Neural Aided Statistical Attack for Cryptanalysis

Yi Chen, Yantian Shen, Hongbo Yu

***Abstract*—In CRYPTO'19, Gohr proposed the first key recovery attack based on deep learning, which opens the direction of neural aided cryptanalysis. Up to date, neural aided cryptanalysis still faces two problems: (1) its attack complexity estimations rely purely on practical experiments. There is no theoretical framework for estimating theoretical complexity. (2) it can not work when there are no enough neutral bits that exist in the prepended differential. To our best knowledge, we are the first to solve these two problems. In this paper, our work contains three aspects: (1) we propose a Neural Aided Statistical Attack (NASA) that is a generic neural aided cryptanalysis technique. NASA allows us to estimate the theoretical complexities without performing practical experiments. Moreover, NASA does not rely on any special properties including the neutral bit. (2) we propose three methods for reducing the attack complexities of NASA. One of the methods is based on a newly proposed concept named Informative Bit that reveals an important phenomenon related to neural distinguishers. (3) We prove the superiorities of NASA based on applications to round reduced DES and Speck32/64. Our work arguably raises a new direction for neural aided cryptanalysis.**

***Index Terms*—Neural Aided Statistical Attack, Deep Learning, Cryptanalysis, Informative Bit, DES, Speck**

I. Introduction

Deep learning has received much expectation in the cryptography community since the last century. Rivest in [1] 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 [2].

Although deep learning has shown its superiorities in various fields such as computer vision [3], natural language processing [4], and smart medical [5], its application in the field of conventional cryptanalysis has been stagnant. A few valuable applications are only concentrated in the side-channel analysis [6]–[8].

In CRYPTO'19, Gohr proposed a deep learning-based distinguisher [9] that is also called neural distinguisher ($\mathcal{ND}$). By placing a differential before $\mathcal{ND}$, Gohr developed a key recovery attack on 11-round Speck32/64. Gohr's attack shows considerable advantages in terms of attack complexities over the traditional differential attack, which opens the direction of neural aided cryptanalysis.

However, Gohr's attack faces two problems. First, it can not be applied to the theoretical security analysis of a cipher. More exactly, we do not know what the required data complexity is to attack a specific cipher. We can estimate the attack complexities and success rate empirically only if the attack is finished within an acceptable running time. Second, when we add a differential before $\mathcal{ND}$ for attacking a cipher covering more rounds, it requires that enough neutral bits [10] must exist in the prepended differential. Up to date, there are no solutions for these two problems.

In this paper, we have explored neural aided cryptanalysis and made contributions as follows.

- We figure out the reason why Gohr's attack does not allow the adversary to estimate the theoretical complexities. In [9], each key guess corresponds to a key rank score that is directly determined by the output of $\mathcal{ND}$. A key guess is returned as a candidate when its key rank score exceeds a threshold. Since the output of $\mathcal{ND}$ is unpredictable, the threshold is set without any theoretical basis. Then it is impossible to estimate the attack success rate and theoretical complexities unless practical experiments are executed.
- We propose a Neural Aided Statistical Attack (NASA), which supports theoretical complexity estimation and does not rely on neutral bits. The theoretical basis of NASA is as follows. By analyzing the key recovery process, we find that there are only four different scenarios. A statistic designed in this article obeys a normal distribution in each scenario. Based on this statistic, the key recovery is transformed into the distinguishing between two normal distributions, which tells the required data complexity of NASA.
- We propose three methods to reduce the attack complexities of NASA. The first one is reducing the key space by building $\mathcal{ND}$ on partial ciphertext bits. The initial $\mathcal{ND}$ proposed by Gohr takes the complete ciphertext pair as input, which forces the adversary to guess all the key bits simultaneously. Adopting the new $\mathcal{ND}$, the adversary guesses partial key bits at a time. The second one is a highly selective Bayesian key search algorithm. It allows the adversary to search key guesses that are most likely to be the right key instead of traversing all the key guesses. The third one is reducing the data complexity by exploiting neutral bits. When there are available neutral bits, the data complexity of NASA can also be reduced.
- We perform experiments on DES and Speck32/64 to fully analyze NASA. First, experiments on DES prove that NASA can attack a cipher covering more rounds than Gohr's attack when there are no enough neutral bits. Second, the experiment on Speck32/64 shows that NASA can achieve comparable performance with Gohr's attack when the proposed three optimization methods are available.

Organization Section III presents the neural aided statistical distinguisher. Section IV summarizes NASA and provides the correctness verification. The three optimization methods are introduced in Section V, VI, VII. Applications to DES and Speck32/64 are presented in Section VIII, IX.

II. Related Work

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).

A. Neutral Bit

Consider a differential $\Delta P \rightarrow \Delta S$. Let E denote the encryption function covering the differential. For any plaintext pair (P_0, P_1) conforming to the differential, if the following condition always holds

$$E(P_0 \oplus e^j) \oplus E(P_1 \oplus e^j) = \Delta S, \quad e^j = 1 \ll j,$$

the j-th bit is called a neutral bit [10].

Based on k neutral bits $\{j_1, \cdots, j_k\}$ and a plaintext pair $(P_0, P_1)|P_0 \oplus P_1 = \Delta P$, we can generate a plaintext structure consisting of 2^k plaintext pairs. Once (P_0, P_1) satisfies the differential, the remaining $2^k - 1$ plaintext pairs also conform to the differential.

B. Neural Distinguisher

The target of $\mathcal{ND}$ [9] is to distinguish two classes of ciphertext pairs

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

where $Y = 1$ or $Y = 0$ is the 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 an $\mathcal{ND}$ if the distinguishing accuracy over a testing database is higher than 0.5.

Let $\mathcal{ND}_h$ denotes a neural distinguisher against the cipher reduced to h rounds. Given a sample (C_0, C_1), $\mathcal{ND}$ will output a score Z which is used as the posterior probability

$$Pr(Y = 1\,|(C_0, C_1)) = Z = \mathcal{ND}(C_0, C_1), \quad 0 \leqslant Z \leqslant 1 \tag{2}$$

When $Z > 0.5$, the predicted label of (C_0, C_1) is 1 [9].

In [9], a Residual Network (ResNet) is adopted by Gohr. The training pipeline can refer to [9].

C. Gohr's Key Recovery Attack

Algorithm 1 summarizes the core idea of the basic version (unaccelerated version) of Gohr's key recovery attack [9].

Decrypting 2^k ciphertext pairs drawn from the target or uniform distribution with a key guess kg, the adversary uses 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.

Algorithm 1 Basic version of Gohr's key recovery attack

Require: k neutral bits that exist in $\Delta P \rightarrow \Delta S$;
An $\mathcal{ND}$ built over 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 ⊕ P_1^1 = ΔP;
    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 ∈ {1, ··· , 2^k};
    for each key guess kg do
        Partially decrypt 2^k ciphertext pairs with kg;
        Feed decrypted ciphertext pairs to ND and collect the outputs;
        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 rank score is likely to exceed c_1 only when the plaintext structure passes the prepended differential and kg is the right key. If the plaintext structure does not pass the differential or the key guess is wrong, the rank score should be very low. Thus, the right key can be identified by comparing the rank score with a threshold. When the performance of $\mathcal{ND}$ is weak, 2^k needs to be large. Then more neutral bits are required.

The application of Gohr's attack is limited by two aspects:

- The adversary can not estimate the required attack complexities and success rate theoretically. Since the output Z of $\mathcal{ND}$ is unpredictable, the threshold c_1 is set without any clear theoretical basis. Then it is unknown how many plaintext pairs a plaintext structure should contain. As a result, the success rate under an attack setting is also unknown.
- If the number of neutral bits is not large enough, Gohr's attack does not work.

D. Distinguishing between Two Normal Distributions

Here, we present the details of the distinguishing between two normal distributions.

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:

$$\begin{aligned} \beta_r &= Pr\left\{\mathcal{N}(\mu_w, \sigma_w) \,|s \in \mathcal{N}(\mu_r, \sigma_r)\right\}, \\ \beta_w &= Pr\left\{\mathcal{N}(\mu_r, \sigma_r) \,|s \in \mathcal{N}(\mu_w, \sigma_w)\right\}. \end{aligned} \tag{4}$$

When a sample s is sampled from $\mathcal{N}(\mu_r, \sigma_r)$, the probability that the decision is $s \in \mathcal{N}(\mu_w, \sigma_w)$ is β_r.

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 [11], [12].

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

$$\frac{z_{1-\beta_r} \times \sigma_r + z_{1-\beta_w} \times \sigma_w}{|\mu_r - \mu_w|} = 1 \tag{5}$$

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

Proposition 1 is the theoretical basis of our work in this article.

III. Neural Aided Statistical Distinguisher

A. A Chosen Plaintext Statistical Distinguisher

Consider a cipher E with a block size of m, and a differential $\Delta P \xrightarrow{p_0} \Delta S | \Delta P, \Delta S \in \mathbb{F}_2^m$ where p_0 is the transition probability. Build an $\mathcal{ND}$ over ΔS. Randomly generate N plaintext pairs with a difference ΔP, and collect corresponding ciphertext pairs. The adversary needs to distinguish between this cipher and a random permutation.

