-round key-recovery attack on `SKINNY-64-64` using a 7-round distinguisher discovered by a neural network, combined with a structure that performs 3 rounds of forward decryption and 5 rounds of backward encryption. This surpasses the previous best result by one round. Moreover, in Section 7.2, we show that using probabilistic integral distinguishers can further reduce the time complexity of the attack. However, this particular attack is not valid for all key values.

In summary, neural networks serve as a valuable auxiliary tool in cryptanalysis. Although the features in our training data are constrained to those related to integral properties—limiting the discovery of entirely new patterns—neural networks effectively reveal deficiencies in existing automated search models. By addressing these deficiencies, we achieve stronger cryptanalytic results. To a certain extent, this demonstrates that neural networks can significantly contribute to the advancement of classical cryptanalytic techniques.

4 Construct Short-Round Integral Distinguisher Using Neural Network

Neural networks have the potential to learn additional non-random features from ciphertexts beyond what can be extracted by classical cryptanalysis methods. In CRYPTO 2019, Gohr used ciphertext pairs generated from plaintext pairs with a fixed input difference as inputs to a neural network, enabling it to learn difference-related features [9]. Bao *et al.* further demonstrated that neural networks not only learn differential features from ciphertext pairs of SPECK, but

also extract XOR-related information between the left and right branches of the ciphertext [2]. Inspired by these findings, we aim to explore whether neural networks can also discover novel integral properties directly from data.

4.1 Definition of Data Types

Zahednejad *et al.* used multisets as inputs to a neural network, training it to distinguish between ciphertexts with specific integral properties and random values [21]. The positive instances are obtained by encrypting plaintext multisets generated through partial-bit enumeration, while the negative instances are generated by encrypting randomly constructed plaintext multisets. However, training integral distinguishers using neural networks presents several limitations:

1. In classical integral cryptanalysis, extending the length of an integral distinguisher typically requires activating a large number of plaintext bits. However, encrypting such a lot of plaintexts is computationally expensive. Consequently, when training an integral distinguisher using a neural network, only a small number of plaintext bits can be activated, which limits the neural distinguisher to fewer rounds.
2. Since the multisets consist of concrete ciphertext values, they may contain various types of non-random features. Neural networks may unintentionally learn to exploit a mixture of these features during training, whereas the original goal is to guide the network to focus exclusively on integral-related properties.

The data requirements described in Limitation 1 are inherent to cryptanalysis in general—not only integral cryptanalysis, but also differential and linear cryptanalysis face similar challenges. The motivation for introducing neural networks into cryptanalysis is precisely to uncover more non-random features from a limited amount of data. We aim to address Limitation 2 by exploring methods to reduce the influence of unintended non-random features. In CRYPTO 2016, Boura *et al.* proposed the notion of the parity set to characterize the division property of a ciphertext set.

Definition 5 (Parity Set [6]). *Let $\mathbb{X}$ be a set of elements in $\mathbb{F}_2^m$, the parity set of $\mathbb{X}$, denoted by $\mathcal{U}(\mathbb{X})$, is the subset of $\mathbb{F}_2^m$ defined by*

$$\mathcal{U}(\mathbb{X}) = \{u \in \mathbb{F}_2^m : \bigoplus_{x \in \mathbb{X}} x^u = 1\}.$$

We propose leveraging the concept of the parity set to generate training data that is better aligned with the learning of integral-related features using neural networks. In practice, it is generally infeasible to characterize the division property of the entire cipher state in a single step. Instead, the analysis is typically performed at the granularity of individual S-boxes. Consequently, the division property of the full state can be derived by aggregating the division properties of each individual cell. Therefore, we define two types of data formats as follows.

Definition 6 (Division Suquence). *Let $\mathbb{X}$ be a set of elements in $\mathbb{F}_2^c$, the division sequence of $\mathbb{X}$, denoted by $\mathbb{DS}$, is a sequence defined by*

$$\mathbb{DS} = [\bigoplus_{x \in \mathbb{X}} x^u, u \in \mathbb{F}_2^c].$$

Definition 7 (Vectorial Division Suquence). *Let $\mathbb{X}_1, \mathbb{X}_2, \cdots \mathbb{X}_i$ $(1 \leq i \leq \frac{n}{c})$ be the set of elements in $\mathbb{F}_2^c$, the vectorial division sequence of $\mathbb{X}_1, \mathbb{X}_2, \cdots \mathbb{X}_s$, denoted by $\mathbb{VDS}$, is a sequence defined by*

$$\mathbb{VDS} = [\bigoplus_{x \in \mathbb{X}_1} x^u] \parallel [\bigoplus_{x \in \mathbb{X}_2} x^u] \parallel \cdots \parallel [\bigoplus_{x \in \mathbb{X}_i} x^u], u \in \mathbb{F}_2^c.$$

We use the division sequence $\mathbb{DS}$ to describe the integral property of a single cell, and extend this notion to the vectorial division sequence $\mathbb{VDS}$ to represent the collective integral properties of multiple cells.

4.2 The Neural Network Learns the Integral-Related Property

The following procedure enables the neural network to learn integral-related properties by using the (vectorial) division sequence as input.

Data Generation Let N and M denote the sizes of the training set and the test set, respectively. The dataset is generated through the following procedure:

- Label generation. Generate N (resp., M) labels Y for the N (resp., M) instances, with approximately half labeled as 0 (i.e., negative instances) and the other half as 1 (i.e., positive instances).
- Plaintext multiset generation. Use the Linux random number generator to generate N (resp., M) uniformly distributed plaintexts P_i. Then, obtain a plaintext multiset by traversing a subset of plaintext bits while keeping the remaining bits fixed.
- Ciphertext multiset generation. If the label Y is 0, replace the traversed plaintext bits with random values. Use the Linux random number generator to generate N (resp., M) uniformly distributed plaintexts K_i. Encrypt the resulting plaintext multiset using the corresponding key K_i to obtain the ciphertext multiset.
- Instance formatting. Construct the input instance from the ciphertext multiset according to Definition 6 or Definition 7.

Network Architecture We employ a simple three-layer fully connected neural network. The three layers contain 256, 64, and 1 neurons, respectively. Each fully connected layer is followed by a batch normalization layer and a ReLU activation function. Finally, a sigmoid activation function is applied to produce a scalar output in the range $[0, 1]$. If the output is greater than 0.5, the input is classified as a positive instance; otherwise, it is classified as a negative instance.

Training Process The neural network was trained for 20 epochs using a training dataset of size N and a test dataset of size M. The batch size was dynamically adjusted to optimize GPU performance. Optimization was performed using the Adam algorithm with the mean square error (MSE) as the loss function. To improve training efficacy, a cyclic learning rate schedule was adopted, defined as: $l_i = \alpha + \frac{(9-i) \bmod 10}{10} \cdot (\beta - \alpha)$, where $\alpha = 10^{-4}$ and $\beta = 2 \times 10^{-3}$. After each epoch, the model was saved, and the best-performing network was selected based on validation loss for subsequent evaluation on the test set.

Performance Evaluation of the Integral-Neural Distinguisher Accuracy (Acc) serves as the primary metric for evaluating the performance of the integral-neural distinguisher. In addition, the true positive rate (TPR) and true negative rate (TNR) are reported to assess the proportions of correctly classified positive and negative instances, respectively.

4.3 Integral Distinguisher Based on Linear Combination of Bits

In this section, the neural network learns the integral properties in Definition 4 where the function g is linear. As a result, we obtain extended integral distinguishers under a given plaintext structure previously proposed by Lambin *et al.* [14] and further developed by Derbez *et al.* [8], which we refer to as integral distinguisher based on linear combination of bits.

We use the vectorial division sequence $\mathbb{VDS}$ as the input of the neural network to train an integral-neural distinguisher. The size of $\mathbb{VDS}$ is 256, as $u \in \mathbb{F}_2^4$ and there are 16 cells in `SKINNY-64-64`. To account for the computational cost of data generation, the sizes of the training and test sets, N and M, are both set to 10^6. The network architecture and training process are detailed in Section 4.2. The result of the 7-round integral-neural distinguisher for `SKINNY-64-64` is shown in Table 3.

Table 3. The integral-neural distinguisher of 7-round `SKINNY-64-64` using $\mathbb{VDS}$

Activated Plaintext Cell	Acc	TPR	TNR
15th	99.8%	100%	99.6%

Observation of Performance Metrics The true positive rate (TPR) reaches 100%, indicating that the neural network has learned deterministic features, as all positive instances conform to these features. For negative instances, there is a $1 - 0.096 = 2^{-8}$ probability that a negative instance is incorrectly classified as positive. Since negative instances make up half of the dataset, the integral-neural distinguisher misclassifies approximately 2^{-9} of the total dataset, which corresponds to an overall error rate of 0.2% and an accuracy of 99.8%. In other words, the feature learned by the 7-round integral-neural distinguisher is associated with only 8 bits.

