applied sciences

MDPI

Article

Improved Neural Differential Distinguisher Model for Lightweight Cipher Speck

Xiaoteng Yue [1,2,*,†] and Wanqing Wu [1,2,†]

[1] School of Cyber Security and Computer, Hebei University, Baoding 071002, China; wuwanqing8888@126.com
[2] Key Laboratory on High Trusted Information System in Hebei Province, Baoding 071002, China
* Correspondence: 15028874462@163.com
† These authors contributed equally to this work.

Abstract: At CRYPTO 2019, Gohr proposed the neural differential distinguisher using the residual network structure in convolutional neural networks on round-reduced Speck32/64. In this paper, we construct a 7-round differential neural distinguisher for Speck32/64, which results in better than Gohr's work. The details are as follows. Firstly, a new data format $(C\_r, C\_r', d\_l, C_l, C_r, C_l', C_r')$ is proposed for the input data of the differential neural distinguisher, which can help the distinguisher to identify the features of the previous round of ciphertexts in the Speck algorithm. Secondly, this paper modifies the convolution layer of the residual block in the residual network, inspired by the Inception module in GoogLeNet. For Speck32/64, the experiments show that the accuracy of the 7-round differential neural distinguisher is 97.13%, which is better than the accuracy of Gohr's distinguisher of 9.1% and also higher than the currently known accuracy of 89.63%. The experiments also show that the data format and neural network in this paper can improve the accuracy of the distinguisher by 2.38% and 2.1%, respectively. Finally, to demonstrate the effectiveness of the distinguisher in this paper, a key recovery attack is performed on 8-rounds of Speck32/64. The results show that the success rate of recovering the correct key is 92%, with no more than two incorrect bits. Finally, this paper briefly discussed the effect of the number of ciphertext pairs in a sample on the training results of the differential neural distinguisher. When the total number of ciphertext pairs is kept constant, the accuracy of the distinguisher increases with s, but it also leads to the occurrence of overfitting.

Keywords: deep learning; differential cryptanalysis; Speck; key rcovery attack

Citation: Yue, X.; Wu, W. Improved Neural Differential Distinguisher Model for Lightweight Cipher Speck. *Appl. Sci.* **2023**, *13*, 6994. https://doi.org/10.3390/app13126994

Academic Editor: Xianpeng Wang

Received: 28 April 2023
Revised: 25 May 2023
Accepted: 8 June 2023
Published: 9 June 2023

1. Introduction

With the rapid development of computer networks [1,2] and Internet of Things [3] (IoT) technology, IoT devices have been applied in many fields and have achieved constructive results. However, in the production of IoT devices, storage and computing resources are compressed to improve their productivity and convenience, which makes traditional algorithms such as DES and AES ineffective in IoT devices and thus reduces the security of the devices.

For this problem, the National Security Agency (NSA) [4] designed the lightweight block cipher Speck, which offers better performance on both hardware and software platforms compared with other existing ciphers. However, the designers of Speck neither provided the design rationale nor gave any security evaluation or cryptanalysis results.

This has spurred further research on Speck in the cryptographic community to deepen the understanding and refine it, such as an ultra-lightweight cryptographic Speck-R proposed by Sleem and Couturier [5] based on Speck. Furthermore, among this cryptanalytic research, differential cryptanalysis is the most promising attack.

Differential cryptanalysis was proposed by Biham and Shamir [6] in 1990 for breaking Data Encryption Standard [7] (DES) block ciphers, and today it is considered one of the most robust techniques in cryptanalysis of symmetric key cryptographic primitives. Abed et al. [8] introduced the first differential attack for almost all Speck variants.

Biryukov et al. [9] searched for better differential features of Speck. In [10], Dinur improved the key recovery attack for all variants of Speck and Biryukov [11] showed an automatic algorithm for searching the best differential trajectory with improved MILP-based differential features results.

The development of deep learning has brought some new minds to cryptanalysis. In recent years, deep learning has spread across almost every field of science and technology (medical [12], agriculture [13], etc.) and has made remarkable progress on many difficult tasks. Several researchers have begun to apply deep learning to the study of cryptanalysis of block ciphers.

At CRYPTO 2019, Gohr [14] successfully trained (5∼8)-round of differential neural distinguisher by using the residual network (ResNet) [15] to create a precedent for neural-aided cryptanalysis. It was used to capture the distribution of output pairs when the input pairs of round-reduced Speck32/64 have specific differences.

In Gohr's work, a sample consists of only one ciphertext pair. To improve the prediction accuracy of the differential neural differentiator, Chen and Yu [16] combine multiple ciphertext pairs generated by encrypting multiple plaintext pairs with the same key into a matrix as one sample of the neural network input. By using more output differences in the matrix, the neural network is made to learn more features. As a result, they improved the prediction accuracy of the (5∼7)-round differential neural distinguisher for Speck32/64 to some extent. Zhang et al. [17] modified the initial convolutional layer by borrowing ideas from the Inception module of GoogLeNet, which was used to construct a new neural network structure. Thus, they trained differential neural distinguishers for (5∼9)-round of Speck32/64.

In this paper, we further improve the data format of the input data and the neural network framework in the differential neural distinguisher, which helps the differential distinguisher identify ciphertext pairs with specific differential features more correctly.

(1) By analyzing the features of Speck's cipher and modifying the data format in the input data of the differential neural distinguisher according to Speck's round function. The data format (C_r, C_r', d_l, C_l, C_r, C_l', C_r') is proposed, which combines information integrity and domain knowledge, and enables the neural network to recognize a large amount of information contained in the previous round of the Speck cipher. This paper also used multiple ciphertext pairs as input to the neural network. Experiments (see Section 4.2) show that this data format can significantly improve the accuracy of the differential neural distinguisher.

(2) This paper adopts the idea of GoogLeNet by replacing the convolutional layers in the residual blocks with Inception modules, which consist of multiple parallel convolutional layers, to capture more dimensional information in the ciphertext pairs and train a better neural distinguisher. As a result, for the 6-round Speck and the 7-round Speck, the accuracy of the neural distinguisher reached 99.97% and 97.13%, respectively, which is higher than the accuracy of the above work. The results and comparisons of the differential neural distinguisher for Speck32/64 are listed in Tables 1 and 2.

(3) To demonstrate the advantage of our distinguisher, this paper performs a key recovery attack on the 8-round Speck32/64.

Table 1. Summary of distinguish accuracy on Speck32/64 using different number of instances.

Number of Speck Rounds	Ours	Zhang [17]	Gohr [14]	Hou [18]
6	99.97%	99.92%	78.50%	97.67%
7	97.13%	89.63%	59.10%	70.74%

Table 2. Experiment with different neural network model.

