Neural Aided Statistical Attack for Cryptanalysis

No Author Given

No Institute Given

Abstract. In Crypto'19, Gohr proposed the first deep learning-based key recovery attack on 11-round Speck32/64, which opens the direction of neural aided cryptanalysis. Until now, neural aided cryptanalysis still faces two problems: (1) the attack complexity estimations rely purely on practical experiments. There is no theoretical framework for estimating theoretical complexity. (2) it does not work when there are not enough neutral bits that exist in the prepended differential. To the best of our knowledge, we are the first to solve these two problems. In this paper, we propose a Neural Aided Statistical Attack (NASA) that has the following advantages: (1) NASA supports estimating the theoretical complexity. (2) NASA does not rely on any special properties including neutral bits. (3) NASA is applicable to large-size ciphers. Moreover, we propose three methods for reducing the attack complexity of NASA. One of the methods is based on a newly proposed concept named Informative Bit that reveals an important phenomenon.
Four attacks on 9-round or 10-round Speck32/64 are executed to verify the correctness of NASA. To further highlight the advantages of NASA, we have performed a series of experiments. At first, we apply NASA and Gohr's attack to round reduced DES. Since NASA does not rely on neutral bits, NASA breaks 10-round DES while Gohr's attack breaks 8-round DES. Then, we compare the time consumption of attacks on 11-round Speck32/64. When the newly proposed three methods are used, the time consumption of NASA is almost the same as that of Gohr's attack. Besides, NASA is applied to 13-round Speck32/64. At last, we introduce how to analyze the resistance of large-size ciphers with respect to NASA, and apply NASA to 14-round Speck96/96. The code of this paper is available at `https://github.com/AI-Lab-Y/NASA`. Our work arguably raises a new direction for neural aided cryptanalysis.

Keywords: Cryptanalysis · deep learning · Informative Bit · Bayesian

1 Introduction

Deep learning has received much expectation in the cryptography community since the last century. Rivest in [17] 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 [14].

Although deep learning has shown its superiorities in various fields such as computer vision [16], natural language processing [3], and smart medical [8], its application in the field of conventional cryptanalysis has been stagnant. A few valuable applications are only concentrated in the side-channel analysis [7, 15].

In Crypto'19, Gohr proposed a deep learning-based distinguisher [13] 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, which shows considerable advantages in terms of attack complexity over the differential attack [5]. In Eurocrypto'20, Benamira et al [2] presented a deeper analysis of $\mathcal{ND}$.

In [13], 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 and the threshold is set without any theoretical basis, the adversary does not foresee the required data complexity to attack a specific cipher. As a result, the estimation of the attack complexity and success rate must rely on practical experiments that are finished within an acceptable runtime. This is unfavorable for evaluating the security of ciphers against machine learning. Besides, when a differential is placed before $\mathcal{ND}$, enough neutral bits [4] must exist in this prepended differential. Otherwise, the attack in [13] does not work.

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

- We propose a Neural Aided Statistical Attack (NASA) which supports theoretical complexity estimation and does not rely on neutral bits. NASA is based on a neural aided statistical distinguisher. And the key recovery is transformed into the distinguishing between two normal distributions, which tells the required data complexity of NASA. Four attacks on 9-round or 10-round Speck32/64 proves the correctness of NASA. Experiments on round reduced DES further prove that NASA has more potential than Gohr's attack when there are not enough neutral bits.
- We propose three methods to reduce the attack complexity of NASA. The first one is reducing the key space by building $\mathcal{ND}$ on partial ciphertext bits. This method comes from the truth that only partial ciphertext bits have a significant influence on $\mathcal{ND}$. We call these bits informative bits and propose a Bit Sensitivity Test to identify them. 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. By building $\mathcal{ND}$ on partial informative bits, 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 for the most promising key guesses 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 be reduced. When these three methods are adopted, the average time consumption of NASA on 11-round Speck32/64 is almost the same as that of Gohr's attack.

– At last, we introduce how to analyze the resistance of large-size ciphers with respect to NASA by applying NASA to 14-round Speck96/96. A practical attack on 10-round Speck96/96 is provided together.

Organization Sections 3, 4 presents the neural aided statistical distinguisher and NASA respectively. The three optimization methods are introduced in sections 5, 6, 7. Applications to DES, Speck32/64, and Speck96/96 are presented in sections 8, 9, 10. At last, we summarize this paper and provide more discussion.

2 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).

2.1 Neutral Bit

Consider a differential $\Delta P \to \Delta S$. Let E denote the encryption function covering the differential. We denote the probability that the following condition holds as the neutrality of the j-th bit

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

where (P_0, P_1) stands for plaintext pairs conforming to the differential. If the neutrality is 1, the j-th bit is called a neutral bit [4].

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.

2.2 Neural Distinguisher

The target of $\mathcal{ND}$ [13] 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. Otherwise, (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. The training pipeline refers to [13].

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 [13]. In this paper, let $\mathcal{ND}_h$ denote an h-round neural distinguisher.

2.3 Gohr's Key Recovery Attack

Algorithm 1 summarizes the core idea of the basic version (unaccelerated version) of Gohr's key recovery attack [13].

