mathematics

MDPI

Article

Mixture Differential Cryptanalysis on Round-Reduced SIMON32/64 Using Machine Learning

Zehan Wu [1], Kexin Qiao [1,2,*], Zhaoyang Wang [1], Junjie Cheng [1] and Liehuang Zhu [1]

[1] School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China; zehanwu@bit.edu.cn (Z.W.); wangzhaoyang1@bit.edu.cn (Z.W.); junjiecheng@bit.edu.cn (J.C.); liehuangz@bit.edu.cn (L.Z.)
[2] State Key Laboratory of Cryptology, P. O. Box 5159, Beijing 100878, China
* Correspondence: qiao.kexin@bit.edu.cn

Abstract: With the development of artificial intelligence (AI), deep learning is widely used in various industries. At CRYPTO 2019, researchers used deep learning to analyze the block cipher for the first time and constructed a differential neural network distinguisher to meet a certain accuracy. In this paper, a mixture differential neural network distinguisher using ResNet is proposed to further improve the accuracy by exploring the mixture differential properties. Experiments are conducted on SIMON32/64, and the accuracy of the 8-round mixture differential neural network distinguisher is improved from 74.7% to 92.3%, compared with that of the previous differential neural network distinguisher. The prediction accuracy of the differential neural network distinguisher is susceptible to the choice of the specified input differentials, whereas the mixture differential neural network distinguisher is less affected by the input difference and has greater robustness. Furthermore, by combining the probabilistic expansion of rounds and the neutral bit, the obtained mixture differential neural network distinguisher is extended to 11 rounds, which can realize the 12-round actual key recovery attack on SIMON32/64. With an appropriate increase in the time complexity and data complexity, the key recovery accuracy of the mixture differential neural network distinguisher can be improved to 55% as compared to 52% of the differential neural network distinguisher. The mixture differential neural network distinguisher proposed in this paper can also be applied to other lightweight block ciphers.

Keywords: SIMON; ResNet; mixture differential; cryptanalysis

MSC: 94A60

Citation: Wu, Z.; Qiao, K.; Wang, Z.; Cheng, J.; Zhu, L. Mixture Differential Cryptanalysis on Round-Reduced SIMON32/64 Using Machine Learning. *Mathematics* **2024**, *12*, 1401. https://doi.org/10.3390/math12091401

Academic Editor: Pedro A. Castillo Valdivieso

Received: 31 March 2024
Revised: 28 April 2024
Accepted: 30 April 2024
Published: 3 May 2024

1. Introduction

The development of machine learning has impacted people's lives in every aspect, while deep learning has performed well in various tasks in most of the existing domains. This includes computer vision [1], smart driving [2], machine translation [3] and the latest interactive chat assistants [4]. Regarding cryptography, machine learning has been used for side-channel attacks [5,6] and cryptanalysis [7], demonstrating its superiority over traditional techniques [8].

Block ciphers [9] are widely used to ensure confidentiality in all kinds of information systems. Lightweight block ciphers [10], as one of the fast-growing ciphers, are gaining more and more attention and are widely used in IoT devices. IoT security is critical because these devices handle and transmit large amounts of sensitive data. The openness of IoT makes the information stored and transmitted in the network face great security threats [11]. So the security of lightweight block ciphers used in IoT devices is crucial. The SIMON lightweight block ciphers are proposed by the National Security Agency (NSA) of the US, aiming to be the sort of generalist block ciphers that will be required for future applications in the IoT era [12]. SIMON is well implemented on a variety of constrained

platforms, including ASICs, FPGAs, and microcontrollers. The family uses only basic bitwise arithmetic operations of XOR, And, and rotation. It is crucial to evaluate the security of such lightweight block ciphers by subjecting them to rigorous cryptanalysis with all known cryptanalytic techniques.