Round	Accuracy	Time	Source
7	86.46%	1200 s	Gohr [14]
	67.73%	300 s	Hou [18]
	87.98%	3800 s	Zhang [17]
	90.08%	3600 s	Ours

Training a neural network to distinguish 7-round Speck32/64 output for the input difference $\Delta = (0x0040, 0)$ from random data. Only the neural network model is different in these experiments while the other experimental conditions are the same.

2. Preliminaries

2.1. Brief Description of Speck Cipher

The lightweight family of ARX block ciphers Speck designed by the NSA [4] to build a cipher efficient in software implementations in Internet of Things (IoT) devices, which adopts a very simple Fesitel structure combining bitwise XOR operation$(\oplus)$, modular addition $(\boxplus)$ and bit-wise rotation. In ref. [4], various versions of Speck are presented that differ from the number of rounds (r), the block size (n), and the key size (m). Generally, Speckn/m will denote Speck with n bits block size and m bits key size. This paper will focus mainly on Speck32/64 and abbreviate it as Speck32.

Let (C_l, C_r) be the input of *i*-round of Speck32. Then the output of *i*-round is (C_l, C_r), and (C_l, C_r) is computed as follows:

$$C_l := ((C\_l \ggg \alpha) \boxplus C\_r) \oplus K$$
$$C_r := (C\_r \lll \beta) \oplus C_l$$

The round function of Speck is shown in Figure 1 where k_i represents the subkey at *i*-round and where $\alpha = 7, \beta = 2$.

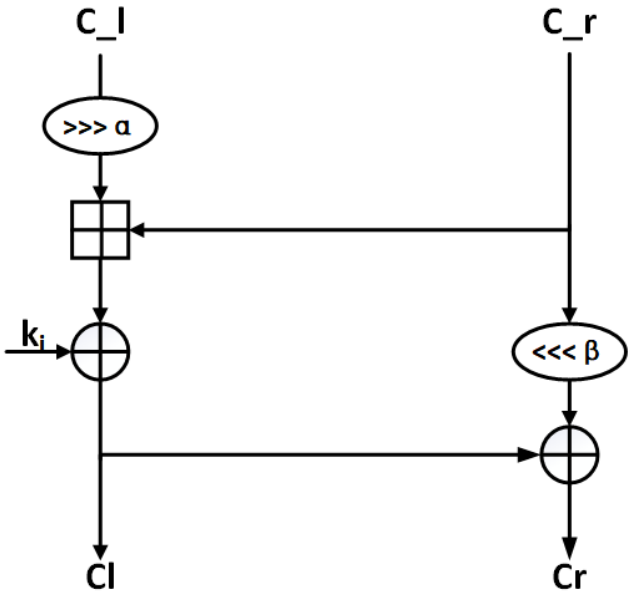


Figure 1. The round function of Speck.

2.2. Brief Description of ResNet

Residual neural network (ResNet), which is one of the most well-represented convolutional neural networks (CNN) currently, was proposed by He Kaiming et al. at CVPR 2016 [15]. ResNet can solve the problem of gradient disappearance when training convolutional neural network models and train deeper CNN models to reach higher accuracy.

The core concept is introducing a so-called "shortcuts(skip) connection" to a normal convolutional neural network, that is, the data output from the previous layer is superimposed directly on the input of the data layer that follows, skipping one or more convolutional layers. It is composed of a set of residual blocks. A residual block can be expressed as:

$$H(x) = F(x) + x$$

where $H(x)$ is the desired underlying mapping and x is the direct mapping, denoting that the stacked nonlinear layers fit another mapping as $F(x) := H(x) - x$[15]. By rearranging the linking order of the convolution layer(Conv), batch normalization(BN), and activation functions of ReLU, many residual block variants can be designed. Figure 2 shows the residual block.

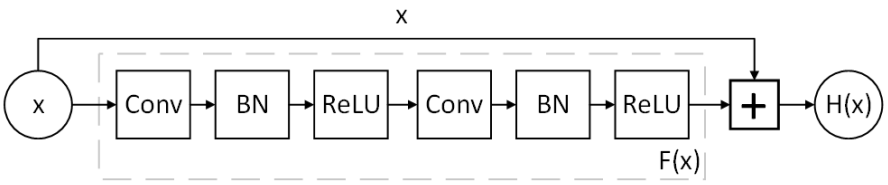


Figure 2. The residual block.

The batch normalization layer in the figure is to transform the output data of the neural network layer into a standard normal distribution with mean zero and variance one by some methods of normalization, which can effectively prevent the gradient disappearance problem and accelerate the network training speed. ReLU is a one-sided saturating activation function, which is defined as $f(x) = max(0, x)$; the gradients of the ReLU activation function become constant at positive values and no longer vanish [19]. This means that the gradient vanishing problem can effectively be avoided by using ReLU.

2.3. Brief Description of Inception Module

The Inception module is the core module of GoogLeNet proposed by Christian Szegedy [20], which takes into account the enlarged depth and width of the model. The Inception module was an impressive milestone in the development of CNN classifiers.

As shown in Figure 3, an Inception module has multiple convolutional layers with different convolutional kernel sizes, such as a 1 × 1 convolutional layer, a 2 × 2 convolutional layer, and a 4 × 4 convolutional layer. The 1 × 1 convolutional layer is equivalent to a fully connected layer, which is used to adjust the number of channels usually between each network layer to achieve cross-channel interaction and information integration of convolutional kernels.

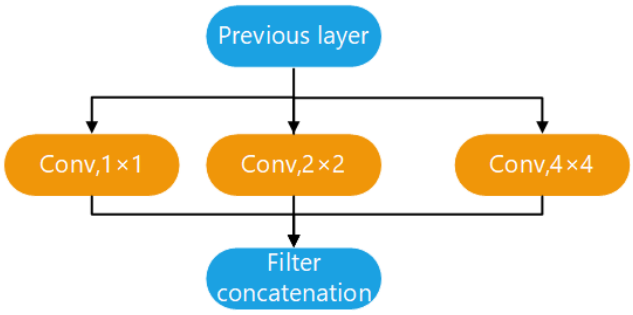


Figure 3. The Inception Module.

3. Improved Neural Distinguishers Model for Speck32

This Section Attempts to Teach Neural Networks to Identify the Differential Features of the Round-Reduced Speck as a Way to Construct a Differential Neural Distinguisher for Speck.

3.1. Data Format

As the number of Speck cipher rounds increases, the features of the Speck cipher algorithm are not easily recognized by the neural network. Therefore, as the data format contains more features from previous cipher rounds, it will help improve the accuracy of the neural network.