Algorithm 1 Basic version of Gohr's key recovery attack

```
Require: $k$ neutral bits that exist in $\Delta P \to \Delta S$; An $\mathcal{ND}$ built over $\Delta S$;
    A key rank score threshold, $c_1$; A maximum number of iterations.
Ensure: A possible key candidate.
 1: repeat
 2:   Random generate a plaintext pair $(P_0^1, P_1^1)|P_0^1 \oplus P_1^1 = \Delta P$;
 3:   Create a plaintext structure consisting of $2^k$ plaintext pairs by $k$ neutral bits;
 4:   Collect corresponding ciphertext pairs, $(C_0^i, C_1^i), i \in \{1, \cdots, 2^k\}$;
 5:   for each key guess $kg$ do
 6:     Partially decrypt $2^k$ ciphertext pairs with $kg$;
 7:     Feed decrypted ciphertext pairs to $\mathcal{ND}$ and collect the outputs;
 8:     Calculate the key rank score $v_{kg}$ based on collected outputs
            $$v_{kg} = \sum_{i=1}^{2^k} \log_2 \left( \frac{Z_i}{1 - Z_i} \right), \tag{3}$$
        where $Z_i$ is the output of $\mathcal{ND}$.
 9:     if $v_{kg} > c_1$ then
10:       stop the key search and return $kg$ as the key candidate;
11:     end if
12:   end for
13: until a key candidate is returned or the maximum number of iterations is reached.
```

The rank score v_{kg} 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.

2.4 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.

3 Neural Aided Statistical Distinguisher

3.1 A Chosen Plaintext Statistical Distinguisher

Consider a cipher E and a differential $\Delta P \xrightarrow{p_0} \Delta S$ where $\Delta P, \Delta S \in \mathbb{F}_2^m$ and 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 the $\mathcal{ND}$ and obtains its output Z_i. Setting a threshold value c_2, the adversary calculates the statistic T

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

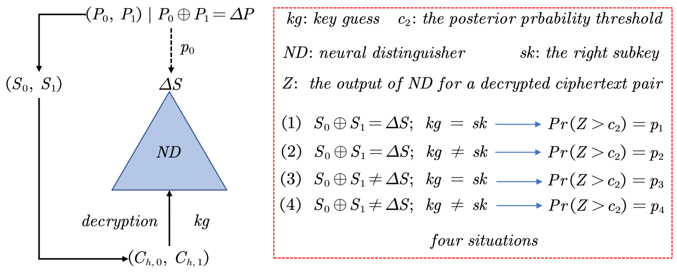
When $p_0 > 2^{-m}$ holds, it's expected that the value of the statistic T for the cipher is higher than that for a random permutation. In a key recovery setting, the right key will result in the statistic T being among the highest values for all candidate keys if N is large enough. Next, we give this a theoretical analysis.

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 c_2, it is equivalent to creating a stable classification hyperplane in this space using an $\mathcal{ND}$. Thus the classification over a 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 kg 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.
- **Decrypting a positive sample with wrong keys:** the ciphertext pair satisfies the differential but the key guess is wrong.



$(P_0, P_1) \mid P_0 \oplus P_1 = \Delta P$
p_0
(S_0, S_1)
ΔS
ND
decryption
kg
$(C_{h,0}, C_{h,1})$
kg: key guess
c_2: the posterior prbability threshold
ND: neural distinguisher
sk: the right subkey
Z: the output of ND for a decrypted ciphertext pair
(1) $S_0 \oplus S_1 = \Delta S$; $kg = sk$ → $Pr(Z > c_2) = p_1$
(2) $S_0 \oplus S_1 = \Delta S$; $kg \neq sk$ → $Pr(Z > c_2) = p_2$
(3) $S_0 \oplus S_1 \neq \Delta S$; $kg = sk$ → $Pr(Z > c_2) = p_3$
(4) $S_0 \oplus S_1 \neq \Delta S$; $kg \neq sk$ → $Pr(Z > c_2) = p_4$
four situations

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

- **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

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

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

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

If the probability of the differential $\Delta P \to \Delta S$ is p_0 and N ciphertext pairs are collected randomly, then

$$N_1 = N_2 = N \times p_0, \quad N_3 = N_4 = N \times (1 - p_0). \tag{8}$$

Besides, the distributions of the statistic (formula 6) 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 6) under the right key guess is:

$$T_r = T_1 + T_3 \sim \mathcal{N}(\mu_r, \sigma_r) \tag{9}$$

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

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

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:

$$T_w = T_2 + T_4 \sim \mathcal{N}(\mu_w, \sigma_w) \tag{12}$$

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

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

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 $\mathcal{ND}$ is higher than 0.5, $p_1 > p_2$ also holds with a high probability. When we set $c_2 = 0.5$, we ensure $p_1 > p_2$. Thus $\mu_r > \mu_w$ also holds.

Since the distributions of T_r, T_w are different, the right key can be recovered based on **Proposition 1**.

3.3 Data Complexity of the Statistical Distinguisher

Based on **Proposition 1**, one obtains the condition:

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

where the values of μ_r, σ_r, μ_w, σ_w refer to formula 10, 11, 13, 14 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 do not know the real classification hyperplane learned by $\mathcal{ND}$, 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 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 15) is simplified as

$$\sqrt{N} = \frac{z_{1-\beta_r} \times \sqrt{p_0 a_1 + (1 - p_0) a_3} + z_{1-\beta_w} \times \sqrt{p_0 a_2 + (1 - p_0) a_3}}{(p_1 - p_2) \times p_0}, \tag{16}$$

where

$$a_1 = p_1(1 - p_1), \quad a_2 = p_2(1 - p_2), \quad a_3 = p_n(1 - p_n). \tag{17}$$

The decision threshold t is:

$$t = \mu_r - z_{1-\beta_r}\sigma_r = \mu_w + z_{1-\beta_w}\sigma_w. \tag{18}$$

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 A.

3.4 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.

3.5 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,

$$\begin{aligned} C^1_{h-1} &= DecOneRound(C_h, kg_1) \\ C^2_{h-1} &= DecOneRound(C_h, kg_2). \end{aligned}$$

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 $F(\cdot)$. If two input samples s_1, s_2 are very close to each other in the input space, two outputs $F(s_1), F(s_2)$ of the neural network may satisfy $F(s_1) \approx F(s_2)$ in high probability.

Although the distance metric in the input space of neural networks is complex and unknown, the Hamming distance is 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

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

where kg_{h+j} is the subkey guess of the $(h+j)$-th round. $(C_{h,0}, C_{h,1})$ is fed into an $\mathcal{ND}$ for estimating the probability of $Z > c_2$.