1.1. Related Work on Differential Cryptanalysis Using Machine Learning

In Eurocrypt 2017, L. Grassi et al. [13] discovered a "multiple of-8" property for 5-round AES. In FSE/ToSC 2019, this property is further refined as mixture differential cryptanalysis [14] and developed to a probabilistic version [15]. The main idea is given that the 4-round ciphertexts from a chosen plaintext pair lie in a particular subspace, the probability of a specially constructed pair has the same property with a probability of 1, while this is not the case for a random permutation. In [16], the exchange attack [17] and the mixture differential attack is unified as a variant of a differential attack where quadruples of plaintexts are constructed and the distinguisher can be produced by a Mixed Integer Linear Programming (MILP) [18] tool.

Discovering a distinguisher is the very first step in breaking a block cipher. Traditional techniques of the ultimate key-recovery attack equipped with a distinguisher are to use statistics to distinguish if a guess is right or wrong about the subkey used in appended rounds. The situation changed in 2019 when Gohr [7] presented a deep learning method to launch key recovery attacks using SPECK [12] as the target block cipher. Ghor designed a differential neural distinguisher for the SPECK and performed key recovery experiments with reduced data complexity. Inspired by Ghor's work, in 2021, Fu et al. [19] combined the idea of a polytopic differential [20], especially the 2-differential, developed 3-polytopic neural distinguishers, and applied it to recover keys for 13-round SIMECK32/64, a lightweight block cipher that resembles SIMON very much. Bao et al. [21] perform 12- and 16-round key recovery attacks against SPECK32/64 and SIMON32/64 using differential neural distinguisher, respectively, and also provide a more detailed analysis of the process of key recovery attacks using machine learning. Some rules of thumb are provided for tuning the key parameters and making better trade offs. Baksi et al. explore more on choosing an efficient machine learning model and observe that only a three-layer neural network can be used [22]. Bao et al. find that the power of differential-neural cryptanalysis in the related-key setting can exceed that in the single-key setting by successfully conducting a 14-round key recovery attack on SPECK32/64 [23]. Benamira et al. [24] provide a more in-depth interpretation of Ghor's neural differentiator. They found that the differential neural network distinguisher learns not only the difference distribution of the output pairs but also the difference distribution of the penultimate and pre-penultimate rounds.

1.2. Problem Statement and Our Contribution

Existing work studies the inner workings of machine learning-based distinguishers based on differential analysis or try to train higher rounds of the differential neural network distinguisher. The combination of machine learning-based key recovery techniques with other classes of differential analysis is lacking except for the polytopic differential, and the effect of combining mixture differential ideas with machine learning-based key recovery techniques is still unknown. In order to extend the applicability of machine learning-based key recovery techniques in differential-like cryptanalysis, and also to analyze the security of lightweight block ciphers against key recovery attacks by a mixture differential neural network distinguisher, we proposed a mixture differential neural network distinguisher.

Our contribution: In this paper, we for the first time incorporate the mixture differential idea with machine learning techniques to carry out key recovery attacks. Firstly, we construct 7- and 8-round mixture differential neural network (*MDNN*) distinguishers. Compared with the classical differential neural network distinguisher, *MDNN* has higher prediction accuracy; the accuracy of 7-round and 8-round *MDNN* reaches 99.4% and 92.3%. The accuracy comparison of the specific distinguisher is shown in Table 1. In Table 1, the mixture differential neural network distinguisher has similar accuracy for different

choices of input difference, whereas the fluctuations in the differential neural network distinguisher are larger, and thus the $MDNN$ is more robust to the choice of the specified input difference. Secondly, we utilize the obtained $MDNN$ to carry out a 12-round key recovery attack on SIMON32/64, and the accuracy of the last round key recovery reaches 55% with data complexity 2^{18} and time complexity $2^{31.19}$. Compared with the differential neural network distinguisher in [7], it has a key recovery success rate of about 52.1%, while the data complexity is $2^{14.5}$ and time complexity is 2^{38}. $MDNN$ has higher key recovery accuracy than the differential neural network distinguisher.

Table 1. Accuracy comparison between classical differential neural network distinguisher and mixture differential neural network distinguisher.

