Neural Aided Statistical Attack for Cryptanalysis

Yi Chen and Hongbo Yu *

Department of Computer Science and Technology, Tsinghua University, P.R. China,
chenyi19@mails.tsinghua.edu.cn
yuhongbo@mail.tsinghua.edu.cn

Abstract. At CRYPTO 2019, Gohr proposed a neural aided attack on 11-round Speck32/64, which is the first work of neural aided cryptanalysis that is competitive to the state-of-the-art attacks against reduced versions of modern block ciphers. But such an attack can only work when there are plenty of neutral bits and relies purely on experiments for complexity evaluations. In this paper, we propose a *neural aided statistical attack* that almost can be as generic as the differential cryptanalysis. It has no special requirements about the attacked cipher and allows us to estimate the theoretical complexities and success rate. For reducing the key space to be searched, we propose a *Bit Sensitivity Test* to identify which ciphertext bit is informative. Then specific key bits can be recovered by building neural distinguishers on related ciphertext bits. Applications to round reduced Speck32/64, DES prove the correctness and superiorities of our neural aided statistical attack.

Keywords: Cryptanalysis · Neural network · Normal distribution · Statistical attack · Bit sensitivity · Speck32/64 · DES.

1 Introduction

1.1 Background and motivation

Block ciphers are the most widely used class of symmetric-key primitives nowadays. Our confidence in the security of block ciphers stems from analyzing their resistance with respect to all known cryptanalytic techniques. Developing generic cryptanalysis techniques with great potential is the common practice for further understanding the practical security of block ciphers.

Differential cryptanalysis [8] is one of the generic cryptanalysis techniques. In the key recovery attack for a cipher with a block size of L, a *differential transition* with a probability p_0 higher than 2^{-L} is the only requirement. The data complexity, computation complexity, and success rate can be directly calculated once the needed differential transition is found. Besides, the adversary can focus on specific key bits instead of the complete subkey. These good properties make

* Corresponding author.

it generic and result in many advanced developments such as truncated differential [9, 25], higher-order differential [17, 25], impossible differential attack [14], boomerang attack [15, 33], collision attack [31] etc.

Neural aided cryptanalysis is another cryptanalysis technique that has received much expectation since the last century. Rivest in [30] reviewed various connections between machine learning and cryptography. Some possible directions of research in cryptanalytic applications of machine learning were also suggested. Greydanus proved that a simplified version of Enigma can be simulated by recurrent neural networks [21]. Deep learning has shown its superiorities in various fields including computer vision [26], natural language processing [7], smart medical [12], and so on. But the application of deep learning in the field of conventional cryptanalysis has been stagnant. A few valuable applications are only concentrated in the side-channel analysis [5, 11, 24].

At Crypto 2019, Gohr firstly proposed a distinguisher model based on deep learning [20]. By prepending a differential transition before neural distinguishers, Gohr improved the key recovery attack on 11-round Speck32/64. Gohr's attack requires enough neutral bits [16] in the differential transition. The neural distinguisher's output for ciphertext pairs is directly linked with the rank score for key guesses in [20]. A key guess is returned as a candidate when its rank score exceeds a threshold. However, a clear theoretical basis isn't provided for the choice of the threshold in [20]. Then the attack complexities and success rate can only be estimated by practical experiments. These properties limit its application.

In this paper, our target is to develop a new neural aided cryptanalysis technique that is as generic as *differential cryptanalysis*. Our *neural aided statistical attack* (**NASA**) almost completely achieves this target. Applications of our attack also prove its superiorities.

1.2 Our Contribution

In this paper, we mainly explore neural aided cryptanalysis.

Neural aided statistical attack. From a Bayesian perspective, solving binary classification problems with a neural network can be regarded as a stable posterior probability estimation [27]. Once we set a posterior probability threshold, it's equivalent to creating a stable classification hyperplane in the input space. Thus the probability that the posterior probability of samples coming from a certain distribution is higher than the threshold is stable and estimable.

Inspired by the fact above, we propose a *neural aided statistical distinguisher* based on a statistic related to multiple samples' classification results instead of the network's raw outputs (Section 3). In the key recovery setting, the distribution of the statistic resulted from the right key is different from that resulted from the wrong keys. Thus a basic attack model based on the statistical distinguisher for key recovery is proposed and verified (Section 4).

Our attack model has no extra requirements (eg. neutral bits) about the attacked cipher. The distinguishing of two distributions provides a theoretical framework for calculating the attack complexities and expected success rate.

Reduce key guess space by identifying informative bits. The input of Gohr's neural distinguisher is a complete ciphertext pair. In the key recovery setting, the adversary needs to guess all the bits of the subkey simultaneously. This is a serious bottleneck when the subkey has a large size.

To significantly reduce the key guess space, a *Bit Sensitivity Test* is designed to identify which ciphertext bit is informative for neural distinguishers (Section 5). Identified informative bits can guide us build new neural distinguishers on partial ciphertext bits for recovering specific subkey bits. An improved attack model adopting this technique is proposed and verified (Section 6). This technique can also be applied to improving Gohr's work.

Applications to round reduced Speck. Speck [6] is a family of block ciphers containing 10 variants. We perform attacks on round-reduced Speck32/64. The summary of our attacks together with the previous best ones is shown in Table 1.

Table 1. Summary of key recovery attacks on round reduced Speck. SD: neural aided statistical distinguisher. CP: Chosen-Plaintext

cipher	Distinguisher Type	Rounds	Complexity	Data	Source
Speck32/64	Differential Distinguisher	11	$2^{46.7}$	$2^{30.1}CP$	[2]
	Differential Distinguisher	11	2^{46}	$2^{14}CP$	[13]
	Neural Distinguisher	11	2^{38}	$2^{14.5}CP$	[20]
	SD	11	$2^{32.29}$	$2^{23.44}CP$	Section 7.3
	Differential Distinguisher	12	2^{51}	$2^{19}CP$	[13]
	Neural Distinguisher	12	-	-	[20]
	SD	12	$2^{40.35}$	$2^{27.93}CP$	Section 7.3
	Differential Distinguisher	13	2^{57}	$2^{25}CP$	[13]
	SD	13	2^{58}	$2^{28.7}CP$	Section 7.3

[1] Complexity is given in terms of the full decryption of the attacked cipher. And the complexity measure in [20] is a little different from others.

[2] Gohr also provided an attack on 12-round Speck32/64, but the data, computation complexity were not presented in [20].

For Speck32/64 reduced to $11, 12$ rounds, we can obtain the best attack. Gohr's attack model can't attack 13-round Speck32/64, but we can attack that with similar complexities as in [13].

Applications to round reduced DES. We have also performed attacks on DES reduced to 6, 7, 8 rounds. The previous best attack on 6-round DES is presented in [8], which needs 240 chosen plaintexts. Our attack on 6-round DES only requires 136 chosen plaintexts. The previous best attack on 8-round DES is presented in [4], which needs 20000 chosen plaintexts. But our attacks on 7-round and 8-round DES both require 5052 chosen plaintexts.

All the code used in this paper will be released on Github.

2 Review of Gohr's Key Recovery Attack

We briefly review the core techniques of Gohr's key recovery attack. More detailed descriptions can be found in [20].

2.1 Neural Distinguisher Model

Let (P_0, P_1) denote a plaintext pair with difference ΔP. The corresponding intermediate states, ciphertexts are (S_0, S_1), (C_0, C_1). The target of the neural distinguisher [20] is to distinguish two classes of ciphertext pairs

$$Y\left(C_0, C_1\right) = \begin{cases} 1, if\ S_0 \oplus S_1 = \Delta S \\ 0, if\ S_0 \oplus S_1 \neq \Delta S \end{cases} \tag{1}$$

$Y = 1$ or $Y = 0$ is the corresponding label of (C_0, C_1). If the difference between S_0 and S_1 is the target difference ΔS, the pair (C_0, C_1) is regarded as a positive sample drawn from the target distribution. Or (C_0, C_1) is regarded as a negative sample that comes from a uniform distribution.

A neural network is trained over $\frac{N}{2}$ positive samples and $\frac{N}{2}$ negative samples. The neural network can be used as a distinguisher if the distinguishing accuracy over a testing database is higher than 0.5.

Let ND_h denotes a neural distinguisher against the cipher reduced to h rounds. Given a sample (C_0, C_1), the neural distinguisher will output a score Z which is used as the posterior probability

$$Pr(Y = 1\,|(C_0, C_1)\,) = Z = F(C_0, C_1), \quad Z \in [0, 1] \tag{2}$$

where $F(\cdot)$ stands for the posterior probability estimation function learned by the neural distinguisher.

2.2 Gohr's Key Recovery Attack on 11-round Speck32/64

Algorithm 1 summarizes the core idea of the basic version (unaccelerated version) of Gohr's key recovery attack.

Plaintext structure created from k neutral bits. In order to extend the neural distinguishers for key recovery, a 3-round differential transition $\Delta P \rightarrow \Delta S$ extended from a 2-round differential transition $0x211/0xa04 \rightarrow 0x0040/0$ is prepended.

There are 6 high probabilistic neutral bits $\{14, 15, 20, 21, 22, 23\}$ in this 2-round differential transition. Neutral bits don't influence the differential transition. Thus a plaintext structure consisting of 64 plaintext pairs can be generated. These plaintext pairs can pass the differential transition together. Such a good property is one of the key factors resulting in Gohr's practical attack on 11-round Speck32/64.

Algorithm 1 Basic version of Gohr's key recovery attack model

Require: k neutral bits [16] that exist in the differential transition $\Delta P \rightarrow \Delta S$;
A neural distinguisher $F(\cdot)$ built from ΔS;
A key rank score threshold, c_1;
A maximum number of iterations.
Ensure: A possible key candidate.
repeat
  Random generate a plaintext pair $(P_0^1, P_1^1), P_0^1 \oplus P_1^1 = \Delta P$;
  Create a plaintext structure consisting of 2^k plaintext pairs by k neutral bits;
  Collect corresponding ciphertext pairs, $(C_0^i, C_1^i), i \in [1, 2^k]$;
  for each key guess kg **do**
    Partially decrypt 2^k ciphertext pairs with kg;
    Feed decrypted ciphertext pairs to $F(\cdot)$ and collect the outputs $Z_i, i \in [1, 2^k]$;
    Calculate the key rank score v_{kg} based on collected outputs;
    if $v_{kg} > c_1$ **then**
      stop the key search and return kg as the key candidate;
    end if
  end for
until a key candidate is returned or the maximum number of iterations is reached.

The link between key guess and score from a neural distinguisher. Decrypting 2^k ciphertext pairs drawn from the target distribution with the same key guess kg, we use the following formula

$$v_{kg} = \sum_{i=1}^{2^k} \log_2 \left(\frac{Z_i}{1 - Z_i} \right) \tag{3}$$

to combine the scores Z_i of individual decrypted ciphertext pairs into a rank score for kg. When the rank score v_{kg} exceeds a threshold c_1, kg is regarded as a key candidate.

The rank score is likely to be very high only when the plaintext structure passes the differential transition and the key guess is right. If the plaintext structure doesn't pass the differential transition or the key guess is wrong, the rank score should be very low. Then the right key can be identified by comparing the rank score with a threshold. When the neural distinguisher is weak, 2^k needs to be large. Then more neutral bits are required.