When the last $x-1$ subkey guesses $kg_{h+j}, j \in [2, \cdots, x]$ are all right, $(C_{h+1,0}, C_{h+1,1})$ is still a positive sample. Then the final probability of $Z > c_2$ would be high if kg_{h+1} is different from the right subkey at few bits. However, if $kg_{h+j}, j \in \{2, \cdots, x\}$ are not all right, $(C_{h+1,0}, C_{h+1,1})$ is not a positive sample anymore. Then the final probability of $Z > c_2$ is closer to p_n.

Thus, we consider x Hamming distances for estimating p_2 at first. Let d_j denotes the Hamming distance between the right subkey and subkey guess in the

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

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

$(h + j)$-th round, and $p_{2|d_1,\cdots,d_x}$ denotes the probability of $Z > c_2$. Algorithm 2 is proposed to estimate $p_{2|d_1,\cdots,d_x}$.

Verification. Gohr provided $\mathcal{ND}_5, \mathcal{ND}_6, \mathcal{ND}_7, \mathcal{ND}_8$ against Speck32/64 [13], which are built over a plaintext difference $(0x0040, 0)$. We have performed tests on these four distinguishers. Let $M = 10^7$, Table 1 and Table 2 show the estimation results of $p_{2|d_1}$ and $p_{2|d_1,d_2}$ respectively.

Table 1. 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 $\mathcal{ND}_8$, $c_2 = 0.5$. $p_{2|d_1=0} = p_1$. Only four decimal places are presented in this paper. Actually, we kept more decimal places during follow-up experiments.

$\mathcal{ND}_5$	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	$8 \sim 16$
	$p_{2\|d_1}$	0.6784	0.4430	0.3135	0.2394	0.1958	0.1691	0.1522	0.1410	$\leqslant 0.1336$
$\mathcal{ND}_7$	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	$8 \sim 16$
	$p_{2\|d_1}$	0.5183	0.5056	0.4993	0.4958	0.4939	0.4927	0.4925	0.4918	$\leqslant 0.4917$

The test results have verified the analysis of p_2. When two subkeys are guessed simultaneously, $p_{2|d_1,d_2}$ decreases sharply even if the subkey guess of the last round is wrong at only 1 bit.

Table 2. 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
$9 \sim 16$	$\leqslant Y$																

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 subkey guess and the right subkey does not exceed a threshold d, the value of p_2 should be

$$p_2 = \max\left\{p_{2|d_1,\cdots,d_x} | d_1 + \cdots + d_x > d\right\} \tag{19}$$

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

$$p_{2|d_1,\cdots,d_x} \leqslant 0.5, \quad if \quad d_1 + \cdots + d_x > d. \tag{20}$$

According to formula 16, 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 subkey is different from the right subkey 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