Once the ciphertext pair of the i-round (C_l, C_r, C_l', C_r') is known, one can straightforwardly compute (C_r, C_r') without knowing the $(i-1)$-round subkey (K) according to the algorithmic structure of the Speck cipher. Expressed in the formula as:

$$\begin{cases} C\_r = (C_l \oplus C_r) \ggg \beta \\ C\_r' = (C_l' \oplus C_r') \ggg \beta \end{cases} \tag{1}$$

But one needs to know the $(i-1)$-round subkey(K) to calculate (C_l, C_l'), which is expressed in the formula:

$$\begin{cases} C\_l = ((C_l \oplus K) \boxminus C\_r) \lll \alpha \\ C\_l' = ((C_l' \oplus K) \boxminus C\_r') \lll \alpha \end{cases} \tag{2}$$

where $\boxminus$ is modular minus. The difference d_l_{real} of C_l and C_l' is:

$$d\_l_{real} = C\_l \oplus C\_l' = (((C_l \oplus K) \boxminus C\_r) \oplus ((C_l' \oplus K) \boxminus C\_r')) \lll \alpha \tag{3}$$

Definition 1. *Denote $(C_l \oplus K), C\_r, (C_l' \oplus K), C\_r'$ by vectors, respectively:*

$$\begin{cases} C_l \oplus K = (a_1 \oplus k_1, a_2 \oplus k_2, \ldots, a_n \oplus k_n) \in \{0,1\}^n, n = 16 \\ C\_r = (b_1, b_2, \ldots, b_n) \in \{0,1\}^n, n = 16 \\ C_l' \oplus K = (x_1 \oplus k_1, x_2 \oplus k_2, \ldots, x_n \oplus k_n) \in \{0,1\}^n, n = 16 \\ C\_r' = (y_1, y_2, \ldots, y_n) \in \{0,1\}^n, n = 16 \end{cases} \tag{4}$$

Definition 2. *Denote d_l as the difference of C_l and C_l′ without the effect of the $(i-1)$-round subkey (K):*

$$d\_l = ((C_l \boxminus C\_r) \oplus (C_l' \boxminus C\_r')) \lll \alpha \tag{5}$$

Proposition 1. *Part of the bits of d_l_{real} that are not affected by $(i-1)$-round subkey can be captured by d_l.*

Proof. Without loss of generality, let $a_i \oplus k_i > b_i (i \neq 2), a_2 \oplus k_2 \leq b_2, n = 16$.

Converting $(C_l \oplus K) \boxminus C\_r$ to number field will obtain:

$$\begin{aligned} (C_l \oplus K) \boxminus C\_r = ((a_1 \oplus k_1)2^{n-1} + (a_2 \oplus k_2)2^{n-2} + (a_3 \oplus k_3)2^{n-3} \\ + \cdots + (a_n \oplus k_n) - b_1 2^{n-1} - b_2 2^{n-2} - \cdots - b_n) mod 2^n \end{aligned} \tag{6}$$

We can obtain:

$$\begin{aligned} (C_l \oplus K) \boxminus C\_r = ((1 + a_1 \oplus k_1 \oplus b_1 \oplus a_2 \oplus k_2 \oplus b_2)2^{n-2} \\ + (a_3 \oplus k_3 \oplus b_3)2^{n-3} + \cdots + (a_n \oplus k_n \oplus b_n)) mod 2^n \end{aligned} \tag{7}$$

In the same way, we get:

$$\begin{aligned} (C_l' \oplus K) \boxminus C\_r' = ((1 + x_1 \oplus k_1 \oplus y_1 \oplus x_2 \oplus k_2 \oplus y_2)2^{n-2} \\ + (x_3 \oplus k_3 \oplus y_3)2^{n-3} + \cdots + (x_n \oplus k_n \oplus y_n)) mod 2^n \end{aligned} \tag{8}$$

When the XOR operation is performed on $((C_l \oplus K) \boxminus C\_r)$ and $((C_l' \oplus K) \boxminus C\_r')$, it is obtained that the other bits of K has no effect on the operation except for k_1, k_2. That

is, the results of $(C_l \boxminus C\_r) \oplus (C_l' \boxminus C\_r')$ and $((C_l \oplus K) \boxminus C\_r) \oplus ((C_l' \oplus K) \boxminus C\_r')$ are the same for all bits except k_1, k_2.

The bit-wise rotation operation does not affect the number of captured bits in the vector. Therefore d_l captures the rest of the bits in d_l_{real} that are not affected by $(i-1)$-round subkey (K). □

Extending this proof, d_l can capture part of the bits of the d_l_{real} and the number of bits captured is different in each sample. It is certain that the accuracy of the neural network will improve as d_l captures more bits of d_l_{real}. The experiments in Section 4.2 show that the inclusion of d_l in the data format can improve the accuracy of the neural network by 2.34% under the same conditions.

Integrating the above data formats, this paper proposes a new data format

$$(C\_r, C\_r', d\_l, C_l, C_r, C_l', C_r')$$

which contains a number of cryptographic features from the $(i-1)$-round.

3.2. Data Structure

The data structure of multiple ciphertext pairs can achieve higher differentiation accuracy than the data structure of a single ciphertext pair. The dataset required to construct the differential distinguisher is shown in Figure 4.

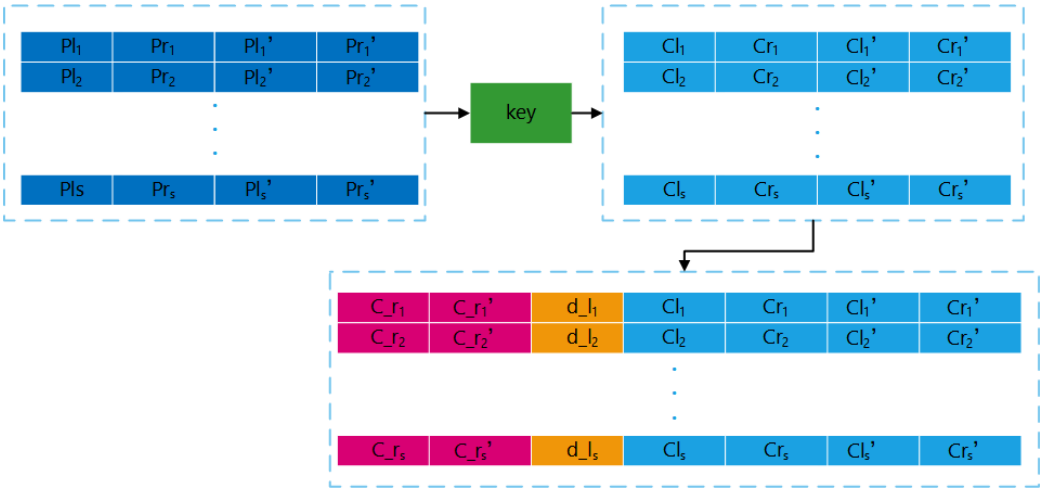


Figure 4. The structure of dataset.

