On the Explanation and Enhancement of Neural-inspired Differential Cryptanalysis

Weixi Zheng[a], Liu Zhang[a], Zilong Wang[a,*]

[a]*School of Cyber Engineering, Xidian University, 266 Xinglong Section of Xifeng Road, Xi'an, 710026, Shaanxi, China*

Abstract

Neural networks have been applied to symmetric cryptanalysis, and Gohr demonstrated that a neural-network-based distinguisher achieves higher accuracy than classical differential distinguishers at CRYPTO 2019. In this work, we analyze ciphertext data through the lens of probability distributions, identifying non-random features that provide indirect insights into how neural networks distinguish ciphertexts from random data. In parallel, we improve the key-recovery attack process by adopting the Bayesian-UCB method, which achieves a better balance between exploration and exploitation of ciphertext structures. These enhancements reduce the runtime of key-recovery attacks while simultaneously increasing their success rate.

Keywords: Neural Networks, Probability Distributions, Interpretability, Bayesian-UCB, Key Recovery Attacks

1. Introduction

Recent advances in deep learning have led to remarkable progress across diverse and challenging tasks, ranging from machine translation and autonomous driving to achieving superhuman performance in abstract board games. On the theoretical side, the long-recognized connection between cryptography and machine learning, as highlighted in the survey of [1], continues to provide valuable insights.

*Corresponding author.
Email address: zlwang@xidian.edu.cn (Zilong Wang)

At CRYPTO 2019, Gohr [2] introduced differential-neural cryptanalysis, where a neural network is trained as a differential-neural distinguisher ($\mathcal{DND}$) to separate ciphertext pairs with a fixed input difference from random ones. Since then, numerous works have investigated how neural networks achieve this. Gohr showed that they capture features beyond those exploited in classical differential cryptanalysis. Benamira *et al.* [3] related this ability to approximating the differential distribution table (DDT). Gohr *et al.* [4] further proved that, for `SIMON32/64`, neural networks can rely solely on differential features. More recently, Bao *et al.* [5] demonstrated that their strong performance on `SPECK32/64` stems from leveraging additional features such as *Intra-XOR* and *Cross-XOR*.

To extend neural distinguishers to key-recovery attacks, a short classical differential ($\mathcal{CD}$) is often prepended and combined with techniques such as the BayesianKeySearch algorithm, wrong-key response profiling, and the upper confidence bound (UCB) strategy to improve efficiency.

In this work, we pursue two directions: deepening the understanding of neural network mechanisms and improving the efficiency of key-recovery attacks.

- **Explanation of the Working Mechanism.** From the perspective of probability distributions, we analyze the features embedded in the ciphertext pair of `SPECK32/64` and infer that, in addition to classical difference information, ciphertext pairs also contain extra XOR-based features. Based on this observation, we reduce both the dimensionality of the training data and the size of the neural network, while maintaining its performance.

- **Improved Key Recovery Attacks.** To mitigate the tendency of UCB to become trapped in wrong ciphertext structures, we employ Bayesian-UCB as a new ciphertext-structure selection strategy to strengthen the key-recovery attack process. Furthermore, we refine the BayesianKeySearch algorithm by retaining the highest-scoring key, which reduces errors caused by incorrect ciphertext structures. Combined, these improvements lower the time complexity and substantially increase the success rate of key-recovery attacks on `SPECK32/64`, as shown in Table 1.

Table 1: Summary of key-recovery attacks on SPECK32/64

Round	c_1	c_2	n_{cts}	n_{it}	N_e	Run Time	Success Rate	Ref.
11	5	10	100	500	1000	216.61 s	36.1 %	[2]
						168.07 s	52.8 %	Our work
12	20	500	500	2000	100	1842.43 s	39 %	[2]
						1786.86 s	50 %	Our work
13	8	-500	2^{11}	2^{12}	100	17 011.22 s	21 %	[6]
						16 145.12 s	24 %	Our work

2. Preliminary

2.1. Brief Description of SPECK32/64

Let n be the word size in bits and $2n$ the state size. Denote the internal state after round i as (C_L^i, C_R^i) and the i-th round subkey as k^i. We use $\oplus$ for XOR, $\boxplus$ for addition modulo 2^n, and $\ggg$ / $\lll$ for right/left rotation.