Since Gohr's attack model recovers the right key based on a subset sampled from the same distribution, such a strategy is highly dependent on neutral bits. If the number of neutral bits is not large enough, Gohr's attack can't work. The key rank score threshold is set without any clear theoretical basis. Thus we can't estimate the attack complexities and success rate theoretically.

2.3 Comparison between Gohr's Attack Model and Classic Differential Cryptanalysis

Gohr's attack model differs from the differential cryptanalysis in the following aspects:

1. Enough neutral bits must exist in the differential transition $\Delta P \rightarrow \Delta S$.
2. The data complexity, computation complexity, and final success rate can only be obtained by performing practical attacks.
3. All the decryption key bits are considered simultaneously.

These three properties lead to the fact that Gohr's attack model can't be applied as generic as the differential cryptanalysis. We guess this is also the main reason why Gohr argued "we do not think that machine learning methods will supplant traditional cryptanalysis" in [20]. We show that our **NASA** can do better.

3 Neural Aided Statistical Distinguisher

We now start to describe our neural aided statistical distinguisher. It should be noticed that all the neural distinguishers to be used are built based on Gohr's distinguisher model(Section 2.1). The implementation of our neural distinguishers is only slightly different from Gohr's: the depth of the residual block is set 1. This change doesn't influence the conclusions of our work. If there are no special instructions, the number of training samples is 10^7. Other settings about the training can refer to [20].

3.1 A Chosen Plaintext Statistical Distinguisher

Generate N chosen-plaintext pairs $(P_0^i, P_1^i), P_0^i \oplus P_1^i = \Delta P, i \in [1, N]$ randomly and collect corresponding ciphertext pairs $\left(C_0^i, C_1^i\right), i \in [1, N]$ using a cipher with a block size of L. Given a neural distinguisher $F\left(C_0, C_1\right)$, the adversary needs to distinguish between this cipher and a random permutation.

The concrete process is as follows. For each ciphertext pair $\left(C_0^i, C_1^i\right)$, the adversary feeds it into the neural distinguisher and obtains its output Z_i. Setting a threshold value c_2, the adversary calculates the statistic T

$$T = \sum_{i=1}^{N} \phi\left(Z_i\right), \quad \phi\left(Z_i\right) = \begin{cases} 1, if\ Z_i > c_2 \\ 0, if\ Z_i \leqslant c_2 \end{cases} \tag{4}$$

When the probability of the differential transition $\Delta P \rightarrow \Delta S$ is higher than 2^{-L}, it's expected that the value of the statistic T for the cipher is higher than that for a random permutation. In a key recovery setting, the right key will result in the statistic T being among the highest values for all candidate keys if N is large enough. In the sequel, we give this a theoretical analysis.

Remark 1. The value of the posterior probability threshold c_2 is selected experimentally in this paper. The value of c_2 has an indirect influence on the data complexity.

3.2 Distribution of the Statistic under Right and Wrong keys

First, we regard a ciphertext pair as a point in a high-dimensional space. For a given threshold, it's equivalent to create a stable classification hyperplane in this space using a neural distinguisher. Thus the classification over a random ciphertext pair can be modeled as a Bernoulli experiment. It provides us with a theoretical analysis framework.

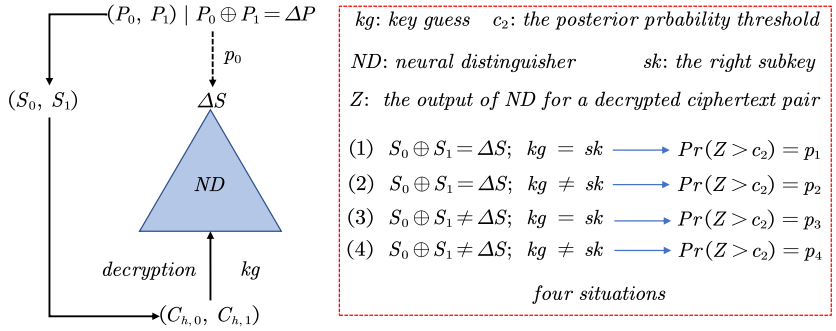

Fig. 1. Four situations of decrypting a ciphertext pair with a key guess.

According to the key recovery process, there are four possible situations when we decrypt a ciphertext pair with a key guess as shown in Fig.1:

(1) Decrypting a positive sample with the right key: the ciphertext pair satisfies the differential transition and the key guess is right.

(2) Decrypting a positive sample with wrong keys: the ciphertext pair satisfies the differential transition but the key guess is wrong.

(3) Decrypting a negative sample with the right key: the ciphertext pair doesn't satisfy the differential transition but the key guess is right.

(4) Decrypting a negative sample with wrong keys: the ciphertext pair doesn't satisfy the differential transition and the key guess is wrong.

Given a neural distinguisher, we denote the probability of $Z > c_2$ as p_1, p_2, p_3, p_4 for the four situations respectively. Then the distributions of the statistic (formula (4)) in these four situations are

$$\begin{aligned} T_1 &\sim \mathcal{N}(\mu_1, \sigma_1), \mu_1 = N_1 \times p_1, \sigma_1 = \sqrt{N_1 \times p_1 \times (1-p_1)} \\ T_2 &\sim \mathcal{N}(\mu_2, \sigma_2), \mu_2 = N_2 \times p_2, \sigma_2 = \sqrt{N_2 \times p_2 \times (1-p_2)} \\ T_3 &\sim \mathcal{N}(\mu_3, \sigma_3), \mu_3 = N_3 \times p_3, \sigma_3 = \sqrt{N_3 \times p_3 \times (1-p_3)} \\ T_4 &\sim \mathcal{N}(\mu_4, \sigma_4), \mu_4 = N_4 \times p_4, \sigma_4 = \sqrt{N_4 \times p_4 \times (1-p_4)} \end{aligned} \tag{5}$$

if N_1, N_2, N_3, N_4 are high enough. $\mathcal{N}(\mu_i, \sigma_i)$ is a normal distribution with mean μ_i and standard deviation $\sigma_i, i \in [1, 4]$. An empirical condition is

$$N_i \times p_i > 5, \quad N_i \times (1 - p_i) > 5, \quad i \in [1, 4] \tag{6}$$

Now come back to the key recovery attack. If the probability of the differential transition $\Delta P \rightarrow \Delta S$ is p_0 and N ciphertext pairs are collected randomly, then

$$N_1 = N_2 = N \times p_0, \quad N_3 = N_4 = N \times (1 - p_0) \tag{7}$$

Besides, the distributions of the statistic (formula (4)) under the right key and wrong keys are both a mixture of two normal distributions.

Right key guess. This case contains two situations in which corresponding distributions are $\mathcal{N}(\mu_1, \sigma_1)$ and $\mathcal{N}(\mu_3, \sigma_3)$. Since a mixture of two independent normal distributions is still a normal distribution, the distribution of the statistic (formula (4)) under the right key guess is:

$$T_r = T_1 + T_3 \sim \mathcal{N}(\mu_r, \sigma_r) \tag{8}$$

$$\mu_r = N \times (p_0 p_1 + (1 - p_0) p_3) \tag{9}$$

$$\sigma_r = \sqrt{N \times p_0 \times p_1 \times (1 - p_1) + N (1 - p_0) \times p_3 \times (1 - p_3)} \tag{10}$$

Wrong key guess. This case also contains two situations in which corresponding distributions are $\mathcal{N}(\mu_2, \sigma_2)$ and $\mathcal{N}(\mu_4, \sigma_4)$. Then the distribution of the statistic (formula (4)) under wrong key guesses is:

$$T_w = T_2 + T_4 \sim \mathcal{N}(\mu_w, \sigma_w) \tag{11}$$

$$\mu_w = N \times (p_0 p_2 + (1 - p_0) p_4) \tag{12}$$

$$\sigma_w = \sqrt{N \times p_0 \times p_2 \times (1 - p_2) + N (1 - p_0) \times p_4 \times (1 - p_4)} \tag{13}$$

Negative samples in the high-dimensional space approximately obey uniform distribution, thus p_3 and p_4 are theoretically equal. $p_3 \approx p_4$ can also be verified by experiments. Since the accuracy of neural distinguishers is higher than 0.5, $p_1 > p_2$ also holds with a high probability. When we set $c_2 = 0.5$, we can ensure $p_1 > p_2$. Thus $\mu_r > \mu_w$ also holds.

From the analysis above, we know that the distributions of T_r, T_w are different. Then the key recovery attack can be performed based on this finding.

3.3 Distinguishing between Two Normal Distributions

The distinguishing between two normal distributions is also adopted in zero correlation cryptanalysis [10]. Here, we still give the details.

Consider two normal distributions: $\mathcal{N}(\mu_r, \sigma_r)$, and $\mathcal{N}(\mu_w, \sigma_w)$. A sample s is sampled from either $\mathcal{N}(\mu_r, \sigma_r)$ or $\mathcal{N}(\mu_w, \sigma_w)$. We have to decide if this sample is from $\mathcal{N}(\mu_r, \sigma_r)$ or $\mathcal{N}(\mu_w, \sigma_w)$. The decision is made by comparing the value s to some threshold t. Without loss of generality, assume that $\mu_r > \mu_w$. If $s \geqslant t$, the decision is $s \in \mathcal{N}(\mu_r, \sigma_r)$. If $s < t$, the decision is $s \in \mathcal{N}(\mu_w, \sigma_w)$. Then there are error probabilities of two types:

$$\begin{aligned} \beta_r &= Pr\{s \in \mathcal{N}(\mu_w, \sigma_w) \,|\, s \in \mathcal{N}(\mu_r, \sigma_r)\} \\ \beta_w &= Pr\{s \in \mathcal{N}(\mu_r, \sigma_r) \,|\, s \in \mathcal{N}(\mu_w, \sigma_w)\} \end{aligned} \tag{14}$$

Here a condition is given on μ_r, μ_w, σ_r, σ_w such that the error probabilities are β_r and β_w. The proof can refer to related research [18] [19].

Proposition 1. *For the test to have error probabilities of at most β_r and β_w, the parameters of the normal distribution $N(\mu_r, \sigma_r)$ and $N(\mu_w, \sigma_w)$ with $\mu_r \neq \mu_w$ have to be such that*

$$\frac{z_{1-\beta_r} \times \sigma_r + z_{1-\beta_w} \times \sigma_w}{|\mu_r - \mu_w|} = 1 \tag{15}$$

where $z_{1-\beta_r}$ and $z_{1-\beta_w}$ are the quantiles of the standard normal distribution.

3.4 Data Complexity of the Statistical Distinguisher

Based on Proposition 1, one obtains the condition:

$$\frac{z_{1-\beta_r}\sigma_r + z_{1-\beta_w}\sigma_w}{\mu_r - \mu_w} = 1 \tag{16}$$

where the values of $\mu_r, \sigma_r, \mu_w, \sigma_w$ refer to formula $(9), (10), (12), (13)$ respectively. In a key recovery setting, $1 - \beta_r$ is the probability that the right key survives, β_w is the probability that the wrong keys survive.

Since we can't know the real classification hyperplane learned by the neural distinguisher, p_1, p_2, p_3, and p_4 can only be estimated experimentally. Then the estimated values of p_3 and p_4 will be slightly different even they should be theoretically equal. When the probability p_0 of the differential transition is very low, the slight distinction $p_3 - p_4$ may dominate $\mu_p - \mu_n$, which is wrong. Thus we neglect the minor difference and replace p_3, p_4 with p_n.