The training and validation data are obtained by using the Linux random number generator (/dev/urandom) to get uniformly plaintext pairs P and distributed keys k with the input difference $\Delta = 0x0040/0000$, followed by a vector of binary-valued real/random labels Y, where s neighboring plaintext pairs are formed into one sample:

$$(P_{l1}, P_{r1}, {P_{l1}}', {P_{r1}}'), (P_{l2}, P_{r2}, {P_{l2}}', {P_{r2}}'), \cdots, (P_{ls}, P_{rs}, {P_{ls}}', {P_{rs}}').$$

To generate training or validation data for i-round ciphertext pairs, the s plaintext pairs P in a sample were then encrypted for i rounds if $Y = 1$, otherwise the second plaintext of the pair was replaced with a newly generated random plaintext and encrypted for i rounds

$$(C_{l1}, C_{r1}, {C_{l1}}', {C_{r1}}'), (C_{l2}, C_{r2}, {C_{l2}}', {C_{r2}}'), \cdots, (C_{ls}, C_{rs}, {C_{ls}}', {C_{rs}}').$$

Then the generated s ciphertext pairs are linearly transformed into (C_r, C_r', d_l, C_l, C_r, C_l', C_r') to obtain the samples needed for neural network training:

$$(C\_r_1, {C\_r_1}', d\_l_1, C_{l1}, C_{r1}, {C_{l1}}', {C_{r1}}'),$$

$$(C\_r_2, C\_r_2', d\_l_2, C_{l2}, C_{r2}, C_{l2}', C_{r2}'),$$
$$\vdots$$
$$(C\_r_s, C\_r_s', d\_l_s, C_{ls}, C_{rs}, C_{ls}', C_{rs}'),$$

Finally attach a label $Y = 1$ to the sample with $(P_l', P_r') = (P_l, P_r) \oplus \Delta P$ and a label $Y = 0$ to the sample with $(P_l', P_r') = random$.

3.3. Design the Network Structure

In this paper, the model of Gohr [14] is improved in order to smoothly converge the residual neural network to an optimal solution. A network model with higher accuracy is proposed, which significantly reduces the training time and makes it more efficient to attack the Speck. The framework of the neural network is shown in Figure 5.

The neural network is divided into four parts: an input layer consisting of multiple ciphertext pairs, an initial convolutional layer made of a one-layer convolutional neural network, a residual tower consisting of three layers of convolutional neural network optimized by the Inception module, and a prediction head consisting of multiple fully connected layers (distinguished by the colors in Figure 5).

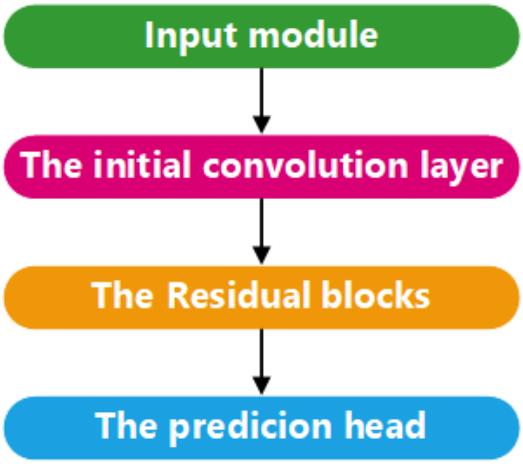


Figure 5. The framework of the neural network.

Initial convolution. After transforming the one-dimensional initial ciphertext data from Section 4.2 into $[s, w, \frac{2L}{w}]$ three-dimensional input data, the train data enters the initial convolution layer, where L represents the block size of the target cipher, w is the size of a basic unit, and $s = 16$, $w = 16$ for Speck32. The number of channels in the initial convolutional layer is $3N_f$, where $3N_f$ is the number of filters in the convolutional layer and $N_f = 32$. The convolutional layer is first convolved by a convolutional kernel of size 1, and then the convolved results are batch normalized. Finally, rectifier nonlinearity is applied to the output of batch normalization, and the resulting $[s, w, 3N_f]$ matrix is passed to the residual block.

Residual block. The residual neural network model constructed in this paper contains five residual blocks. Compared with the neural network model of Gohr [14], the residual block in this paper contains three convolutional blocks of $3N_f$ channels. The first convolutional block is convolved by a convolutional kernel of size one and then the output data is directly transferred to the second convolutional block. The second convolutional block consists of a 2×2 convolutional layer, a 4×4 convolutional layer, and an 8×8 convolutional layer; each convolutional layer has N_f channels. The three convolutional layers are concatenated in the channel dimension, similar to the Inception module in GoogLeNet. Batch normalization is applied to the output of the concatenated layers. The nonlinearity of the rectifier is applied to the output of batch normalization. The output from the second

convolution block is applied to the convolution of kernel size one, then a batch normalization layer, and finally a rectifier layer. At the end of the convolution block, the output of the last rectification layer in that block is added to the input of the convolution block, and the result is transferred to the next block.

Prediction head. The prediction head consists of a GlobalAveragePooling layer, two hidden layers, and one output cell. The three fully connected layers consist of 64, 64, and 1 neural unit, followed by batch normalization and rectifier layers. Finally, in order to constrain the final output neural unit between 0 and 1, the output neural unit is activated using the Sigmoid activation function. The Sigmoid activation function is a logistic function defined as $f(x) = \frac{1}{1+e^{-x}}$.

The overall structure of the neural network for training the differential-neural distinguisher is shown in Figure 6.

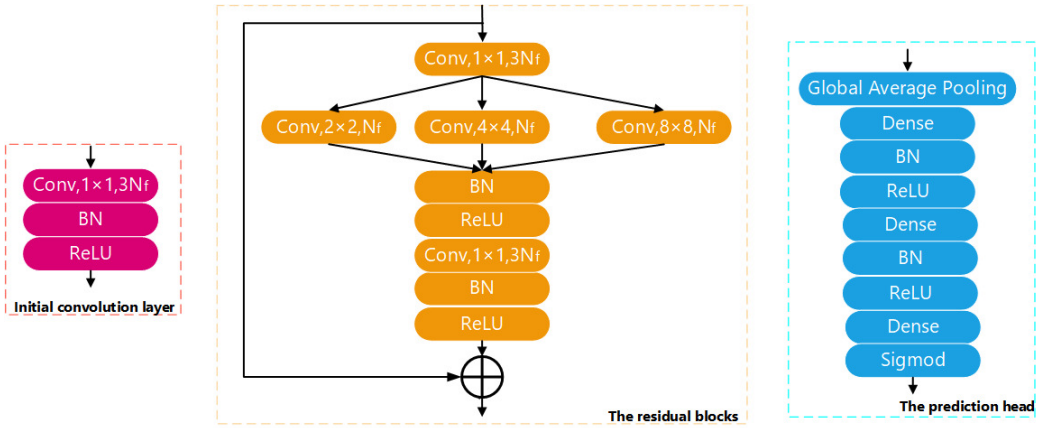


Figure 6. The overall structure of the neural network.