SPECK32/64 is a lightweight block cipher from the SPECK family [7]. It encrypts 32-bit blocks with a 64-bit key, using 16-bit words for both state and subkeys. In round i, the state update is:

$$C_L^{i+1} = ((C_L^i \ggg 7) \boxplus C_R^i) \oplus k^i, \quad C_R^{i+1} = (C_R^i \lll 2) \oplus C_L^{i+1}.$$

2.2. Differential Scenario

Given an encryption function $E : \mathbb{F}_2^{n_p} \times \mathbb{F}_2^{n_k} \to \mathbb{F}_2^{n_c}$, a differential distinguisher separates:

- **Encryption samples:** $(E(p_1, k), E(p_2, k))$, with $p_1 \oplus p_2 = \Delta$;
- **Random samples:** $(E(p_1, k), E(p_2, k))$, with p_1, p_2 chosen independently at random.

Here, Δ is a fixed input difference, and p_1, p_2, k are uniformly random. In differential-neural cryptanalysis, these are labeled as positive (encryption) and negative (random) examples to train the $\mathcal{DND}$.

3. Explanation of the Working Mechanism

The encryption sample is $(c_1, c_2) = (E(k, p), E(k, p \oplus \Delta))$, where k and p are uniform. If $E(k, \cdot)$ is bijective, c_1 and c_2 are each uniform, so distinguishing requires analyzing them jointly. Since the key is random, ciphertext bits are independent; however, when c_1 and c_2 are considered together, their differences cancel out the randomness from the last key addition.

3.1. Probability Distribution of Ciphertext Pair for SPECK32/64

According to the round function of SPECK32/64, both X and Y depend on the same subkey K, as illustrated in Equation 1. We consider $(X, Y) = (C_L^i, C_R^i)$ as the known ciphertext state, and analyze the probability distribution of the ciphertext pair (X^1, Y^1, X^2, Y^2).

$$\begin{aligned} X = C_L^i &= ((C_L^{i-1} \ggg 7) \boxplus C_R^{i-1}) \oplus K = x \oplus K, \\ Y = C_R^i &= ((C_R^{i-1} \lll 2) \oplus x) \oplus K = y \oplus K. \end{aligned} \tag{1}$$

Lemma 1. *Let $X^1, X^2, Y^1, Y^2 \in \mathbb{F}_2^n$ be random vectors with Bernoulli components, and let $K = (K_1, \ldots, K_n)$ have independent components $K_i \sim \mathrm{Bern}(1/2)$, independent of X^1, X^2, Y^1, Y^2. For $x^1, x^2, y^1, y^2, \delta^1, \delta^2, \delta^3 \in \mathbb{F}_2^n$ satisfying*

$$x^2 = x^1 \oplus \delta^1, y^1 = x^1 \oplus \delta^2, y^2 = y^1 \oplus \delta^3 \ (= x^1 \oplus \delta^2 \oplus \delta^3),$$

we have

$$\begin{aligned} &\Pr[X^1 = x^1 \oplus K,\ X^2 = x^2 \oplus K,\ Y^1 = y^1 \oplus K,\ Y^2 = y^2 \oplus K] \\ =&\Pr\Big[X^1 \oplus X^2 = \delta^1,\ Y^1 \oplus X^1 = \delta^2,\ Y^2 \oplus Y^1 = \delta^3,\ Y^1 \oplus X^2 = \delta^1 \oplus \delta^2, \\ &\qquad\qquad Y^2 \oplus X^1 = \delta^2 \oplus \delta^3,\ Y^2 \oplus X^2 = \delta^1 \oplus \delta^2 \oplus \delta^3\Big]. \end{aligned}$$

The detailed proof is shown in Appendix A.