Then the final condition can be simplified:

$$\sqrt{N} = \frac{z_{1-\beta_r} \times \sqrt{p_0 a_1 + (1-p_0)a_3} + z_{1-\beta_w} \times \sqrt{p_0 a_2 + (1-p_0)a_3}}{(p_1 - p_2) \times p_0} \tag{17}$$

$$a_1 = p_1(1-p_1), \quad a_2 = p_2(1-p_2), \quad a_3 = p_n(1-p_n) \tag{18}$$

p_1, p_2, p_n are all constant values that are only related to the neural distinguisher. p_0 is the probability of the differential transition. The data complexity N can be directly calculated when β_r and β_w are set.

The decision threshold t is:

$$t = \mu_r - z_{1-\beta_r}\sigma_r = \mu_w + z_{1-\beta_w}\sigma_w \tag{19}$$

The analysis about how the data complexity is affected by the neural distinguisher and the differential transition is presented in appendix A.

3.5 Estimation of p_1, p_n

Consider a neural distinguisher $F(\cdot)$ against a cipher reduced to h rounds, the value of p_1, p_n can be estimated as:

1. Randomly generate M positive (negative) samples and decrypt them for 1 round with the right (random) subkeys.
2. Feed partially decrypted samples into $F(\cdot)$.
3. Calculate the final ratio of $Z > c_2$.

The ratio is the statistical expectation of p_1 or p_n. A large M can help make the statistical expectation accurate enough.

3.6 Further Analysis and Estimation of p_2

When we decrypt a positive sample with a wrong key guess (Figure 1(2)), the final value of p_2 is rather complex and related to characteristics of the wrong key guess. Such a phenomenon is based on *Property 1* and *Property 2*.

Property 1. Decrypt a ciphertext for one round with two different subkeys,

$$C^1_{h-1} = DecOneRound(C_h, sk_h), \quad C^2_{h-1} = DecOneRound(C_h, kg) \tag{20}$$

where sk_h is the right subkey, and kg is the key guess. If kg and sk_h are only different at a few bits (e.g. just 1 bit or 2 bits), C^1_{h-1} and C^2_{h-1} may be very similar in high probability. In other words, the Hamming distance between C^1_{h-1} and C^2_{h-1} will be very small.

Property 2. Given a neural network $F(\cdot)$ for solving a binary classification problem, if two input samples X_1, X_2 are very close to each other in the input space, two outputs $F(X_1)$ and $F(X_2)$ obtained from the neural network may satisfy $F(X_1) \approx F(X_2)$ with a high probability.

Although the distance metric in the input space of neural networks is complex and unknown, the Hamming distance is still a good alternative. Thus it is expected that p_2 is related to the Hamming distance between the right key and wrong key guesses. Besides, we need to consider multiple Hamming distances when the decryption covers multiple rounds.

For example, we decrypt a positive sample $(C_{h+x,0}, C_{h+x,1})$ with x subkey guesses simultaneously

$$C_{h+j-1,0/1} = DecOneRound(C_{h+j,0/1}, kg_{h+j}), \quad j \in [1, x] \tag{21}$$

where kg_{h+j} is the key guess of the $(h+j)$-th round. $(C_{h,0}, C_{h,1})$ is fed into an h-round neural distinguisher for estimating p_2.

When the last $x-1$ key guesses $kg_{h+j}, j \in [2, x]$ are all right, $(C_{h+1,0}, C_{h+1,1})$ is a positive sample. The probability of $Z > c_2$ is p_2. If $kg_{h+j}, j \in [2, x]$ are not all right, then $(C_{h+1,0}, C_{h+1,1})$ isn't a positive sample anymore. The resulted probability of $Z > c_2$ is closer to p_n.

Since x key guesses have different influences on the probability of $Z > c_2$, we need to consider x Hamming distances for estimating p_2. Let d_j denotes the Hamming distance between the right key and key guess in the $(h+j)$-th round, and $p_{2|d_1,\cdots,d_x}$ denotes the probability of $Z > c_2$. Algorithm 2 is proposed for the estimation of $p_{2|d_1,\cdots,d_x}$.

Verification. We have performed tests on 5 neural distinguishers against round reduced Speck32/64. The difference constraint is $\Delta S = (0x0040, 0)$. We train four neural distinguishers ND_4, ND_5, ND_6, ND_7 from scratch. The 8-round neural distinguisher ND_8 has been provided in [20]. Let $M = 10^7$, Table 2, 3 show the estimation results of $p_{2|d_1}$ and $p_{2|d_1,d_2}$ respectively.

Tests above have verified the analysis of p_2. Besides, when two subkeys are guessed simultaneously, $p_{2|d_1,d_2}$ will decrease sharply even if the key guess of the last round is wrong at only 1 bit.

Algorithm 2 Estimation of $p_{2|d_1,\cdots,d_x}$

Require: a cipher with a subkey size of L;
a neural distinguisher against this cipher reduced to r rounds, $F(\cdot)$;
M random plaintext pairs, $(P_0^i, P_1^i), P_0^i \oplus P_1^i = \Delta P, i \in [1, M]$;
M random master keys, $MK_i, i \in [1, M]$;
The threshold c_2;
Ensure: $p_{2|d_1,\cdots,d_x}$.

```
Encrypt each plaintext pair (P_0^i, P_1^i) with a master key MK_i for h + x rounds.
Save resulting ciphertext pair (C_0^i, C_1^i);
Save the (h + j)-th subkey sk_{h+j}^i, j ∈ [1, x];
for d_1 = 0 to L, ···, d_x = 0 to L do
    for i = 1 to M do
        Randomly draw x key guesses kg_j^i, j ∈ [1, x] where the Hamming distance
        between kg_j^i and sk_{h+j}^i is d_j;
        Decrypt (C_0^i, C_1^i) with kg_j^i, j ∈ [1, x] for x rounds;
        Feed the decrypted ciphertext pair into F(·) and save the output as Z_{i|d_1,···,d_x};
    end for
    Count the number of Z_{i|d_1,···,d_x} > c_2, and denote it as T_{d_1,···,d_x};
    Save p_{2|d_1,···,d_x} = T_{d_1,···,d_x} / M;
end for
```

Table 2. The estimation of $p_{2|d_1}$ of the neural distinguishers against round reduced Speck32/64. For ND_4, ND_5, ND_6, ND_7, $c_2 = 0.55$. For ND_8, $c_2 = 0.5$. $p_{2|d_1=0} = p_1$.

ND_4	d_1	0	1	2	3	4	5	6	7	8
	$p_{2\|d_1}$	0.995	0.5065	0.2815	0.1686	0.1088	0.0735	0.0521	0.039	0.0301
	d_1	9	10	11	12	13	14	15	16	
	$p_{2\|d_1}$	0.0239	0.0198	0.0169	0.0146	0.0129	0.0117	0.0107	0.01	
ND_5	d_1	0	1	2	3	4	5	6	7	8
	$p_{2\|d_1}$	0.8889	0.5151	0.3213	0.2168	0.1556	0.1189	0.0956	0.08	0.0694
	d_1	9	10	11	12	13	14	15	16	
	$p_{2\|d_1}$	0.0617	0.056	0.0516	0.0483	0.0456	0.0436	0.0419	0.0407	
ND_6	d_1	0	1	2	3	4	5	6	7	8
	$p_{2\|d_1}$	0.6785	0.4429	0.3135	0.2384	0.1947	0.1684	0.1518	0.1408	0.1334
	d_1	9	10	11	12	13	14	15	16	
	$p_{2\|d_1}$	0.1283	0.1247	0.1219	0.1201	0.1183	0.117	0.1171	0.1188	
ND_7	d_1	0	1	2	3	4	5	6	7	8
	$p_{2\|d_1}$	0.4183	0.3369	0.2884	0.2607	0.2442	0.234	0.2276	0.2236	0.2211
	d_1	9	10	11	12	13	14	15	16	
	$p_{2\|d_1}$	0.2193	0.2183	0.2175	0.2172	0.2167	0.2159	0.2161	0.209	
ND_8	d_1	0	1	2	3	4	5	6	7	8
	$p_{2\|d_1}$	0.5184	0.5056	0.4993	0.4957	0.4939	0.4927	0.4925	0.4918	0.4917
	d_1	9	10	11	12	13	14	15	16	
	$p_{2\|d_1}$	0.4914	0.4913	0.4913	0.4911	0.4913	0.4914	0.491	0.4914	

Thus, **the choice of p_2 depends on the target of the key recovery attack**. If we think the attack is successful as long as the Hamming distance between the key guess and the right key is not higher than a threshold d, the

Table 3. The estimation of $p_{2|d_1,d_2}$ of the 7-round distinguisher [20] against Speck32/64. $c_2 = 0.55$. Elements in the same column have the same Hamming distance d_2. Elements in the same row have the same Hamming distance d_1. Limited to the width of the table, all results only retain two decimal places. The same value is replaced by an uppercase letter. $Y = 0.21$, $E = 0.22$, $J = 0.23$, $U = 0.25$, and $V = 0.26$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.42	V	E	E	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
1	0.33	U	E	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	E
2	0.29	J	E	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
3	V	J	E	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
4	J	J	E	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
5	E	E	E	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
6	E	Y	E	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
7	E	E	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	E
8	E	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
$9 \sim 16$	$\leqslant Y$																

value of p_2 should be

$$p_2 = \max\left\{p_{2|d_1,\cdots,d_x} | d_1 + \cdots + d_x > d\right\} \tag{22}$$

This choice is based on the following truth. By setting a proper threshold c_2 such as $c_2 \geqslant 0.5$, we can ensure

$$p_{2|d_1,\cdots,d_x} \leqslant 0.5, \quad if \quad d_1 + \cdots + d_x > d \tag{23}$$

According to formula (17), the higher p_2 is, the higher the required data complexity is. The decision threshold also increases when p_2 increases. Thus we only need to focus on the highest data complexity needed for filtering wrong keys.

Take the 7-round neural distinguisher as an example. Let $d = 2$, it means that the attack is successful if the recovered key is different from the right key at most 2 bits. Then $p_2 = p_{2|3} = 0.2607$ or $p_2 = p_{2|0,1} = p_{2|3,0} = 0.26$.

4 Neural Aided Statistical Attack

Algorithm 3 Statistical test for a key guess

Require: A neural distinguisher; A key guess, kg;
A posterior probability threshold, c_2; The decision threshold, t;
N ciphertext pairs (C_0^i, C_1^i) encrypted from $(P_0^i, P_1^i), P_0^i \oplus P_1^i = \Delta P, i \in [1, N]$.

1: Decrypt N ciphertext pairs with kg;
2: Feed decrypted ciphertext pairs into the neural distinguisher;
3: Collect the neural distinguisher's outputs $Z_i, i \in [1, N]$;
4: Calculate the statistic T in formula (4);
5: **if** $T \geqslant t$ **then**
6: Return kg as a key candidate
7: **end if**

This neural aided statistical distinguisher can be used to determine whether a key guess may be the right key. This is accomplished by the *Statistical Test* as shown in Algorithm 3.

4.1 Basic Attack Model

Let's take the key recovery with 1-round decryption as the example, Algorithm 4 summarizes the basic attack model based on the neural aided statistical distinguisher.