The Working Mechanism of the Integral-Neural Distinguisher Once an integral-neural distinguisher is trained, the decision rules it relies on are implicitly fixed. By modifying the input, we can probe the underlying patterns learned by the neural network. Yi Chen *et al.* [7] proposed an algorithm, called the *bit sensitivity test*, to evaluate the impact of individual bits in a given instance on the performance of the neural distinguisher, as outlined in Algorithm 4. We apply the bit sensitivity test to identify which specific bits significantly influence the accuracy of the integral-neural distinguisher. The results are shown in Fig. 2. The horizontal axis represents the ciphertext cell corresponding to each bit in the $\mathbb{VDS}$, while the vertical axis indicates the value of u associated with that bit. The ■ shading highlights the bits for which randomization causes a significant drop in the distinguisher's accuracy.

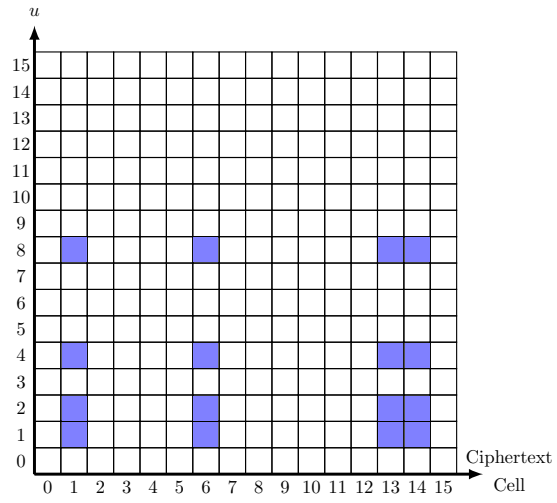

Fig. 2. The bit sensitivity test for the 7-round integral-neural distinguisher

From Fig. 2, it is evident that the 7-round integral-neural distinguisher learns features from 16 bits of the $\mathbb{VDS}$. However, based on the performance metrics, the feature actually utilized by the neural network appears to be related to only 8 bits. To resolve this discrepancy, we extract the values of these 16 bits and observe several interesting phenomena. Specifically, the 4 bits corresponding to the four different u values on ciphertext cells $\{1, 13\}$ are identical. In other words, the 4-bit XOR sum over ciphertext cells $\{1, 13\}$ is always `0b0000`. A similar pattern is observed for ciphertext cells $\{6, 14\}$. Therefore, it can be deduced that the integral-neural distinguisher classifies instances based on this 8-bit pattern. The working mechanism of the 7-round integral-neural distinguisher can thus be summarized as follows:

Pattern 1 *For 7-round* `SKINNY-64-64`*, when the 15th plaintext cell is activated, then* $\bigoplus_{x_1 \in \mathbb{X}_1} \pi_u(x_1) \oplus \bigoplus_{x_2 \in \mathbb{X}_2} \pi_u(x_2) = 0$*, where* $wt(u) = 1$ *and ciphertext cells* $(\mathbb{X}_1, \mathbb{X}_2) \in \{(1, 13), (6, 14)\}$.

Moreover, the Pattern also applies to SKINNY-128-128. We convert Pattern 1 into the representation of a integral distinguisher.

Distinguisher 1 (Key-Independent Integral Distinguisher Based on Linear Combination of Bits Against 7-round `SKINNY-n-n`**).** *When the 15th plaintext cell is activated, the linear combination of bits* $b_i \oplus b_{i+12c}, i \in \{4, 5, 6, 7\}$ *and* $b_i \oplus b_{i+8c}, i \in \{24, 25, 26, 27\}$ *are balanced, where* c *is the size of the S-box.*

Using the same training method and analysis procedure, we obtained an 8-round integral distinguisher based on a linear combination of bits by activating two plaintext cells.

Distinguisher 2 (Key-Independent Integral distinguishers Based on Linear Combination of Bits Against 8-round `SKINNY-64-64`**.)** *If 14th and 15th plaintext cells are activated, then the linear combination of bits* $b_{28} \oplus b_{44} \oplus b_{60}$ *is balanced.*

4.4 Transforming Neural Distinguisher to Boolean Function

We introduce an alternative approach to analyzing the working mechanism of the integral-neural distinguisher. This method is introduced here in preparation for its application in the subsequent sections. The working mechanism of the 7- (or 8-) round integral-neural distinguisher can be understood by directly observing the values of the sensitive bits, primarily because the underlying rules it relies on are relatively simple.

Since both the input and output of the neural network can be interpreted as binary values, the integral-neural distinguisher can be transformed into a Boolean function. By deriving its conjunctive normal form (CNF), we can precisely characterize the rules learned by the neural network. The transformation process is described in Algorithm 1, as shown below.

Explaining the Working Mechanism of the Integral-Neural Distinguisher The size of the $\mathbb{VDS}$ is 256, which makes it infeasible to construct a Boolean function over $\mathbb{F}_2^{256}$. Moreover, according to the results shown in Fig. 2, the features learned by the neural network are related to only 16 bits. Therefore, we retrain the integral-neural distinguisher using these 16 sensitive bits (indexed as $\boldsymbol{b}^u_{cth}$, where cth denotes the index of the cell), and transform the resulting model into a Boolean function, along with its CNF, using Algorithm 1. We then apply the `Espresso` logic minimizer[4] to obtain the minimized CNF corresponding to the 7-round integral-neural distinguisher, which is shown below:

[4] https://github.com/classabbyamp/espresso-logic

Algorithm 1 Get the CNF of the Neural Distinguisher

Input: a trained neural distinguisher $\mathcal{ND}$, the size of the input to the distinguisher d
Output: The CNF of integral-nerual distinguisher $\text{CNF}_{\mathcal{ND}}$

$T_t \leftarrow [\,]$ ▷ Truth table
for input in 0 to 2^d **do**
  if $\mathcal{ND}(\text{input}) > 0.5$ **then**
    add 1 to T_t
  else
    add 0 to T_t
  end if
end for
$\mathcal{BF} = \text{BooleanFunction}(T_t)$ ▷ Construct a boolean function
Get $\text{CNF}_{\mathcal{ND}}$ of $\mathcal{BF}$
return $\text{CNF}_{\mathcal{ND}}$

$(\neg \boldsymbol{b}_1^1 \vee \boldsymbol{b}_{13}^1)$ & $(\boldsymbol{b}_1^1 \vee \neg \boldsymbol{b}_{13}^1)$ & $(\neg \boldsymbol{b}_1^2 \vee \boldsymbol{b}_{13}^2)$ & $(\boldsymbol{b}_1^2 \vee \neg \boldsymbol{b}_{13}^2)$ & $(\neg \boldsymbol{b}_1^4 \vee \boldsymbol{b}_{13}^4)$ & $(\boldsymbol{b}_1^4 \vee \neg \boldsymbol{b}_{13}^4)$ & $(\neg \boldsymbol{b}_1^8 \vee \boldsymbol{b}_{13}^8)$ & $(\boldsymbol{b}_1^8 \vee \neg \boldsymbol{b}_{13}^8)$ & $(\neg \boldsymbol{b}_6^1 \vee \boldsymbol{b}_{14}^1)$ & $(\boldsymbol{b}_6^1 \vee \neg \boldsymbol{b}_{14}^1)$ & $(\neg \boldsymbol{b}_6^2 \vee \boldsymbol{b}_{14}^2)$ & $(\boldsymbol{b}_6^2 \vee \neg \boldsymbol{b}_{14}^2)$ & $(\neg \boldsymbol{b}_6^4 \vee \boldsymbol{b}_{14}^4)$ & $(\boldsymbol{b}_6^4 \vee \neg \boldsymbol{b}_{14}^4)$ & $(\neg \boldsymbol{b}_6^8 \vee \boldsymbol{b}_{14}^8)$ & $(\boldsymbol{b}_6^8 \vee \neg \boldsymbol{b}_{14}^8)$

To make the CNF evaluate to true, all clauses must be satisfied, which implies that the ciphertext must satisfy every clause. Based on this, we infer that the following expressions must all evaluate to 0:

$$\boldsymbol{b}_1^1 \oplus \boldsymbol{b}_{13}^1, \quad \boldsymbol{b}_1^2 \oplus \boldsymbol{b}_{13}^2, \quad \boldsymbol{b}_1^4 \oplus \boldsymbol{b}_{13}^4, \quad \boldsymbol{b}_1^8 \oplus \boldsymbol{b}_{13}^8, \quad \boldsymbol{b}_6^1 \oplus \boldsymbol{b}_{14}^1, \quad \boldsymbol{b}_6^2 \oplus \boldsymbol{b}_{14}^2, \quad \boldsymbol{b}_6^4 \oplus \boldsymbol{b}_{14}^4, \quad \boldsymbol{b}_6^8 \oplus \boldsymbol{b}_{14}^8.$$

This result is consistent with the observed pattern in Pattern 1, which clearly demonstrates the effectiveness of the proposed method.