4.1 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 Statistical Attack (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, $\beta_r, \beta_w$;
    A posterior probability threshold, $c_2$.
Ensure: All possible key candidates.
1: Train an $\mathcal{ND}$ over $\Delta S$;
2: Estimate $p_1, p_n, p_2$ using $\mathcal{ND}$ (Section 3.4, Algorithm 2);
3: Calculate the data complexity $N$ and the decision threshold $t$ (Section 3.3);
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.
```

4.2 Verification of the Key Recovery Attack Model

Four practical attacks on h-round Speck32/64 are performed to verify NASA. The target is to recover the last subkey sk_h. It's expected that returned subkey guesses are different from sk_h 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.

Four distinguishers $\mathcal{ND}_5, \mathcal{ND}_6, \mathcal{ND}_7, \mathcal{ND}_8$ provided by Gohr [13] are adopted. Table 3 shows two different differentials 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 (formula 16).
2. Perform NASA 100 times with N samples.
3. Check the following observation indexes:
   (a) The ratio that the right subkey ($d_1 = 0$) survives.

Table 3. two options of the prepended differential of Speck32/64. nr is the number of encryption rounds covered by the differential.

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

(b) The average number of surviving subkey guesses in 100 trails.
(c) The ratio that the number of surviving subkeys does not exceed the expected upper bound.

Table 4 summarizes the settings related to four attacks. Table 1 shows the estimations of $p_{2|d_1}$ related to $\mathcal{ND}_5, \mathcal{ND}_6, \mathcal{ND}_7, \mathcal{ND}_8$. The value of p_2 is $p_{2|d_1=3}$ in four attacks.

Table 4. Settings of the four attacks on round reduced Speck32/64. DID is the differential's ID in Table 3.

Attack ID	Attack rounds	$\mathcal{ND}$	DID	p_0	c_2	p_1	d	p_2	p_n	β_r	β_w
1	9	$\mathcal{ND}_5$	2	2^{-6}	0.55	0.8889	2	0.2168	0.0384	0.005	2^{-16}
2	9	$\mathcal{ND}_6$	1	2^{-2}	0.55	0.6784	2	0.2394	0.1162	0.005	2^{-16}
3	10	$\mathcal{ND}_7$	1	2^{-2}	0.55	0.4183	2	0.2607	0.2163	0.005	2^{-16}
4	10	$\mathcal{ND}_8$	-	1	0.5	0.5183	2	0.4958	0.4914	0.001	2^{-16}

Attack 1: recover sk_9 of 9-round Speck32/64 In the first attack setting, we get $N = 15905 \approx 2^{13.957}$ (see formula 16). The decision threshold is $t = 758$. The right subkey ($d_1 = 0$) should survive with a $1 - \beta_r = 0.995$ probability at least. Wrong subkey guesses ($d_1 \geqslant 3$) should survive with a $\beta_w = 2^{-16}$ probability at most. The number of surviving subkey guesses should not exceed $137 + (2^{16} - 137) \times 2^{-16} = 137.998$.

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 subkey guesses is 18.41.
- The number of surviving subkey guesses does not exceed 137.998 in 100 experiments.

Attack 2: recover sk_9 of 9-round Speck32/64 In the second attack setting, $N = 475 \approx 2^{8.893}$ and $t = 101$. The number of surviving subkey guesses should not exceed $137 + (2^{16} - 137) \times 2^{-16} \approx 137.998$.

After performing this attack 100 times, we find:

- The right subkey has survived in all the 100 experiments.
- The average number of surviving subkey guesses is 33.43.
- The number of surviving subkey guesses does not exceed 137.998 in 100 experiments.

Attack 3: recover sk_{10} of 10-round Speck32/64 In the third attack setting, $N = 5272 \approx 2^{12.364}$ and $t = 1325$. The number of surviving subkey guesses should not exceed $137 + (2^{16} - 137) \times 2^{-16} \approx 137.998$.

After performing this attack 100 times, we find:

- The right subkey ($d_1 = 0$) has survived in 99 experiments.
- The average number of surviving subkey guesses is 63.54.
- The number of surviving subkey guesses does not exceed 137.998 in 98 experiments.

Attack 4: 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 fourth attack setting, $N = 25680 \approx 2^{14.65}$ and $t = 13064$. The number of surviving subkey guesses should not exceed $137 + (2^{16} - 137) \times 2^{-16} \approx 137.998$.

After performing this attack 100 times, we find:

- The right subkey ($d_1 = 0$) has survived in all the 100 experiments.
- The average number of surviving subkey guesses is 77.47.
- The number of surviving subkey guesses does not exceed 137.998 in 85 experiments. In the other 15 experiments, the ratio that subkey guesses with $d_1 = 3$ survive is a little higher than that in the 85 experiments.

It's clear that these four attacks [1] have achieved the most important two targets of NASA. This proves the Hamming distance is a good distance metric for estimating p_2. The correctness of NASA is also well verified.

5 Reduce the Key Space

So far we 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.

5.1 An Intuitive Method for Reducing the Key Space

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

$$C_i = C_i[L-1]||\cdots||C_i[0], i \in [0,1] \tag{21}$$

$$\Gamma = \{x_1, x_2, \cdots, x_k\}, x_1 > \cdots > x_k, k <= L \tag{22}$$

$$\varphi(C_i, \Gamma) = C_i[x_1]||C_i[x_2]||C_i[x_k], i \in [0,1] \tag{23}$$

[1] It takes about 10 hours to 2 days to execute any one of the four attacks 100 times when a graphics card is used. To facilitate readers to perform the experiment, we provide accelerated versions, which are completed within 1 hour to 4 hours.

$$Y(\varphi(C_0,\varGamma),\varphi(C_1,\varGamma)) = \begin{cases} 1, & if\ S_0 \oplus S_1 = \varDelta S \\ 0, & if\ S_0 \oplus S_1 \neq \varDelta S \end{cases} \tag{24}$$

$$Pr\{Y = 1\,|(\varphi(C_0,\varGamma),\varphi(C_1,\varGamma))\,\} = \mathcal{ND}(\varphi(C_0,\varGamma),\varphi(C_1,\varGamma)) \tag{25}$$

where $C_i[0]$ is the least significant bit of the ciphertext C_i, $\varGamma$ 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*** $\mathcal{ND}^t_h$*, the neural distinguisher trained on partial ciphertext bits* $(\varphi(C_0,\varGamma),\varphi(C_1,\varGamma))$ *is denoted as the* ***student distinguisher*** $\mathcal{ND}^s_h$*. The teacher distinguisher is 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 an* $\mathcal{ND}^t$*, if the distinguishing accuracy is significantly 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 does not ensure each informative bit that obeys **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 [13] 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

$$C_0 = C_0 \oplus (\eta \ll j) \quad or \quad C_1 = C_1 \oplus (\eta \ll j) \tag{26}$$

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

We have performed an extreme test on $\mathcal{ND}_h^t, h \in \{5, 6, 7, 8\}$ against Speck32/64. If we XOR each bit of C_0 or C_1 with a random mask, $\mathcal{ND}_h^t, h \in \{5, 6, 7, 8\}$ 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 26.

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.

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.
 1: Test the distinguishing accuracy of $\mathcal{ND}^t$ on the test dataset. Save it to $sen[m]$;
 2: for $j = 0$ to $m - 1$ do
 3:     for $i = 1$ to $M$ do
 4:         Generate a random mask $\eta \in \{0, 1\}$;
 5:         $C_0^{i,new} = C_0^i \oplus (\eta \ll j)$;
 6:         Feed the new sample $(C_0^{i,new}, C_1^i)$ to $\mathcal{ND}^t$;
 7:     end for
 8:     Count the current accuracy $cp$;
 9:     $sen[j] = sen[m] - cp$;
10: end for
```

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

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 $\mathcal{ND}_7^t$ against Speck32/64 as an example, we have performed the reverse verification based on results in Fig. 3(b). Table 5 shows the distinguishing accuracies under two settings. For Speck32/64, the j-th and $(j + 16)$-th bit are

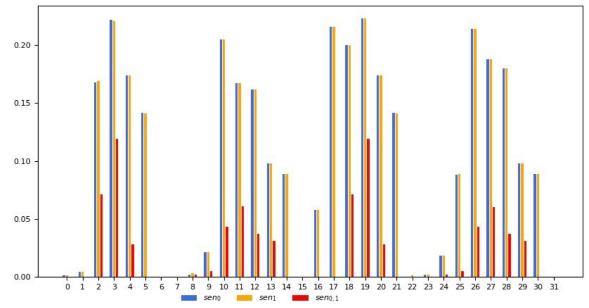


Fig. 2. Results of BST of $\mathcal{ND}_5^t$ against Speck32/64, $M = 10^6$. sen_0 is the results of performing $C_0 \oplus (\eta \lll j)$, sen_1 is the results of performing $C_1 \oplus (\eta \lll j)$, $sen_{0,1}$ is the results of performing two operations simultaneously, $j \in \{0, \cdots, 31\}$. Only three decimal places are kept.