Algorithm 4 Basic model of our neural aided statistical attack

Require: The attacked cipher;
The differential transition with a probability of p_0, $\Delta P \rightarrow \Delta S$;
Two maximum error probabilities, β_r, β_w;
A posterior probability threshold, c_2.
Ensure: All possible key candidates.
1: Train a teacher distinguisher $F(C_0, C_1)$ based on ΔS;
2: Estimate p_1, p_n, p_2 using $F(C_0, C_1)$ (Section 3.5, Algorithm 2);
3: Calculate the data complexity N and the decision threshold t (Section 3.4);
4: Randomly generate N plaintext pairs $(P_0^i, P_1^i), P_0^i \oplus P_1^i = \Delta P, i \in [1, N]$;
5: Collect corresponding N ciphertext pairs, $(C_0^i, C_1^i), i \in [1, N]$;
6: **for** each key guess kg **do**
7: Perform the statistical test (Algorithm 3);
8: **end for**
9: Test surviving key candidates against a necessary number of plaintext-ciphertext pairs according to the unicity distance for the attacked cipher.

4.2 Verification of Basic Attack Model

In order to verify our **NASA**, four practical key recovery attacks on round reduced Speck32/64 are performed. For Speck32/64 reduced to h rounds, our target is to recover the last subkey sk_h. It's expected that returned key guesses are different from the right key at most $d = 2$ bits.

Our attack model should work as long as the neural distinguisher has an accuracy higher than 0.5. Besides, the data complexity should be correctly estimated once $\Delta P \rightarrow \Delta S$, ND, d, β_r, and β_w are provided. Thus, different settings about these factors are considered.

Three neural distinguishers ND_5, ND_7, ND_8 are adopted. Table 4 shows two different differential transitions of Speck32/64 adopted in the verification. Since no key addition happens in Speck before the first nonlinear operation, these two differential transitions can be extended to a 2/3-round differential respectively.

Table 4. Two options of the prepended differential transition of Speck32/64. nr is the number of encryption rounds covered by the differential transition.

ID	$\Delta P \rightarrow \Delta S$	p_0	nr
1	$(0x2800, 0x10) \rightarrow (0x0040, 0)$	2^{-2}	1
2	$(0x211, 0xa04) \rightarrow (0x0040, 0)$	2^{-6}	2

The verification plan consists of three steps:
1. Set the value of β_r and β_w. Calculate the data complexity N.
2. Perform the neural aided statistical attack 100 times with N samples.
3. Check the following observation indexes:
   (a) The ratio that the right key passes the statistical test.

(b) The average number of surviving keys in 100 trails.
(c) The ratio that the number of surviving keys is smaller than the expected upper bound.

Table 5 summarizes the settings related to four attacks. Table 2 shows the estimations of $p_{2|d_1}$ related to ND_5, ND_7, ND_8. The value of p_2 should be $p_{2|d_1=3}$ in four attacks.

Table 5. Settings of the four attacks against round reduced Speck32/64. DID is the differential transition's ID in Table 4.

Attack ID	Attack rounds	ND	DID	p_0	c_2	p_1	d	p_2	p_n	β_r	β_w
1	10	ND_7	1	2^{-2}	0.55	0.4183	2	0.2607	0.2162	0.005	0.003
2	10	ND_7	1	2^{-2}	0.55	0.4183	2	0.2607	0.2162	0.005	2^{-16}
3	10	ND_8	-	1	0.5	0.5184	2	0.4957	0.4914	0.001	2^{-16}
4	9	ND_5	2	2^{-6}	0.55	0.8889	2	0.2168	0.0384	0.005	2^{-16}

Attack 1: recover sk_{10} of 10-round Speck32/64. In the first attack setting, we get $N = 3309 \approx 2^{11.69}$ (see formula (17)). The decision threshold is $t = 818$. The right key ($d_1 = 0$) should survive with a $1-\beta_r = 99.5\%$ probability at least. Wrong keys ($d_1 \geqslant 3$) should survive with a $\beta_w = 0.3\%$ probability at most. The number of surviving keys shouldn't exceed $137 + (2^{16} - 137) \times 0.003 = 333.197$.

After performing this attack 100 times with 3309 plaintext pairs, we find:
1. The right key ($d_1 = 0$) has passed the test in all the 100 experiments.
2. The average number of surviving keys is 124.21 that is smaller than 333.197.
3. The number of surviving keys is smaller than 333.197 in 97 experiments.

Based on the Hamming distance between the right key and key guess, the whole subkey space can be divided into 17 subspaces. We further calculate the average ratio that keys in each subspace survive the attack. Table 6 shows the average surviving ratios of 17 key subspaces.

Table 6. Average surviving ratios (SR) of key guesses in 17 subspaces.

d_1	0	1	2	3	4	5	6	7	8
SR	1	0.47063	0.17858	0.05911	0.01697	0.00412	0.00097	0.00025	0.00008
d_1	9	10	11	12	13	14	15	16	
SR	0.00003	0.00002	0.00001	0.00001	0	0	0	0	

Attack 2: recover sk_{10} of 10-round Speck32/64. In the second attack setting, $N = 5274 \approx 2^{12.34}$ and $t = 1325$. The number of surviving keys shouldn't exceed $137 + (2^{16} - 137) \times 2^{-16} \approx 137.998$.

After performing this attack 100 times with 5274 plaintext pairs, we find:
1. The right key ($d_1 = 0$) has passed the test in 99 experiments.
2. The average number of surviving keys is 63.54 that is smaller than 137.998.
3. The number of surviving keys is smaller than 137.998 in 98 experiments.

Table 7 shows the average surviving ratios of 17 subspaces.

Table 7. Average surviving ratios (SR) of key guesses in 17 subspaces.

d_1	0	1	2	3	4	5	6	7	$8 \sim 16$
SR	0.99	0.37875	0.1245	0.03429	0.00795	0.00144	0.00018	0.00001	0

Attack 3: recover sk_{10} of 10-round Speck32/64. ND_8 is a very weak distinguisher. Its accuracy is only about 0.518. In the third attack setting, $N = 25680 \approx 2^{14.65}$ and $t = 13064$. The number of surviving keys shouldn't exceed $137 + (2^{16} - 137) \times 2^{-16} \approx 137.998$.

After performing this attack 100 times with 25680 plaintext pairs, we find:

1. The right key ($d_1 = 0$) has passed the test in all the 100 experiments.
2. The average number of surviving keys is 77.47 that is smaller than 137.998.
3. The number of surviving keys is smaller than 137.998 in 85 experiments.

Table 8 shows the average surviving ratios of 17 subspaces.

Table 8. Average surviving ratios (SR) of key guesses in 17 subspaces.

d_1	0	1	2	3	4	5	6	7	8	$9 \sim 16$
SR	1	0.44438	0.14592	0.04055	0.00949	0.00194	0.00032	0.00006	0.00001	0

Attack 4: recover sk_9 of 9-round Speck32/64. In the fourth attack setting, $N = 15905 \approx 2^{13.957}$ and $t = 758$. The number of surviving keys shouldn't exceed $137 + (2^{16} - 137) \times 2^{-16} \approx 137.998$.

After performing this attack 100 times with 15905 plaintext pairs, we find:

1. The right key ($d_1 = 0$) has passed the test in all the 100 experiments.
2. The average number of surviving keys is 18.41 that is smaller than 137.998.
3. The number of surviving keys is smaller than 137.998 in 100 experiments.

Table 9 shows the average surviving ratios of 17 subspaces.

Table 9. Average surviving ratios (SR) of key guesses in 17 subspaces.

d_1	0	1	2	3	4	5	6	$7 \sim 16$
SR	1	0.27813	0.05617	0.0082	0.00071	0.00006	0.00001	0

It's clear that these four attacks have achieved the most important two targets of the basic attack model. Although the ratio that keys of each subset pass the test is not strictly consistent with the theoretical value, the total number of surviving keys is within the upper bound. This shows the Hamming distance is a good distance metric for the estimation of p_2. The correctness of our neural aided statistical distinguisher can be also well verified.

To ensure the accuracy of our **NASA**, the data complexity N and the decision threshold t should be as accurate as possible. Thus the estimations of p_0, p_1, p_2, p_n are very important. Here we assume that the probability p_0 of the differential transition is accurate. For p_1, p_2, p_n, it's better to keep more decimal places. After adequate experiments, we suggest that at least 4 decimal places should be kept. More in-depth analysis about the data complexity of the **NASA** can see appendix A.

5 Reduce the Key Space

So far we still need to guess all the bits of the subkey simultaneously, since the neural distinguisher takes the complete ciphertext pairs (C_0, C_1) as the input. When the subkey has a large size, this is a serious bottleneck.

5.1 An Intuitive Method for Reducing the Key Space

An intuitive method for reducing the key space is building the neural distinguisher on partial ciphertext bits

$$C_i = C_i[L-1]||\cdots||C_i[0], i \in [0,1] \tag{24}$$

$$\Gamma = \{x_1, x_2, \cdots, x_k\}, x_1 > \cdots > x_k, k <= L \tag{25}$$

$$\varphi(C_i, \Gamma) = C_i[x_1]||C_i[x_2]||C_i[x_k], i \in [0,1] \tag{26}$$

$$Y(\varphi(C_0,\Gamma), \varphi(C_1,\Gamma)) = \begin{cases} 1, & if\ S_0 \oplus S_1 = \Delta S \\ 0, & if\ S_0 \oplus S_1 \neq \Delta S \end{cases} \tag{27}$$

$$Pr(Y = 1\,|(\varphi(C_0,\Gamma), \varphi(C_1,\Gamma))) = Z = f(\varphi(C_0,\Gamma), \varphi(C_1,\Gamma)) \tag{28}$$

where $C_i[0]$ is the least significant bit of the ciphertext C_i, Γ is the subscript set of selected ciphertext bits, $f(\cdot)$ is the neural distinguisher built on selected ciphertext bits.

Such a method can significantly reduce the key space to be searched. But **which ciphertext bits should we select for building $f(\cdot)$? Can we develop a generic and efficient framework for guiding this selection?** In order to better introduce our work for solving these problems, three new concepts are proposed first.

Definition 1 *An **informative bit** is the ciphertext bit that is helpful to distinguish the cipher and a pseudo-random permutation.*

Definition 2 *For a cipher reduced to h rounds, the neural distinguisher $F(\cdot)$ trained on the complete ciphertexts (C_0, C_1) is denoted as the **teacher distinguisher** ND_h^t, the neural distinguisher $f(\cdot)$ trained on the selected ciphertext bits $(\varphi(C_0,\Gamma), \varphi(C_1,\Gamma))$ is denoted as the **student distinguisher** ND_h^s. The teacher distinguisher can be viewed as a special student distinguisher.*

5.2 Identify Informative Bits by Bit Sensitivity Test

It's clear that student distinguishers should be built on informative bits. However, it's hard to identify informative bits according to **Definition 1**. Thus we propose an approximate definition of the informative bit.

Definition 3 *For a teacher distinguisher $F(C_0, C_1)$, if the distinguishing accuracy is greatly affected by the j-th bit of C_0, C_1, the j-th ciphertext bit is an **informative bit**.*

The reason why a teacher distinguisher can work is that it has learned knowledge from ciphertext bits. According to **Definition 1**, only the informative bit can provide knowledge. Thus the ciphertext bit that has a significant influence on the performance of the teacher distinguisher must be the *informative bit*.

Definition 3 can't ensure each informative bit that obeys the **Definition 1** is identified successfully. But we only care about informative bits that can

be captured by a teacher distinguisher. This approximate definition can help develop a simple but effective framework for identifying informative bits.