As shown in Lemma 1, ciphertext pairs contain not only classical difference information but also XOR relations between the left and right branches within a single ciphertext or between two ciphertexts. This observation is consistent with the conclusion in [5].

3.2. Lemma Verification through Modified Network Architecture

New Data Format. Previously, the ciphertext pair (X^1, Y^1, X^2, Y^2) was used directly as network input. By Lemma 1, `SPECK32/64` ciphertexts yield only six distinct difference combinations. Thus, we use $(X^1 \oplus X^2,\ X^1 \oplus Y^1,\ X^1 \oplus Y^2)$ as input, from which the network can infer the other three differences.

Modified the Network Architecture. In Gohr's network [2] (Fig. B.3), the initial convolutional layer (Module 1) was used to capture simple bit-sliced operations. Benamira *et al.* [3] later showed that this layer mainly performs linear combinations of ciphertext-pair differences. Since Lemma 1 already characterizes these features, the layer is redundant. We therefore remove it, as shown in Fig. 1. The modified network (15,383 parameters) is much smaller than Gohr's original (44,865 parameters), with detailed hyperparameters given in the Appendix B.

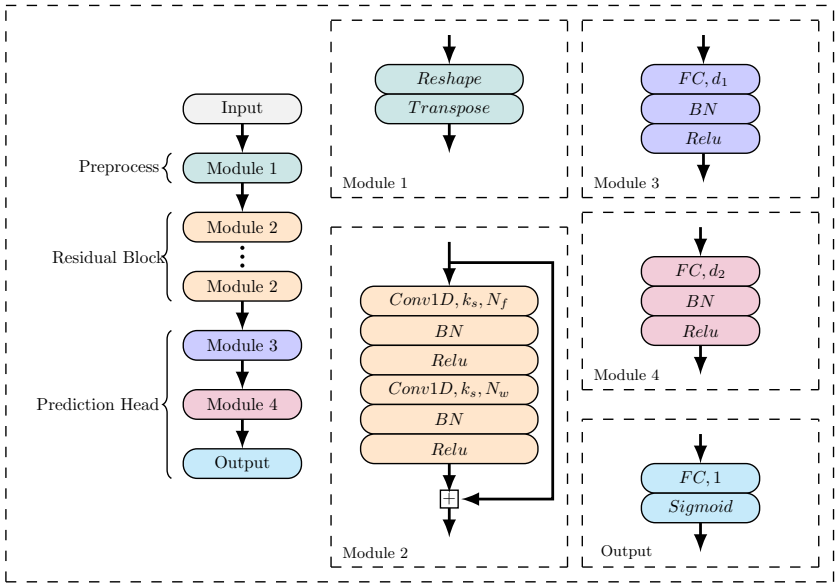

Figure 1: Modified network architecture

Performance Evaluation of $\mathcal{DND}$. Table 2 summarizes the accuracy (acc) of our $\mathcal{DND}$ compared with previous work.

Notably, removing the initial convolutional layer and reformulating the input data format according to the derived lemma does not lead to a significant loss in accuracy, thereby confirming the validity of Lemma 1.

Table 2: Accuracy comparison of $\mathcal{DND}$ on different rounds

Round	Accuracy	Reference
6	78.8 %	Our work
	78.8 %	[2]
	78.4 %	[8]
7	61.7 %	[4]
	61.6 %	[2]
	61.4 %	Our work
	60.7 %	[8]

4. Improved Key Recovery Attack

By refining the UCB-based selection of ciphertext structures and the BayesianKeySearch algorithm, we accelerate key recovery and improve the attack success rate.[1]

4.1. Improved Select Strategy of Ciphertext Structure

In differential-neural cryptanalysis, a $\mathcal{CD}$ is combined with a $\mathcal{DND}$ to extend the reach of key recovery attacks. The overall process is shown in Fig. 2; further details are provided in Appendix C.