The concrete process is as follows. For each ciphertext pair $\left(C_0^i, C_1^i\right), i \in \{1, \cdots, N\}$, the adversary feeds it into $\mathcal{ND}$ and obtains its output Z_i. Setting a threshold value c_2, the adversary calculates the statistic T

$$T = \sum_{i=1}^{N} \phi(Z_i), \quad \phi(Z_i) = \begin{cases} 1, if\ Z_i > c_2 \\ 0, if\ Z_i \leqslant c_2 \end{cases}. \tag{6}$$

When $p_0 > 2^{-m}$ holds, 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.

B. 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 c_2, it's equivalent to creating a stable classification hyperplane in this space using an $\mathcal{ND}$. Thus the classification over a random ciphertext pair is modeled as a Bernoulli experiment. It provides us with a theoretical analysis framework.

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:

- **Decrypting a positive sample with the right key:** the ciphertext pair satisfies the differential and the key guess is right.

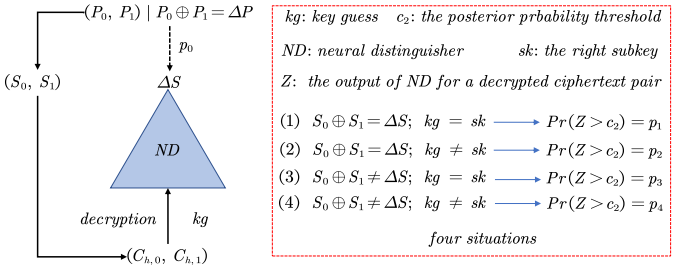

Fig. 1. Four situations of decrypting a ciphertext pair with a key guess.

- **Decrypting a positive sample with wrong keys:** the ciphertext pair satisfies the differential but the key guess is wrong.
- **Decrypting a negative sample with the right key:** the ciphertext pair does not satisfy the differential but the key guess is right.
- **Decrypting a negative sample with wrong keys:** the ciphertext pair does not satisfy the differential and the key guess is wrong.

Given an $\mathcal{ND}$, 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 6) in these four situations are

$$\begin{aligned} T_1 &\sim \mathcal{N}(\mu_1, \sigma_1), \mu_1 = N_1 \times p_1, \sigma_1 = \sqrt{N_1 \times p_1(1-p_1)} \\ T_2 &\sim \mathcal{N}(\mu_2, \sigma_2), \mu_2 = N_2 \times p_2, \sigma_2 = \sqrt{N_2 \times p_2(1-p_2)} \\ T_3 &\sim \mathcal{N}(\mu_3, \sigma_3), \mu_3 = N_3 \times p_3, \sigma_3 = \sqrt{N_3 \times p_3(1-p_3)} \\ T_4 &\sim \mathcal{N}(\mu_4, \sigma_4), \mu_4 = N_4 \times p_4, \sigma_4 = \sqrt{N_4 \times p_4(1-p_4)} \end{aligned} \tag{7}$$

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 $\sigma_i, i \in \{1, 2, 3, 4\}$. An empirical condition is

$$N_i \times p_i > 5, \quad N_i \times (1 - p_i) > 5, \quad i \in \{1, 2, 3, 4\} \tag{8}$$

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

$$N_1 = N_2 = N \times p_0, \quad N_3 = N_4 = N \times (1 - p_0). \tag{9}$$

Besides, the distributions of the statistic (formula 6) under the right key and wrong keys are both a mixture of two normal distributions.

1) 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 6) under the right key guess is:

$$T_r = T_1 + T_3 \sim \mathcal{N}(\mu_r, \sigma_r) \tag{10}$$

$$\mu_r = N \times (p_0 p_1 + (1 - p_0) p_3) \tag{11}$$

$$\sigma_r = \sqrt{N \times p_0 \times p_1 (1 - p_1) + N (1 - p_0) p_3 (1 - p_3)} \tag{12}$$

2) 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 6) under wrong key guesses is:

$$T_w = T_2 + T_4 \sim \mathcal{N}(\mu_w, \sigma_w) \tag{13}$$

$$\mu_w = N \times (p_0 p_2 + (1 - p_0) p_4) \tag{14}$$

$$\sigma_w = \sqrt{N \times p_0 \times p_2 (1 - p_2) + N (1 - p_0) p_4 (1 - p_4)} \tag{15}$$

Negative samples in the high-dimensional space approximately obey uniform distribution, thus $p_3 = p_4$ holds theoretically and experiments also verify it. Since the accuracy of NDs 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.

Since the distributions of T_r, T_w are different, the key recovery attack can be performed based on *Proposition 1*.

C. 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 μ_r, σ_r, μ_w, σ_w refer to formula 11, 12, 14, 15 respectively. In a key recovery setting, $1 - \beta_r$ is the minimum probability that the right key survives, β_w is the maximum probability that wrong keys survive.

Since we can not know the real classification hyperplane learned by the ND, p_1, p_2, p_3, and p_4 are 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 $\mu_r - \mu_w$, which is wrong. Thus we neglect the minor difference and replace p_3, p_4 with p_n.

Then the condition (formula 16) is simplified

$$\sqrt{N} = \frac{z_{1-\beta_r} \times X_1 + z_{1-\beta_w} \times X_2}{(p_1 - p_2) \times p_0}, \tag{17}$$

where

$$X_1 = \sqrt{p_0 a_1 + (1 - p_0) a_3}, \tag{18}$$

$$X_2 = \sqrt{p_0 a_2 + (1 - p_0) a_3}, \tag{19}$$

$$a_1 = p_1(1-p_1), \quad a_2 = p_2(1-p_2), \quad a_3 = p_n(1-p_n). \tag{20}$$

The decision threshold t is:

$$t = \mu_r - z_{1-\beta_r}\sigma_r = \mu_w + z_{1-\beta_w}\sigma_w. \tag{21}$$

The data complexity N is directly calculated when β_r and β_w are set. The impacts of p_0, p_1, p_2, p_n on N are about $\mathcal{O}(p_0^{-2})$, $\mathcal{O}((p_1 - p_2)^{-2})$, $\mathcal{O}(p_n)$ respectively. The proof is presented in Appendix.

D. Estimation of p_1, p_n

Consider an $\mathcal{ND}$ against a cipher, the values of p_1, p_n are estimated as:

1) Randomly generate M positive / negative samples and decrypt them for 1 round with the right / wrong subkeys.
2) Feed partially decrypted samples into $\mathcal{ND}$.
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 make the statistical expectation accurate enough.

E. Further Analysis and Estimation of p_2

When we decrypt a positive sample with a wrong key guess (Fig. 1(2)), the final value of p_2 is related to the Hamming distance between the wrong key guess and the right key. Such a phenomenon is based on *Property 1* and *Property 2*.

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

$$C^1_{h-1} = DecOneRound(C_h, kg_1)$$
$$C^2_{h-1} = DecOneRound(C_h, kg_2).$$

If kg_1 and kg_2 are only different at a few bits (e.g. just 1 bit or 2 bits), the Hamming distance between C^1_{h-1} and C^2_{h-1} will be very small in high probability.

Property 2: Consider a neural network $F(\cdot)$ solving a binary classification problem. If two input samples I_1, I_2 are very close to each other in the input space, two outputs $F(I_1), F(I_2)$ of the neural network may satisfy $F(I_1) \approx F(I_2)$ 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.

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

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

where kg_{h+j} is the key guess of the $(h+j)$-th round. $(C_{h,0}, C_{h,1})$ is fed into an $\mathcal{ND}_h$ 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 $kg_{h+j}, j \in \{2, \cdots, x\}$ are not all right, then $(C_{h+1,0}, C_{h+1,1})$ is not 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 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. Gohr provided $\mathcal{ND}_5$, $\mathcal{ND}_6$, $\mathcal{ND}_7$, $\mathcal{ND}_8$ against Speck32/64 [9], which are built over a plaintext difference $(0x0040, 0)$. We have performed tests on these four distinguishers. Let $M = 10^7$, Table I and Table II show the estimation results of $p_{2|d_1}$ and $p_{2|d_1,d_2}$ respectively.

TABLE I
THE ESTIMATION OF $p_{2|d_1}$ OF 4 NEURAL DISTINGUISHERS AGAINST ROUND REDUCED SPECK32/64. FOR $\mathcal{ND}_5, \mathcal{ND}_6, \mathcal{ND}_7$, $c_2 = 0.55$. FOR $\mathcal{ND}_8$, $c_2 = 0.5$. $p_{2|d_1=0} = p_1$.