Basic training scheme. In this paper, the training is run for 20 epochs on the training dataset with a $SN = 10^6$ sample size. The batch size for dataset processing is adjusted according to parameter s to maximize GPU performance, where s is the number of ciphertext pairs in a single sample, so the number of ciphertext pairs in the training dataset is $CN = SN \times s$. The $SM = 10^5$ samples will be used for validation, containing $CM = SM \times s$ ciphertext pairs. Optimization was performed against mean square error (MSE) loss plus a small penalty based on the L2 weights regularization parameter $c = 10^{-5}$ using the Adam algorithm. A cyclic learning rate schedule was adopted, setting the learning rate L_i for epoch i to

$$L_i := \alpha + \frac{(n-i) mod (n+1)}{n}(\beta - \alpha)$$

with $\alpha = 10^{-4}$, $\beta = 2 \times 10^{-3}$ and $n = 9$. The best neural network model is saved by a callback function triggering the ModelCheckPiont method.

3.4. Design the Differential Neural Distinguisher

In this section, a neural differential distinguisher is designed for the round-reduced Speck. The distinguisher uses the data structures generated in Sections 3.1 and 3.2 as input data and the neural network structure of Section 3.3 as the structure of the distinguisher. The training algorithm for the differential neural distinguisher is shown in Algorithm 1.

Algorithm 1 The training algorithm for the differential neural distinguisher.

Require: Speck cipher $Oracle$, Number of randomly selected plaintext pairs n, Linear transformation $Transform$.
Ensure: Differential neural distinguisher N

$TD \leftarrow \emptyset$
$Y \leftarrow n$ random sample labels, assigned to 0 or 1
$(P_l, P_r, P_l', P_r') \leftarrow s$ random plaintext pairs with a difference of $\Delta = 0x0040/0000$;
for $i = 0$ to $n - 1$ **do**
  if $Y_i = 0$ **then**
    $(P_l', P_r') \leftarrow s$ random plaintexts
  end if
  $(C_l, C_r, C_l', C_r') \leftarrow Oracle(P_l, P_r, P_l', P_r')$
  $(C\_r, C\_r', d\_l, C_l, C_r, C_l', C_r') \leftarrow Transform(C_l, C_r, C_l', C_r')$
end for
$TD \leftarrow (X(C\_r, C\_r', d\_l, C_l, C_r, C_l', C_r'), Y)$
$N \leftarrow trainNetwork(TD)$
return N

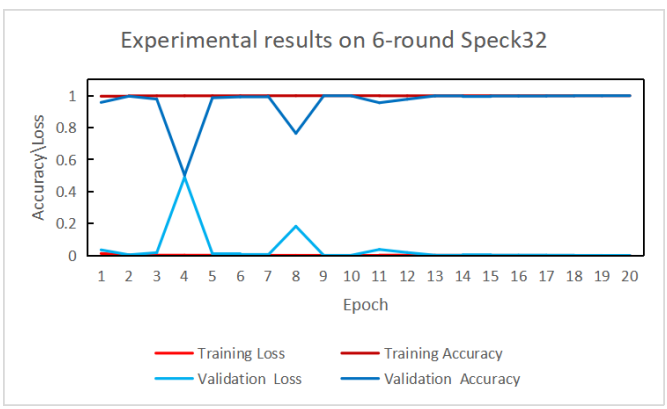
4. Results

In this paper, the experiments were conducted with Python 3.8 on Ubuntu 20.04 OS. The model we use is implemented by Tensorflow 2.9.0. For our experiments, we used a server with an Intel(R) Xeon(R) Platinum 8255C CPU at 2.50 GHz, 80 GB of RAM, and an RTX 3080 10 GB.

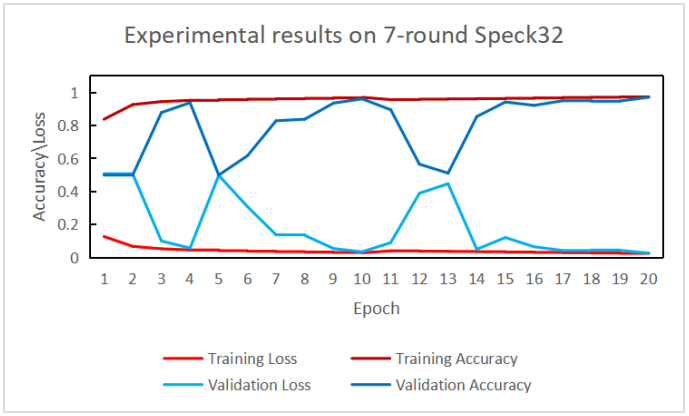
4.1. Experiments on Speck32

In this paper, we choose $\Delta = (0x0040, 0x0000)$ as the difference of the distinguisher when training the neural network, which transitions deterministically after one round to a difference with low Hamming weights [14], helping the neural network distinguisher to obtain the difference with the highest probability [21]. The parameter s is set to 32, and the batch size is set to 500. Then the plaintext pairs are encrypted by the Speck algorithm, and finally, the input data of the neural network is obtained after format conversion. Figure 7 gives the accuracy and loss rate of the training and validation sets in 6- and 7-rounds of Speck32 with 20 rounds.

It is worth noting that accuracy is used as a measure of distinguisher effectiveness in this paper because it is related to the distinguishing advantage of classical password distinguishers, and when the accuracy rate is higher, it means that the distinguisher is more effective.



(**a**) Expermential results on 6-round Speck32



(**b**) Expermential results on 7-round Speck32

Figure 7. Training neural networks to distinguish 6-, 7-round Speck32 output for the input difference $0x0040/0000$ from random data.

In Figure 7, the horizontal axes represent the number of rounds, and the vertical axes represent the accuracy and loss rate of the dataset results. The collapsed line shows the accuracy and loss rate of the data set during the training process of the differential neural distinguisher. From Figure 7a, the validation set accuracy of the 6-round distinguisher using the Speck algorithm trained by the neural network is 99.97% with a loss rate of 0.04%, while from Figure 7b, the 7-round distinguisher validation set accuracy is 97.13% with a loss rate of 2.67%, which is the highest known accuracy rate.

Since the data format and neural network structure proposed in this paper were not the same as in Zhang [17], Gohr [14], and Hou [18]. In their papers, differential neural distinguishers were obtained by changing the number of ciphertext pairs in a single sample (s) and the number of samples (SN).

In this section, the distinguisher with the highest accuracy in each paper is selected for comparison with the distinguisher proposed in this paper.