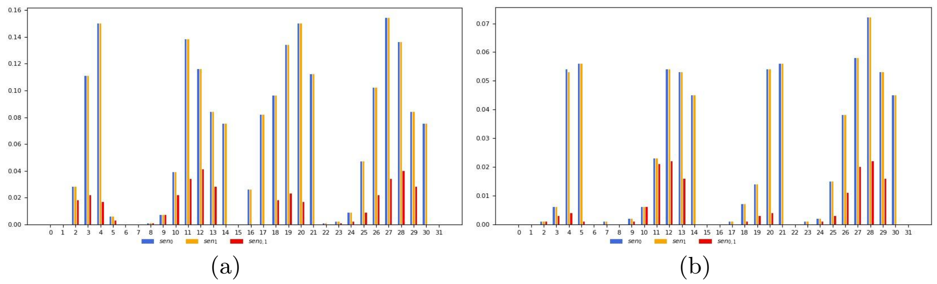


Fig. 3. Results of BST of $\mathcal{ND}_6^t$ (a) and $\mathcal{ND}_7^t$ (b) against Speck32/64, $M = 10^6$.

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

Table 5. 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.

6 Selective Key Search

In Algorithm 4, each possible key guess kg is tested. Inspired by the analysis of p_2 in Section 3.5, 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 key guesses that have been tested.

6.1 Distribution of the Statistic Under Different Keys

We discuss the distribution of the statistic (formula 6) under different keys again. We still take Fig. 1 as an example.

Suppose that the size of the key guess kg is m. According to the analysis in Section 3.5, we know there are the following $m + 1$ probabilities

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

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

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

$$T_{d_1} \sim \mathcal{N}(\mu_{d_1}, \sigma_{d_1}), \tag{27}$$

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

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

The parameters of these $m + 1$ distributions are obtained offline. These distributions are used as prior knowledge to develop a Bayesian key search strategy.

6.2 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.

7 Reduce the Data Complexity

Consider the prepended differential $\varDelta P \xrightarrow{p_0} \varDelta S$. As we have presented in section 3.3, 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.

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 $\mu_{d_1}$ 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 key guesses and statistics.
 1: $\mathcal{K} = \{kg_1, \cdots, kg_{cand}\} \leftarrow$ choose $n_{cand}$ values at random without replacement from
    the set of all subkey candidates.
 2: $L \leftarrow \{\}$
 3: for $t = 1$ to $n_{iter}$ do
 4:     for each $kg \in \mathcal{K}$ do
 5:         for $i = 1$ to $N$ do
 6:             Decrypt $C_0^i, C_1^i$ with $kg$.
 7:             Feed partially decrypted ciphertext pair into $\mathcal{ND}$.
 8:             Collect the output $Z_{i,kg}$ of $\mathcal{ND}$.
 9:         end for
10:         Compute the statistic (formula 6), $T^{kg}$ ;
11:         $L \leftarrow L||(kg, T^{kg})$.
12:     end for
13:     for $sk \in \{0, \cdots, 2^m - 1\}$ do
14:         $\lambda_{sk} = \sum_{kg \in \mathcal{K}} \left(T^{kg} - \mu_{hd(kg \oplus sk)}\right)^2 / \sigma^2_{hd(kg \oplus sk)}$.
            /* $hd(kg \oplus sk)$ is the Hamming distance between $kg$ and $sk$ */
15:     end for
16:     $\mathcal{K} \leftarrow argsort_{sk}(\lambda)[0 : n_{cand} - 1]$.
        /* Pick $n_{cand}$ key guesses with the $n_{cand}$ smallest score to form the new set of
        key guess candidates $\mathcal{K}$ */
17: end for
18: Return $L$
```

7.1 Improved Neural Aided Statistical Attack

We still take the key recovery attack with 1-round decryption as an example to introduce the improved NASA.

Its core idea is to divide the 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 $\Delta B \to \Delta S$. Neutral bits that exist in the first part $\Delta P \to \Delta B$ are exploited. Algorithm 7 summarizes the details of the improved NASA.

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}.

7.2 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 $\Delta P \to \Delta B$.

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 $\Delta P \to \Delta B$;
    Two maximum error probabilities, $\beta_r, \beta_w$;
    A posterior probability threshold, $c_2$.
Ensure: All possible key candidates.
1: Train an $\mathcal{ND}$ over $\Delta S$;
2: Estimate $p_1, p_n, p_2$ using $\mathcal{ND}$ (Section 3.4, Algorithm 2);
3: Calculate the data complexity $N_1$ based on $p, p_1, p_n, p_2$ (Section 3.3);
4: for $j$ from 1 to $\frac{1}{q}$ do
5:     Based on $\Delta P$ 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.
```

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

$$T = \sum_{i=1}^{M} \phi(Z_i), \quad \phi(Z_i) = \begin{cases} 1, if\ Z_i > c_2 \\ 0, if\ Z_i \leqslant c_2 \end{cases}. \tag{28}$$

The four situations as shown in Fig. 1 also exist in this identification process. The following notations 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 $d_1 \in \{0, \cdots, m\}$.
- p_n : the probability $Pr\{Z > c_2 | S_0 \oplus S_1 \neq \Delta S\}$.