$\mathcal{ND}_5$	d_1	0	1	2	3	4	5	6	7	$8 \sim 16$
	$p_{2\|d_1}$	0.8889	0.5151	0.3213	0.2168	0.1556	0.1189	0.0956	0.08	$\leqslant 0.0694$
$\mathcal{ND}_6$	d_1	0	1	2	3	4	5	6	7	$8 \sim 16$
	$p_{2\|d_1}$	0.6785	0.4429	0.3135	0.2384	0.1947	0.1684	0.1518	0.1408	$\leqslant 0.1334$
$\mathcal{ND}_7$	d_1	0	1	2	3	4	5	6	7	$8 \sim 16$
	$p_{2\|d_1}$	0.4183	0.3369	0.2884	0.2607	0.2442	0.234	0.2276	0.2236	$\leqslant 0.2211$
$\mathcal{ND}_8$	d_1	0	1	2	3	4	5	6	7	$8 \sim 16$
	$p_{2\|d_1}$	0.5184	0.5056	0.4993	0.4957	0.4939	0.4927	0.4925	0.4918	$\leqslant 0.4917$

TABLE II
THE ESTIMATION OF $p_{2|d_1,d_2}$ OF $\mathcal{ND}_7$ AGAINST SPECK32/64. $c_2 = 0.55$. THE COLUMNS CORRESPOND TO d_2. THE ROWS CORRESPOND TO d_1. 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
$9 \sim 16$	$\leqslant Y$																

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

Require: a cipher with a subkey size of m;
M random plaintext pairs, $(P_0^i, P_1^i), P_0^i \oplus P_1^i = \Delta P, i \in \{1, \cdots, M\}$;
an $\mathcal{ND}_h$ built over ΔP.
M random master keys, $MK_i, i \in \{1, \cdots, M\}$;
The threshold c_2.
Ensure: $p_{2|d_1,\cdots,d_x}$.
1: Encrypt each plaintext pair (P_0^i, P_1^i) with a master key MK_i for $h + x$ rounds;
2: Save resulting ciphertext pair (C_0^i, C_1^i);
3: Save the $(h + j)$-th subkey $sk_{h+j}^i, j \in \{1, \cdots, x\}$;
4: **for** $d_1 = 0$ to m, $\cdots$, $d_x = 0$ to m **do**
5: **for** $i = 1$ to M **do**
6: Randomly draw x key guesses $kg_j^i, j \in \{1, \cdots, x\}$ where the Hamming distance between kg_j^i and sk_{h+j}^i is d_j;
7: Decrypt (C_0^i, C_1^i) with $kg_j^i, j \in \{1, \cdots, x\}$ for x rounds;
8: Feed the decrypted ciphertext pair into $\mathcal{ND}_h$ and save the output as $Z_{i|d_1,\cdots,d_x}$;
9: **end for**
10: Count the number of $Z_{i|d_1,\cdots,d_x} > c_2$, and denote it as $T_{d_1,\cdots,d_x}$;
11: Save $p_{2|d_1,\cdots,d_x} = \frac{T_{d_1,\cdots,d_x}}{M}$.
12: **end for**

The test results 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.

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 larger than a threshold d, the 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 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 required for filtering wrong keys.

Take $\mathcal{ND}_7$ 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 $p_2 = p_{2|0,1} = p_{2|3,0} = 0.26$.

IV. NEURAL AIDED STATISTICAL ATTACK

A. Key Recovery Attack Model

This neural aided statistical distinguisher is used to determine whether a key guess may be the right key. This is done by the *Statistical Test* as shown in Algorithm 3. Algorithm 4 summarizes the Neural Aided StatisticalAttack (NASA) based on the statistical distinguisher.

Algorithm 3 Statistical test for a key guess

Require: An $\mathcal{ND}$; 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 $\mathcal{ND}$, and collect the outputs $Z_i, i \in [1, N]$;
3: Calculate the statistic T in formula 6;
4: **if** $T \geqslant t$ **then**
5: Return kg as a key candidate.
6: **end if**

Algorithm 4 Neural Aided Statistical Attack

Require: The attacked cipher;
The differential with a probability of p_0, $\Delta P \xrightarrow{p_0} \Delta S$;
Two maximum error probabilities, β_r, β_w;
A posterior probability threshold, c_2.

Ensure: All possible key candidates.

1: Train an $\mathcal{ND}$ over ΔS;
2: Estimate p_1, p_n, p_2 using $\mathcal{ND}$ (Section III-D, Algorithm 2);
3: Calculate the data complexity N and the decision threshold t (Section III-C);
4: Randomly generate N plaintext pairs $(P_0^i, P_1^i), P_0^i \oplus P_1^i = \Delta P, i \in \{1, \cdots, N\}$;
5: Collect corresponding N ciphertext pairs, $(C_0^i, C_1^i), i \in \{1, \cdots, N\}$;
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.

B. Verification of the Key Recovery Attack Model

In order to verify NASA, four practical 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.

NASA should work as long as the adopted $\mathcal{ND}$ has a distinguishing accuracy higher than 0.5. Besides, the data complexity should be correctly estimated once $\Delta P \xrightarrow{p_0} \Delta S$, $\mathcal{ND}$, d, β_r, and β_w are provided. Thus, different settings about these factors are considered.

Three distinguishers $\mathcal{ND}_5$, $\mathcal{ND}_7$, $\mathcal{ND}_8$ provided by Gohr [9] are adopted. Table III shows two different differentials of Speck32/64 adopted in the verification. Since no key addition happens in Speck before the first nonlinear operation, these two differentials can be extended to a 2/3-round differential respectively.

The verification plan consists of three steps:

1) Set the value of β_r, β_w. Calculate the data complexity N.
2) Perform NASA 100 times with N samples.

TABLE III
TWO OPTIONS OF THE PREPENDED DIFFERENTIAL OF SPECK32/64. nr IS THE NUMBER OF ENCRYPTION ROUNDS COVERED BY THE DIFFERENTIAL.

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

3) Check the following observation indexes:
   a) The ratio that the right key ($d_1 = 0$) 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 IV summarizes the settings related to four attacks. Table I shows the estimations of $p_{2|d_1}$ related to $\mathcal{ND}_5$, $\mathcal{ND}_7$, $\mathcal{ND}_8$. The value of p_2 should be $p_{2|d_1=3}$ in four attacks.

1) 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 should not exceed $137 + (2^{16} - 137) \times 0.003 = 333.197$.

After performing this attack 100 times, we find:

- The right key ($d_1 = 0$) has survived in all the 100 experiments.
- The average number of surviving keys is 124.21 that is smaller than 333.197.
- 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 is divided into 17 subspaces. We further calculate the average ratio that keys in each subspace survive the attack. Table V shows the average surviving ratios of 17 key subspaces.

2) 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 should not exceed $137 + (2^{16} - 137) \times 2^{-16} \approx 137.998$.

After performing this attack 100 times, we find:

- The right key ($d_1 = 0$) has survived in 99 experiments.
- The average number of surviving keys is 63.54 that is smaller than 137.998.
- The number of surviving keys is smaller than 137.998 in 98 experiments.

Table VI shows the average surviving ratios of 17 subspaces.

3) Attack 3: recover sk_{10} of 10-round Speck32/64: $\mathcal{ND}_8$ is a very weak distinguisher. Its distinguishing 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 should not exceed $137 + (2^{16} - 137) \times 2^{-16} \approx 137.998$.

TABLE IV
SETTINGS OF THE FOUR ATTACKS AGAINST ROUND REDUCED SPECK32/64. DID IS THE DIFFERENTIAL TRANSITION'S ID IN TABLE 4.

Attack ID	Attack rounds	$\mathcal{ND}$	DID	p_0	c_2	p_1	d	p_2	p_n	β_r	β_w
1	10	$\mathcal{ND}_7$	1	2^{-2}	0.55	0.4183	2	0.2607	0.2162	0.005	0.003
2	10	$\mathcal{ND}_7$	1	2^{-2}	0.55	0.4183	2	0.2607	0.2162	0.005	2^{-16}
3	10	$\mathcal{ND}_8$	-	1	0.5	0.5184	2	0.4957	0.4914	0.001	2^{-16}
4	9	$\mathcal{ND}_5$	2	2^{-6}	0.55	0.8889	2	0.2168	0.0384	0.005	2^{-16}

TABLE V
AVERAGE SURVIVING RATIOS (SR) OF KEY GUESSES IN ATTACK 1.

d_1	0	1	2	3	4	$5 \sim 16$
SR	1	0.4706	0.1786	0.0591	0.017	$\leqslant 0.0041$

TABLE VI
AVERAGE SURVIVING RATIOS (SR) OF KEY GUESSES IN ATTACK 2.

d_1	0	1	2	3	4	$5 \sim 16$
SR	0.99	0.3788	0.1245	0.0343	0.008	$\leqslant 0.0014$

After performing this attack 100 times, we find:

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

Table VII shows the average surviving ratios of 17 subspaces.

TABLE VII
AVERAGE SURVIVING RATIOS (SR) OF KEY GUESSES IN ATTACK 3.

d_1	0	1	2	3	4	$5 \sim 16$
SR	1	0.4444	0.1459	0.0406	0.0095	$\leqslant 0.0019$

4) 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 should not exceed $137 + (2^{16} - 137) \times 2^{-16} \approx 137.998$.

After performing this attack 100 times, we find:

- The right key ($d_1 = 0$) has survived in all the 100 experiments.
- The average number of surviving keys is 18.41 that is smaller than 137.998.
- The number of surviving keys is smaller than 137.998 in 100 experiments.