The s-round $\mathcal{CD}$ ($\delta \rightarrow \Delta$) is probabilistic: a plaintext structure with input difference δ may or may not yield a ciphertext structure with output difference Δ. We call the former *correct* and the latter *wrong*. To efficiently select promising structures, Gohr [2] applied the UCB strategy:

$$priority_p(i) = X_{\max}(i) + \gamma \cdot \frac{\sqrt{\log(p)}}{\sqrt{N_p(i)}},$$

where $X_{\max}(i)$ is the maximum score of structure i before iteration p, $N_p(i)$ its usage count, and $\gamma = \sqrt{n_{cts}}$ with n_{cts} denoting the number of ciphertext structures.

The UCB strategy can identify promising ciphertext structures but depends heavily on the weight γ, making it hard to balance exploration and

[1]Experiments were conducted on Ubuntu 20.04 with Python 3.7.15 and TensorFlow 2.5.0, using dual Intel Xeon Gold 6226R CPUs (2.90 GHz), 256 GB RAM, and five NVIDIA RTX 2080Ti GPUs (12 GB each).

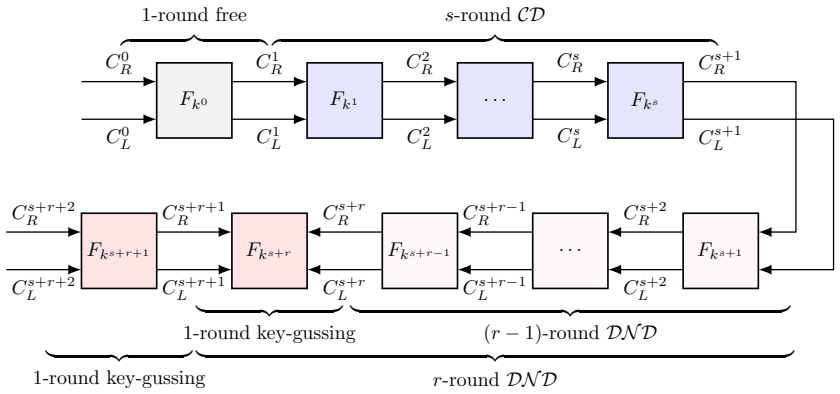

Figure 2: $(1 + s + r + 1)$-round key recovery attack of differential-neural cryptanalysis in [2].

exploitation. It may also get stuck in wrong structures with deceptively high $\mathcal{DND}$ scores, wasting computation and reducing success rates. Separately, Kaufmann *et al.* [9] proposed Bayesian Upper Confidence Bound (Bayes-UCB), a general bandit algorithm. Unlike frequency-based UCB, Bayes-UCB assigns each arm a prior distribution and selects structures based on cumulative scores.

Let X_j be the $\mathcal{DND}$ score at the j-th selection. For structure i, define $\mathbb{1}(I_p = i) = 1$ if $I_p = i$ at iteration p, and 0 otherwise. The cumulative score before iteration p is $s_p(i)$. With $Q(1-\nu, \rho)$ the quantile of distribution ρ and $\mathcal{T}(d)$ the t-distribution with d degrees of freedom, the priority of structure i is

$$priority_p(i) = \frac{s_p(i)}{N_p(i)} + \sqrt{w_p(i)} \cdot \frac{Q\Big(1 - \frac{1}{p}, \mathcal{T}(N_p(i) - 1)\Big)}{\sqrt{N_p(i)}},$$

where

$$s_p(i) = \sum_{j=1}^{p-1} \mathbb{1}(I_j = i) X_j, w_p(i) = \frac{\sum_{j=1}^{p-1} \mathbb{1}(I_j = i) X_j^2 - s_p^2(i)/N_p(i)}{N_p(i) - 1}.$$

The first term estimates the mean score, and the second adds an exploration bonus. At each iteration, the structure with the highest priority is selected.