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

At first, we do not set d_1 clearly and denote $p_{2|d_1}$ as p_V. The distribution of the statistic(formula 28) is

$$T_V \sim \mathcal{N}(\mu_V, \sigma_V), \tag{29}$$

$$\mu_V = M[p \times p_V + (1-p)p_n], \tag{30}$$

$$\sigma_V = \sqrt{M \times p \times p_V(1-p_V) + M(1-p)p_n(1-p_n)}. \tag{31}$$

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

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 28) is

$$T_I \sim \mathcal{N}(\mu_I, \sigma_I) \tag{32}$$

$$\mu_I = M \times p_n, \quad \sigma_I = \sqrt{M \times p_n(1-p_n)} \tag{33}$$

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

Distinguishing between $\boldsymbol{T_V}$ and $\boldsymbol{T_I}$ Since T_V and T_I are two different normal distributions, the technique in Section 2.4 is used to distinguish these two distributions. According to **Proposition 1**, the condition for distinguishing T_V and T_I is

$$\frac{z_{1-\beta_V} \times \sigma_V + z_{1-\beta_I} \times \sigma_I}{\mu_V - \mu_I} = 1 \tag{34}$$

where

$$\begin{aligned} \beta_V &= Pr\left\{\mathcal{N}\left(\mu_I, \sigma_I\right) | s \in \mathcal{N}\left(\mu_V, \sigma_V\right)\right\}, \\ \beta_I &= Pr\left\{\mathcal{N}\left(\mu_V, \sigma_V\right) | s \in \mathcal{N}\left(\mu_I, \sigma_I\right)\right\}, \end{aligned} \tag{35}$$

and s stands for a sample.

We present a deeper 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 $kg \in \mathcal{K_V}$ survive the attack is $1 - \beta_V$.

By simplifying formula 34, we know that the required data complexity M is

$$a_V = p_V(1-p_V), \quad a_n = p_n(1-p_n), \tag{36}$$

$$\sqrt{M} = \frac{z_{1-\beta_V} \times \sqrt{p \times a_V + (1-p)a_n} + z_{1-\beta_I} \times \sqrt{a_n}}{(p_V - p_n) \times p}. \tag{37}$$

And the decision threshold t_M is

$$t_M = \mu_V - z_{1-\beta_V}\sigma_V = \mu_I + z_{1-\beta_I}\sigma_I, \tag{38}$$

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

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

$$|\mathcal{K}_{\mathcal{R}}| \times (1-\beta_V) \gg |\mathcal{K}| \times \beta_I. \tag{39}$$

Let t_P satisfy the following condition

$$|\mathcal{K}_{\mathcal{R}}| \times (1-\beta_V) \geqslant t_P \gg |\mathcal{K}| \times \beta_I, \tag{40}$$

where $x \gg y$ means that x is much larger than y here. Then valid plaintext structures is identified by by comparing the number of surviving subkey guesses with t_P [2].

Algorithm 8 Identify valid plaintext structures

```
Require: a plaintext structure $\mathcal{P}$ with a size of $M$(formula 37);
    an $\mathcal{ND}$ trained over $\Delta S$;
    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 $\mathcal{P}$.
 1: Collect the $M$ ciphertext pairs corresponding to $\mathcal{P}$;
 2: Initialize a counter $cp \leftarrow 0$;
 3: for each possible subkey guess $kg$ do
 4:     Decrypt $M$ ciphertext pairs with $kg$;
 5:     Feed partially decrypted ciphertext pairs into $\mathcal{ND}$;
 6:     Save the outputs of $\mathcal{ND}$, $Z_i, i \in [1, M]$;
 7:     Count the number of $Z_i > c$, and denote it as $T_M$;
 8:     if $T_M > t_M$ then
 9:         $cp \leftarrow cp + 1$;
10:     end if
11: end for
12: if $cp \geqslant t_P$ then
13:     Return 1 ($\mathcal{P}$ is a valid plaintext structure).
14: else
15:     Return 0 ($\mathcal{P}$ is an invalid plaintext structure).
16: end if
```

Algorithm 8 summarizes the concrete identification process. Since the identification is 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.

[2] We provide the code to verify Algorithm 8. The identification success rate over valid plaintext structures is 100%.

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 37) decreases since

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

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

When the Hamming distance d_1 decreases, $p_{2|d_1}$ will increase in high probability. But the number of subkey guesses in the subspace may decrease sharply when d_1 decreases, which may make the condition (formula 39) not hold. Thus, there is a trade-off. As long as the condition (formula 39) holds, we advise $p_V = p_{2|d_1}$ where d_1 should be as small as possible.

8 Application to DES

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

DES [9] is a block cipher with a block size of 64 bits. The structure of DES is the classical Feistel structure. Its round function f is given by eight different S-boxes. More details refer to [9], please. We perform key recovery attacks on round reduced DES.

8.1 Prepended Differentials

Two optimal 2-round iterative differentials found in [6] are

$$\begin{aligned} 0x19600000/0 &\xrightarrow{Pr=\frac{1}{234}} 0x19600000/0, \\ 0x1B600000/0 &\xrightarrow{Pr=\frac{1}{234}} 0x1B600000/0. \end{aligned} \tag{41}$$

Based on these 2-round differentials, we can get longer iterative differentials. 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.

8.2 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.

Based on the BST, we find that 4 bits $\{39, 50, 56, 61\}$ related to the fifth S-box $S5$ and 4 bits $\{59, 37, 43, 49\}$ related to the eighth S-box $S8$ are all informative bits.

In order to introduce the next experiment more clearly, we focus on the student distinguisher $\mathcal{ND}_5^s$ built over the bits $\{39, 50, 56, 61\}$ for now. Let the posterior probability threshold be $c_2 = 0.5$, we get $p_1 = 0.6041$ and $p_n = 0.4890$. Table 6 shows the estimation of $p_{2|d_1}$.