Table VIII shows the average surviving ratios of 17 subspaces.

It's clear that these four attacks have achieved the most important two targets of NASA. Although the ratio that keys of each subset survive 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 NASA is also well verified.

TABLE VIII
AVERAGE SURVIVING RATIOS (SR) OF KEY GUESSES IN ATTACK 4.

d_1	0	1	2	3	4	$5 \sim 16$
SR	1	0.2781	0.0562	0.0082	0.0007	$\leqslant 0.0001$

V. REDUCE THE KEY SPACE

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

A. An Intuitive Method for Reducing the Key Space

An intuitive method for reducing the key space is building $\mathcal{ND}$ on partial ciphertext bits

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

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

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

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

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

where $C_i[0]$ is the least significant bit of the ciphertext C_i, Γ is the subscript set of selected ciphertext bits.

Such a method significantly reduces the key space to be searched. But **which ciphertext bits should we select for building $\mathcal{ND}$? 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 between the cipher and a pseudo-random permutation.

Definition 2: For a cipher reduced to h rounds, the neural distinguisher trained on the complete ciphertexts (C_0, C_1) is denoted as the **teacher distinguisher** $\mathcal{ND}_h^t$, the neural distinguisher trained on partial ciphertext bits $(\varphi(C_0,\Gamma), \varphi(C_1,\Gamma))$ is denoted as the **student distinguisher** $\mathcal{ND}_h^s$. The teacher distinguisher is viewed as a special student distinguisher.

B. 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 an $\mathcal{ND}^t$, if the distinguishing accuracy is greatly affected by the j-th bit of C_0 or C_1, the j-th ciphertext bit is an **informative bit**.

An $\mathcal{ND}^t$ works since it has learned knowledge from ciphertext bits. According to *Definition 1*, only informative bits provide knowledge. Thus the ciphertext bit that has a significant influence on the distinguishing accuracy of $\mathcal{ND}^t$ must be an *informative bit*.

Definition 3 can not ensure each informative bit that obeys the *Definition 1* is identified successfully. But we only care about informative bits that are captured by an $\mathcal{ND}^t$. This approximate definition helps develop a simple but effective framework for identifying informative bits.

The proposed framework is named **Bit Sensitivity Test** (BST). Its core idea is to test whether the distinguishing accuracy of an $\mathcal{ND}^t$ drops after we remove some knowledge related to the specific bit.

Gohr in [9] has proved that $\mathcal{ND}^t_h, h \in \{5,6,7,8\}$ against Speck32/64 captures the knowledge about the ciphertext difference and some unknown features. Consider the j-th ciphertext bit. We remove the knowledge about the j-th ciphertext bit difference by

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

where η is a random mask that could be 0 or 1.

We have performed an extreme test on $\mathcal{ND}^t_h, h \in \{5,6,7,8\}$ against Speck32/64. If we XOR each bit of C_0 or C_1 with a random mask, $\mathcal{ND}^t_h, h \in \{5,6,7,8\}$ can not distinguish positive samples and negative samples anymore. These tests imply that knowledge about unknown features is also removed by one of the two operations presented in formula 29.

After the knowledge related to a ciphertext bit is removed, the accuracy decrease of $\mathcal{ND}^t$ is named **Bit Sensitivity**, which is used to identify informative bits. Algorithm 5 summarizes the BST.

Examples and analysis. We have applied the BST to $\mathcal{ND}^t_h, h \in \{5,6,7\}$ against Speck32/64. The results of the BST under three scenarios are shown in Fig. 2, Fig. 3, and Fig. 4 respectively.

We observe that $sen_0 \approx sen_1$. This proves that $C_0 \oplus (\eta \ll j)$ is equivalent to $C_1 \oplus (\eta \ll j)$. Besides, we know

- If $sen_0[j] > 0$, the j-th ciphertext bit is an informative bit.
- 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.
- If $sen_0[j] \approx sen_{0,1}[j]$, the j-th ciphertext bit difference has little influence on $\mathcal{ND}^t_h$.

Algorithm 5 Bit Sensitivity Test

Require: a cipher with a block size of m;
an $\mathcal{ND}^t$ against this cipher;
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 m ciphertext bits.

```
Test the distinguishing accuracy of ND^t on the test dataset. Save it to sen[m];
for j = 0 to m − 1 do
    for i = 1 to M do
        Generate a random mask η ∈ {0, 1};
        C_0^{i,new} = C_0^i ⊕ (η ≪ j);
        Feed the new sample (C_0^{i,new}, C_1^i) to ND^t;
    end for
    Count the current accuracy cp;
    sen[j] = sen[m] − cp;
end for
```

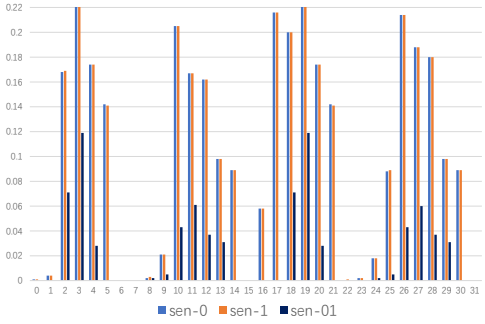

Fig. 2. Results of BST of $\mathcal{ND}^t_5$ against Speck32/64, $M = 10^6$. sen_0 is the results of performing $C_0 \oplus (\eta \ll j)$, sen_1 is the results of performing $C_1 \oplus (\eta \ll j)$, $sen_{0,1}$ is the results of performing two operations simultaneously, $j \in \{0, \cdots, 31\}$.

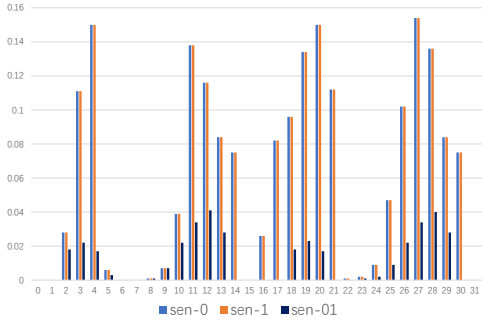

Fig. 3. Results of BST of $\mathcal{ND}^t_6$ against Speck32/64, $M = 10^6$.

Reverse verification about identified informative bits. To further verify *Definition 3*, a reverse verification about identified informative bits is performed. First, select some informative bits. Second, train an $\mathcal{ND}^s$ on selected informative bits and observe the distinguishing accuracy.

Taking $\mathcal{ND}^t_7$ against Speck32/64 as an example, we have performed the reverse verification based on results in Fig. 4. Table IX shows the distinguishing accuracies under two settings. For Speck32/64, the j-th and $(j+16)$-th bit are directly

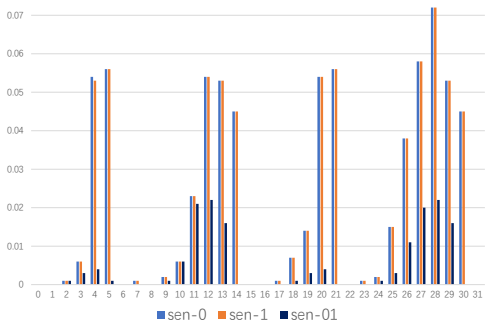

Fig. 4. Results of BST of $\mathcal{ND}_7^t$ against Speck32/64, $M = 10^6$.

related to the same subkey bit. Thus the 8-th and 1st ciphertext bits are also considered.

TABLE IX
ACCURACIES OF NEURAL DISTINGUISHERS TRAINED ON SELECTED CIPHERTEXT BITS

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

The accuracy of $\mathcal{ND}_7^t$ is 0.6067. When all the identified informative bits are considered, the resulted $\mathcal{ND}_7^s$ obtains 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.

Once informative bits are identified by the *Bit Sensitivity Test*, the whole key space can be divided into several subspaces. In each subspace, NASA is performed to recover specific key bits.

This informative-bit-based method is the first generic technique for reducing the attack complexities of NASA.

VI. Selective Key Search

In the basic NASA as shown in Algorithm 4, each possible key guess kg is tested. Inspired by the analysis of p_2 in Section III-E, we develop a highly selective key search strategy for further reducing the attack complexity.

Specifically, we do not need to traverse all the key guesses. Some key guesses that are most likely to be the right key are recommended based on the keys that have been tested.

A. Distribution of the Statistic Under Different Keys

In this section, we first discuss the distribution of the statistic (formula 6) under different keys again.

Here we still take Fig. 1 as the example. Suppose that the size of the key guess kg is m. According to the analysis in Section III-E, we know there are the following $m+1$ probabilities

$$p_{2|d_1} = Pr\{z > c_2 | S_0 \oplus S_1 = \Delta S\}, \quad 0 \leqslant d_1 \leqslant m,$$

where d_1 is the Hamming distance between kg and the right key.