The proposed framework is named **Bit Sensitivity Test**. Its core idea is to test whether the teacher distinguisher's accuracy drops after we remove some knowledge related to the specific bit. This framework is based on the following property

Property 3. Let X denote a variable that obeys a Bernoulli distribution. By XOR X with a random mask $\eta \in [0,1]$, X can be randomized since

$$Pr(X=1)=p; \quad Pr(X=0)=1-p$$
$$Pr(X \oplus \eta = 1) = Pr(X \oplus \eta = 0) = 0.5 \times (p+1-p) = 0.5$$

Property 3 provides a way to remove the knowledge related to a specific ciphertext bit. The input of a teacher distinguisher is a ciphertext pair (C_0, C_1). Gohr in [20] has proved that teacher distinguishers can capture the knowledge about the ciphertext difference and some unknown features. We can remove the knowledge about the j-th ciphertext bit's difference by

$$C_0 = C_0 \oplus (\eta << j) \quad or \quad C_1 = C_1 \oplus (\eta << j) \tag{29}$$

where η is a random mask that could be 0 or 1. We have performed an extreme test on teacher distinguishers against 4/5/6/7/8-round Speck32/64. If we XOR each bit of C_0 or C_1 with a random mask, teacher distinguishers can't distinguish positive samples and negative samples anymore. These tests imply that knowledge about unknown features can also be removed by one of the two operations above. Subsequent reverse verification can further prove it.

After the knowledge related to a ciphertext bit is removed, the decrease of the teacher distinguisher's accuracy is named **bit sensitivity**, which is used to identify informative bits. Algorithm 5 summarizes the **Bit Sensitivity Test**.

Algorithm 5 Bit Sensitivity Test

Require: a cipher with a block size of L;
a teacher distinguisher against this cipher, $F(C_0, C_1)$;
a test dataset consisting of $\frac{M}{2}$ positive samples and $\frac{M}{2}$ negative samples;
Ensure: An array *sen* that saves the bit sensitivity of L ciphertext bits.
1: Test the distinguishing accuracy of the neural distinguisher on the test dataset. Save it to $sen[L]$.
2: **for** $j = 0$ to $L-1$ **do**
3: **for** $i = 1$ to M **do**
4: Generate a random mask $\eta \in [0,1]$;
5: $C_0^{i,new} = C_0^i \oplus (\eta << j)$;
6: Feed the new sample $(C_0^{i,new}, C_1^i)$ to the neural distinguisher;
7: **end for**
8: Count the current accuracy *cp*;
9: $sen[j] = sen[L] - cp$;
10: **end for**

Table 10. Results of *Bit Sensitivity Test* of teacher distinguishers against Speck32/64 under three scenarios. sen_0 is the results of performing $C_0^i \oplus (\eta << j)$, sen_1 is the results of performing $C_1^i \oplus (\eta << j)$, $sen_{0,1}$ is the results of performing two operations simultaneously, $i \in [1, 10^6], j \in [0, 31]$. All results are only to three decimal places.

Bit index	ND_7^t			ND_6^t			ND_5^t		
	sen_0	sen_1	$sen_{0,1}$	sen_0	sen_1	$sen_{0,1}$	sen_0	sen_1	$sen_{0,1}$
0	0	0	0	0	0	0	0.001	0.001	0
1	0	0	0	0	0	0	0.004	0.004	0
2	0.001	0.001	0.001	0.028	0.028	0.018	0.168	0.169	0.071
3	0.006	0.006	0.003	0.111	0.111	0.022	0.222	0.221	0.119
4	0.054	0.053	0.004	0.15	0.15	0.017	0.174	0.174	0.028
5	0.056	0.056	0.001	0.006	0.006	0.003	0.142	0.141	0
6	0	0	0	0	0	0	0	0	0
7	0.001	0.001	0	0	0	0	0	0	0
8	0	0	0	0.001	0.001	0.001	0.002	0.003	0.002
9	0.002	0.002	0.001	0.007	0.007	0.007	0.021	0.021	0.005
10	0.006	0.006	0.006	0.039	0.039	0.022	0.205	0.205	0.043
11	0.023	0.023	0.021	0.138	0.138	0.034	0.167	0.167	0.061
12	0.054	0.054	0.022	0.116	0.116	0.041	0.162	0.162	0.037
13	0.053	0.053	0.016	0.084	0.084	0.028	0.098	0.098	0.031
14	0.045	0.045	0	0.075	0.075	0	0.089	0.089	0
15	0	0	0	0	0	0	0	0	0
16	0	0	0	0.026	0.026	0	0.058	0.058	0
17	0.001	0.001	0	0.082	0.082	0	0.216	0.216	0
18	0.007	0.007	0.001	0.096	0.096	0.018	0.2	0.2	0.071
19	0.014	0.014	0.003	0.134	0.134	0.023	0.223	0.223	0.119
20	0.054	0.054	0.004	0.15	0.15	0.017	0.174	0.174	0.028
21	0.056	0.056	0	0.112	0.112	0	0.142	0.141	0
22	0	0	0	0.001	0.001	0	0	0.001	0
23	0.001	0.001	0	0.002	0.002	0.001	0.002	0.002	0
24	0.002	0.002	0.001	0.009	0.009	0.002	0.018	0.018	0.002
25	0.015	0.015	0.003	0.047	0.047	0.009	0.088	0.089	0.005
26	0.038	0.038	0.011	0.102	0.102	0.022	0.214	0.214	0.043
27	0.058	0.058	0.02	0.154	0.154	0.034	0.188	0.188	0.06
28	0.072	0.072	0.022	0.136	0.136	0.04	0.18	0.18	0.037
29	0.053	0.053	0.016	0.084	0.084	0.028	0.098	0.098	0.031
30	0.045	0.045	0	0.075	0.075	0	0.089	0.089	0
31	0	0	0	0	0	0	0	0	0

Examples and analysis. We have applied the bit sensitivity test to teacher distinguishers against Speck32/64. Table 10 shows the results sen_0, sen_1, $sen_{0,1}$ of the bit sensitivity test under three scenarios.

According to Table 10, we can observe that $sen_0 \approx sen_1$. This can prove that $C_0 \oplus (\eta << j)$ is equivalent to $C_1 \oplus (\eta << j)$. Besides, we can know

1. If $sen_0[j] > 0$, the j-th ciphertext bit is an informative bit.
2. If $sen_{0,1}[j] > 0$, the j-th ciphertext bit provides some useful unknown features. Since the knowledge about the bit difference is not removed, then only useful unknown features can lead to a decrease in the accuracy.

3. If $sen_0[j] \approx sen_{0,1}[j]$, the j-th ciphertext bit's difference has little influence on the neural distinguisher.

The bit sensitivity test doesn't need any human knowledge except for the teacher distinguisher itself. In fact, it can help us understand what knowledge has been learned by the neural distinguisher. But the related work has gone beyond the subject of this paper, so we leave it to future reports.

Reverse verification about identified informative bits. To further verify **Definition 3**, a reverse verification about identified informative bits can be performed. First, select some informative bits. Second, train a student distinguisher on selected informative bits and observe the distinguishing accuracy.

Taking the ND_7^t against Speck32/64 as an example, we have performed the reverse verification based on the result in Table 10. Table 11 shows the distinguishing accuracies under two settings. For Speck32/64, the j-th and $(j+16)$-th bit are directly related to the same subkey bit. Thus the 8-th and 1st ciphertext bits are also considered.

Table 11. Accuracies of neural distinguishers trained on selected ciphertext bits

Γ	$\{30 \sim 23, 14 \sim 7\}$	$\{30 \sim 23, 21 \sim 17, 14 \sim 7, 5 \sim 1\}$	$\{31 \sim 0\}$
Accuracy	0.5414	0.6065	0.6067

The accuracy of the teacher distinguisher is 0.6067. When all the identified informative bits are considered, the resulted student distinguisher can obtain a distinguishing accuracy of 0.6065, which is almost the same as 0.6067. Such an experiment shows that **Definition 3** can help identify all the ciphertext bits that have a significant influence on teacher distinguishers.

5.3 Improve the Student Distinguisher

Student distinguishers are trained on selected ciphertext bits which only provide partial knowledge about the attacked cipher. One way to improve student distinguishers is pretraining the student distinguisher with the help of the teacher distinguisher.

Algorithm 6 Training of the student distinguisher

Require: A training dataset consisting of $\frac{M}{2}$ positive samples and $\frac{M}{2}$ negative samples, $(C_0^i, C_1^i, Y^i), i \in [1, M]$, Y^i is the real sample label;
The subscript set of selected ciphertext bits Γ;
A teacher distinguisher, $F(\cdot)$.
Ensure: The student distinguisher trained on selected ciphertext bits, $f(\cdot)$.
1: **for** $i = 1$ to M **do**
2: get the prediction of (C_0^i, C_1^i) from the teacher distinguisher, $F(C_0^i, C_1^i)$;
3: $\tilde{Y^i} = F(C_0^i, C_1^i) > 0.5 \ ? \ 1 : 0$;
4: **end for**
5: pretrain the student distinguisher on $(\varphi(C_0^i, \Gamma), \varphi(C_1^i, \Gamma), \tilde{Y^i}), i \in [1, M]$
6: train the student distinguisher on $(\varphi(C_0^i, \Gamma), \varphi(C_1^i, \Gamma), Y^i), i \in [1, M]$

This is inspired by the idea of knowledge distillation [23]. Hinton in [23] found that the output of a neural network contains the knowledge learned by itself. More details about knowledge distillation can refer to [3, 23, 28, 34]. If we take the output of the teacher distinguisher as the sample label for pretraining the student distinguisher, the student distinguisher is likely to learn extra knowledge that is failed to be captured when dropping some ciphertext bits. Algorithm 6 summarizes the training method.

We have tested Algorithm 6 on several ciphers. Let $\Gamma = \{30 \sim 23, 14 \sim 7\}$, for Speck32/64 reduced to 5/6/7 rounds, if we don't adopt the pretraining, the accuracies of the student distinguisher are $0.7981, 0.6388, 0.5414$ respectively. By adopting the pretraining, the accuracies are $0.7982, 0.6391, 0.5437$ respectively. The training/test dataset, the cyclic learning rate scheme, and other learning parameters are the same.

For Speck128/128 reduced to 8 rounds, the teacher distinguisher built with $\Delta S = (0, 0x80)$ [2] has an accuracy of 0.8304. Let $\Gamma = \{93 \sim 72, 29 \sim 8\}$, if pre-training is not adopted, we have encountered a problem that the training is very unstable. Even if the training epoch is set to 200, it is difficult to obtain a student distinguisher with an accuracy higher than 0.5. After restarting the training many times, we finally get a student distinguisher with an accuracy of 0.6281. If we adopt the pretraining, we can easily obtain a student distinguisher with an accuracy of 0.6406. The training epochs in two stages are both 10.

6 Improved Neural Aided Statistical Attack

6.1 Improved Attack Model

The technique in Section 5 can be used to improve the basic attack model (Algorithm 4). Here we still take the key recovery with 1-round decryption as the example for explanation. Algorithm 7 summarizes the improved attack model.

6.2 Verification of Improved Attack Model

Two practical key recovery attacks are performed on round reduced Speck32/64 using the improved attack model.

Attack 5: recover sk_9, sk_{10} of 10-round Speck32/64. By setting $\Gamma = \{30 \sim 23, 14 \sim 7\}$, we have trained two student distinguishers ND_7^s, ND_6^s using Algorithm 5. Let $c_2 = 0.55$, Table 12 shows the estimation of $p_{2|d_1}$ of two student distinguishers.