We carried out 11-round key recovery attacks on `SPECK32/64` using Bayes-UCB with the $\mathcal{DND}$s from [2], keeping all parameters the same except for the ciphertext structure selection strategy. A key guess is deemed successful

if the last-round subkey is correct and the Hamming distance between the penultimate-round subkey and the true subkey does not exceed two. The results are presented in Table 3.

Table 3: The result of key recovery attack using the Bayes-UCB Algorithm

Round	c_1	c_2	n_{cts}	n_{it}	N_e	Run Time	Success Rate	Ref.
11	5	10	100	500	100	227.07 s	51 %	[2]
						182.57 s	55 %	Bayes-UCB

1. c_1, c_2: cutoffs for last-round and penultimate-round subkeys.
2. N_e: number of experiments
3. n_{it}: number of iterations in key recovery.

These results confirm the effectiveness of Bayes-UCB in enhancing both the efficiency and reliability of key recovery attacks.

4.2. Improved BayesianKeySearch *Algorithm*

In key recovery, we find that the correct key with an incorrect ciphertext structure may yield a high score. To mitigate this, we propose an improved BayesianKeySearch Algorithm 1 (refer to Appendix D), which retains the best key and adds it to the new candidate key. The experiment share identical parameters except for the search method, and the results are shown in Table 4.

Table 4: The key recovery attack using the improved BayesianKeySearch Algorithm

Round	c_1	c_2	n_{cts}	n_{it}	N_e	Run Time	Success Rate	Ref.
11	5	10	100	500	100	227.07 s	51 %	[2]
						158.84 s	52 %	Alg. 1

Experimental results show that the improved BayesianKeySearch algorithm achieves comparable success rates while reducing time complexity relative to [2].

4.3. Bayes-UCB + Improved BayesianKeySearch

We evaluated Bayes-UCB and improved BayesianKeySearch through 11-, 12-, and 13-round key recovery attacks on `SPECK32/64`, using the $\mathcal{DND}$s of `SPECK32/64` trained in [2]. For comparison, the results of [2] and [10] were reproduced on our setup (Table 1).

These results demonstrate that combining Bayes-UCB with the improved BayesianKeySearch algorithm not only reduces time complexity but also achieves a significant improvement in success rate, highlighting the effectiveness of our approach.

5. Conclusion

In this work, we revisited differential-neural cryptanalysis on `SPECK32/64`. We characterized non-random XOR-based features in ciphertext pairs beyond classical differentials and used these insights to redesign the data format and simplify the neural network, yielding a smaller model without loss of accuracy. For key recovery, we introduced Bayes-UCB for ciphertext structure selection and improved the BayesianKeySearch algorithm. Together, these refinements lower time complexity and raise success rates, offering both theoretical insight into learned features and practical gains in attack efficiency.

References

[1] R. L. Rivest, Cryptography and machine learning, in: International Conference on the Theory and Application of Cryptology, Springer, 1991, pp. 427–439.

[2] A. Gohr, Improving attacks on round-reduced speck32/64 using deep learning, in: CRYPTO (2), Vol. 11693 of Lecture Notes in Computer Science, Springer, 2019, pp. 150–179.

[3] A. Benamira, D. Gérault, T. Peyrin, Q. Q. Tan, A deeper look at machine learning-based cryptanalysis, in: EUROCRYPT (1), Vol. 12696 of Lecture Notes in Computer Science, Springer, 2021, pp. 805–835.

[4] A. Gohr, G. Leander, P. Neumann, An assessment of differential-neural distinguishers, Cryptology ePrint Archive (2022).

[5] Z. Bao, J. Lu, Y. Yao, L. Zhang, More insight on deep learning-aided cryptanalysis, in: ASIACRYPT (3), Vol. 14440 of Lecture Notes in Computer Science, Springer, 2023, pp. 436–467.

[6] L. Zhang, Z. Wang, B. Wang, Improving differential-neural cryptanalysis, IACR Commun. Cryptol. 1 (3) (2024) 13.