Then there are $m+1$ resulting distributions of the statistic (formula 6)

$$T_{d_1} \sim \mathcal{N}(\mu_{d_1}, \sigma_{d_1}), \tag{30}$$

$$\mu_{d_1} = N \times (p_0 \times p_{2|d_1} + (1-p_0) p_n),$$

$$\sigma_{d_1} = \sqrt{N \times p_0 \times p_{2|d_1}\left(1-p_{2|d_1}\right) + N\left(1-p_0\right) p_n \left(1-p_n\right)}.$$

The parameters of these $m+1$ distributions are obtained off-line. These distributions are used as prior knowledge to improve the key search strategy of NASA.

B. Bayesian Key Search Strategy

Algorithm 6 summarizes the newly proposed Bayesian key search algorithm, which is the second technique for reducing the attack complexities of NASA.

Algorithm 6 Bayesian Key Search Algorithm

Require: Ciphertext pairs, $(C_0^i, C_1^i), i \in \{1, \cdots, N\}$;
A neural distinguisher, $\mathcal{ND}$;
Prior knowledge μ_{d_1} and $\sigma_{d_1}, d_1 \in \{0, \cdots, m\}$;
The number of key guess candidates to be generated within each iteration, n_{cand};
The number of iterations, n_{iter}.

Ensure: The list L of tuples of recommended keys and corresponding statistics.

$\mathcal{K} = \{kg_1, \cdots, kg_{cand}\} \leftarrow$ choose n_{cand} values at random without replacement from the set of all subkey candidates.
$L \leftarrow \{\}$
for $t = 1$ to n_{iter} **do**
  for each $kg \in \mathcal{K}$ **do**
    for $i = 1$ to N **do**
      Decrypt C_0^i, C_1^i with kg.
      Feed partially decrypted ciphertext pair into $\mathcal{ND}$.
      Collect the output $Z_{i,kg}$ of $\mathcal{ND}$.
    end for
    Compute the statistic T^{kg} (formula 6).
    $L \leftarrow L || (kg, T^{kg})$.
  end for
  for $sk \in \{0, \cdots, 2^m - 1\}$ **do**
    $\lambda_{sk} = \sum_{kg \in \mathcal{K}} \left(T^{kg} - \mu_{hd(kg \oplus sk)}\right)^2 / \sigma^2_{hd(kg \oplus sk)}$.
    /* $hd(kg \oplus sk)$ is the Hamming distance between kg and sk */
  end for
  $\mathcal{K} \leftarrow argsort_{sk}(\lambda)[0 : n_{cand} - 1]$.
  /* Pick n_{cand} key guesses with the n_{cand} smallest score to form the new set of key guess candidates $\mathcal{K}$ */
end for
Return L

VII. Reduce the Data Complexity

Consider the prepended differential $\Delta P \xrightarrow{p_0} \Delta S$. As we have presented in Section III-C, the impact of p_0 on the data complexity is about $\mathcal{O}(p_0^{-2})$.

Neutral bits seldom exist in a long differential characteristic. But there usually are numerous neutral bits in a short

differential characteristic. This section shows how to reduce the data complexity of NASA by neutral bits.

A. Improved Neural Aided Statistical Attack

We still take the key recovery attack with 1-round decryption as an example. Fig. 5 shows the scheme of the improved NASA.

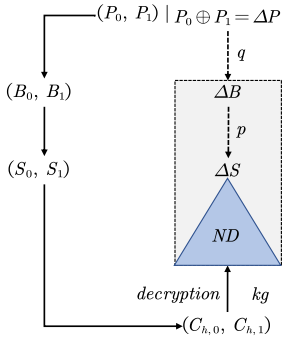

Fig. 5. The scheme of the improved neural aided statistical attack. The statistical distinguisher only covers the second differential $\Delta B \rightarrow \Delta S$. Neutral bits that exist in $\Delta P \rightarrow \Delta B$ are exploited.

Its core idea is dividing the initial long differential into two short ones: $\Delta P \xrightarrow{q} \Delta B$, and $\Delta B \xrightarrow{p} \Delta S$ where $p_0 = q \times p$. The statistical distinguisher only covers the second differential $\Delta B \rightarrow \Delta S$. Neutral bits that exist in the first part $\Delta P \rightarrow \Delta B$ are exploited. Algorithm 7 summarizes the details of the improved NASA.

Algorithm 7 Improved neural aided statistical attack

Require: The attacked cipher;
The prepended differential, $\Delta P \xrightarrow{q} \Delta B \xrightarrow{p} \Delta S$;
Neutral bits that exist in $\Delta P \rightarrow \Delta B$;
Two maximum error probabilities, β_r, β_w;
A posterior probability threshold, c_2.
Ensure: All possible key candidates.
1: Train an $\mathcal{ND}$ over ΔS;
2: Estimate p_1, p_n, p_2 using $\mathcal{ND}$ (Section III-D, Algorithm 2);
3: Calculate the data complexity N_1 based on p, p_1, p_n, p_2 (Section III-C);
4: **for** j from 1 to $\frac{1}{q}$ **do**
5: Based on ΔP and neutral bits, randomly generate a plaintext structure $\mathcal{P}$ consisting of N_1 plaintext pairs.
6: Perform the basic NASA based on $\mathcal{P}$ (Algorithm 4).
7: **end for**
8: Test surviving key candidates against a necessary number of plaintext-ciphertext pairs according to the unicity distance for the attacked cipher.

Now, only the impact of p on the total data complexity is $\mathcal{O}(p^{-2})$. The impact of q on the total data complexity is $\mathcal{O}(q^{-1})$. Thus, the total impact of the prepended differential is $\mathcal{O}(q^{-1}p^{-2})$ instead of $\mathcal{O}(p_0^{-2}) = \mathcal{O}(q^{-2}p^{-2})$. The data complexity is reduced by a factor of about q^{-1}.

B. Further Improvement Based On Early Stopping

In Algorithm 7, $\frac{1}{q}$ plaintext structures are generated. But only one plaintext structure $\mathcal{P}$ is expected to satisfy the differential $\Delta P \rightarrow \Delta B$.

This plaintext structure is called **valid** plaintext structure while other plaintext structures are called **invalid** plaintext structures. If a valid plaintext structure is identified once it arises, Algorithm 7 can be early stopped at step 4.

We propose an identification method that does not change the process of Algorithm 7. At a high level, the identification method is as follows:

1) Generate a plaintext structure $\mathcal{P}$ consisting of M plaintext pairs.
2) Filter key guesses based on the statistic T (formula 6) and a decision threshold t_M.
3) If the number of surviving key guesses exceeds a threshold t_P, $\mathcal{P}$ is a valid plaintext structure.

By setting proper parameters t_M, the number of surviving key guesses exceeds t_P only when $\mathcal{P}$ is a valid plaintext structure.

Next, we present an theoretical analysis of M, t_M, t_P. For convenience, we rewrite the statistic T (formula 6) as

$$T = \sum_{i=1}^{M} \phi(Z_i), \quad \phi(Z_i) = \begin{cases} 1, if\ Z_i > c_2 \\ 0, if\ Z_i \leqslant c_2 \end{cases}. \tag{31}$$

The four situations shown in Fig. 1 also exist in the improved NASA (see Fig. 5). The following notations introduced in Section III are adopted again:

- $p_{2|d_1}$: the probability $Pr(Z > c_2 | S_0 \oplus S_1 = \Delta S)$ when the Hamming distance between the key guess and the right key is $d_1 \in [0, m]$.
- p_n : the probability $Pr(Z > c_2 | S_0 \oplus S_1 \neq \Delta S)$.

1) Distribution of the statistic under valid plaintext structures: When $\mathcal{P}$ is a valid plaintext structure that satisfies $\Delta P \rightarrow \Delta B$, there are $M \times p$ positive samples and $M \times (1-p)$ negative samples.

When d_1 is not set clearly, we denote $p_{2|d_1}$ as p_V. The distribution of the statistic(formula 31) is

$$T_V \sim \mathcal{N}(\mu_V, \sigma_V), \tag{32}$$

$$\mu_V = M[p \times p_V + (1-p)p_n], \tag{33}$$

$$\sigma_V = \sqrt{M \times p \times p_V(1-p_V) + M(1-p)p_n(1-p_n)}. \tag{34}$$

Select a specific d_1, we have $p_V = p_{2|d_1}$. Let $\mathcal{K}_\mathcal{V}$ denote the set of key guesses with a Hamming distance d_1 from the right key. Then only $kg \in \mathcal{K}_\mathcal{V}$ makes the above T_V hold.

2) Distribution of the statistic under invalid plaintext structures: When $\mathcal{P}$ is an invalid plaintext structure, all the M samples are negative samples.

The distribution of the statistic(formula 31) is

$$T_I \sim \mathcal{N}(\mu_I, \sigma_I) \tag{35}$$

$$\mu_I = M \times p_n, \quad \sigma_I = \sqrt{M \times p_n(1-p_n)} \tag{36}$$

Let $\mathcal{K}$ denote the set of all possible key guesses. Then any $kg \in \mathcal{K}$ makes the above T_I hold.