5 More Precise Automated Search Model

In Sect. 5.1, by comparing existing classical results with those obtained from integral-neural distinguishers, we observe that, for the same number of rounds, integral distinguishers found by existing automated methods typically require more active plaintext bits than those identified by neural networks. This indicates that current automated modeling techniques may lack precision and fail to identify optimal distinguishers. Motivated by this observation, we refine the automated search model to align its results with those of the neural approach. With the improved model, we are able to discover more effective integral distinguishers in Sect. 5.2.

5.1 Comparison of Integral Distinguishers Based on Neural Networks and Classical Methods

In CRYPTO 2016, Beierle *et al.* proposed a 6-round integral distinguisher for both `SKINNY-64-64` and `SKINNY-128-128` by activating one plaintext cell [3]. In 2019, Wening Zhang *et al.* discovered a 7-round integral distinguisher for

SKINNY by activating two plaintext cells [22]. At FSE 2020, Derbez *et al.* obtained an 8-round integral distinguisher based on linear combinations of bits, using 15 active plaintext bits, by employing the Superbox-Sbox technique and linear mappings [8].

Table 4. The integral distinguisher against `SKINNY-64-64`

Round	Method	Type	Data	Reference
6	Classical	Single Ciphertext Bit	2^4	[3]
7	Classical	Single Ciphertext Bit	2^8	[22]
	Neural	Linear Combination of Bits	2^4	Sect.4.3
8	Classical	Linear Combination of Bits	2^{15}	[8]
	Neural	Linear Combination of Bits	2^8	Sect.4.3

Based on the results in Table 4, we present the following observations and conjectures:

- According to [3] and Distinguisher 2 in Section 4.3, the use of different types of distinguishers can increase the number of rounds achievable by integral distinguishers under the same number of activated plaintext bits.
- By comparing the integral distinguisher in [8] with Distinguisher 2 in Section 4.3, both of which are based on linear combinations of bits, we observe a difference in the number of required active plaintext bits.

These discrepancy may be attributed to the lack of precision in the existing automated modeling method.

5.2 Improved Automated Search Model

We attempt to reproduce result in Table 4 using existing modeling methods:

- **Bit-Based Division Property with Two Subsets** [20]: Using this method, we are only able to find a 6-round integral distinguisher when activating 4 plaintext bits.
- **Monomial Prediction Technique** [12]: This method fails due to the high algebraic degree and the numerous copy operations in the `SKINNY` model. The number of candidate monomials becomes so large that it is extremely difficult to determine whether a specific monomial appears in the ANF.

The automated search model based on the bit-based division property with two subsets is simple but lacks precision. In contrast, monomial prediction techniques offer high precision but suffer from high computational complexity. To balance these trade-offs, we developed an automated search model based on the

meet-in-the-middle approach. This method applies backward monomial extension for q rounds and performs forward modeling using the bit-based division property with two subsets for p rounds, thereby enabling the search for a $(p+q)$-round integral distinguisher. The core idea of the improved automated search model is illustrated as follows:

$$\underbrace{\text{Input division property } \boldsymbol{d}_0 \to \mathbb{D}_p}_{p \text{ rounds forward modeling}} \to \textbf{Check} \leftarrow \underbrace{\text{Monomials} \leftarrow \text{Ciphertext bit}}_{q \text{ rounds backward extension}}.$$

Notations Let $\boldsymbol{x} = (x_0, \cdots, x_{63})$, $\boldsymbol{s} = (s_0, \cdots, s_{63})$, $\boldsymbol{b} = (b_0, \cdots, b_{63})$, and $\boldsymbol{k} = (k_0, \cdots, k_{63})$ denote the plaintext, the output after p rounds, the $p+q$ rounds ciphertext, and the key of `SKINNY-64-64`, respectively where bits are indexed from left to right. The encryption from round r_i to r_j is denoted as $E_{\boldsymbol{k}}^{(r_i, r_j)}$, so $\boldsymbol{b} = E_{\boldsymbol{k}}^{(0,p+q)}(\boldsymbol{x}) = E_{\boldsymbol{k}}^{(p,p+q)}(\boldsymbol{s})$. The input division property is denoted by $\boldsymbol{d}_0$, and the set of division properties after p rounds is denoted as $\mathbb{D}_p$.

The MILP-based automated modeling of division properties is already well-established in [20] and will not be elaborated here. The basic procedure of the improved automated search model is presented as follows:

- **q-Round Backward Monomial Extension.** For the linear combination of bits $\mathcal{C}(\boldsymbol{b})$, we recursively expand it in reverse to its ANF over the p-round intermediate variables $\boldsymbol{s}$:

$$\mathcal{C}(\boldsymbol{b}) = \mathcal{C}(E_{\boldsymbol{k}}^{(0,p+q)}(\boldsymbol{x})) = \mathcal{C}(E_{\boldsymbol{k}}^{(p,p+q)}(\boldsymbol{s})) = \sum_i \pi_{\boldsymbol{u}_i}(\boldsymbol{s})\, \pi_{\boldsymbol{v}_i}(\boldsymbol{k}).$$

  We focus only on the balance property of each $\pi_{\boldsymbol{u}_i}(\boldsymbol{s})$, treating $\pi_{\boldsymbol{v}_i}(\boldsymbol{k})$ as constants. To accelerate the search, we apply Reducing Rule 1 from Lambin's work [14] to reduce the number of monomials that need to be examined.

- **p-Round Forward Modeling Using the Bit-Based Division Property with Two Subsets.** We model the propagation of the division property over p rounds, storing all possible division properties in $\mathbb{D}_p$. For details on the modeling process of the bit-based division property with two subsets, please refer to [20]. The MixColumn operation is treated as a single S-box. Finally, the model is solved using the Gurobi optimizer [10].

- **Checking the Balance Property of the Linear Combination of Bits.** For any monomial $\pi_{\boldsymbol{u}_i}(\boldsymbol{s})$ in the ANF, its parity can be determined as follows:

$$\sum_i \pi_{\boldsymbol{u}_i}(\boldsymbol{s}) = \begin{cases} \textit{unknown} & \text{if there exists } \boldsymbol{d} \in \mathbb{D}_p \text{ such that } \boldsymbol{u}_i \succeq \boldsymbol{d}, \\ 0 & \text{otherwise.} \end{cases}$$

  If all $\pi_{\boldsymbol{u}_i}(\boldsymbol{s})$ in the ANF is balanced, then the linear combination of bits $\mathcal{C}(\boldsymbol{b})$ is considered to be balanced.

Reducing Rule 1 ([14]) *Given a monomial set $\mathbb{U}$, if there exist $\boldsymbol{u}_i, \boldsymbol{u}_j \in \mathbb{U}$ such that $\boldsymbol{u}_i \succeq \boldsymbol{u}_j$, then for all $\boldsymbol{u}_s$ such that $\boldsymbol{u}_j \succeq \boldsymbol{u}_s$, it always holds that $\boldsymbol{u}_i \succeq \boldsymbol{u}_s$. Therefore, it suffices to check the reduced set:*

$$\mathbb{U}' = \{\boldsymbol{u}_i \in \mathbb{U} \mid \nexists\, \boldsymbol{u}_j \in \mathbb{U} \textit{ such that } \boldsymbol{u}_j \succeq \boldsymbol{u}_i\}.$$

The algorithm for checking the balance property of a linear combination of bits is presented in Algorithm 2.