Distinguishers	Difference	*N*7 Acu.	*N*8 Acu.	Ref.
Differential	0×00000001	94.3%	74.7%	this
	0×00000004	94.1%	73.3%	this
	0×00000008	94.1%	73.1%	this
	0×00400000	74.5%	63.6%	this
	0×00400000	61.6%	51.4%	[7]
Mixture differential	0×00000001, 0×00000004, 0×00000008	99.4%	92.3%	this
	0×00000001, 0×00000004, 0×00400000	99.6%	91.4%	this

*N*7: 7-round neural network distinguisher, *N*8: 8-round neural network distinguisher.

This paper is organized as follows. In Section 2, after a brief description of SIMON, we have intruduced differential and mixture differential cryptanalysis, and we have also given a brief overview of the ResNet used in our model. In Section 3, we have briefly introdeced our neural network model and given seven rounds and eight rounds of the classical/mixture differential neural network distinguisher. In Section 4, we have presented methods related to key recovery using the neural network distinguisher and performed 12 rounds of key recovery attacks against SIMON32/64. The paper has been summarized in Section 5.

2. Preliminaries

2.1. A Brief Description of SIMON

The SIMON [12] lightweight block ciphers are in Feistel structure with branch sizes n = 16, 24, 32, 48 and 64 bits. The block sizes are $2n$. The key is composed of m n-bit words for m = 2, 3, 4 (i.e., the key size mn varies between 64 and 256 bits) depending on the word size n. The block cipher instances corresponding to a fixed word size n (block size $2n$) and key size mn are denoted by SIMON$2n/mn$. The number of rounds, block size and key size of the SIMON block ciphers are summarized in Table 2.

The SIMON round function is an AND-RX construction, i.e., constructions that only make use of the bitwise operations And, XOR and rotation, denoted by &, $\oplus$ and S^i (left rotation by i bits), respectively. The round function under a round key k defined on inputs x and y is

$$R_k(x, y) = (y \oplus f(x) \oplus k, x),$$

where $f(x) = (S^1(x) \& S^8(x)) \oplus S^2(x)$, and $S^i(x) = (x \lll i)$. The round function of SIMON is shown in Figure 1. For SIMON32/64, the subkeys of 16-bit for each round are generated from a master key of 64-bit by the linear key schedule using simple rotations and XOR components.

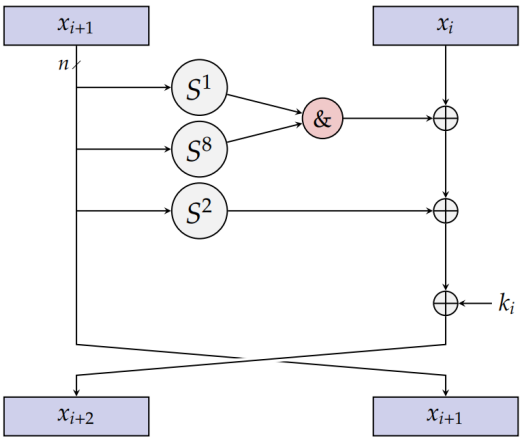

Figure 1. SIMON round function [12].

Table 2. Parameters for SIMON.

Block Size	Key Size	Rounds
32	64	32
48	72	36
	96	36
64	96	42
	128	44
96	96	52
	144	54
128	128	68
	192	69
	256	72

2.2. Differential and Mixture Differential Cryptanalysis

Differential cryptanalysis [25] studies the propagation of plaintext difference to ciphertext difference through the iterations of round functions. Let a function $f : F_2^n \to F_2^n$ denote a certain number of rounds, and x, x' two different inputs of f with a difference $\Delta_{in} = x \oplus x'$. Let $y = f(x)$ and $y' = f(x')$ and a difference $\Delta_{out} = y \oplus y'$. Then, a transition from Δ_{in} to Δ_{out} ($\Delta_{in} \xrightarrow{f} \Delta_{out}$) with probability $p > 2^{-n}$ is a differential distinguisher:

$$\Pr(\Delta_{in} \xrightarrow{f} \Delta_{out}) = \frac{\{x \in F_2^n | f(x) \oplus f(x \oplus \Delta_{in}) = \Delta_{out}\}}{2^n} > 2^{-n}.$$