Table 12. The estimation of $p_{2|d_1}$ of ND_6^s, ND_7^s against Speck32/64. The subscript set of selected ciphertext bits is $\Gamma = \{30 \sim 23, 14 \sim 7\}$

ND_6^s	d_1	0	1	2	3	4	5	6	7	8
	$p_{2\|d_1}$	0.5132	0.4057	0.3402	0.3025	0.2817	0.2706	0.265	0.2617	0.2594
ND_7^s	d_1	0	1	2	3	4	5	6	7	8
	$p_{2\|d_1}$	0.3576	0.3230	0.3036	0.2940	0.2893	0.2873	0.2866	0.2863	0.2862

Then this attack is divided into four stages. Table 13 summarizes the attack settings of four stages. The expected maximum Hamming distance between surviving keys and the right key is d.

Algorithm 7 Improved model of our neural aided statistical attack

Require: The attacked cipher;
The differential transition with a probability of p_0, $\Delta P \rightarrow \Delta S$;
Ensure: All possible key candidates.
1: Train a teacher distinguisher $F(C_0, C_1)$ based on ΔS;
2: Perform the *Bit Sensitivity Test* (Algorithm 5) for identifying informative bits;
3: Design several stages for the entire key recovery;
4: The maximum number of generated ciphertext pairs, $N_{max} \leftarrow 0$;
5: **for** each stage **do**
6: Determine the attack target in the current stage;
7: Find the subscript set Γ of ciphertext bits related to the current attack target;
8: Train the student distinguisher $f(\varphi(C_0, \Gamma), \varphi(C_1, \Gamma))$ (Algorithm 6);
9: Estimate p_1, p_n, p_2 using the student distinguisher (Section 3.5, 3.6);
10: Set β_r, β_w;
11: Calculate the data complexity N and the decision threshold t (Section 3.4);
12: **if** $N \leqslant N_{max}$ **then**
13: Randomly draw N ciphertext pairs from N_{max} ciphertext pairs;
14: **else**
15: Add $N - N_{max}$ plaintext pairs that have the difference ΔP;
16: Collect corresponding ciphertext pairs, $(C_0^i, C_1^i), i \in [1, N - N_{max}]$;
17: $N_{max} \leftarrow N$;
18: **end if**
19: **for** each key guess kg in the key space **do**
20: Perform the statistical test (Algorithm 3);
21: **end for**
22: **end for**
23: Test surviving key candidates against a necessary number of plaintext-ciphertext pairs according to the unicity distance for the attacked cipher.

Table 13. Attack settings of the key recovery attack on 10-round Speck32/64. SD_i^t/SD_i^s: the neural aided statistical distinguisher built on ND_i^t/ND_i^s.

stage	SD	β_r	β_w	d	c_2	p_0	p_1	p_2	p_n	N	t	Related keys
1	SD_7^s	0.005	2^{-8}	2	0.55	2^{-2}	0.3576	0.2940	0.2863	22540	6677	$sk_{10}[7 \sim 0]$
2	SD_7^t	0.005	2^{-16}	2	0.55	2^{-2}	0.4183	0.2607	0.2162	5274	1325	sk_{10}
3	SD_6^s	0.001	2^{-14}	1	0.55	2^{-2}	0.5132	0.3402	0.2603	5228	1589	$sk_{10}, sk_9[7 \sim 0]$
4	SD_6^t	0.001	2^{-16}	1	0.55	2^{-2}	0.6785	0.3135	0.1164	830	181	sk_{10}, sk_9

In stage 1, we guess the 8 key bits $sk_{10}[7 \sim 0]$. In stage 2, we guess the complete sk_{10} based on each surviving $sk_{10}[7 \sim 0]$. Based on **Attack 2**, we know the number of surviving sk_{10} is about 2^6. In stage 3, we guess $sk_9[7 \sim 0]$ and test each possible pair $(sk_{10}, sk_9[7 \sim 0])$. After stage 4, it's expected that sk_{10} is recovered and sk_9 has at most 1 wrong bit. Finally, the number of surviving (sk_{10}, sk_9) shouldn't exceed $1 + 16 + (2^{16} - 17) \times 2^{-16} \approx 18$. In stage 2, 3, 4, the required samples are randomly drawn from 22540 samples.

After performing the attack 100 times with 22540 plaintext pairs, we find:

1. The right subkey pair (sk_{10}, sk_9) survives in 99 experiments.
2. In four stages, the average numbers of surviving subkey guesses are 12.53, 50.43, 20.52, and 14.73 respectively. Compared with that of the basic attack model, the average key space (in stage 1 and 2) is reduced from 2^{16} to

$2^8 + 12.53 \times 2^8 \approx 2^{11.76}$. In stage 3 and 4, the average key space is reduced from $51 \times 2^{16} \approx 2^{21.67}$ to $51 \times 2^8 + 20.52 \times 2^8 \approx 2^{14.16}$.

3. In each experiment, **only the pair which has the right sk_{10} survives the attack**. This can further prove our analysis about the p_2 in Section 3.6.

Attack 6: recover sk_{11} of 11-round Speck32/64. The attack settings for stage 1, 2 in **Attack 5** can be used to attack 11-round Speck32/64. Now the prepended 3-round differential transition is the second differential transition in Table 4. Table 14 summarizes the settings of the two-stage attack.

Table 14. Settings of the key reovery attack on 11-round Speck32/64.

stage	SD	β_r	β_w	d	c_2	p_0	p_1	p_2	p_n	N	t	Related keys
1	SD_7^s	0.005	2^{-8}	2	0.55	2^{-6}	0.3576	0.2940	0.2863	5678510	1629087	$sk_{11}[7 \sim 0]$
2	SD_7^t	0.005	2^{-16}	2	0.55	2^{-6}	0.4183	0.2607	0.2162	1275708	278679	sk_{11}

In the first stage, the number of surviving keys shouldn't exceed 37.86. In the second stage, the number of surviving keys shouldn't exceed 137.998. After performing the attack 100 times with 5678510 plaintext pairs, we find:

1. The right subkey sk_{11} has survived in 100 experiments.
2. In two stages, the average numbers of surviving keys are 13.23 and 55.21 respectively. Compared with that of the basic attack model, the average key space is reduced from 2^{16} to $2^8 + 13.23 \times 2^8 \approx 2^{11.83}$.

We have also performed an attack to recover the last two subkeys of 11-round Speck32/64. Due to the hardware limitation, such an attack has been performed 30 times in about two weeks. The results are the same as those in **Attack 5**. Especially, only the subkey guess pair which has the right sk_{11} survives.

7 Applications

Our **NASA** is applied to round-reduced DES (practical attacks) and Speck32/64 (theoretical analysis). The master key of h-round Speck32/64 can be directly recovered once the last 4 subkeys are recovered. This is based on the key schedule inversion algorithm [13].

7.1 Computation Complexity

The basic operation of a key recovery attack usually contains two parts. The first is the decryption of a ciphertext with a key guess. The second is the verification of a decrypted ciphertext. The key guess space and data complexity determine the total computation complexity. In conventional cryptanalysis, the verification (eg. finding the right ciphertext pair) is usually much simpler than the decryption. Then the time consumption of the second part is neglected.

For neural aided cryptanalysis, the verification is performed by neural networks that contain massive computations. Although there are many efforts in speed optimization [1, 22, 29, 32], the time consumption of neural networks is higher than that of the decryption. Table 15 shows the time consumption of neural distinguishers against round-reduced Speck32/64. A SIMD-parallelized implementation of Speck32/64 is used.

Table 15. Time consumption of neural distinguishers against round reduced Speck32/64. M: the number of tested ciphertexts. t_1: the average time of decrypting M ciphertexts for 1-round. t_2: the average time of distinguishers' verification for M ciphertexts. The CPU is Intel(R) Core(TM) i5-7500. One graphics card (NVIDIA GeForce GTX 1060(6GB)) is used. The number(batch size) of ciphertext pairs fed into a neural distinguisher each time is 2^{18}. All the results are based on 100 experiments.

M	2^{21}	2^{23}	2^{32}
t_1(seconds)	0.029	0.121	60.392
t_2(seconds, ND^t)	1.779	7.091	3643.392
t_2(seconds, ND^s)	0.945	3.725	1935.36

The neural network for implementing our neural distinguishers is not too complex (Section 3). Our graphics card is weak. If different neural networks or graphics cards are adopted, the time difference will change.

For neural aided cryptanalysis, we recommend the time consumption of neural distinguishers should be put aside. First, from the perspective of cryptanalysis, we should spend our efforts on reducing the data complexity or the key guess space, which also determines the time consumption of neural distinguishers. Second, the verification of neural distinguishers can be executed in parallel if there are more graphics cards. Third, a neural network's inference speed can be optimized by better hardware.

Thus in the sequel attacks, the complexity is in terms of the full decryption of the attacked cipher. This is consistent with previous research.

7.2 Key Recovery Attacks on Round Reduced DES

Key recovery attack on 6-round DES. In order to attack 6-round DES, we have trained a 5-round teacher distinguisher ND_5^t with $\Delta S = (0x200008, 0x0400)$ [8]. The distinguishing accuracy is 0.6289.

Table 16 shows the the result of *Bit Sensitivity Test*. Here we try to recover the 6 bits $sk_6[23 \sim 18]$ that will affect the 4-th Sbox's output in the 6-th round. Then $\Gamma = \{63, 54, 44, 38, 31, 22, 12, 6\}$, and we have trained a student distinguisher ND_5^s using Algorithm 6. The accuracy of the student distinguisher is 0.62. Let $c_2 = 0.55$, Table 17 shows the estimation of $p_{2|d_1}$.

Table 16. Results of *Bit Sensitivity Test* of ND_5^t against 5-round DES

Index	$0 \sim 31$	32	33	34	35	36	37	38	39	40	41
sen_0	0	0.008	0.002	0	0.021	0.018	0.001	0.021	0.005	0.004	0.001
Index	42	43	44	45	46	47	48	49	50	51	52
sen_0	0.005	0.001	0.019	0.008	0.012	0.001	0	0.001	0.004	0.005	0.006
Index	53	54	55	56	57	58	59	60	61	62	63
sen_0	0.015	0.024	0.002	0.004	0.004	0	0.001	0.012	0.005	0.005	0.018

Table 17. The estimation of $p_{2|d_1}$ of ND_5^s against 5-round DES when $\Gamma = \{63, 54, 44, 38, 31, 22, 12, 6\}$.

d_1	0	1	2	3	4	5	6
$p_{2\|d_1}$	0.3036	0.0857	0.0745	0.0748	0.0747	0.0836	0.0823

In this attack, we try to recover the right key. Then $p_2 = p_{2|d_1=1}$ since $d = 0$. After test, $p_1 = 0.3036, p_n = 0.0631$. Let $\beta_r = 0.005, \beta_w = 2^{-6}$. we get $N = 68 \approx 2^{6.09}$ and the decision threshold is $t = 10.69$. The number of surviving keys shouldn't exceed $1 + (64 - 1) \times 2^{-6} = 1.98$. We have performed the attack 100 times, the results are:

1. The right key has passed the test in 100 experiments.
2. The average number of surviving keys is 2.18 that is a little higher than 1.98.
3. The number of surviving keys is 1 or 2 in 65 experiments. The number of surviving keys is 3 in 20 experiments.