Table 6. The estimation of $p_{2|d_1}$ of $\mathcal{ND}_5^s$ against round reduced DES. $c_2 = 0.5$. $\mathcal{ND}_5^s$ is built over 4 bits $\{39, 50, 56, 61\}$.

d_1	0	1	2	3	4	5	6
$p_{2\|d_1}$	0.6041	0.5042	0.5065	0.5083	0.5113	0.5043	0.4957

8.3 Attack DES with Gohr's Attack

By placing the 2-round differential $0x19600000/0 \xrightarrow{234^{-1}} 0x19600000/0$ before $\mathcal{ND}_5^s$, 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 differential $0x19600000/0 \xrightarrow{234^{-2}} 0x19600000/0$, **10-round DES can not be broken by Gohr's attack under the current setting.**

8.4 Neural Aided Statistical Attack on DES

Consider the basic NASA (Algorithm 4) on 10-round DES [3]. We also adopt the 4-round prepended differential $0x19600000/0 \xrightarrow{234^{-2}} 0x19600000/0$ and $\mathcal{ND}_5^s$.

Let $d = 0$, we have $p_2 = 0.5113$ based on Table 6. Besides, let $\beta_r = 0.005$, and $\beta_w = 2^{-6}$. Since $p_0 = 234^{-2}$, the required data complexity is $N = 2^{40.824}$ chosen plaintext pairs.

Since the output of an S-box only contains 4 bits, we build a look-up table offline 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 this attack is $N \times 2 \times 2^6 = 2^{47.824}$. Thus, **10-round DES is broken by the basic NASA under the same setting.**

[3] A practical attack on 6-round DES is first executed to confirm that the Hamming distance is a good distance metric for $\mathcal{ND}_5^s$ and NASA is applicable.

9 Application to Speck32/64

Consider the prepended differential $\varDelta P \xrightarrow{p_0} \varDelta S$. For Gohr's attack, the impact of p_0 on the data complexity is only $\mathcal{O}(p_0^{-1})$. For the 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 competitive performance when the three optimization techniques introduced in sections 5, 6, 7 are available. Speck32/64 is one variant of the Speck family that is designed by the NSA Research Directorate [1]. Its round function is

$$\begin{aligned} C_L &= (C_L \lll 7) \boxplus C_R, \\ C_L &= C_L \oplus sk, \\ C_R &= (C_R \ggg 2) \oplus C_L. \end{aligned}$$

where $C_L||C_R$ is the input / output, and sk is the subkey.

9.1 Prepended Differential

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

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

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

Dividing this differential into two 1-round differentials

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

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

9.2 Build Neural Distinguishers Against Speck32/64

Two teacher distinguishers $\mathcal{ND}_6^t, \mathcal{ND}_7^t$ built by Gohr over $\varDelta S = (0x40, 0)$ are adopted in this section. Table 1 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 BST results of $\mathcal{ND}_6^t, \mathcal{ND}_7^t$ as shown in Fig. 3, we build 2 student distinguishers $\mathcal{ND}_6^s, \mathcal{ND}_7^s$ by setting $\varGamma = \{30 \sim 23, 14 \sim 7\}$. 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 7 shows the estimation of $p_{2|d_1}$ of $\mathcal{ND}_6^s, \mathcal{ND}_7^s$. Table 8 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 7. 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	0.3232	0.3036	$\leqslant 0.2940$

Table 8. 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	0.5132	0.3575	0.6784	0.4183
p_n	0.2604	0.2863	0.1162	0.2163

9.3 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 (Algorithm 1) on 11-round Speck32/64. Besides, Gohr provided some optimization techniques for accelerating it.

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 about **70** seconds.

9.4 Neural Aided Statistical Attack on Speck32/64

At first, we consider the neural aided statistical attack on 11-round Speck32/64. At a high level, the attack contains 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 $8 \times (1 - 0.1) = 7.2$ when $\mathcal{P}$ is a valid plaintext structure. Otherwise, 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 $M = 37938 \approx 2^{15.211}$ plaintext pairs. The decision threshold for filtering subkey guesses is $t_M = 11103$. When the number of surviving subkey guess exceeds 7 ($t_P = 8$), $\mathcal{P}$ is a valid plaintext structure.

In stage 2, we set $p_2 = p_{2|d_1=3}, \beta_r = 0.005, \beta_w = 2^{-8}$. The required data complexity is $N = 22586 \approx 2^{14.463}$ plaintext pairs, and the decision threshold for filtering subkey guess is $t = 6690$.

In stage 3, we filter kg_{11} based on each surviving $kg_{11}[7 \sim 0]$. Let $p_2 = p_{2|d_1=3}, \beta_r = 0.005, \beta_w = 2^{-16}$, we have $N = 5271 \approx 2^{12.364}, t = 1325$.

In stage 4, we filter $(kg_{10}[7 \sim 0], kg_{11})$ based on each surviving kg_{11}. Let $p_2 = p_{2|d_1=2}, \beta_r = 0.001, \beta_w = 2^{-14}$, we have $N = 5228 \approx 2^{12.36}, t = 1589$.

In stage 5, we filter (kg_{10}, kg_{11}) based on each surviving $(kg_{10}[7 \sim 0], kg_{11})$. Let $p_2 = p_{2|d_1=2}, \beta_r = 0.001, \beta_w = 2^{-16}$, we have $N = 829 \approx 2^{9.697}, t = 180$.

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 $M = 37938$ 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 24 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 $\mathcal{P}$. Moreover, $22586, 5271, 5228, 829$ plaintext pairs are selected from this valid plaintext structure respectively. In stage $1 \sim 5$, the proposed Bayesian key search strategy (Algorithm 6) is applied in each stage.

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 $n_{iter} = 3, n_{cand} = 32$.

We count a key guess as successful if the right subkey pair (sk_{10}, sk_{11}) survives. We have performed 500 experiments under the same hardware environment used in section 9.3, . Valid plaintext structures occurred in 339 experiments and were all identified successfully. The attack was successful in 265 out of 339 trials. Besides, the attack was successful in 4 out of the remaining 161 trials. We find that the invalid plaintext structure in the 4 trials contains many plaintext pairs that pass the differential $\Delta P \rightarrow \Delta B$. The average numbers of surviving key guesses in the last four stages were $8.2, 32.9, 14.8, 11.1$ respectively. The average number of generated plaintext structures is 10.73. The average time consumption of this attack is about **77.9** seconds, which is very close to the time consumption of Gohr's attack.