[7] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, The SIMON and SPECK families of lightweight block ciphers, IACR Cryptol. ePrint Arch. (2013) 404.

[8] Y. Chen, Y. Shen, H. Yu, S. Yuan, A new neural distinguisher considering features derived from multiple ciphertext pairs, The Computer Journal 66 (6) (2023) 1419–1433.

[9] E. Kaufmann, O. Cappé, A. Garivier, On bayesian upper confidence bounds for bandit problems, in: Artificial intelligence and statistics, PMLR, 2012, pp. 592–600.

[10] Z. Bao, J. Guo, M. Liu, L. Ma, Y. Tu, Conditional differential-neural cryptanalysis, IACR Cryptol. ePrint Arch. (2021) 719.

Appendix A. The Proof of Lemma 1

Proof. Expand over all possible key values:

$$\begin{aligned}
&\Pr[X^1 = x^1 \oplus K,\ X^2 = x^2 \oplus K,\ Y^1 = y^1 \oplus K,\ Y^2 = y^2 \oplus K] \\
&= \sum_{k \in \mathbb{F}_2^n} \Pr[K = k]\ \Pr[X^1 = x^1 \oplus k,\ X^2 = x^2 \oplus k,\ Y^1 = y^1 \oplus k, \\
&\qquad\qquad Y^2 = y^2 \oplus k] \\
&= \sum_{k \in \mathbb{F}_2^n} 2^{-n}\ \Pr[X^1 = x^1 \oplus k,\ X^2 = (x^1 \oplus \delta^1) \oplus k, \\
&\qquad\qquad Y^1 = (x^1 \oplus \delta^2) \oplus k,\ Y^2 = (y^1 \oplus \delta^3) \oplus k] \\
&= \sum_{k \in \mathbb{F}_2^n} 2^{-n}\ \Pr[X^1 = x^1 \oplus k,\ X^2 = (x^1 \oplus \delta^1) \oplus k, \\
&\qquad\qquad Y^1 = (x^1 \oplus \delta^2) \oplus k,\ Y^2 = (x^1 \oplus \delta^2 \oplus \delta^3) \oplus k].
\end{aligned}$$

Let $v = x^1 \oplus k$. As k ranges over $\mathbb{F}_2^n$, so does v, and we get

$$\sum_{v \in \mathbb{F}_2^n} 2^{-n} \Pr[X^1 = v,\ X^2 = v \oplus \delta^1,\ Y^1 = v \oplus \delta^2,\ Y^2 = v \oplus (\delta^2 \oplus \delta^3)].$$

Thus the event is exactly

$$\{\, X^2 = X^1 \oplus \delta^1,\ Y^1 = X^1 \oplus \delta^2,\ Y^2 = Y^1 \oplus \delta^3 \,\},$$

which, by straightforward XOR combinations of the defining equalities, yield the following six pairwise relations:

$$X^2 \oplus X^1 = \delta^1,\ Y^1 \oplus X^1 = \delta^2,\ Y^2 \oplus Y^1 = \delta^3,$$
$$Y^1 \oplus X^2 = \delta^1 \oplus \delta^2,\ Y^2 \oplus X^1 = \delta^2 \oplus \delta^3,\ Y^2 \oplus X^2 = \delta^1 \oplus \delta^2 \oplus \delta^3.$$

Conversely, these six pairwise relations imply the original three defining equations, so the corresponding events are equivalent and have identical probability. □

Appendix B. About the Neural Network

The network architecture used by Gohr in [2] is shown in Figure B.3.