From Table 1, it can be seen that by improving the data format and neural network structure, the differential neural distinguisher can identify ciphertext pairs with specific differences more effectively. As a result, the accuracy of the differential neural distinguisher in this paper is higher than the above work, especially the 7-round distinguisher, which exceeds 95% accuracy for the first time.

4.2. Experiment with Different Data Format

In the work of Gohr [14], the data format (C_l, C_r, C'_l, C'_r) was used as the input data for the neural network. Subsequently, Benamira et al. [21] transformed the input (C_l, C_r, C'_l, C'_r) of Gohr's neural network into $(dl, dv, V0, V1)$ and a linear combination of these terms to achieve better performance, where $dl = C_l \oplus C'_l, dv = C_r \oplus C'_r, V0 = C_l \oplus C_r, V1 = C'_l \oplus C'_r$. The data format is simplified to (dl, dv) and the single ciphertext pair structure is converted to a multi-ciphertext pair structure in the work of Hou et al. [18]. Zhang et al. [17]. proposed $(C\_r, C\_r', C_l, C_r, C'_l, C'_r)$ considering that the key recovery attack requires ciphertext pairs according to Speck's round function, which effectively identifies the features of ciphertext pairs and enhances the performance of the distinguisher.

To demonstrate that the data format of this paper outperforms other data formats, this paper designed a comparison experiment that fixed all other parameters but only the data format is variable, where the neural network uses the model from Section 3.3 The results are shown in Table 3.

Table 3. Experiment with different data format.

Round	Data Fromat	Accuracy	Source
7	(C_l, C_r, C'_l, C'_r)	86.35%	Gohr [14]
	(dl, dv)	81.95%	Hou [18]
	$(dl, dv, V0, V1)$	86.43%	Benamira [21]
	$(C\_r, C\_r', C_l, C_r, C'_l, C'_r)$	87.74%	Zhang [17]
	$(C\_r, C\_r', d\_l, C_l, C_r, C'_l, C'_r)$	90.08%	Ours

Training a neural network to distinguish 7-round Speck32/64 output for the input difference $\Delta = (0x0040, 0)$ from random data. Only the data format is different in these experiments while the other experimental conditions are the same.

As shown in Table 3, since the data format proposed in this paper contains more features of the previous round of the Speck cipher, the accuracy of the neural network for the Speck has improved to 90.08%, which is the highest accuracy we know so far.

4.3. Experiment with Different Neural Network Model

Gohr [14] showed that the residual network could be trained to capture the non-randomness of the value distribution of output pairs when the input pairs of round-reduced Speck32/64 have a certain difference. Subsequently, Benamira [21] used the same neural network, while Hou et al. [18] reduced it to 5 iterations and removed a hidden layer.

In order to further improve the accuracy, better differential neural distinguishers have also recently been investigated. Zhang et al. [17] changed the input data of the neural network to three dimensions and modified the initial convolutional layer using the Inception module instead of the width-1 convolutional layer.

In this section, a comparative experiment was designed to investigate the effect of different neural networks on the distinguisher, with $(C\_r, C\_r', d\_l, C_l, C_r, C_l', C_r')$ as the input data format. Fixing all other parameters and only changing the neural network model, the neural network in this paper is compared with the neural networks of Gohr [14], Hou [18], and Zhang [17]. The comparison results are shown in Table 2.

It can be found that the accuracy increases when the dimensionality of the input data and the model complexity grow, while the training time also adds up. In order to make a trade-off between training time and accuracy, this paper improves Gohr's neural network and increases the accuracy of the neural network to 90.08%, and the training time is less than Zhang's model but longer than Gohr's and Hou's models.

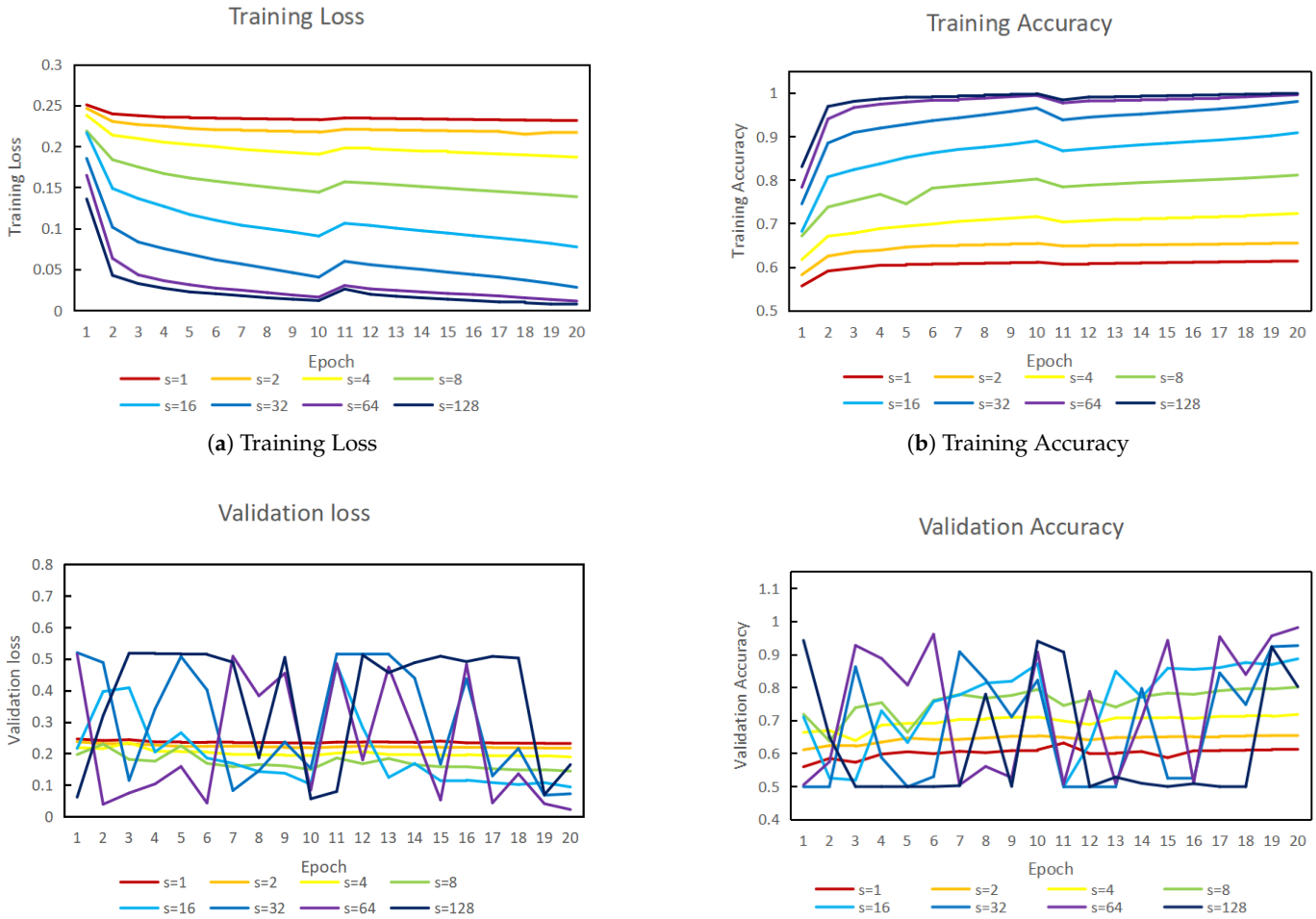
4.4. Effect of the Number of Ciphertext Pairs in a Single Sample (s) on the Neural Network