3) Distinguishing between T_V and T_I: Since T_V and T_I are two different normal distributions, the technique in Section II-D is used to distinguish these two distributions. According to *Proposition 1*, the condition for distinguishing T_V and T_I is

$$\frac{z_{1-\beta_V} \times \sigma_V + z_{1-\beta_I} \times \sigma_I}{\mu_V - \mu_I} = 1 \tag{37}$$

where

$$\begin{aligned} \beta_V &= Pr\left\{\mathcal{N}\left(\mu_I, \sigma_I\right) | s \in \mathcal{N}\left(\mu_V, \sigma_V\right)\right\}, \\ \beta_I &= Pr\left\{\mathcal{N}\left(\mu_V, \sigma_V\right) | s \in \mathcal{N}\left(\mu_I, \sigma_I\right)\right\}, \end{aligned} \tag{38}$$

and s stands for a sample.

Here, we present an explanation about the two error probabilities β_I, β_V. When $\mathcal{P}$ is an invalid plaintext structure, the maximum probability that key guesses $kg \in \mathcal{K}$ survive the attack is β_I. When $\mathcal{P}$ is a valid plaintext structure, the minimum probability that key guesses $kg \in \mathcal{K}_\mathcal{V}$ survive the attack is $1 - \beta_V$.

By simplifying formula 37, the data complexity M is

$$a_V = p_V(1 - p_V), \quad a_n = p_n(1 - p_n), \tag{39}$$

$$\sqrt{M} = \frac{z_{1-\beta_V} \times \sqrt{p \times a_V + (1-p)a_n} + z_{1-\beta_I} \times \sqrt{a_n}}{(p_V - p_n) \times p}. \tag{40}$$

And the decision threshold t_M is

$$t_M = \mu_V - z_{1-\beta_V}\sigma_V = \mu_I + z_{1-\beta_I}\sigma_I, \tag{41}$$

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

4) Identify valid plaintext structures by counting surviving keys: When $\mathcal{P}$ is a valid plaintext structure, The lower bound of the number of surviving subkeys is $|\mathcal{K}_\mathcal{V}| \times (1 - \beta_V)$. When $\mathcal{P}$ is an invalid plaintext structure, The upper bound of the number of surviving subkeys is $|\mathcal{K}| \times \beta_I$.

By setting two proper error probabilities β_V, β_I, we ensure the following condition always holds

$$|\mathcal{K}_\mathcal{R}| \times (1 - \beta_V) >> |\mathcal{K}| \times \beta_I. \tag{42}$$

Let t_S satisfy the following condition

$$|\mathcal{K}_\mathcal{R}| \times (1 - \beta_V) \geqslant t_S >> |\mathcal{K}| \times \beta_I. \tag{43}$$

Then valid plaintext structures is identified by by comparing the number of surviving subkey guesses with t_S.

Algorithm 8 summarizes the concrete identification process. Since the identification is also based on the same statistic as the key recovery, Algorithm 8 and Algorithm 7 are able to be performed simultaneously. The necessary condition is that the size of a plaintext structure $\mathcal{P}$ should exceed N_1 and M.

5) Further analysis about p_V.: The data complexity M is related to p_V. And p_V is related to the Hamming distance d_1.

When p_V increases, M (Equation 40) decreases since

$$\begin{aligned} \sqrt{M} &= \frac{z_{1-\beta_V} \times \sqrt{p \times a_V + (1-p)a_n} + z_{1-\beta_I} \times \sqrt{a_n}}{(p_V - p_n) \times p} \\ &= \frac{z_{1-\beta_V} \times \sqrt{p(1-p_V) + \frac{(1-p)a_n}{p_V}} + \frac{z_{1-\beta_I} \times \sqrt{a_n}}{p_V}}{(\sqrt{p_V} - \frac{p_n}{\sqrt{p_V}}) \times p}. \end{aligned}$$

Algorithm 8 Identify valid plaintext structures

Require: a plaintext structure $\mathcal{P}$ with a size of M(formula 40);
an $\mathcal{ND}$ trained over ΔS;
the posterior probability threshold c_2;
a decision threshold t_M for filtering subkey guesses;
a decision threshold t_P for identifying valid plaintext structures.
Ensure: the classification of $\mathcal{P}$.

```
Collect the M ciphertext pairs corresponding to P;
Initialize a counter cp ← 0;
for each possible subkey guess kg do
    Decrypt M ciphertext pairs with kg;
    Feed partially decrypted ciphertext pairs into ND;
    Save the outputs of ND, Z_i, i ∈ [1, M];
    Count the number of Z_i > c, and denote it as T_M;
    if T_M > t_M then
        cp ← cp + 1;
    end if
end for
if cp ⩾ t_P then
    Return 1 (P is a valid plaintext structure).
else
    Return 0 (P is an invalid plaintext structure).
end if
```

If p_V increases, the numerator will decrease and the denominator will increase. Then M will decrease.

When the Hamming distance d_1 increases, $p_{2|d_1}$ will decrease in high probability. But the number of subkey guesses in the subspace may increase sharply when d_1 increases. As long as the condition (formula 42) holds, we advise $p_V = p_{2|d_1}$ where d_1 should be as small as possible.

VIII. Application to DES

This section proves that NASA surpasses Gohr's attack when enough neutral bits do not exist in the prepended long differential.

DES [13] is a block cipher with a block size of 64 bits. The structure of DES is the classical Feistel structure as shown in Fig. 6. Its round function f is composed of eight different S-boxs. More details refer to [13], please. We perform key recovery attacks on round reduced DES.

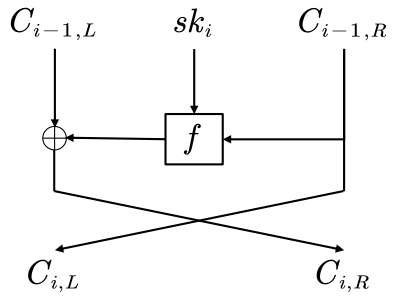

Fig. 6. A round of encryption of DES.

A. Prepended Differentials

Two optimal 2-round iterative differentials found in [14] are

$$\begin{aligned} 0x19600000/0 &\xrightarrow{Pr=\frac{1}{234}} 0x19600000/0, \\ 0x1B600000/0 &\xrightarrow{Pr=\frac{1}{234}} 0x1B600000/0. \end{aligned} \tag{44}$$

Based on these 2-round differentials, we can get the iterative differential of any even-numbered round. These iterative differentials are used as the prepended differential $\Delta P \rightarrow \Delta S$ for attacking round reduced DES.

According to the definition of the neutral bit, We measure the neutrality of each ciphertext bit experimentally. We find that 18 neutral bits $\{33, \cdots, 50\}$ exist in the above 2-round iterative differentials. As for 4-round iterative differentials, no neutral bits exist anymore.

B. Build Neural Distinguishers Against DES

Let $\Delta S = 0x19600000/0$, we build teacher distinguishers against DES up to 5 rounds. The distinguishing accuracy of the 5-round teacher distinguisher is 0.58.

The *Bit Sensitivity Test* identifies 12 informative bits $\{39, 50, 56, 61, 59, 37, 43, 49, 42, 32, 53, 52\}$. The 4 bits $\{39, 50, 56, 61\}$ are related to the fifth S-box $S5$, and $\{59, 37, 43, 49\}$ are related to the eighth S-box $S8$.

In order to introduce the next experiment more clearly, we focus on the student distinguisher ND_5^s built over the bits $\{39, 50, 56, 61\}$ for now. Let the posterior probability threshold be $c_2 = 0.5$, we get $p_1 = 0.6032$ and $p_n = 0.4883$. Table X shows the estimation of $p_{2|d_1}$.

TABLE X
THE ESTIMATION OF $p_{2|d_1}$ OF ND_5^s AGAINST ROUND REDUCED DES. $c_2 = 0.5$. ND_5^s IS BUILT OVER 4 BITS $\{39, 50, 56, 61\}$.

d_1	0	1	2	3	4	5	6
$p_{2\|d_1}$	0.6034	0.505	0.507	0.509	0.5101	0.504	0.496

C. Gohr's Attack on DES

By placing the 2-round differential $0x19600000/0 \xrightarrow{234^{-1}} 0x19600000/0$ before the student distinguisher ND_5^s, with the help of the 18 neutral bits $\{33, \cdots, 50\}$, 8-round DES is broken by Gohr's attack. The 6 key bits related to $S5$ of the last round are recovered.

Since no neutral bits exist in the 4-round iterative differential $0x19600000/0 \xrightarrow{234^{-2}} 0x19600000/0$, 10-round DES can not be attacked by Gohr's attack.

D. Neural Aided Statistical Attack on DES

Consider the basic NASA (Algorithm 4). The 4-round differential $0x19600000/0 \xrightarrow{234^{-2}} 0x19600000/0$ is the prepended differential. The target is also recovering the 6 key bits related to $S5$ of the last round by $\mathcal{ND}_5^s$.

Let $p_2 = 0.5101$, $\beta_r = 0.005$, and $\beta_w = 2^{-6}$. Since $p_0 = 234^{-2}$, the required data complexity is $N = 2^{40.814}$ chosen-plaintext pairs. Thus, 10-round DES is broken.

If we adopt the improved NASA, the data complexity is smaller. Thus, when there are no enough neutral bits, NASA surpasses Gohr's attack.