Although the average number of surviving keys is a little higher than the upper bound, actually we still achieve the attack target. If we choose $\Gamma = \{60, 53, 45, 35, 28, 21, 13, 3\}$, the student distinguisher can obtain an accuracy of 0.619. The 6 key bits $sk_6[35 \sim 30]$ that affect the 6-th Sbox's output can also be recovered using 68 chosen-plaintext pairs. Then the left $56 - 12 = 44$ key bits can be recovered through two stages. First, filter the wrong key guesses for each Sbox using the 68 samples. Second, test surviving key guesses with several plaintext-ciphertext pairs. Thus, the 6-round DES can be practically broken with $68 \times 2 = 136$ chosen plaintexts.

The best key recovery attack against 6-round DES provided in [8] needs 240 chosen plaintexts. Our attack can reduce the data complexity by a factor of about 1.76.

Key recovery attack on 7-round DES. In order to attack 7-round DES, we have trained a 6-round teacher distinguisher ND_6^t with $\Delta S = (0x200008, 0x0400)$ [8]. The distinguishing accuracy is 0.5499.

Table 18. Results of *Bit Sensitivity Test* of ND_6^t against 6-round DES

Index	$0 \sim 31$	32	33	34	35	36	37	38	39	40	41
sen_0	0	0	0.018	0.007	0	0.001	0.008	0	0.005	0.011	0.018
Index	42	43	44	45	46	47	48	49	50	51	52
sen_0	0.001	0.008	0	0	0.001	0.018	0.011	0.007	0.004	0.001	0.001
Index	53	54	55	56	57	58	59	60	61	62	63
sen_0	0	0	0.018	0.005	0.001	0.01	0.009	0	0.005	0.001	0

Table 19. The estimation of $p_{2|d_1}$ of ND_6^s against 6-round DES when $\Gamma = \{55, 47, 41, 33, 23, 15, 9, 1\}$.

d_1	0	1	2	3	4	5	6
$p_{2\|d_1}$	0.0908	0.0655	0.0648	0.0645	0.0644	0.0641	0.0642

Table 18 shows the the result of *Bit Sensitivity Test*. Here we try to recover the 6 bits $sk_7[5 \sim 0]$ that will affect the 1st Sbox's output in the 7-th round. Then $\Gamma = \{55, 47, 41, 33, 23, 15, 9, 1\}$, and we have trained a student distinguisher using Algorithm 5. The distinguishing accuracy is 0.5249. Let $c_2 = 0.55$, Table 19 shows the estimation of $p_{2|d_1}$.

In this attack, we try to recover the right key. Thus $p_2 = p_{2|d_1=1}$ since $d = 0$. After test, $p_1 = 0.0908, p_n = 0.0624$. Let $\beta_r = 0.005, \beta_w = 2^{-6}$, we get $N = 2526 \approx 2^{11.3}$ and $t = 192$. The number of surviving keys shouldn't exceed 1.98. We have performed the attack 100 times, the results are:

1. The right key has passed the test in 99 experiments.
2. The average number of surviving keys is 1.67 that is smaller than 1.98.
3. The number of surviving keys is 1 or 2 in 84 experiments. The number of surviving keys is 3 in 10 experiments.

If we choose $\Gamma = \{58, 48, 40, 34, 26, 16, 8, 2\}$, the corresponding student distinguisher can obtain an accuracy of 0.519. The 6 key bits $sk_7[17 \sim 12]$ that affect the 3rd Sbox's output can also be recovered using 2526 chosen-plaintext pairs. Then the remaining 44 key bits can also be recovered through two stages.

This attack can be extended to the attack on 8-round DES. We just need to guess 36 bits of the subkey in the 8-th round. The data complexity is still $N = 2526$. The best key recovery attack on 8-round DES presented in [4] needs 20000 chosen plaintexts. Our attack can reduce the data complexity by a factor of about 3.96.

7.3 Key Recovery Attacks on Round Reduced Speck32/64.

To attack Speck32/64 reduced to $11, 12, 13$ rounds, the following 8 neural aided statistical distinguishers are built as Table 20 shows.

Table 20. Eight neural aided statistical distinguishers for attacking Speck32/64. p_2 is selected based on d, Table 3/13. p_1, p_n are estimated with $M = 10^7$(Section 3.5).

SD	$\Delta P \rightarrow \Delta S$	p_0	ND	c_2	p_1	d	p_2	p_n
SD_8^t	$(0x211, 0xa04) \rightarrow (0x0040, 0)$	2^{-6}	ND_8^t	0.5	0.5184	1	0.4993	0.4914
SD_7^s	$(0x211, 0xa04) \rightarrow (0x0040, 0)$	2^{-6}	ND_7^s	0.55	0.3576	2	0.2940	0.2863
SD_7^t	$(0x211, 0xa04) \rightarrow (0x0040, 0)$	2^{-6}	ND_7^t	0.55	0.4183	2	0.2607	0.2162
SD_6^s	$(0x211, 0xa04) \rightarrow (0x0040, 0)$	2^{-6}	ND_6^s	0.55	0.5132	1	0.3402	0.2603
SD_6^t	$(0x211, 0xa04) \rightarrow (0x0040, 0)$	2^{-6}	ND_6^t	0.55	0.6785	1	0.3135	0.1161
SD_5^s	$(0x211, 0xa04) \rightarrow (0x0040, 0)$	2^{-6}	ND_5^s	0.55	0.7192	0	0.5498	0.1291
SD_5^t	$(0x211, 0xa04) \rightarrow (0x0040, 0)$	2^{-6}	ND_5^t	0.55	0.8889	0	0.5151	0.0384
SD_4^t	$(0x211, 0xa04) \rightarrow (0x0040, 0)$	2^{-6}	ND_4^t	0.55	0.995	0	0.5065	0.0069

ND_7^s, ND_6^s, ND_5^s are trained using Algorithm 6. The subscript set of selected ciphertext bits is $\Gamma = \{30 \sim 23, 14 \sim 7\}$. The number of encryption rounds covered by $\Delta P \rightarrow \Delta S$ is 2. It is extended to 3 rounds without loss of probability.

Key recovery attack on 11-round Speck32/64. To recover the last 4 subkeys, the attack is divided into seven stages. Table 21 shows the attack settings.

The probability that the right master key can survive is about $(1-0.005)^2 \times (1-0.001)^5 \approx 0.985$. The computation complexity contains seven parts. In stage 2, 3, 5,$(2^{21.28} \times (2^{14} + 2^{12}) + 2^{20.43} \times 2^{12}) \times \frac{1}{11} = 2^{32.29}$. In stage $1, 4, 6, 7$, the computation complexity is negligible. Thus the total complexity is $2^{32.29}$ and the data complexity is $2^{23.44}$.

In stage 1, 2, the number of surviving subkeys is presented based on **Attack 6**. The computation complexity in **Attack 6** is $(2^{23.44+8}+2^{21.28+12}) \times \frac{1}{11} \approx 2^{30.18}$, which is very low. But the time consumption is a little high for our hardware (Table 15). In **Attack 6**, the average time of decryption is about 187 seconds. The average time of neural distinguishers' verification is about 10160 seconds.

Table 21. Settings of the key recovery attack on 11-round Speck32/64. EUB: expected upper bound.

stage	SD	β_r	β_w	N	key space	related keys	surviving key space
1	SD_7^s	0.005	2^{-8}	$2^{22.44}$	2^8	$sk_{11}[7 \sim 0]$	2^4(Attack 6)
2	SD_7^t	0.005	2^{-16}	$2^{20.28}$	2^{4+8}	sk_{11}	2^6(Attack 6)
3	SD_6^s	0.001	2^{-14}	$2^{20.28}$	2^{6+8}	$sk_{11}, sk_{10}[7 \sim 0]$	2^4(EUB)
4	SD_6^t	0.001	2^{-16}	$2^{17.37}$	2^{4+8}	sk_{11}, sk_{10}	2^4(EUB)
5	SD_5^s	0.001	2^{-12}	$2^{19.43}$	2^{4+8}	$sk_{11}, sk_{10}, sk_9[7 \sim 0]$	2(EUB)
6	SD_5^t	0.001	2^{-16}	$2^{15.89}$	2^{1+8}	sk_{11}, sk_{10}, sk_9	2(EUB)
7	SD_4^t	0.001	2^{-17}	$2^{13.04}$	2^{1+16}	$sk_{11}, sk_{10}, sk_9, sk_8$	2(EUB)

Key recovery attack on 12-round Speck32/64. Table 22 shows the attack settings.

Table 22. Settings of the key recovery attack on 12-round Speck32/64.

stage	SD	β_r	β_w	N	key space	related keys	surviving key space
1	SD_8^t	0.005	2^{-16}	$2^{26.93}$	2^{16}	sk_{12}	2^4(EUB)
2	SD_7^s	0.005	2^{-12}	$2^{22.86}$	2^{4+8}	$sk_{12}, sk_{11}[7 \sim 0]$	2^7(EUB)
3	SD_7^t	0.005	2^{-16}	$2^{20.28}$	2^{7+8}	sk_{12}, sk_{11}	2^8(EUB)
4	SD_6^s	0.001	2^{-16}	$2^{20.41}$	2^{8+8}	$sk_{12}, sk_{11}, sk_{10}[7 \sim 0]$	2^4(EUB)
5	SD_6^t	0.001	2^{-16}	$2^{17.37}$	2^{4+8}	$sk_{12}, sk_{11}, sk_{10}$	2^4(EUB)
6	SD_5^t	0.001	2^{-20}	$2^{16.12}$	2^{4+16}	$sk_{12}, sk_{11}, sk_{10}, sk_9$	2(EUB)

The probability that the right master key can survive is about $(1-0.005)^3 \times (1-0.001)^3 \approx 0.982$. The total computation complexity is about $2^{40.35}$ and the data complexity is $2^{27.93}$.

Key recovery attack on 13-round Speck32/64. Table 23 shows the attack settings.

Table 23. Settings of the key recovery attack on 13-round Speck32/64.

stage	SD	β_r	β_w	N	key space	related keys	surviving key space
1	SD_8^t	0.005	2^{-32}	$2^{27.7}$	2^{32}	sk_{12}, sk_{11}	2^5(EUB)
2	SD_7^t	0.005	2^{-21}	$2^{20.58}$	2^{5+16}	$sk_{12}, sk_{11}, sk_{10}$	2^8(EUB)
3	SD_6^t	0.001	2^{-24}	$2^{17.78}$	2^{8+16}	$sk_{12}, sk_{11}, sk_{10}, sk_9$	2^5(EUB)

The probability that the right master key can survive is about $(1-0.005)^2 \times (1-0.001) \approx 0.989$. The total computation complexity is about 2^{58} and the data complexity is $2^{28.7}$.

8 A Special Case Caused by Continuous Non-informative Bits

The connections between the guessed subkey bits and selected ciphertext bit include direct connections and indirect ones. Take the XOR and modular addition as the example, Figure 2 shows the two kinds of connection.

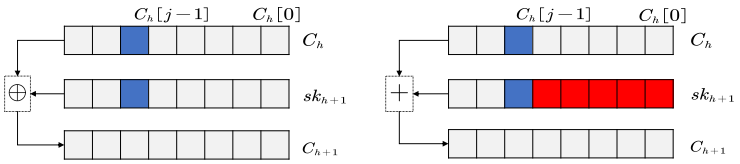

Fig. 2. The blue bit $C_h[j]$ is the informative bit in the h-th round ciphertext. $C_h[j-1 \sim 0]$ are not informative bits. The blue subkey bit $sk_{h+1}[j]$ is direct related to $C_h[j]$. The red subkey bits $sk_{h+1}[j-1 \sim 0]$ are indirect related to $C_h[j]$.