Chen et al. [16] explain the high accuracy of neural networks in recognizing multiple ciphertext pairs compared with a single ciphertext pair: if the ciphertext pairs obtained by encrypting plaintext pairs with specific plaintext differences obey a non-uniform distribution, then some derived features are derived from the multiple ciphertext pairs. Once the network captures these features, the accuracy of the neural distinguisher is improved.

Increasing the number of ciphertext pairs in a single sample (s) can improve the accuracy of the neural network when keeping the number of samples (SN) in the training dataset constant. And the number of cipher text pairs in the training dataset $(CN = SN \times s)$ also increases. Similarly, keeping s constant and increasing the number of samples in the training dataset (SN) can also improve the performance of the neural network.

In order to investigate the effect of the number of ciphertext pairs in a single sample (s) on the neural network with a constant number of total ciphertext pairs (CN), this paper designs a comparative experiment to explore the effect of s on neural networks: the numbers of ciphertext pairs in the training dataset are $CN = 10^7$ and in the validation dataset are $CM = 10^6$. Keeping the other parameters constant and changing the parameter s, the number of samples in the training and validation sets are $SN = CN/s$ and $SM = CM/s$. The batch size for dataset processing is adjusted according to the number of ciphertext pairs in a single sample (s) to maximize GPU performance, where $s \in \{1, 2, 4, 8, 16, 32, 64, 128\}$.

It can be found that the accuracy is growing when the parameter s increases in Figure 8. At the same time, the number of samples in the training dataset (SN) and validation dataset (SM) is decreasing, which leads to the overfitting phenomenon of the neural network when the parameter s is higher. Mitigating the overfitting phenomenon can be achieved by increasing the number of ciphertext pairs. Taking into account time and cost, in this paper, the parameter s is set to 32, the batch size is 500, and the number of samples in the training dataset (SN) and validation dataset (SM) are 10^6 and 10^5, respectively.



(**a**) Training Loss

(**b**) Training Accuracy

(**c**) Validation Loss

(**d**) Validation Accuracy

Figure 8. Training a neural network to distinguish 7-round Speck32 output for parameters $s \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ from random data.

5. Key-Recovery Attack against Speck32

To demonstrate the utility of the neural distinguisher, this paper constructs a partial key recovery attack based on a 7-round distinguisher. The basic idea is the decryption of the resultant ciphertext under all final subkeys for each plaintext pair with a difference $\Delta = 0x0040/0000$, and the sorting of each partially decrypted ciphertext using the neural distinguisher in this paper. Then the scores of the returned individual ciphertext pairs are combined into the scores of the keys, and finally, the keys are sorted in descending order according to their scores. A brief description of the attack steps is detailed below in Algorithm 2.

1. Generate n randomly chosen plaintext pairs $(P_1, P_1{}')$ with a difference $\Delta = 0x0040/0000$, such that it obtains the corresponding sample data ciphertext pair $(C_1, C_1{}')$ in encryption 8 rounds.

2. For each last-round subkey k, decrypt the C_i under k to get $(C_k, C_k{}')$.

3. Using the neural distinguisher in this paper, the score $x_i{}^k$ is obtained for each partially decrypted ciphertext pair $(C_k, C_k{}')$.

4. For each k, the scores $x_i{}^k$ are combined into one score x^k and arranged in descending order.

Algorithm 2 Key-recovery Attack Against Speck32.

Require: Speck cipher *Oracle*, Number of randomly selected plaintext pairs n, an 8 round neural distinguisher N.
Ensure: A descending sorted list of candidate keys L_{ck}.
1: $(P_1, P_1{}') \leftarrow n$ random plaintext pairs with a difference of $\Delta = 0x0040/0000$
2: $(C_k, C_k{}') \leftarrow Oracle(P_1, P_1{}')$
3: $L_{ck} \leftarrow \{\cdot\}$
4: **for** k in subkeys **do**
5: **for** $i = 0$ to $n - 1$ **do**
6: $(C_k, C_k{}') \leftarrow DecryptOneRound((C_i, C_i{}'), k)$
7: $x_i{}^k \leftarrow N(C_k, C_k{}')$
8: **end for**
9: $x^k \leftarrow \sum_{i=0}^{n-1} log_2(x_i{}^k/(1 - x_i{}^k))$
10: Append $(k, x^k) to L_{ck}$
11: **end for**
12: **return** L_{ck}

In this attack, the step 3 key ranking score is likely to be high when the 8-round subkey is guessed correctly. So, when a key guess is returned, the distinguisher can be sure that the correct 8-round key has been found.

In this paper, we repeated the key recovery attack 50 times for different keys. The attack is considered successful if the correct key is in the top five in L_{ck}. Finally, 46 keys were successfully recovered, with a success rate of about 92% for the experimental attack. The key ranking results are shown in Table 4. Of course, the success rate obtained with different success criteria can be different, so this success rate can only be used as a reference for the effectiveness of the key recovery algorithm.

Table 4. The result of the ranking of the correct subkeys in the list L_{ck}.

Ranking of the Correct Subkeys	1st	2nd	3rd	4th	5th	Others
Number of trials	23	11	7	4	1	4

In addition, in order to obtain in detail the correctness of each bit during the key recovery attack. In this paper, the subkey with the highest score is selected as the candidate key for guessing compared with the correct key, and the guess is considered successful if the last subkey is incorrect within only two bits. Then an exhaustive method can be used to eliminate the incorrect bits. The experimental results are shown in Tables 5 and 6.

Table 5. Guess the number of errors in the subkey bits.

The Number of Errors in the Subkey Bits	0	1	2	Others
Number of trials	23	15	12	0

Table 6. Guess the success rate of each subkey bit.