The mixture differential cryptanalysis is a variant of the classical differential cryptanalysis. Unlike the classical differential, the mixture differential focuses on the difference propagation among four plaintexts. For bit-wise block ciphers, the bit-wise mixture differential pattern reflects the bit-wise equality relation among a quadruple of bits in the same position in a plaintext quadruple, denoted by $\mathbb{P} = (\Delta^1, \Delta^2, \Delta^3)$. A bit-wise mixture differential distinguisher is a pair of bit-wise mixture patterns $(\mathbb{P}_{in}, \mathbb{P}_{out})$ such that given plaintext quadruples (P^1, P^2, P^3, P^4) conforming to $\mathbb{P}_{in}$, and the ciphertext quadruples (C^1, C^2, C^3, C^4) conforming to $\mathbb{P}_{out}$ with probability $p > 2^{-3n}$.

$$\begin{aligned}\Pr(\mathbb{P}_{in} \xrightarrow{f} \mathbb{P}_{out}) =& \Pr((\Delta^1_{in}, \Delta^2_{in}, \Delta^3_{in}) \xrightarrow{f} (\Delta^1_{out}, \Delta^2_{out}, \Delta^3_{out})) \\ =& \{x \in F_2^n | (f(x) \oplus f(x \oplus \Delta^1_{in}) = \Delta^1_{out}) \wedge (f(x) \oplus f(x \oplus \Delta^2_{in}) = \Delta^2_{out}) \wedge \\ & (f(x) \oplus f(x \oplus \Delta^3_{in}) = \Delta^3_{out})\} / 2^n \\ >& 2^{-3n}.\end{aligned}$$

When utilizing a differential distinguisher $\Delta_{in} \rightarrow \Delta_{out}$ that covers r rounds to carry out a key recovery attack, the attacker appends additional rounds to the distinguisher. Then, the attacker enumerates the subkeys used in the appended rounds, relevant to only part of the master key. It is crucial to determine whether an enumeration is good or bad. By generating plaintexts that satisfy the specified input difference Δ_{in} or input mixture differential pattern $\mathbb{P}_{in}$ and collecting the ciphertexts, the enumerated subkey is used to decrypt the ciphertexts to the end of the distinguisher. In traditional methods, if the partially decrypted ciphertext pair or quadruple satisfies the output difference Δ_{out}, the enumerated subkey gets a vote. The one with the most votes is regarded as the right subkey, while in the machine learning-based methods [7], there may be more features in the pairs with the specified input difference that can be learned other than the specific output difference, so a neural network is trained on output pairs of the distinguisher to tell if such a pair is encrypted from a plaintext pair with the input difference or from a random pair. Then, each candidate subkey is assigned a score that combines the neural network prediction scores over all ciphertext pairs. The score is

$$V(k) = \sum_{i=1}^{n} \log_2(Z_i^k/(1 - Z_i^k)) \tag{1}$$

where $Z_i^k \in (0, 1)$ is the prediction score with subkey candidate k decrypting the i-th ciphertext pair [7,19].

2.3. ResNet

ResNet, short for Residual Network, stands as a pioneering architectural innovation in deep learning, particularly in the realm of convolutional neural networks (CNNs). Introduced by K. He et al. [26] in 2015, ResNet addresses the degradation problem encountered in very deep neural networks. It is used by Gohr [7] as the network in differential distinguisher training.

The core insight behind ResNet is the introduction of residual connections, which enable the training of extremely deep networks (up to hundreds of layers) without suffering from degradation. In ResNet, each layer is equipped with a "shortcut" connection, known as a skip connection or identity mapping, which bypasses one or more layers. These shortcuts facilitate the flow of gradients during training, alleviating the vanishing gradient problem and enabling effective training of very deep networks.

ResNet is featured for the residual block. Suppose x is the input of the residual block, $\mathcal{F}(x)$ is the residual function parameterized by some weights W_i, usually a combination of a series of convolution, batch normalization, and activation functions, etc. Then $\mathcal{F}(x) + x$ is the output of the residual block. The formulation of $\mathcal{F}(x) + x$ can be realized by feedforward neural networks with "shortcut connections" (Figure 2).