In the modular operation, key bits $sk_{h+1}[j-1 \sim 0]$ are also related to the informative bit $C_h[j]$. Such a connection is an indirect connection. Since $C_h[j-1 \sim 0]$ are not informative bits, $C_h[j]$ may not be influenced even some bit guesses of $sk_{h+1}[j-1 \sim 0]$ are wrong. When the number of continuous non-informative bits is too large, the data complexity for recovering key bits (eg. $sk_{h+1}[0]$) that have very little influence on $C_h[j]$ may be underestimated.

So far, we find this phenomenon occurs in the Speck128/128 reduced to 8 rounds. Let the difference constraint be $\Delta S = (0x0, 0x80)$ [2], we have trained a 8-round teacher distinguisher. $C_8[29 \sim 22]$ are informative bits but $C_8[21 \sim 8]$ are non-informative bits. A student distinguisher is built by setting $\Gamma = \{93 \sim 86, 29 \sim 22\}$. The data complexity is estimated by considering $sk_9[21 \sim 0]$. Then $sk_9[21 \sim 5]$ can be recovered successfully. But the 5 key bits $sk_9[4 \sim 0]$ that are related to $C_8[12 \sim 8]$ can't be recovered. Until now, this special case can only be solved by an x-round attack where $x > 1$. As we have proved in the estimation of $p_{2|d_1,\cdots,d_x}$, all the bits of sk_{h+1} have a significant influence on C_{h-1}.

9 Conclusions

In this paper, we have proposed a neural aided statistical attack for cryptanalysis. It has no extra requirements about the attacked cipher except for a high probabilistic differential transition. Besides, it provides a theoretical framework for estimating the needed attack complexities and success rate. In order to reduce the key space to be searched in the key recovery attack, a bit sensitivity test is proposed to help build neural distinguishers flexibly. Applications to round reduced Speck and DES have proved the correctness and superiorities of our attack model.

Although our neural aided statistical attack is already as generic as the differential cryptanalysis, there is still more work to do. The first is filtering random ciphertext pairs. It is helpful for reducing attack complexities. The second is understanding the knowledge learned by the neural distinguisher. It can help us get rid of the dependence on neural distinguishers and improve the theoretical framework. The practical time consumption can also be reduced significantly. The third is exploring the properties of the neural distinguisher. If we know how to build neural distinguishers against more rounds, the statistical attack model can be greatly improved. Besides, the concept of the informative bit is full of potential. We believe neural aided cryptanalysis has great potential for better assessing the security of ciphers.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16). pp. 265–283 (2016)
2. Abed, F., List, E., Lucks, S., Wenzel, J.: Differential cryptanalysis of round-reduced simon and speck. In: International Workshop on Fast Software Encryption. pp. 525–545. Springer (2014)
3. Aguilar, G., Ling, Y., Zhang, Y., Yao, B., Fan, X., Guo, C.: Knowledge distillation from internal representations. In: AAAI. pp. 7350–7357 (2020)
4. Bar-On, A., Dunkelman, O., Keller, N., Weizman, A.: Dlct: a new tool for differential-linear cryptanalysis. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 313–342. Springer (2019)
5. Batina, L., Bhasin, S., Jap, D., Picek, S.: Poster: Recovering the input of neural networks via single shot side-channel attacks. computer and communications security pp. 2657–2659 (2019)
6. Beaulieu, R., Shors, D., Smith, J., Treatmanclark, S., Weeks, B., Wingers, L.: The simon and speck lightweight block ciphers. design automation conference p. 175 (2015)
7. Bengio, Y., Ducharme, R., Vincent, P.: A neural probabilistic language model. Neural Information Processing Systems (NeurIPS) pp. 932–938 (2000)
8. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. Journal of CRYPTOLOGY **4**(1), 3–72 (1991)
9. Blondeau, C., Bay, A., Vaudenay, S.: Protecting against multidimensional linear and truncated differential cryptanalysis by decorrelation. In: International Workshop on Fast Software Encryption. pp. 73–91. Springer (2015)
10. Bogdanov, A., Wang, M.: Zero correlation linear cryptanalysis with reduced data complexity. fast software encryption pp. 29–48 (2012)
11. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 45–68. Springer (2017)
12. Chen, Y., Yu, L., Ota, K., Dong, M.: Robust activity recognition for aging society. IEEE Journal of Biomedical and Health Informatics **22**(6), 1754–1764 (2018)
13. Dinur, I.: Improved differential cryptanalysis of round-reduced speck. international conference on selected areas in cryptography pp. 147–164 (2014)
14. Dunkelman, O., Keller, N.: An improved impossible differential attack on misty1. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 441–454. Springer (2008)
15. Dunkelman, O., Keller, N., Ronen, E., Shamir, A.: The retracing boomerang attack. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 280–309. Springer (2020)
16. Eli, B., Rafi, C.: Near-collisions of sha-0. Annual International Cryptology Conference pp. 290–305 (2004)
17. Espitau, T., Fouque, P.A., Karpman, P.: Higher-order differential meet-in-the-middle preimage attacks on sha-1 and blake. In: Annual Cryptology Conference. pp. 683–701. Springer (2015)
18. Feller, W.: An introduction to probability theory and its applications. vol. ii. Population **23**(2), 375 (1968)

19. Gisselquist, R., Hoel, P.G., Port, S.C., Stone, C.J.: Introduction to probability theory. American Mathematical Monthly **81**(9), 1041 (1974)
20. Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. international cryptology conference pp. 150–179 (2019)
21. Greydanus, S.: Learning the enigma with recurrent neural networks. arXiv: Neural and Evolutionary Computing (2017)
22. Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M.A., Dally, W.J.: Eie: efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News **44**(3), 243–254 (2016)
23. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
24. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise: Unleashing the power of convolutional neural networks for profiled side-channel analysis. cryptographic hardware and embedded systems **2019**(3), 148–179 (2019)
25. Knudsen, L.R.: Truncated and higher order differentials. In: International Workshop on Fast Software Encryption. pp. 196–211. Springer (1994)
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Neural Information Processing Systems (NeurIPS) pp. 1097–1105 (2012)
27. MacKay, D.J.: Introduction to gaussian processes. NATO ASI Series F Computer and Systems Sciences **168**, 133–166 (1998)
28. Mirzadeh, S.I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., Ghasemzadeh, H.: Improved knowledge distillation via teacher assistant. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 5191–5198 (2020)
29. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems. pp. 8026–8037 (2019)
30. Ronald, L.R.: Cryptography and machine learning. International Conference on the Theory and Application of Cryptology pp. 427–439 (1991)
31. Schramm, K., Wollinger, T., Paar, C.: A new class of collision attacks and its application to des. In: International Workshop on Fast Software Encryption. pp. 206–222. Springer (2003)
32. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE **105**(12), 2295–2329 (2017)
33. Wagner, D.: The boomerang attack. In: International Workshop on Fast Software Encryption. pp. 156–170. Springer (1999)
34. Yuan, L., Tay, F.E., Li, G., Wang, T., Feng, J.: Revisiting knowledge distillation via label smoothing regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3903–3911 (2020)

A Analysis of the Data Complexity

For the conventional differential attack, the data complexity is only related to the differential. For the **NASA**, the data complexity is also related to the neural distinguisher. In this appendix, we present the analysis of each part's impact on the data complexity.

A.1 The Differential's Impact on the Data Complexity

According to formula (17), the data complexity N is affected by the probability p_0 of the differential transition as

$$\begin{aligned}\sqrt{N} &= \frac{z_{1-\beta_r} \times \sqrt{p_0 a_1 + (1-p_0)a_3} + z_{1-\beta_w} \times \sqrt{p_0 a_2 + (1-p_0)a_3}}{(p_1 - p_2) \times p_0} \\ &\propto \frac{z_{1-\beta_r} \times \sqrt{a_3 + (a_1 - a_3)p_0} + z_{1-\beta_w} \times \sqrt{a_3 + (a_2 - a_3)p_0}}{p_0} \\ &\propto \frac{\sqrt{a_3 + (a_1 - a_3)p_0} + a_4 \times \sqrt{a_3 + (a_2 - a_3)p_0}}{p_0}\end{aligned}$$

where $a_4 = \frac{z_{1-\beta_w}}{z_{1-\beta_r}}$. We further know

$$N \propto p_0^{-2}\left[\boldsymbol{a_3 + a_4^2 a_3} + (a_1 - a_3 + a_4^2 a_2 - a_4^2 a_3)p_0 + a_5\right]$$

where $a_5 = 2 \times a_4 \times \sqrt{a_3 + (a_1 - a_3)p_0} \times \sqrt{a_3 + (a_2 - a_3)p_0}$. Thus the impact of the probability p_0 of the differential transition is $\mathcal{O}(p_0^{-2})$.

A.2 The Neural Distinguisher's Impact on the Data Complexity

Three probabilities p_1, p_2, p_n are related to the neural distinguisher. Since p_n is related to negative samples and p_1, p_2 are related to positive samples, we can discuss p_n separately.

The impact of p_n. In formula (17), only a_3 is related to p_n.

$$\begin{aligned}\sqrt{N} &= \frac{z_{1-\beta_r} \times \sqrt{p_0 a_1 + (1-p_0)a_3} + z_{1-\beta_w} \times \sqrt{p_0 a_2 + (1-p_0)a_3}}{(p_1 - p_2) \times p_0} \\ &\propto \sqrt{p_0 a_1 + (1-p_0)a_3} + a_4 \times \sqrt{p_0 a_2 + (1-p_0)a_3} \\ \Rightarrow N &\propto p_0 \times a_1 + a_4^2 \times p_0 \times a_2 + (1-p_0)(1+a_4^2)a_3 + a_5\end{aligned}$$

Next, we will focus on attack scenarios where the p_0 is very small. This can make the discussion easier and more concise. This simplification is also reasonable. Because when we attack a cipher for more rounds, the probability of the differential transition is generally small.

When $p_0 \to 0$, $a_5 \to 2 \times a_4 \times a_3$. Thus the impact of a_3 on the data complexity is $\mathcal{O}(a_3)$. Since $p_n < 1$ always holds and $a_3 = p_n - p_n^2$, the impact of p_n on the data complexity is also $\mathcal{O}(p_n)$.

The impact of p_1, p_2. In formula (17), a_1, a_2 are related to p_1, p_2 respectively. The impacts of a_1, a_2 are adjusted by the differential transition's probability p_0. Due to this property, we can also focus on attack scenarios where $p_0 \to 0$.

$$\begin{aligned}\sqrt{N} &= \frac{z_{1-\beta_r} \times \sqrt{p_0 a_1 + (1-p_0)a_3} + z_{1-\beta_w} \times \sqrt{p_0 a_2 + (1-p_0)a_3}}{(p_1 - p_2) \times p_0} \\ &\approx \frac{z_{1-\beta_r} \times \sqrt{(1-p_0)a_3} + z_{1-\beta_w} \times \sqrt{(1-p_0)a_3}}{(p_1 - p_2) \times p_0} \propto (p_1 - p_2)^{-1}\end{aligned}$$

Thus the impact of p_1, p_2 on the data complexity is $\mathcal{O}((p_1 - p_2)^{-2})$.