Hyperparameters of the Network Architecture. The neural network takes as input a bit sequence of size $N_w \times n$, where N_w is the number of words. This sequence is reshaped into an array of dimension (N_w, n) and transposed during preprocessing. Module 2 corresponds to the Residual Network, where $Conv$ denotes a one-dimensional convolution ($Conv1D$) with either N_f or N_w filters, and k_s is the convolution kernel size. Each residual block contains two layers, and the number of input and output filters must match. Thus, the filters first expand from N_w to N_f and then contract back to N_w. The number of residual modules is determined by the parameter *depth*. Finally, the prediction head is composed of Modules 3 and 4, followed by the output layer. The specific parameter values used in the network are listed in Table B.5.

Appendix C. Process of Key Recovery Attack

To extend the distinguisher, a short s-round classical differential ($\delta \to \Delta$ with probability 2^{-p}) is prepended before an r-round $\mathcal{DND}$. Since no

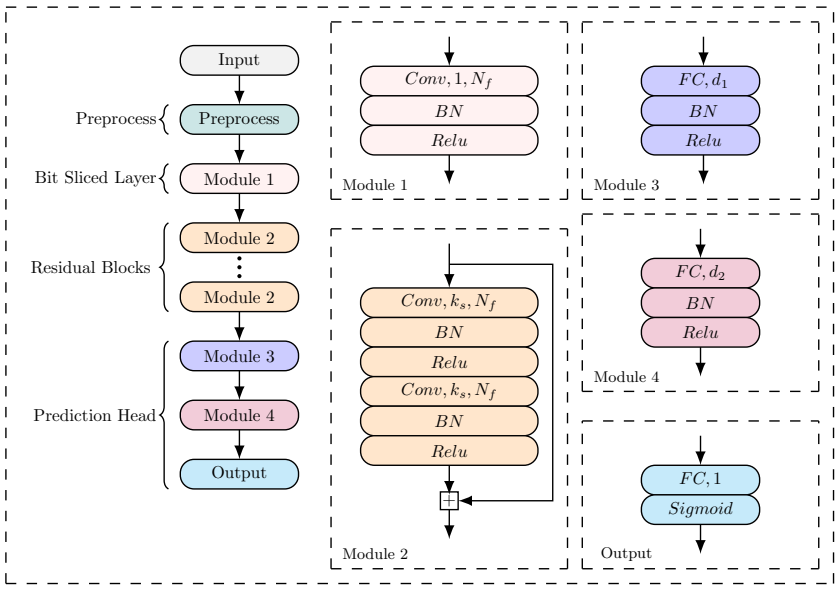

Figure B.3: Network architecture used in [2]

Table B.5: Hyperparameters of the modified neural network for `SPECK32/64`

Batch Size	β	α	$depth$	d_1	d_2	k_s	N_f	N_w	$L2$
2000	3.5×10^{-3}	10^{-4}	10	64	64	3	32	3	5×10^{-7}

key addition occurs in `SPECK32/64` prior to the first non-linear operation, plaintext pairs with input difference δ can be extended by one round without extra cost, enabling a $(1+s+r+1)$-round key recovery attack. This attack requires both the main r-round $\mathcal{DND}$ and an auxiliary $(r-1)$-round $\mathcal{DND}$, each trained on ciphertext pairs whose corresponding plaintext pairs satisfy the input difference Δ.

For ciphertext generation, we first generate about $c \times 2^p$ data pairs with input difference δ (denoted n_{cts}), where c is a small constant. Each pair is then expanded into a structure of n_b pairs using the neutral bits of the s-round $\mathcal{CD}$. Next, the n_{cts} structures are decrypted by one round with subkey 0 to obtain the corresponding plaintext structures. Finally, each plaintext structure is encrypted for $1+s+r+1$ rounds to produce the ciphertext structures. When only one round of trial decryption is used, the wrong-key

randomization hypothesis often fails, especially for lightweight ciphers. To address this, Gohr [2] proposed an efficient Bayesian key search strategy. Here, the expected response of the distinguisher under wrong-key decryption depends on the bitwise difference between a candidate subkey k_i and the true subkey k. The BayesianKeySearch algorithm then refines the candidate set by minimizing the weighted Euclidean distance, using the precomputed WKRP as the criterion.