According to the attack settings in stages $2 \sim 5$, the probability that the right keys survive should be $(1-0.005)^2 \times (1-0.001)^2 \approx 0.988$. We argue that the reason why the attack failed in 74 out of 339 experiments is that the sampling randomness is destroyed by neutral bits. To verify this argument, we randomly generate 22586 plaintext pairs with a difference ΔB to form a plaintext structure and have performed the attack 100 times again. The attack was successful in 99 out of 100 trials.

We wonder how many rounds NASA could attack at most. By adopting $\mathcal{ND}_8^t$ (let $p_2 = p_{2|d_1=3}$) and 19 neutral bits, we guess and recover sk_{13}, sk_{12} simultaneously. The theoretical data complexity is about $2^{22.73}$ chosen-plaintext pairs, which is lower than the data complexity (2^{25} chosen plaintexts) of the previous best attack [10] on 13-round Speck32/64. For NASA, the basic operation contains two steps: (1) partially decrypt a ciphertext pair with a subkey guess, (2) feed the decrypted ciphertext pair into $\mathcal{ND}$ and obtain the output. Under the hardware environment used in section 9.3, it takes about 2.8 seconds to perform 2^{20} basic operations with $\mathcal{ND}_8^t$. As a comparison, it takes about 0.28 seconds to process 2^{20} key guesses by force key search. Thus, the theoretical time complexity is about $\delta \times 2^{22.73+32}$ where $\delta = \frac{2.8}{0.28} = 10$, and 13-round Speck32/64 is broken by NASA [4].

10 Application to Speck96/96

NASA can be used to analyze the resistance of large-size ciphers with respect to deep learning. We introduce the general idea by adopting the application to Speck96/96 [1] as an example.

First, we train a student distinguisher $\mathcal{ND}_7^s$ over $\Delta S = (0x80, 0)$ by setting $\Gamma = \{69 \sim 56, 21 \sim 8\}$. Second, a practical attack [5] based on $\mathcal{ND}_7^s$ is performed to confirm that NASA is applicable. More precisely, the practical attack is used to verify whether the Hamming distance is a good distance metric for estimating p_2 related to $\mathcal{ND}_7^s$. Third, estimate the theoretical complexity of NASA on round reduced Speck96/96.

By placing a prepended 6-round differential extended from the 5-round differential $\Delta P = (0x900900480001, 0x11003084008) \xrightarrow{2^{-32}} \Delta S = (0x80, 0)$, we find that the theoretical data complexity for recovering 14 subkey bits $sk_{14}[13 \sim 0]$ of 14-round Speck96/96 is $2^{70.22}$ chosen plaintext pairs. Thus, the theoretical time complexity is $\delta \times 2^{70.22+14}$ where $\delta \approx 4.85$ under the hardware environment used in section 9.3.

11 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, Speck32/64, and Speck96/96 prove the superiorities of NASA.

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. 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

[4] The code of NASA on 12-round Speck32/64 is provided. The average runtime is under two hours on a GeForce GTX 1080 Ti GPU.

[5] The code of practical NASA on 10-round Speck96/96 is provided.

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. Fourth, if a distance metric is better than the Hamming distance, NASA would give more accurate estimations. At last, the negative influence of neutral bits needs to be further explored.

A Analysis of the Data Complexity of Basic NASA

The data complexity of NASA is related to $\mathcal{ND}$ and the prepended differential. 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 16, the data complexity N is affected by the probability p_0 of the prepended differential as

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

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

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

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 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 discuss p_n separately.

The impact of p_n In formula 16, only a_3 is related to p_n.

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

Next, we focus on attack scenarios where the p_0 is low. This makes 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 low.

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 16, 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 $p_0 \to 0$.

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

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

References

1. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015. pp. 175:1–175:6. ACM (2015)
2. Benamira, A., Gérault, D., Peyrin, T., Tan, Q.Q.: A deeper look at machine learning-based cryptanalysis. In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12696, pp. 805–835. Springer (2021)
3. Bengio, Y., Ducharme, R., Vincent, P.: A neural probabilistic language model. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA. pp. 932–938. MIT Press (2000)
4. Biham, E., Chen, R.: Near-collisions of SHA-0. In: Franklin, M.K. (ed.) Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3152, pp. 290–305. Springer (2004)

5. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1990)
6. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. J. Cryptol. **4**(1), 3–72 (1991)
7. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without preprocessing. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 45–68. Springer (2017)
8. Chen, Y., Yu, L., Ota, K., Dong, M.: Robust activity recognition for aging society. IEEE J. Biomed. Health Informatics **22**(6), 1754–1764 (2018)
9. Davies, D.W.: Some regular properties of the 'data encryption standard' algorithm. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982. pp. 89–96. Plenum Press, New York (1982)
10. Dinur, I.: Improved differential cryptanalysis of round-reduced speck. In: Joux, A., Youssef, A.M. (eds.) Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8781, pp. 147–164. Springer (2014)
11. Feller, W.: An introduction to probability theory and its applications. vol. ii. Population **23**(2), 375 (1968)
12. Gisselquist, R., Hoel, P.G., Port, S.C., Stone, C.J.: Introduction to probability theory. American Mathematical Monthly **81**(9), 1041 (1974)
13. Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11693, pp. 150–179. Springer (2019)
14. Greydanus, S.: Learning the enigma with recurrent neural networks. CoRR **abs/1708.07576** (2017)
15. 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. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(3), 148–179 (2019)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Bartlett, P.L., Pereira, F.C.N., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. pp. 1106–1114 (2012)
17. Rivest, R.L.: Cryptography and machine learning. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.) Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings. Lecture Notes in Computer Science, vol. 739, pp. 427–439. Springer (1991)