Algorithm 2 Search Key-Independnet Integral Distinguisher Based on Combination of Bits

Input: A combination of ciphertext bits $\mathcal{C}(\boldsymbol{b})$, number of rounds p and q, input division property $\boldsymbol{d}_0$
Output: `True` if $\mathcal{C}(\boldsymbol{b})$ is balanced; `False` otherwise

$\mathbb{U} \leftarrow$ **BackwardExtension**$(\mathcal{C}, q)$
$\mathbb{U}' \leftarrow$ **ApplyReducingRule** 1$(\mathbb{U})$
for each $\boldsymbol{u}_i \in \mathbb{U}'$ **do**
    $\mathcal{M} \leftarrow$ **Model**$(p, \boldsymbol{d}_0, \boldsymbol{u}_i)$
    $\mathcal{M}$.`optimize`()
    if $\mathcal{M}$.`status` is `Optimal` **then**
        return `False`
    end if
end for
return `True`

Using Algorithm 2 to Reproduce Distinguishers 1 and Distinguisher 2 For Distinguisher 1, we apply 1 round of backward monomial extension and 6 rounds of forward modeling. As shown in Equation (2), we verify whether each monomial in the ANF is balanced:

$$b_4 \oplus b_{52} = s_{56}s_{57} \oplus s_{56} \oplus s_{57} \oplus s_{59} \oplus 1 \tag{2}$$

For Distinguisher 2, we apply 2 rounds of backward monomial extension and 6 rounds of forward modeling. As shown in Equation (3), we similarly verify the balance property of each monomial.

$$\begin{aligned} & b_{28} \oplus b_{44} \oplus b_{60} \\ =& s_9s_{10}k_{28} \oplus s_8s_9k_{29} \oplus s_8k_{28} \oplus s_9k_{28} \oplus s_{10}k_{28} \oplus s_8k_{29} \oplus s_9k_{29} \oplus s_{11}k_{29} \oplus \\ & k_{28}k_{29} \oplus s_{10} \oplus k_{31} \oplus k_{36} \end{aligned} \tag{3}$$

Finally, using the improved automated search model, we successfully identified Distinguisher 1 and 2, which are consistent with the results obtained from the integral-neural distinguisher.

Using Algorithm 2 to Search for Longer Distinguishers For `SKINNY-64-64`, there are 2^{64} possible linear combinations of bits, making exhaustive search infeasible. However, based on the observed structure of the linear combinations in Distinguisher 1 and 2, we find that each bit involved in the linear combination with balance property is always located at the same position within its respective column. Therefore, it suffices to examine only $4 \times (2^4 - 1)$ linear combinations of

bits. Using this strategy, we successfully obtain an integral distinguisher for 11-round SKINNY-64-64 and SKINNY-128-128 by activating only 60 plaintext bits, which refer to this as **Distinguisher 3** and **Distinguisher 4**. It is worth noting that to obtain an 11-round integral distinguisher for SKINNY-64-64, Derbez *et al.* required 63 activated plaintext bits [8].

Distinguisher 3 (Key-Independent Integral Distinguisher Based on Linear Combination of Bits Against 11-round SKINNY-64-64.) *If the last 60 plaintext bits are activated, then the linear combinations of bits* $b_i \oplus b_{i+16} \oplus b_{i+32}$ *for* $i \in \{16, 17, \ldots, 31\}$ *and* $b_{12} \oplus b_{60}$ *are balanced.*

Distinguisher 4 (Key-Independent Integral Distinguisher Based on Linear Combination of Bits Against 11-round SKINNY-128-128.) *If the last 60 plaintext bits are activated, then the linear combinations of bits* $b_i \oplus b_{i+16} \oplus b_{i+32}$ *for* $i \in \{26, \ldots, 63\}$ *and* $b_{26} \oplus b_{122}$ *are balanced.*

6 Key-Dependent Integral Distinguisher

While our previous analysis has primarily focused on key-independent (i.e., deterministic) integral properties, probabilistic integral properties are also of significant value. In fact, such distinguishers correspond to cube testers, as introduced in [1]. In this section, we demonstrate that neural networks are capable of learning probabilistic integral properties. Moreover, by employing the improved automated search model, we identify a higher-round key-dependent integral distinguisher for SKINNY-64-64.

6.1 Key-Dependent Integral Distinguisher Based on linear Combination of Bits

In Section 4.3, a 7-round integral-neural distinguisher using $\mathbb{VDS}$ with an accuracy close to 100% is obtained by activating a single plaintext cell. In general, the accuracy of a neural distinguisher tends to decrease as the number of rounds increases. This raises a natural question: What is the maximum number of rounds that an integral-neural distinguisher can achieve when only one plaintext cell is activated?

The 8-Round Integral-Neural Distinguisher with One Activated Plaintext Cell The size of the $\mathbb{VDS}$ is reduced to 64, since $u \in \{$0b0001, 0b0010, 0b0100, 0b1000$\}$ and there are 16 cells in total. An 8-round integral-neural distinguisher for SKINNY-64-64 is obtained by activating the 15th plaintext cell, as shown in Table 5. A bit sensitivity test was then performed to analyze the learned features, and the results are presented in Fig. 3.

From Fig. 3, it can be observed that the 8-round integral-neural distinguisher relies on only 6 bits (highlighted in ■ and ■) from the $\mathbb{VDS}$. These bits correspond to the case where $u = 8$ in ciphertext cells $\{4, 7, 8, 11, 12, 15\}$. The 6 bits can be grouped into two categories: ■ and ■.

Table 5. Performance of the 8-round integral-neural distinguisher for `SKINNY-64-64` when the 15th plaintext cell is activated

Activated Plaintext Cell	Acc	TPR	TNR
15th	57.6%	49.2%	65.9%

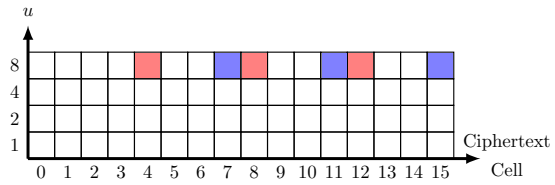

Fig. 3. Bit sensitivity test of the 8-round integral-neural distinguisher for `SKINNY-64-64` with the 15th plaintext cell activated.

- When the ■ bits are masked, the accuracy of the distinguisher decreases by 2.6%.
- When the ■ bits are masked, the accuracy of the distinguisher decreases by 4.7%.

Retraining the Integral-Neural Distinguisher Given the abnormal TPR observed for the distinguisher in Table 5, we retrain the integral-neural distinguisher using only the $\mathbb{VDS}$ corresponding to $u = 8$ in the ciphertext cells $\{4, 8, 12\}$ and $\{7, 11, 15\}$. The retraining results are presented in Table 6.

Table 6. Integral-neural distinguisher for 8-round `SKINNY-64-64` using $\mathbb{VDS}$ entries corresponding to $u = 8$ in ciphertext cells $\{4, 8, 12\}$ and $\{7, 11, 15\}$

Activated Plaintext Cell	u	Observed Ciphertext Cells	Acc	TPR	TNR
15th	8	$\{4, 8, 12\}$	56.3%	62.7%	49.9%
		$\{7, 11, 15\}$	56.3%	62.7%	49.9%

We utilized 10^7 positive instances to calculate the frequency of the equation $b_{16} \oplus b_{32} \oplus b_{48} = 0$, which corresponds to the case of $u = 8$ in ciphertext cells $\{4, 8, 12\}$. The analysis revealed that approximately 62.7% of the instances satisfied this condition. This frequency aligns precisely with the TPR of the integral-neural distinguisher reported in Table 6, indicating that the neural network has successfully learned a probabilistic (i.e., key-dependent) balanced property. Based on this observation, we give a key-dependent integral distinguisher, as presented in Distinguisher 5. A similar 8-round integral distinguisher also exists for `SKINNY-128-128`, as shown in Distinguisher 6.

Distinguisher 5 (Key-Dependent Integral Distinguisher Based on Linear Combination of Bits Against 8-round `SKINNY-64-64`.) *If the 15th plaintext cell is activated, then the linear combinations of bits* $b_{16} \oplus b_{32} \oplus b_{48}$ *and* $b_{28} \oplus b_{44} \oplus b_{60}$ *are probability balanced.*

Distinguisher 6 (Key-Dependent Integral Distinguisher Based on Linear Combination of Bits Against 8-round `SKINNY-128-128`.) *If the 15th plaintext cell is activated, then the linear combinations of bits* $b_{33} \oplus b_{65} \oplus b_{97}$ *and* $b_{57} \oplus b_{89} \oplus b_{121}$ *are probability balanced.*

6.2 Improved Automated Search Model for Key-Dependent Integral Distinguisher

To study the probability of the key-dependent integral distinguisher, we get ANF of the expression $b_{16} \oplus b_{32} \oplus b_{48}$ using a 2-round backward monomial extension. The resulting superpoly is shown in Equation (4), where the variable $\boldsymbol{s}$ denotes the input state of the last two rounds.

$$\begin{aligned} & b_{16} \oplus b_{32} \oplus b_{48} \\ =& \underbrace{s_{14}(1 \oplus k_{16}) \oplus s_{13}(k_{16} \oplus k_{17}) \oplus s_{12}(k_{16} \oplus k_{17}) \oplus s_{15}k_{17} \oplus s_{16}k_{17} \oplus k_{19} \oplus k_{48}}_{balanced} \\ & \oplus \underbrace{s_{13}s_{14}k_{16} \oplus s_{12}s_{13}k_{17}}_{unknown}. \end{aligned} \tag{4}$$

Assuming the key is uniformly random, the probability that both k_{16} and k_{17} are equal to 0 is 0.25. In this case, the monomials $s_{49}s_{50}k_{16} \oplus s_{50}s_{51}k_{17}$ becomes balanced. When k_{16} and k_{17} are not both 0, which occurs with probability 0.75, the probability that monomials $s_{49}s_{50}k_{16} \oplus s_{50}s_{51}k_{17}$ are balanced is 0.5. Therefore, the overall probability that $b_{16} \oplus b_{32} \oplus b_{48}$ is balanced can be estimated as:

$$0.25 \times 1 + 0.75 \times 0.5 = 0.625,$$

which closely matches the TPR of the integral-neural distinguisher reported in Table 6. Notice, the estimated value of 0.625 deviates slightly from the experimental result of 0.627. This discrepancy arises because the probability that a monomial is *unknown* or *balanced* is not necessarily exactly 0.5.