As the number of rounds increases, the accuracy of $\mathcal{DND}$ decreases. To reduce the effect of single misclassifications, Gohr [2] combined the responses of $\mathcal{DND}$ over ciphertext structures generated with neutral bits. For a structure of size n_b, the responses $v_{i,k}$ are aggregated as

$$s_k = \sum_{i=0}^{n_b-1} \log_2\left(\frac{v_{i,k}}{1-v_{i,k}}\right),$$

where s_k is the score of a candidate subkey. This score determines the attack's runtime and success rate. A sufficiently large number of samples is required to compensate for the low accuracy of $\mathcal{DND}$.

Since $\mathcal{CD}$ is inherently probabilistic, a plaintext structure with input difference δ will evolve into a ciphertext structure with output difference Δ after s rounds only with probability 2^{-p}. As a result, the majority of ciphertext structures deviate from the expected distribution, making uniform treatment inefficient. To overcome this, Gohr [2] formulated the selection of ciphertext structures as a multi-armed bandit problem and adopted the Upper Confidence Bound (UCB) approach. The objective is to prioritize the ciphertext structure that yields the highest distinguisher score during key recovery. At iteration p, the priority of structure i is computed as

$$priority_p(i) = X_{\max}(i) + \gamma \cdot \frac{\sqrt{\log(p)}}{\sqrt{N_p(i)}},$$

where $X_{\max}(i)$ denotes the best score observed for structure i, $N_p(i)$ is its selection count up to iteration p, and $\gamma = \sqrt{n_{cts}}$. This formulation directs computational effort toward the most promising structures, striking a balance between exploration and exploitation.

Appendix D. Improved BayesianKeySearch Algorithm

Algorithm 1: Improved BayesianKeySearch

Input: Ciphertext structures $\mathcal{C} = \{C_0, \ldots, C_{n_{cts}-1}\}$; number of candidates n_{cand} per iteration; (optional) current best key K_{best}; number of iterations ℓ; $\mathcal{DND}$; wrong-key response profile (μ, σ).

Output: List L of tuples (key, score)

if $K_{\text{best}} = None$ **then** $S \leftarrow$ sample n_{cand} keys uniformly at random from the key space $\mathcal{K}$ (w/o replacement) ;
else $S \leftarrow \{K_{\text{best}}\} \cup$ (sample $n_{\text{cand}} - 1$ keys from $\mathcal{K} \setminus \{K_{\text{best}}\}$) ;
$L \leftarrow \varnothing$;
for $t = 1$ **to** ℓ **do**
  foreach $k_i \in S$ **do**
    for $j = 0$ **to** $n_{cts} - 1$ **do**
      $C'_{j,k_i} \leftarrow \text{DecryptOneRound}(C_j,\ k_i)$;
      $v_{j,k_i} \leftarrow \mathcal{DND}(C'_{j,k_i})$;
      $z_{j,k_i} \leftarrow \log\left(\dfrac{v_{j,k_i}}{1 - v_{j,k_i}}\right)$; `// log-odds`
    end
    $s_{k_i} \leftarrow \sum_{j=0}^{n_{cts}-1} z_{j,k_i}$; `// combined score via neutral bits`
    $L \leftarrow L \cup \{(k_i,\ s_{k_i})\}$;
    $m_{k_i} \leftarrow \dfrac{1}{n_{cts}} \sum_{j=0}^{n_{cts}-1} z_{j,k_i}$; `// mean log-odds`
  end
  foreach $k \in \mathcal{K}$ **do**
    $\lambda(k) \leftarrow \sum_{i=0}^{n_{\text{cand}}-1} \dfrac{\left(m_{k_i} - \mu_{\,k_i \oplus k}\right)^2}{\sigma^2_{k_i \oplus k}}$;
  end
  $S \leftarrow \underset{k \in \mathcal{K}}{\arg\min}\ \lambda(k)[0 : n_{cand} - 1]$
end
return L