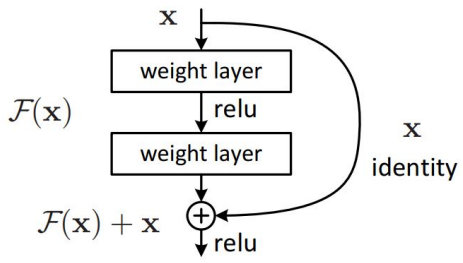

Figure 2. A residual block [26].

3. The 8-Round Mixture Differential Neural Network Distinguisher (MDNN)

In this section, the construction of the mixture differential neural network distinguisher is described in detail, including the generation of data, the specific introduction of the model architecture, and the training of the model.

3.1. Basic Idea of MDNN

The core idea of the differential neural network distinguisher [7] is to learn the characteristics of the output difference distribution of a given cryptographic algorithm with a specified input difference. The trained network is then utilized to predict the unknown ciphertext pairs and determine whether it is generated by encryption of the plaintext pair that satisfies the specified input difference. These output difference distribution features are invisible to conventional pure differential distinguishers. Meanwhile, when we use the differential neural network distinguisher for key recovery attacks, we can effectively reduce the data complexity as well as the time complexity. Combining the idea of mixture differential, we can obtain the $MDNN$.

For $MDNN$, given the input difference of $\mathbb{P} = (\Delta^1, \Delta^2, \Delta^3)$, ciphertext quadruples (C^0, C^1, C^2, C^3) are used for training. When a ciphertext quadruple is generated by encrypting a plaintext quadruple that satisfies the specified input difference, i.e., $(P^0, P^0 \oplus \Delta^1, P^0 \oplus \Delta^2, P^0 \oplus \Delta^3)$, the label is 1. On the other hand, when the ciphertext is generated by encrypting four random plaintexts (P^0, P^1, P^2, P^3), the label is 0. The trained model is then used to make predictions on a set of unknown ciphertext quadruples and outputs a prediction score $Z \in (0, 1)$ for each quadruple, which represents to what extent the ciphertext quadruple is deduced from the plaintext quadruple conforming to the input mixture pattern. In general, only if $Z > 0.5$, it is concluded that the ciphertext is generated by the plaintext encryption that satisfies the difference $(\Delta^1, \Delta^2, \Delta^3)$. The closer Z is to 1, the higher the probability that the ciphertexts satisfy the output differential features.

When the neural network distinguisher is trained, it is essentially learning the distribution characteristics of the ciphertext differences obtained after the plaintext differences have been encrypted and propagated. Furthermore, compared with the classical differential neural network distinguisher, the $MDNN$ acquires more information about the plaintext differences and can acquire more dependencies of the plaintext set when encrypted. Therefore, the accuracy of the $MDNN$ can reach a higher degree.

3.2. Data Generation

In order to train the neural network differential distinguisher, we need to generate a batch of data for training, which contains the plaintext, ciphertext, masterkey and label. For SIMON32/64, the length of the plaintext is 32, and the length of the masterkey is 64. We first generate the masterkey matrix.

$$masterkey = [[k^1[0], k^2[0], k^3[0], k^4[0]], \ldots, [k^1[n-1], k^2[n-1], k^3[n-1], k^4[n-1]]]$$

$k^j[i]$ represents a 16-bit random number, where $i \in [0, n-1]$, $j \in \{1, 2, 3, 4\}$. n denotes the number of samples generated for training. j indicates that the masterkey consists of four 16-bit random numbers. Similarly, we can generate the plaintexts for training:

$$P_l^0 = [P_l^0[0], P_l^0[1], \ldots, P_l^0[n-1]], \quad P_r^0 = [P_r^0[0], P_r^0[1], \ldots, P_r^0[n-1]]$$

$P_l^0[i]$ denotes the left input of the i-th randomly generated first pair of plaintexts. $P_r^0[i]$ denotes the right input of the i-th randomly generated first pair of plaintexts. By differentiating the P_l^0 and P_r^0 from the input difference, we can obtain a set of quadrple plaintexts:

$$\Delta = (\Delta^1, \Delta^2, \Delta^3), \Delta^i = (\Delta_l^i, \Delta_r^i)$$

$$P_l^1 = P_l^0 \oplus \Delta_l^1, P_r^1 = P_r^0 \oplus \Delta_r^1, \ldots, P_r^4 = P_r^0 \oplus \Delta_r^3$$