In previous work, the focus has primarily been on identifying key-independent integral distinguishers. When a *unknown* monomial is key-independent, it means there is no key bit in the monomial. In this case, there is no opportunity to make the *unknown* monomial balanced by setting key bits to 0. Therefore, the monomial is certainly *unknown*. If a key-independent *unknown* monomial exists in the ANF, the entire ANF will be considered *unknown*. However, when specific key values are considered, it is possible for an *unknown* monomial with key variable to become balanced. As a result, a probabilistic (key-dependent) integral distinguisher may be obtained—resembling the notion of a weak-key integral distinguisher.

Automatic Search for Key-Dependent Integral Distinguisher The algorithm for searching key-dependent integral distinguishers is shown in Algorithm 3. Building upon Algorithm 2, we examine each *unknown* monomial individually, collect the key bits involved in the monomial, and estimate the probability that the linear combination of bits is balanced when both key bits and constant plaintext bits are randomly assigned. It is important to note that when key bits are present in the *unknown* monomial, Reducing Rule 1 becomes insufficient. For example, consider two monomials $\pi_{\boldsymbol{u}_i}(\boldsymbol{s})\pi_{\boldsymbol{v}_i}(\boldsymbol{k})$ and $\pi_{\boldsymbol{u}_j}(\boldsymbol{s})\pi_{\boldsymbol{v}_j}(\boldsymbol{k})$ such that $\boldsymbol{u}_i \succeq \boldsymbol{u}_j$, $\boldsymbol{v}_i \neq 0$, and $\boldsymbol{v}_j = 0$. In this case, the monomial $\pi_{\boldsymbol{u}_j}(\boldsymbol{s})\pi_{\boldsymbol{v}_j}(\boldsymbol{k})$ must not be ignored, as it is certainly *unknown*. In contrast, if $\boldsymbol{u}_i = \boldsymbol{u}_j$ and $\boldsymbol{v}_i \preceq \boldsymbol{v}_j$ holds for all relevant monomials, then it is safe to disregard the pair $(\boldsymbol{u}_j, \boldsymbol{v}_j)$. We summarize this new insight as Reducing Rule 2.

Reducing Rule 2 *Given a set of monomials* $\mathbb{M} = \{(\boldsymbol{u}_i, \boldsymbol{v}_i)\}$*, the reduced set is defined as:*

$$\mathbb{M}' = \{(\boldsymbol{u}_i, \boldsymbol{v}_i) \in \mathbb{M} \mid \nexists\, (\boldsymbol{u}_j, \boldsymbol{v}_j) \in \mathbb{M} \text{ such that } \boldsymbol{u}_j \succeq \boldsymbol{u}_i \text{ and } \boldsymbol{v}_j \preceq \boldsymbol{v}_i\}\,.$$

Algorithm 3 Search Key-Dependnet Integral Distinguisher Based on Combination of Bits

Input: Combination of ciphertext bits $\mathcal{C}(\boldsymbol{b})$, number of rounds p, q, input division property $\boldsymbol{d}_0$
Output: Estimated probability that $\mathcal{C}(\boldsymbol{b})$ is balanced

$\mathbb{M} \leftarrow$ **BackwardExtension**$(\mathcal{C}, q)$
$\mathbb{M}' \leftarrow$ **ApplyReducingRule** 2$(\mathbb{M})$
$\mathbb{V} \leftarrow \emptyset$ ▷ Set of key bit indices in the *unknown* monomials
for each $(\boldsymbol{u}_i, \boldsymbol{v}_i) \in \mathbb{M}'$ **do**
  if $\boldsymbol{u}_i = \mathbf{0}$ **then**
    continue
  end if
  $\mathcal{M} \leftarrow$ **Model**$(p, \boldsymbol{d}_0, \boldsymbol{u}_i)$
  $\mathcal{M}$.`optimize`()
  if $\mathcal{M}$.`status` is `Optimal` **then**
    if $\mathbf{0} \in \mathbb{V}$ **then**
      return 0
    end if
    $\mathbb{V}$.`add`$(\{j \mid \boldsymbol{v}_i[j] = 1\})$ ▷ Collect key bits involved
  end if
end for
return $2^{-|\mathbb{V}|} + (1 - 2^{-|\mathbb{V}|}) \times 0.5$ ▷ Key bits $= 0$ with prob $2^{-|\mathbb{V}|}$

Notice, when Algorithm 3 returns a probability of 1, the integral distinguisher is in fact deterministically valid. Using Algorithm 3, we successfully identify Distinguisher 7, which represents the longest known integral distinguisher for `SKINNY-64-64` to date.

Distinguisher 7 (Key-Dependent Integral Distinguisher Based on Linear Combination of Bits Against 12-round `SKINNY-64-64`.) *If the last 60 plaintext bits are activated, then the probability that the linear combinations of bits* $b_{16} \oplus b_{32} \oplus b_{48}$*,* $b_{24} \oplus b_{40} \oplus b_{56}$*, and* $b_{28} \oplus b_{44} \oplus b_{60}$ *are balanced is approximately 0.625.*

6.3 Key-Dependent Integral Distinguisher Based on Nonlinear Combination of Bits

In the early stages of integral cryptanalysis, both the integral property and the division property were originally studied at the byte level. In this section, we shift our focus back to the byte level by using the division sequence $\mathbb{DS}$ corresponding to a ciphertext cell as the input to train the integral-neural distinguisher. The experimental results are presented in Table 7.

Table 7. The integral-neural distinguisher for 7-round `SKINNY-64-64` using $\mathbb{DS}$

Activated Plaintext Cell	Observed Ciphertext Cells	Acc	TPR	TNR
15th	6th	76.3%	94.4%	58.3%

Analyzing the Working Mechanism To further analyze the internal decision process of the integral-neural distinguisher, we adopt the method described in Section 4.4 to directly convert the trained model into a Boolean function. The conjunctive normal form (CNF) representation of the Boolean function consists of 490 clauses in total. For brevity, only the shortest clauses are listed below:
$(\boldsymbol{b}_6^1 \vee \boldsymbol{b}_6^2 \vee \boldsymbol{b}_6^8 \vee \neg \boldsymbol{b}_6^9)$ & $(\boldsymbol{b}_6^1 \vee \neg \boldsymbol{b}_6^2 \vee \boldsymbol{b}_6^8 \vee \boldsymbol{b}_6^9)$ & $(\boldsymbol{b}_6^1 \vee \boldsymbol{b}_6^4 \vee \boldsymbol{b}_6^8 \vee \neg \boldsymbol{b}_6^{12})$ & $(\neg \boldsymbol{b}_6^1 \vee \boldsymbol{b}_6^4 \vee \boldsymbol{b}_6^8 \vee \boldsymbol{b}_6^{12})$ & $(\boldsymbol{b}_6^2 \vee \boldsymbol{b}_6^8 \vee \neg \boldsymbol{b}_6^9 \vee \boldsymbol{b}_6^{12})$ & $(\neg \boldsymbol{b}_6^2 \vee \boldsymbol{b}_6^8 \vee \boldsymbol{b}_6^9 \vee \boldsymbol{b}_6^{12})$