E. Time Complexity

Since the output of an S-box only contains 4 bits, we build a look-up table off-line for saving the tuple $((C_0, C_1), \mathcal{ND}_5^s(C_0, C_1))$. Then the time complexity is not related to $\mathcal{ND}_5^s$ anymore. In other words, the time complexity of the basic NASA is $N \times 2^6 = 2^{46.814}$.

By building $\mathcal{ND}_5^s$ over outputs of different S-box and guessing partial key bits, the complete master key is recovered.

IX. APPLICATION TO SPECK32/64

When there are enough neutral bits in the prepended differential $\Delta P \xrightarrow{p_0} \Delta S$, the attack complexities of Gohr's attack are much lower than the differential attack. For Gohr's attack, the impact of p_0 on the data complexity is only $\mathcal{O}(p_0^{-1})$. For basic NASA, the impact of p_0 on the data complexity is $\mathcal{O}(p_0^{-2})$.

In this section, we prove that NASA can achieve comparable performance as Gohr's attack when the three optimisation techniques introduced in Section V, VI, VII are available.

Speck32/64 is one variant of the Speck family that is designed by the NSA Research Directorate [15]. Fig. 7 shows the round function of Speck.

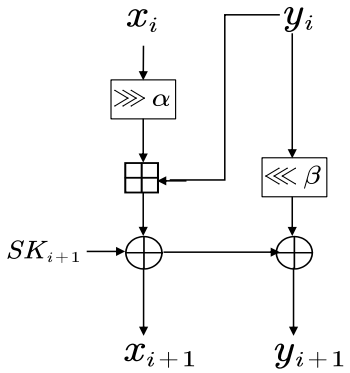

Fig. 7. The round function of Speck32/64. SK_{i+1} : the round key. $L_{i+1}||R_{i+1}$: the state of the $(i+1)$-th round. For Speck32/64, $\alpha = 7, \beta = 2$.

A. Prepended Differential

To attack 11-round Speck32/64, Gohr adopted the following 2-round prepended differential

$$\Delta P = (0x211, 0x204) \xrightarrow{p_0 = 2^{-6}} \Delta S = (0x40, 0x0).$$

There are only 3 neutral bits $\{20, 21, 22\}$ that exist in $\Delta P \rightarrow \Delta S$. Besides, there are 2 high probabilistic neutral bits $\{14, 15\}$ whose neutrality exceeds 0.95.

Dividing this differential into two 1-round differentials

$$\begin{aligned} \Delta P = (0x211, 0x204) &\xrightarrow{q=2^{-4}} \Delta B = (0x2800, 0x10) \\ \Delta B = (0x2800, 0x10) &\xrightarrow{p=2^{-2}} \Delta S = (0x40, 0), \end{aligned}$$

we measure the neutrality of each ciphertext bit. There are 8 neutral bits $\{0, 11, 14, 15, 20, 21, 22, 26, \}$ that exist in $\Delta P \rightarrow \Delta B$. Moreover, there are 8 high probabilistic neutral bits $\{1, 3, 4, 5, 23, 24, 27, 28\}$ whose neutrality exceeds 0.95.

B. Build Neural Distinguishers Against Speck32/64

Two teacher distinguishers $\mathcal{ND}_6^t, \mathcal{ND}_7^t$ built by Gohr over $\Delta S = (0x40, 0)$ are adopted in this section. Table I shows the estimation of $p_{2|d_1}$ of $\mathcal{ND}_6^t, \mathcal{ND}_7^t$ when $c_2 = 0.55$.

Based on the bit sensitivity test results of $\mathcal{ND}_6^t, \mathcal{ND}_7^t$ as shown in Fig. 3 and Fig. 4 (see Section V), we build 2 student distinguishers $\mathcal{ND}_6^s, \mathcal{ND}_7^s$ by setting $\Gamma = \{30 \sim 23, 14 \sim 7\}$.

These 16 ciphertext bits are related to the least significant 8 subkey bits. Later, $\mathcal{ND}_6^s, \mathcal{ND}_7^s$ are used to recover $sk_{10}[7 \sim 0]$ and $sk_{11}[7 \sim 0]$ respectively. Let $c_2 = 0.55$, Table XI shows the estimation of $p_{2|d_1}$ of $\mathcal{ND}_6^s, \mathcal{ND}_7^s$.

TABLE XI
THE ESTIMATION OF $p_{2|d_1}$ OF $\mathcal{ND}_6^s, \mathcal{ND}_7^s$ AGAINST SPECK32/64. $c_2 = 0.55$. THE SUBSCRIPT SET OF SELECTED CIPHERTEXT BITS IS $\Gamma = \{30 \sim 23, 14 \sim 7\}$

$\mathcal{ND}_6^s$	d_1	0	1	2	$3 \sim 8$
	$p_{2\|d_1}$	0.5132	0.4057	0.3402	$\leqslant 0.3025$
$\mathcal{ND}_7^s$	d_1	0	1	2	$3 \sim 8$
	$p_{2\|d_1}$	0.3576	0.3230	0.3036	$\leqslant 0.2940$

Let $c_2 = 0.55$, Table XII summarizes the estimation of p_1, p_n of $\mathcal{ND}_6^s, \mathcal{ND}_7^s, \mathcal{ND}_6^t, \mathcal{ND}_7^t$ against Speck32/64.

TABLE XII
THE ESTIMATION OF p_1, p_n OF $\mathcal{ND}_6^s, \mathcal{ND}_7^s, \mathcal{ND}_6^t, \mathcal{ND}_7^t$ AGAINST SPECK32/64. $c_2 = 0.55$.

	$\mathcal{ND}_6^s$	$\mathcal{ND}_7^s$	$\mathcal{ND}_6^t$	$\mathcal{ND}_7^t$
p_1	0.5129	0.3575	0.6787	0.4183
p_n	0.2604	0.2865	0.1162	0.2162

C. Gohr's Attack on Speck32/64

Based on the 2-round prepended differential $\Delta P \rightarrow \Delta S$ and $\mathcal{ND}_7^t$, $\mathcal{ND}_6^t$, Gohr presented a key recovery attack on 11-round Speck32/64. By adopting some optimization techniques for accelerating the basic key (Section II-C), Gohr broken 11-round Speck32/64 with a time complexity of 2^{38}.

The target is to recover the last two subkeys sk_{10}, sk_{11}. Gohr counted a key guess as successful if the last subkey was guessed correctly and if the second subkey was at hamming distance at most two of the real key sk_{10}. Finally, the success rate of Gohr's attack is about 52%.

We have performed this accelerated attack again based on the code provided by Gohr. By adopting an Intel(R) Core(TM) i5-7500 CPU and one graphics card (NVIDIA GeForce GTX 1060(6GB)), we find that the average time consumption of performing this attack one time is 45 seconds.

D. Neural Aided Statistical Attack on Speck32/64

We also perform the key recovery attack on 11-round Speck32/64 by adopting NASA. At a high level, the attack is composed of five stages:

- stage 1: Identify the valid plaintext structure $\mathcal{P}$ that satisfies $\Delta P \rightarrow \Delta B$ by $\mathcal{ND}_7^s$ (Algorithm 8). The subkey to be searched is $sk_{11}[7 \sim 0]$.
- stage 2: Recover $sk_{11}[7 \sim 0]$ by $\mathcal{ND}_7^s$ (Algorithm 4).
- stage 3: Recover sk_{11} by $\mathcal{ND}_7^t$ (Algorithm 4).
- stage 4: Recover $(sk_{11}, sk_{10}[7 \sim 0])$ by $\mathcal{ND}_6^s$ (Algorithm 4).
- stage 5: Recover (sk_{11}, sk_{10}) by $\mathcal{ND}_6^t$ (Algorithm 4).

In stage 1, we set $p_V = p_{2|d_1=1}, \beta_V = 0.1, \beta_I = 2^{-8}$. It means that the number of surviving subkey guess $kg_{11}[7 \sim 0]$ should exceed $8(1 - 0.1) = 7.2$ when $\mathcal{P}$ is a valid plaintext structure. Or the number of surviving subkey should not exceed $2^8 \times 2^{-8} = 1$. Based on this setting, each plaintext structure $\mathcal{P}$ should contain $M = 2^{15.235}$ plaintext pairs. The decision threshold for filtering subkey guess is $t_M = 11285$. When the number of surviving subkey guess is greater than or equal to $t_P = 8$, $\mathcal{P}$ is a valid plaintext structure.

In stage 2, we set $p_2 = p_{2|d_1=3}, \beta_r = 0.005, \beta_w = 2^{-8}$. Then the number of surviving subkey guess $kg_{11}[7 \sim 0]$ should not exceed 37. The required data complexity is $N = 2^{14.465}$ plaintext pairs, and the decision threshold for filtering subkey guess is $t = 6703$.

In stage 3, we filter kg_{11} based on each surviving $kg_{11}[7 \sim 0]$. Let $p_2 = p_{2|d_1=2}, \beta_r = 0.005, \beta_w = 2^{-16}$, the number of surviving subkey guess kg_{11} should not exceed 137. Besides, we have $N = 2^{12.373}, t = 1333$.