Subkey Bits	k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7
Number of trials	100%	100%	100%	100%	100%	100%	100%	98%
Subkey Bits	k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}	k_{14}	k_{15}
Number of trials	100%	100%	100%	100%	100%	100%	66%	58%

Table 5 shows that the neural network distinguisher in this paper guessed no more than 2 bits of the subkey incorrectly in 8 rounds of Speck experiments, and 23 experiments had exactly correct subkey guesses, accounting for 46%. Among the experiments with incorrect

guesses, 15 experiments (in 30%) have only 1 bit of incorrect guesses, and 12 experiments have 2 bits of incorrect guesses.

Further analysis of the 16 bits of the subkeys in Table 6 shows that the success rate of the neural network distinguisher in this paper reaches 98% for k_7 of the subkey. The success rates of k_{14} and k_{15} are 66% and 58%, respectively, and the rest of the bits are guessed correctly.

6. Conclusions

This paper proposed a new data format and model to further improve neural distinguishers and then performed a practical key recovery attack on Speck. Firstly, by adopting the new data format $(C\_r, C\_r', d\_l, C_l, C_r, C_l', C_r')$ and stitching multiple ciphertext pairs into a matrix as samples to capture more derived features, this can improve the accuracy of the neural distinguishers. In addition, by using the idea of the Inception module to modify the residual block of the neural network structure. As a result, the accuracy of the distinguisher in this paper is 99.97% and 97.13% for 6- and 7-round of Speck, respectively. Finally, the key recovery attack was performed on the 8-round Speck algorithm by the trained differential neural distinguisher. Among the 50 experiments performed, 46 were successful, with a 92% probability of success. In all the key recovery experiments, the number of subkey bits guessed incorrectly did not exceed 2 bits, and the accuracy of 100% was achieved in 13 of them.

To be sure, there are many factors that affect the accuracy of the neural network distinguisher, such as the data format and structure, the neural network structure and methods of model training, and so on. In this paper, we discuss the effects of data format, network structure, and the number of ciphertext pairs in the samples on the accuracy of neural networks. During the experiments, we found that the choice of neural network model affects the accuracy and time complexity of the trained distinguisher. Therefore, in the future, we will try to further eliminate the effect of the $(i-1)$-round subkey on the data format, optimize the neural network to improve the accuracy of the neural distinguisher, and investigate other block ciphers.

Author Contributions: X.Y.: conceptualization, methodology, validation, writing—original draft preparation. W.W.: supervision, writing—reviewing and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hai, Z.; Zhou, J.; Lu, Y.; Jawawi, D.; Wang, D.; Onyema, E.M.; Biamba, C. Enhanced security using multiple paths routine scheme in cloud-MANETs. *J. Cloud Comput.* **2023**, *12*, 68. [CrossRef]
2. Onyema, E.M.; Kumar, M.A.; Balasubaramanian, S.; Bharany, S.; Rehman, A.U.; Eldin, E.T.; Shafiq, M. A security policy protocol for detection and prevention of internet control message protocol attacks in software defined networks. *Sustainability* **2022**, *14*, 11950. [CrossRef]
3. Kavitha, A.; Reddy, V.B.; Singh, N.; Gunjan, V.K.; Lakshmanna, K.; Khan, A.A.; Wechtaisong, C. Security in IoT Mesh Networks based on Trust Similarity. *IEEE Access* **2022**, *10*, 121712–121724. [CrossRef]
4. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptol. EPrint Arch.* **2013**, *404*. Available online: https://eprint.iacr.org/2013/404 (accessed on 27 April 2023).
5. Sleem, L.; Couturier, R. Speck-R: An Ultra Light-Weight Cryptographic Scheme for Internet of Things. *Multimed. Tools Appl.* **2021**, *80*, 17067–17102. [CrossRef]
6. Biham, E.; Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.* **1991**, *4*, 3–27. [CrossRef]

7. FIPS PUB. Data Encryption Standard (DES); NIST, 1999. Available online: https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf (accessed on 27 April 2023).
8. Abed, F.; List, E.; Lucks, S.; Wenzel, J. Differential cryptanalysis of round-reduced SIMON and SPECK. In Proceedings of the Fast Software Encryption: 21st International Workshop, London, UK, 3–5 March 2014; pp. 525–545.
9. Biryuköv, A.; Roy, A.; Velichkov, V. Differential analysis of block ciphers SIMON and SPECK. In Proceedings of the Fast Software Encryption: 21st International Workshop, London, UK, 3–5 March 2014; pp. 546–570.
10. Dinur, I. Improved differential cryptanalysis of round-reduced speck. In Proceedings of the Selected Areas in Cryptography–SAC 2014: 21st International Conference, Montreal, QC, Canada, 14–15 August 2014; pp. 147–164.
11. Biryuköv, A.; Velichkov, V. Automatic search for differential trails in ARX ciphers. In Proceedings of the Cryptology–CT-RSA 2014: The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, 25–28 February 2014; pp. 227–250.
12. Gunjan, V.K.; Singh, N.; Shaik, F.; Roy, S. Detection of lung cancer in CT scans using grey wolf optimization algorithm and recurrent neural network. *Health Technol.* **2022**, *12*, 1197–1210. [CrossRef]
13. Pradhan, A.K.; Swain, S.; Kumar Rout, J. Role of Machine Learning and Cloud-Driven Platform in IoT-Based Smart Farming. *Mach. Learn. Internet Things Soc. Issues* **2022**, 43–54. [CrossRef]
14. Gohr, A. Improving attacks on round-reduced speck32/64 using deep learning. In Proceedings of the Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Barbara, CA, USA, 18–22 August 2019; pp. 150–179.
15. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, 26–30 June 2016; pp. 770–778.
16. Chen, Y.; Yu, H. A new neural distinguisher model considering derived features from multiple ciphertext pairs. *IACR Cryptol. EPrint Arch.* **2021**, *2021*, 310.
17. Zhang, L.; Wang, Z.; Wang, B. Improving differential-neural cryptanalysis with inception blocks. *IACR Cryptol. EPrint Arch.* **2022**, *183*. Available online: https://eprint.iacr.org/2022/183 (accessed on 27 April 2023).
18. Hou, Z.; Ren, J.; Chen, S. Improve neural distinguisher for cryptanalysis. *IACR Cryptol. EPrint Arch.* **2021**, *1017*. Available online: https://eprint.iacr.org/2021/1017 (accessed on 27 April 2023).
19. Ide, H.; Kurita, T. Improvement of learning for CNN with ReLU activation by sparse regularization. In Proceedings of the 2017 international joint conference on neural networks, Anchorage, AK, USA, 14–19 May 2017; pp. 2684–2691.
20. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Dragomir, A.; Dumitru, E.; Vincent, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 8–10 June 2015; pp. 1–9.
21. Benamira, A.; Gerault, D.; Peyrin, T.; Tan, Q.Q. A deeper look at machine learning-based cryptanalysis. In Proceedings of the Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 17–21 October 2021; pp. 805–835.