For positive examples (i.e., the $\mathbb{DS}$ corresponding to a single ciphertext cell), every clause in the CNF must be satisfied. However, due to the large number of clauses, it is infeasible to directly infer the specific decision rules learned by the integral-neural distinguisher from the CNF representation. To address this, we focus on the variables involved in each clause and evaluate their significance. Specifically, we use Algorithm 2 to verify the integral property of various combinations of bits, such as:

$$\boldsymbol{b}_6^1 \oplus \boldsymbol{b}_6^2 \oplus \boldsymbol{b}_6^8 \oplus \boldsymbol{b}_6^9, \quad \boldsymbol{b}_6^1 \oplus \boldsymbol{b}_6^4 \oplus \boldsymbol{b}_6^8 \oplus \boldsymbol{b}_6^{12}, \quad \boldsymbol{b}_6^2 \oplus \boldsymbol{b}_6^8 \oplus \boldsymbol{b}_6^9 \oplus \boldsymbol{b}_6^{12}.$$

To facilitate understanding, we convert b_{cth}^u into its representation as a nonlinear combination of bits, i.e.

$$b_{24} \oplus b_{26} \oplus b_{27} \oplus b_{24}b_{27}, \quad b_{24} \oplus b_{25} \oplus b_{27} \oplus b_{24}b_{25}, \quad b_{24} \oplus b_{26} \oplus b_{24}b_{27} \oplus b_{24}b_{25}.$$

We first perform one round of backward monomial extension on these three nonlinear combinations of bits, followed by six rounds of forward modeling. Each

resulting monomial is then checked for balanc property. The results are presented in Equations (5), (6), and (7).

$$
\begin{aligned}
& b_{24} \oplus b_{26} \oplus b_{27} \oplus b_{24}b_{27} \\
=& \underbrace{1 \oplus k_{30} \oplus k_{28} \oplus s_8 \oplus k_{28}k_{31} \oplus s_{11}k_{31} \oplus s_9k_{31} \oplus s_8k_{31} \oplus s_{10}k_{28} \oplus s_9k_{28} \oplus s_8k_{28}}_{balanced} \\
\oplus & \underbrace{s_{10}s_{11} \oplus s_9s_{11} \oplus s_8s_{11} \oplus s_8s_{10} \oplus s_8s_9k_{31} \oplus s_{10}s_{11}k_{28} \oplus s_9s_{11}k_{28} \oplus s_8s_{11}k_{28}}_{unknown} \\
\oplus & \underbrace{s_8s_{10}k_{28} \oplus s_9s_{10}s_{11} \oplus s_8s_9s_{10} \oplus s_9s_{10}s_{11}k_{28} \oplus s_8s_9s_{10}k_{28}}_{unknown}
\end{aligned}
\tag{5}
$$

$$
\begin{aligned}
& b_{24} \oplus b_{26} \oplus b_{24}b_{27} \oplus b_{24}b_{25} \\
=& \underbrace{k_{31} \oplus k_{30} \oplus k_{29} \oplus s_{11} \oplus s_8 \oplus k_{28}k_{31} \oplus s_{11}k_{31} \oplus s_9k_{31} \oplus s_8k_{31} \oplus k_{28}k_{29}}_{balanced} \\
\oplus & \underbrace{s_{11}k_{29} \oplus s_9k_{29} \oplus s_8k_{29}}_{balanced} \oplus \underbrace{s_{10}s_{11} \oplus s_9s_{11} \oplus s_8s_{11} \oplus s_9s_{10} \oplus s_8s_{10} \oplus s_8s_9}_{unknown} \\
\oplus & \underbrace{s_8s_9k_{31} \oplus s_8s_9k_{29} \oplus s_{10}s_{11}k_{28} \oplus s_9s_{11}k_{28} \oplus s_8s_{11}k_{28} \oplus s_9s_{10}k_{28}}_{unknown} \\
\oplus & \underbrace{\oplus s_8s_{10}k_{28} \oplus s_9s_{10}s_{11} \oplus s_8s_9s_{10} \oplus s_9s_{10}s_{11}k_{28} \oplus s_8s_9s_{10}k_{28}}_{unknown}
\end{aligned}
\tag{6}
$$

$$
\begin{aligned}
& b_{24} \oplus b_{25} \oplus b_{27} \oplus b_{24}b_{25} \\
=& \underbrace{1 \oplus k_{31} \oplus s_{10} \oplus k_{28}k_{29} \oplus s_{11}k_{29} \oplus s_9k_{29} \oplus s_8k_{29} \oplus s_{10}k_{28} \oplus s_9k_{28} \oplus s_8k_{28}}_{balanced} \\
\oplus & \underbrace{s_8s_9k_{29} \oplus s_9s_{10}k_{28}}_{uknown}
\end{aligned}
\tag{7}
$$

From these three equations, we observe that only Equation (7) allows the nonlinear combination of bits to become balanced by controling the key values, while Equations (5) and (6) do not, as they contain *unknown* monomials that do not involve key bits. Specifically, when k_{28} and k_{29} in Equation (7) are set to 0, the probability that the nonlinear combination of bits is balanced becomes approximately 0.625. Therefore, we conclude that the neural network has learned certain probabilistic integral features from nonlinear combinations of bits.

We enumerated all nonlinear combinations of bits for the 6th ciphertext cell, resulting in a total of 2^{2^4} possibilities. Using a process similar to Algorithm 3, we calculated the probability that each of the 2^{16} nonlinear combinations of bits is balanced. Therefore, a key-dependent integral distinguisher for the nonlinear combination of bits is proposed, as shown in Distinguisher 8.

Distinguisher 8 (Key-Dependent Integral Distinguisher Based on Nonlinear combination of bits Against 7-round `SKINNY-64-64`.) *If the 15th plaintext cell is activated, the probability that the nonlinear combination of bits* $b_{24} \oplus b_{26} \oplus b_{24}b_{27}$ *(related to* k_{28}, k_{31}*) and* $b_{24} \oplus b_{25} \oplus b_{27} \oplus b_{24}b_{25}$ *(related to* k_{28}, k_{29}*) are balanced is 0.625.*

Additionally, we derive an 11-round key-dependent integral distinguisher based on a nonlinear combination of bits against `SKINNY-64-64`, as shown in Distinguisher 9.

Distinguisher 9 (Key-Dependent Integral Distinguisher Based on Nonlinear Combination of Bits Against 11-round `SKINNY-64-64`.) *If the last 60 plaintext bits are activated, then the probability that the following nonlinear combinations of bits are balanced is approximately 0.625:*

$b_{24} \oplus b_{26} \oplus b_{24}b_{27}$*(related to* k_{20}, k_{23}*), and* $b_{24} \oplus b_{25} \oplus b_{27} \oplus b_{24}b_{25}$*(related to* k_{20}, k_{21}*).*

The integral properties in Distinguishers 8 and 9 correspond to the integral property defined in Definition 4 where the function g is nonlinear. We restrict the search space to all possible combinations within a single ciphertext cell. Specifically, we verify the balance of nonlinear combinations of bits within a single ciphertext cell for `SKINNY-64-64`, where the number of calls to the automated search model is 2^{2^4}. However, extending this approach to `SKINNY-128-128` would require evaluating 2^{2^8} combinations, which makes automated search computationally infeasible at this scale.

7 Improved Key Recovery Attack

At CRYPTO 2016, Beierle *et al.* introduced the `SKINNY` family of tweakable block ciphers and evaluated its resistance to integral attacks [3]. They identified a 6-round integral distinguisher using single-cell plaintext activation. By extending this distinguisher 4 rounds backward and 4 rounds forward, the authors successfully mounted 14-round key recovery attacks against both `SKINNY-64-64` and `SKINNY-128-128`. Subsequent work by Zhang *et al.* [22] demonstrated a 16-round integral attack on `SKINNY-128-128`, using a 7-round distinguisher extended by 3 backward and 6 forward rounds. However, their analysis focused solely on the key guessing space and did not fully evaluate the time complexity. More recently, Hosein *et al.* proposed an 18-round attack by constructing an integral distinguisher derived from a zero-correlation distinguisher under a chosen-tweak model. Their attack targets the 60-bit and 120-bit key variants of `SKINNY-64-64` and `SKINNY-128-128`, respectively [11].

7.1 Key Recovery Attack Using Key-Independent Integral Distinguisher

The longest integral distinguisher does not necessarily lead to the most effective key-recovery attack. We use a 7-round integral distinguisher based on a linear

combination of bits to perform a key-recovery attack. The detailed procedure is described as follows.