Δ^i denotes one of the mixture difference, it consists of the difference of the left and right inputs, $i \in \{1, 2, 3\}$. This gives us n quadruple plaintexts that satisfy the input difference. Through key expansion algorithms as well as encryption, we can then obtain the round-key for each of the n samples and the corresponding n ciphertexts. The label Y is randomly generated and contains 0 and 1, and the length of Y is n. When Y is 0, the plaintext

is replaced with randomly generated plaintext, when Y is 1, the plaintext is the set of plaintexts that satisfy the specified input difference.

$$Y = [0, 1, 0, 0, 1, \ldots, 1]$$

The ratio of 0 and 1 in Y is about 1:1. In this way, we can obtain the dataset with balanced positive and negative samples. Since we are targeting the training of SIMON32/64 mixture differential neural network distinguishers, the data as a whole consist of two parts: features as well as labels. The features consist of four ciphertexts obtained by encrypting plaintexts that satisfy the specified input differences (the number of encryption rounds is determined by the number of rounds of the neural network distinguisher that we want to train). So the overall number of dimensions of features is $32 \times 4 = 128$. Labels are 0/1 of one dimension, so the overall dimension of training/validation data is 129 dimensions. While training the model, we set the training set size and validation set size. Thus, the overall number of rows and columns in the dataset is $(10^7 + 10^6)$ and 129, respectively.

3.3. Neural Network Distinguisher Model Architecture

Using *MDNN* as an example, the model architecture for SIMON32/64 and the relevant parameters are given. The distinguisher model as a whole consists of three blocks.

- Block 1: the input layer and the initial convolutional layer, the initial convolutional layer contains a 1D-CNN with kernel size 1 (convolution uses 32 filters), batch normalization and ReLU activation function.
- Block 2: the residual layer, which contains i residual blocks and i is the depth of the residual layer (in this model, we let i be 10). Each residual block consists of two 1D-CNN with a kernel size of 3 (convolution uses 32 filters), each followed by batch normalization and a ReLU activation function.
- Block 3: the final classification layer, the classification layer contains three perceptron layers separated by two batch normalization and ReLU functions, and finished with a sigmoid function.

The overall architecture of the model is shown in Figure 3.

When we denote by X the data transferred to the model input layer, X consists of n sets of encrypted data:

$$X = [X_0, X_1, X_2, \ldots, X_{n-1}]$$

$$X_i = [C_l^0, C_r^0, C_l^1, C_r^1, C_l^2, C_r^2, C_l^3, C_r^3]$$

For mixture difference, there are four cipher pairs in total, and when considering left and right inputs, there are eight ciphertexts in total, and in SIMON32/64, the inputs are 16 bits per side. So each sample in the training set is 128 bits of data. These data are represented in binary during training. Thus, in the final output classification layer, the first two layers have 128 neurons each.