In stage 4, we filter $(kg_{10}[7 \sim 0], kg_{11})$ based on each surviving kg_{11}. Let $p_2 = p_{2|d_1=1}, \beta_r = 0.001, \beta_w = 2^{-14}$, the number of surviving subkey guess kg_{11} should not exceed 16. Besides, we have $N = 2^{12.36}, t = 1598$.

In stage 5, we filter (kg_{10}, kg_{11}) based on each surviving $(kg_{10}[7 \sim 0], kg_{11})$. Let $p_2 = p_{2|d_1=1}, \beta_r = 0.001, \beta_w = 2^{-16}$, the number of surviving subkey guess kg_{11} should not exceed 16. Besides, we have $N = 2^{9.694}, t = 180$.

We use 16 high probabilistic neutral bits $\{0, 1, 3 \sim 5, 11, 14, 15, 20 \sim 24, 26 \sim 28\}$ that exist in $\Delta P \rightarrow \Delta B$ to generate plaintext structures $\mathcal{P}$ consisting of $M = 2^{15.235}$ plaintext pairs with a difference $\Delta P = (0x211, 0xa04)$. The probability that $\mathcal{P}$ is a valid plaintext structure is about $2^{-4.2}$. In stage 1, if no valid plaintext structures occur after 32 plaintext structures are generated, the attack is stopped and viewed as a failure.

Once one valid plaintext structure $\mathcal{P}$ is found, the remaining 4 stages are performed based on this structure $\mathcal{P}$. More exactly, $2^{14.465}, 2^{12.373}, 2^{12.36}, 2^{9.694}$ plaintext pairs are randomly selected from this valid plaintext structure respectively.

In stage $1 \sim 5$, we do not traverse each possible subkey guess. The proposed Bayesian key search strategy (Algorithm 6) is applied in each stage, which does not influence the above attack setting.

The settings related to the Bayesian key search strategy are as follows. In stage 1 and stage 2, the number of iterations is $n_{iter} = 3$. For each iteration, we search $n_{cand} = 32$ subkey guesses. In stage 3, we set $n_{iter} = 4, n_{cand} = 32$. In stage 4 and stage 5, we set $n_{iter} = 32, n_{cand} = 32$.

We count a key guess as successful if the right subkey pair (sk_{10}, sk_{11}) survives and if the average number of surviving key guesses in stage 5 does not exceed 16. Under these conditions, the attack was successful 585 out of 1000 trials. The average number of generated plaintext structures is 18. For comparison, under the same hardware environment, the

average time consumption of this attack one time is 87 seconds.

E. Time Complexity

Once the valid plaintext structure is found, the key recovery process is very fast. Compared with the time consumption of recovering (sk_{10}, sk_{11}), the time consumption of recovering the remaining subkeys is negligible.

Under the same hardware environment, the average time consumption of NASA is about twice as long as that of Gohr's attack. The time complexity of Gohr's attack is 2^{38}. Thus, the time complexity of NASA is 2^{39}.

X. Conclusion

In this article, we propose a Neural Aided Statistical Attack (NASA) and three methods for reducing the complexity of NASA. NASA recovers the right key based on distinguishing between two different normal distributions. NASA is the first deep learning-based cryptanalysis technique that supports theoretical complexity estimation and does not rely on any special properties such as neutral bits. Applications to round reduced DES and Speck32/64 also prove that NASA is full of potential.

Our work in this article also provides many inspirations for neural aided cryptanalysis. First, if we replace the neural network with other machine learning models, NASA still works. Compared with neural networks, other shallow machine learning models such as logistic regression are far faster. Thus, it is possible to further accelerate NASA by adopting other machine learning-based distinguishers. Second, when we try to reduce the key space, we find that ciphertext bits have a different influence on the neural distinguisher. This finding is not only useful for neural aided cryptanalysis. The traditional differential attack may be improved by exploiting knowledge extracted from neural distinguishers. Third, the data complexity for distinguishing two normal distributions is very high, which makes the data complexity of basic NASA is also high. If some new probability distributions are more suitable for simulating the key recovery process, new neural aided attacks with a lower complexity are able to be developed.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (2018YFB0803405, 2017YFA0303903).

Appendix

For the conventional differential cryptanalysis, the data complexity is only related to the differential. For 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. 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 as

$$\begin{aligned}\sqrt{N} &= \frac{z_{1-\beta_r}\sqrt{p_0a_1+(1-p_0)a_3}+z_{1-\beta_w}\sqrt{p_0a_2+(1-p_0)a_3}}{(p_1-p_2)\times p_0}\\ &\propto \frac{z_{1-\beta_r}\sqrt{a_3+(a_1-a_3)p_0}+z_{1-\beta_w}\sqrt{a_3+(a_2-a_3)p_0}}{p_0}\\ &\propto \frac{\sqrt{a_3+(a_1-a_3)p_0}+a_4\times\sqrt{a_3+(a_2-a_3)p_0}}{p_0}\end{aligned}$$

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

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

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 is $\mathcal{O}(p_0^{-2})$.

B. 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.

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

$$\begin{aligned}\sqrt{N} &= \frac{z_{1-\beta_r}\sqrt{p_0a_1+(1-p_0)a_3}+z_{1-\beta_w}\sqrt{p_0a_2+(1-p_0)a_3}}{(p_1-p_2)\times p_0}\\ &\propto \sqrt{p_0a_1+(1-p_0)a_3}+a_4\times\sqrt{p_0a_2+(1-p_0)a_3}\\ \Rightarrow N &\propto p_0\times a_1+a_4^2\times p_0\times a_2+(1-p_0)(1+a_4^2)a_3+a_5\end{aligned}$$

Next, we 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 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 $\mathcal{O}(p_n)$.

2) 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 p_0. Due to this property, we also focus on attack scenarios where $p_0\to 0$.

$$\begin{aligned}\sqrt{N} &= \frac{z_{1-\beta_r}\sqrt{p_0a_1+(1-p_0)a_3}+z_{1-\beta_w}\sqrt{p_0a_2+(1-p_0)a_3}}{(p_1-p_2)\times p_0}\\ &\approx \frac{z_{1-\beta_r}\sqrt{(1-p_0)a_3}+z_{1-\beta_w}\sqrt{(1-p_0)a_3}}{(p_1-p_2)\times p_0}\propto(p_1-p_2)^{-1}\end{aligned}$$

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

References

[1] L. R. Ronald, "Cryptography and machine learning," *International Conference on the Theory and Application of Cryptology*, pp. 427–439, 1991.

[2] S. Greydanus, "Learning the enigma with recurrent neural networks." *arXiv: Neural and Evolutionary Computing*, 2017.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems (NeurIPS)*, pp. 1097–1105, 2012.

[4] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *Neural Information Processing Systems (NeurIPS)*, pp. 932–938, 2000.

[5] Y. Chen, L. Yu, K. Ota, and M. Dong, "Robust activity recognition for aging society," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 6, pp. 1754–1764, 2018.

[6] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 45–68.

[7] L. Batina, S. Bhasin, D. Jap, and S. Picek, "Poster: Recovering the input of neural networks via single shot side-channel attacks." *computer and communications security*, pp. 2657–2659, 2019.

[8] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise: Unleashing the power of convolutional neural networks for profiled side-channel analysis," *cryptographic hardware and embedded systems*, vol. 2019, no. 3, pp. 148–179, 2019.

[9] A. Gohr, "Improving attacks on round-reduced speck32/64 using deep learning," *international cryptology conference*, pp. 150–179, 2019.

[10] B. Eli and C. Rafi, "Near-collisions of sha-0," *Annual International Cryptology Conference*, pp. 290–305, 2004.

[11] W. Feller, "An introduction to probability theory and its applications. vol. ii," *Population*, vol. 23, no. 2, p. 375, 1968.

[12] R. Gisselquist, P. G. Hoel, S. C. Port, and C. J. Stone, "Introduction to probability theory." *American Mathematical Monthly*, vol. 81, no. 9, p. 1041, 1974.

[13] R. Howard, "Data encryption standard," *Information Age archive*, vol. 9, no. 4, pp. 204–210, 1987.

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

[15] R. Beaulieu, D. Shors, J. Smith, S. Treatmanclark, B. Weeks, and L. Wingers, "The simon and speck lightweight block ciphers," *design automation conference*, p. 175, 2015.

Yi Chen received his B.E. degree and M.S. degree both from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology(HUST), Wuhan, China, in 2016, 2018 respectively. He is currently pursuing the Ph.D. degree in the school of computer science and technology at Tsinghua University, Beijing, China. His current research interests include cryptanalysis, multimedia encryption and deep learning.

Yantian Shen is currently pursuing the B.E. degree in the department of computer science and technology at Tsinghua University, Beijing, China. His current research interests include cryptanalysis and deep learning.

Hongbo Yu is currently a Associate Professor and a Ph.D. Supervisor with the Department of Computer Science and Technology, Tsinghua University. She has published many high-quality papers in a series of top conferences and journals such as CRYPTO, EUROCRYPT. Her current research interests include cryptanalysis and design.