The Key Recovery Attack for `SKINNY-n-n`. The overall procedure of the key-recovery attack is illustrated in Figure 4 The target cipher E is divided into three parts: E_0 covers rounds $(0 \to \ldots \to 3)$, E_1 (**Distinguisher 1**) covers rounds $(3 \to \ldots \to 10)$, and E_2 covers rounds $(10 \to \ldots \to 15)$. The strategy proceeds as follows:

1. Obtain a pair of plaintext-ciphertext (P, C).
2. Guess 15-cell involved keys,
   - Determine the value of P in ▨ of E_0. Encrypt the plaintext P using the guessed key to obtain the ciphertext after the 3rd round. Then, traverse the 15th cell of the 3rd-round ciphertext to obtain a ciphertext set. Next, perform 3 rounds of decryption to recover the initial plaintext set. Finally, execute 15 rounds of encryption to obtain the final ciphertext set.
   - Determine the values in ■ of round 10: partially decrypt each ciphertext using the guessed involved keys. Check whether the involved ciphertext bits are balanced and obtain the candidate round keys that pass the test.

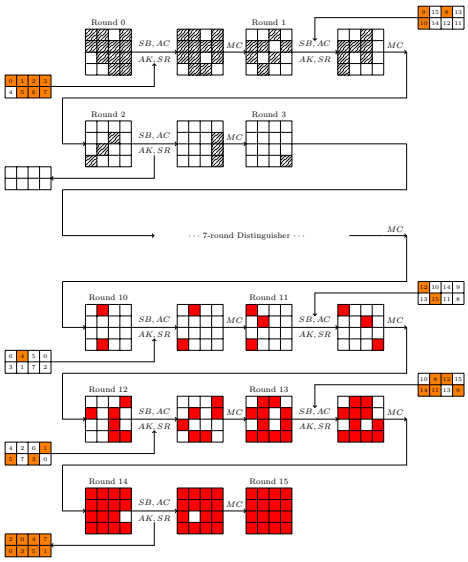

Fig. 4. 15-round key recovery using key-independent integral distinguisher for `SKINNY-n-n` in single tweakey setting

Complexity Analysis. In round 0, a total of 12 cells are activated, resulting in a data complexity of 2^{12c}. A total of 16 plaintexts and ciphertexts are used, and

the entire attack process involves keys from 15 cells and encryption/decryption of 59 cells. Therefore, the time complexity is calculated as $2^c \times 2^{15\times c} \times \frac{59}{16\times 15} = 2^{16c-2.024}$.

The Key Recovery Attack for `SKINNY-n-2n` and `SKINNY-n-3n`. Since the 7-round integral distinguisher is independent of the key, it can be applied to `SKINNY-n` with different key lengths.

- Using a similar procedure, we can perform a 17-round key recovery attack on `SKINNY-n-2n`. The attack process is shown in Appendix B.1 Fig. 6.The data complexity of 2^{12c} and the time complexity is $2^c \times 2^{31\times c} \times \frac{59+32}{16\times 17} = 2^{32c-1.579}$.

- Using a similar procedure, we can perform a 19-round key recovery attack on `SKINNY-n-3n`. The attack process is shown in Appendix B.1 Fig. 7. The data complexity of 2^{12c} and the time complexity is $2^c \times 2^{47\times c} \times \frac{59+64}{16\times 19} = 2^{48c-1.305}$.

7.2 Key Recovery Attack Using Key-Dependent Integral Distinguisher

In Section 7.1, the key recovery attack uses an integral distinguisher based on the linear combination of bits (**Distinguisher** 1) involving two cells, which results in the need to guess two key bytes in the 11th round. When using a 8-round key-dependent integral distinguisher (**Distinguisher** 5), fewer key bytes need to be guessed. Note that the attack is only effective when the two key bytes involved in the distinguisher are set to zero.

The Complexity Analysis of Key Recovery Attack for `SKINNY-n-n`. The overall procedure of the key-recovery attack is illustrated in Figure 5. In round 0, a total of 12 cells are activated, resulting in a data complexity of 2^{12c}. A total of 16 plaintexts and ciphertexts are used, and the entire attack process involves keys from 15 cells and encryption/decryption of 57 cells. Therefore, the time complexity is calculated as $2^c \times 2^{13\times c} \times \frac{57}{16\times 15} = 2^{14c-2.074}$.

The Key Recovery Attack for `SKINNY-n-2n` and `SKINNY-n-3n`. For different parameter settings of the `SKINNY`, the 8-round key-dependent integral distinguishers involve different key positions, but the number of involved key bytes remains the same.

- Using a similar procedure, we can perform a 17-round key recovery attack on `SKINNY-n-2n`. The attack process is shown in Appendix B.2 Fig. 8. The data complexity of 2^{12c} and the time complexity is $2^c \times 2^{29\times c} \times \frac{57+32}{16\times 17} = 2^{30c-1.612}$.

- Using a similar procedure, we can perform a 19-round key recovery attack on `SKINNY-n-3n`. The attack process is shown in Appendix B.2 Fig. 9. The data complexity of 2^{12c} and the time complexity is $2^c \times 2^{45\times c} \times \frac{57+64}{16\times 19} = 2^{46c-1.329}$.

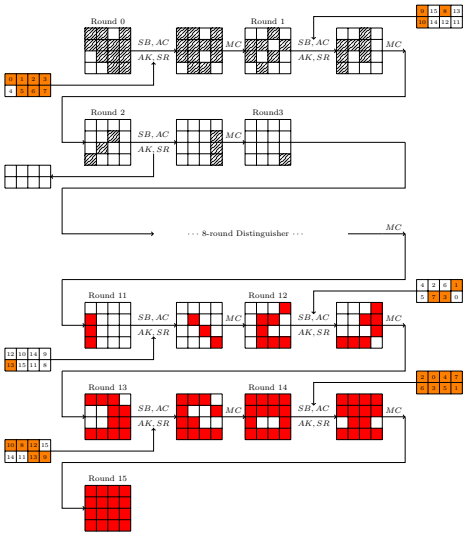

Fig. 5. 15-round key recovery using key-dependent integral distinguisher for `SKINNY-n-n` in single tweakey setting

8 Conclusion

This paper explores the application of neural networks in integral cryptanalysis and presents several significant advancements. By comparing the results obtained through neural networks and classical methods, we observe that existing automated search models are inaccurate. Using an improved automated search model, we extend the length of the integral distinguishers for `SKINNY`. In addition to discovering longer integral distinguishers, we also improve key-recovery attacks. Specifically, the short-round distinguisher learned by the neural network enables a longer-round key-recovery attack on `SKINNY` through a combination of backward decryption and forward encryption. These results demonstrate that artificial intelligence techniques can offer valuable support in advancing cryptanalysis. Unfortunately, identifying integral distinguishers requires an exhaustive enumeration of all possible mapping function. Exploring more efficient search strategies remains an important direction for future research.

References

1. Aumasson, J., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and trivium. In: FSE. Lecture Notes in Computer Science, vol. 5665, pp. 1–22. Springer (2009)
2. Bao, Z., Lu, J., Yao, Y., Zhang, L.: More insight on deep learning-aided cryptanalysis. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 436–467. Springer (2023)

3. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016)
4. Benamira, A., Gérault, D., Peyrin, T., Tan, Q.Q.: A deeper look at machine learning-based cryptanalysis. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12696, pp. 805–835. Springer (2021)
5. Beyne, T., Verbauwhede, M.: Integral cryptanalysis using algebraic transition matrices. IACR Trans. Symmetric Cryptol. **2023**(4), 244–269 (2023)
6. Boura, C., Canteaut, A.: Another view of the division property. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 9814, pp. 654–682. Springer (2016)
7. Chen, Y., Shen, Y., Yu, H.: Neural-aided statistical attack for cryptanalysis. Comput. J. **66**(10), 2480–2498 (2023)
8. Derbez, P., Fouque, P.A.: Increasing precision of division property. IACR Transactions on Symmetric Cryptology pp. 173–194 (2020)
9. Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. In: Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39. pp. 150–179. Springer (2019)
10. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2024), https://www.gurobi.com
11. Hadipour, H., Gerhalter, S., Sadeghi, S., Eichlseder, M.: Improved search for integral, impossible differential and zero-correlation attacks application to ascon, forkskinny, skinny, mantis, PRESENT and qarmav2. IACR Trans. Symmetric Cryptol. **2024**(1), 234–325 (2024)
12. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 12491, pp. 446–476. Springer (2020)
13. Knudsen, L.R., Wagner, D.A.: Integral cryptanalysis. In: FSE. Lecture Notes in Computer Science, vol. 2365, pp. 112–127. Springer (2002)
14. Lambin, B., Derbez, P., Fouque, P.: Linearly equivalent s-boxes and the division property. Des. Codes Cryptogr. **88**(10), 2207–2231 (2020)
15. Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 10624, pp. 128–157. Springer (2017)
16. Todo, Y.: Structural evaluation by generalized integral property. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 9056, pp. 287–314. Springer (2015)
17. Todo, Y.: Integral cryptanalysis on full MISTY1. J. Cryptol. **30**(3), 920–959 (2017)
18. Todo, Y., Morii, M.: Bit-based division property and application to simon family. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. pp. 357–377. Lecture Notes in Computer Science (2016)
19. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: Milp-aided method of searching division property using three subsets and applications. In: ASIACRYPT (3). Lecture Notes in Computer Science, vol. 11923, pp. 398–427. Springer (2019)
20. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying milp method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22. pp. 648–678. Springer (2016)

21. Zahednejad, B., Lyu, L.: An improved integral distinguisher scheme based on neural networks. Int. J. Intell. Syst. **37**(10), 7584–7613 (2022)
22. Zhang, W., Cao, M., Guo, J., Pasalic, E.: Improved security evaluation of SPN block ciphers and its applications in the single-key attack on SKINNY. IACR Trans. Symmetric Cryptol. **2019**(4), 171–191 (2019)

A The Algorithm of Bit Sensitivity Test

Algorithm 4 Bit Sensitivity Test [7]

Input: Test dataset $X = \{X_0, \cdots, X_{N-1}\}$, where half of the dataset is positive, and half is negative instances; n: the size of an instance X_i; $\mathcal{ND}$: neural distinguisher;
Output: An array of the decrease in accuracy Acc_{dec} for each bit;

$\text{Acc}_{orig} \leftarrow \mathcal{ND}(X)$;
$\text{Acc}_{dec} \leftarrow \{\}$;
$X_{masked} \leftarrow \{\}$;
for $i = 0$ to $n - 1$ **do**
    for $j = 0$ to N **do**
        Generate a random mask $\eta \in \{0, 1\}$;
        $X_{masked} \leftarrow X^j \oplus (\eta \ll i)$; ▷ Randomize bit i
    end for
    Add $\text{Acc}_{orig} - \mathcal{ND}(X_{masked})$ to Acc_{dec};
end for
return Acc_{dec};

B Improced Key Recovery Attack

B.1 Improved Key Recovery Attack Using Key-Independent Integral Distinguisher

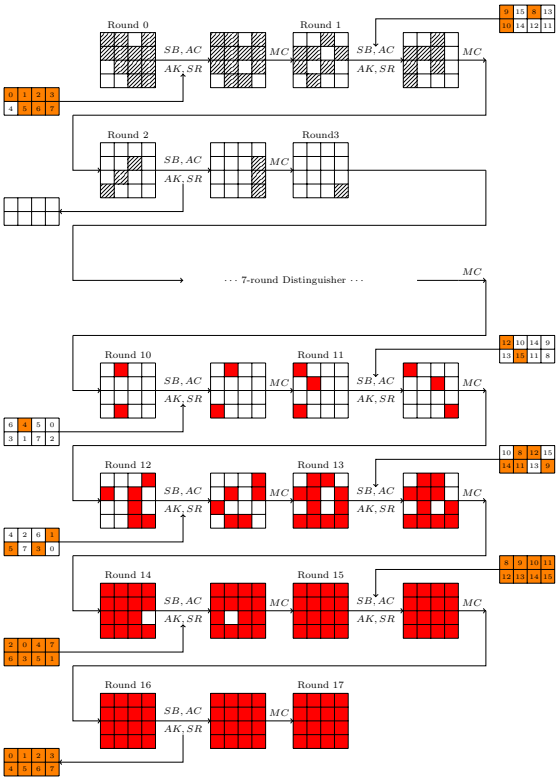

Fig. 6. 17-round key recovery using key-independent integral distinguisher for `SKINNY-n-2n` in single tweakey setting

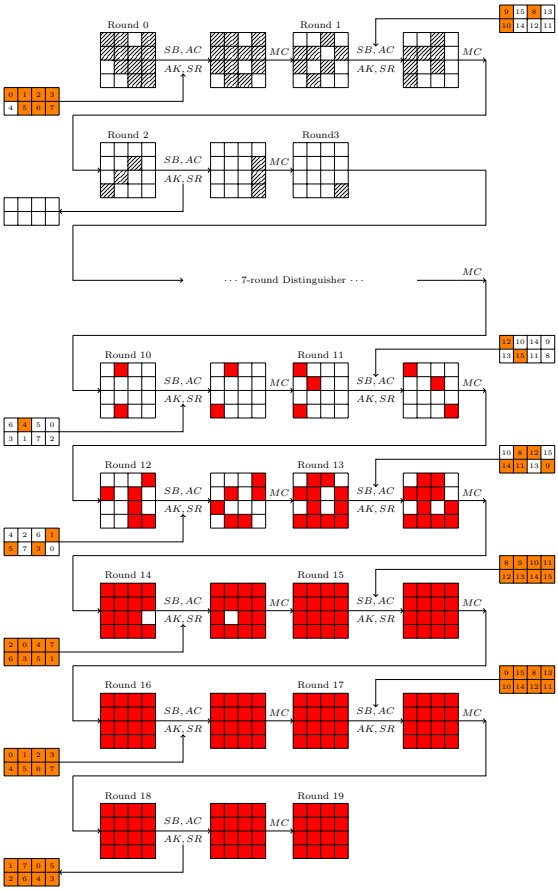

Fig. 7. 19-round key recovery using key-independent integral distinguisher for `SKINNY-n-3n` in single tweakey setting

B.2 Improved Key Recovery Attack Using Key-Dependent Integral Distinguisher

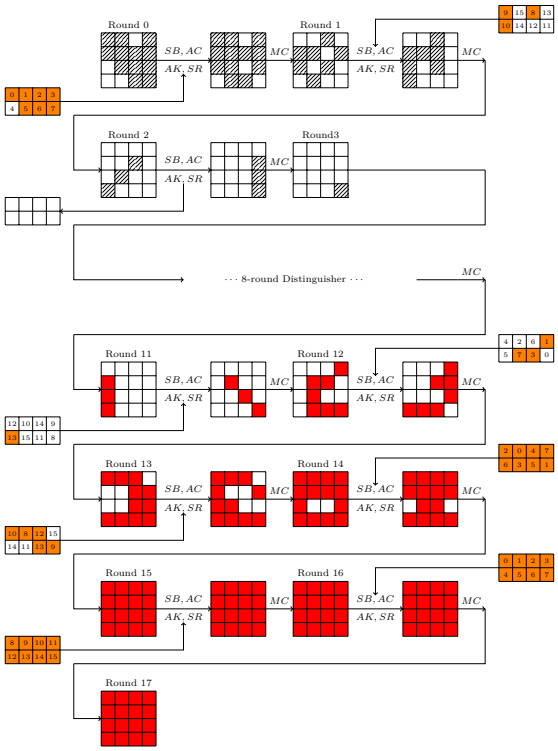

Fig. 8. 17-round key recovery using key-dependent integral distinguisher for `SKINNY-n-2n` in single tweakey setting

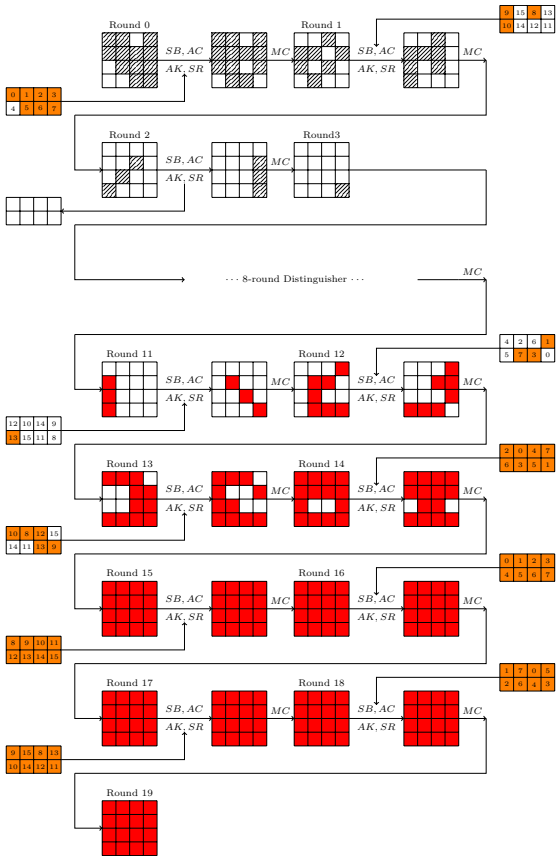

Fig. 9. 19-round key recovery using key-dependent integral distinguisher for `SKINNY-n-3n` in single tweakey setting