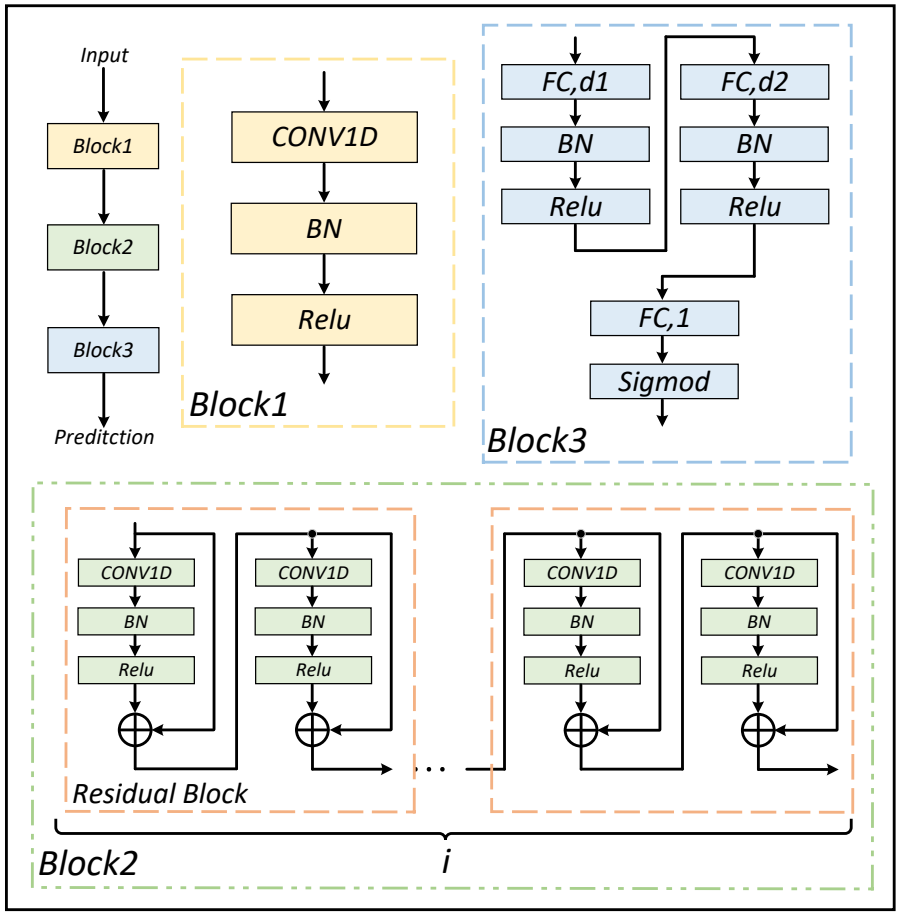

Figure 3. Model architecture.

3.4. Model Training

We mainly trained 7-round as well as 8-round neural network difference distinguishers for SIMON32/64 for the differential as well as the mixture differential. The training data are the previously mentioned features X and labels Y. For the specified input difference that are satisfied by the plaintext, we have chosen $\Delta^1 = (0\times0000001)$, $\Delta^2 = (0\times00000004)$ and $\Delta^3 = (0\times00000008)$. We trained 7 and 8 rounds of classical differential neural network distinguishers for Δ^1, Δ^2 and Δ^3, respectively. We also trained 7 and 8 rounds of mixture differential neural network distinguishers for the input difference $(\Delta^1, \Delta^2, \Delta^3)$. Furthermore, for comparison with the distinguishers in [7], we selected the same difference ($\Delta = 0\times40000000$) and trained the distinguisher with the same number of rounds. The accuracy of the distinguishers is specified in Table 1.

For training, we chose the size of the train set as 10^7 and the size of the validation set as 10^6. We ran 50 epochs for each training, and the batch size was 10,000. For the loss function, we chose the Mean Square Error loss (MSE) and used the Adam optimization algorithm that comes with tensorflow-keras to optimize the loss function as well as the L2 regularization (regularization parameter = 10^{-4}). The learning rate scheduler selects cyclic learning rate scheduling (CyclicalLR). We utilize Rtx3090 for model training. It takes about 2 min per epoch during training. The programming language used for data generation, model construction and model training is python, and the specifications of the machine used for model training are that the number of CPU cores is 64, the CPU frequency is 3699 MHZ, and the size of the RAM is 94 GB. The machine learning framework used is tensorlfow-keras, and the libraries such as numpy and pandas are used. The operating system is Ubuntu 22.04.4.

Figure 4 as well as Figure 5 illustrate the comparison of the success rate of 7-round as well as 8-round classical differential neural network distinguishers with the mixture

differential neural network distinguisher. Taking Figure 5 as an example, the horizontal axis represents the epoch, and the vertical axis represents model accuracy. There are four lines in the figure, which represent the training accuracy and prediction accuracy of the classical differential and mixture differential distinguishers, respectively. From the figure, it can be seen that the accuracy rate tends to stabilize with the improvement in the epoch and compared with the classical differential neural network distinguisher, and the mixture differential neural network distinguisher has a higher accuracy rate.

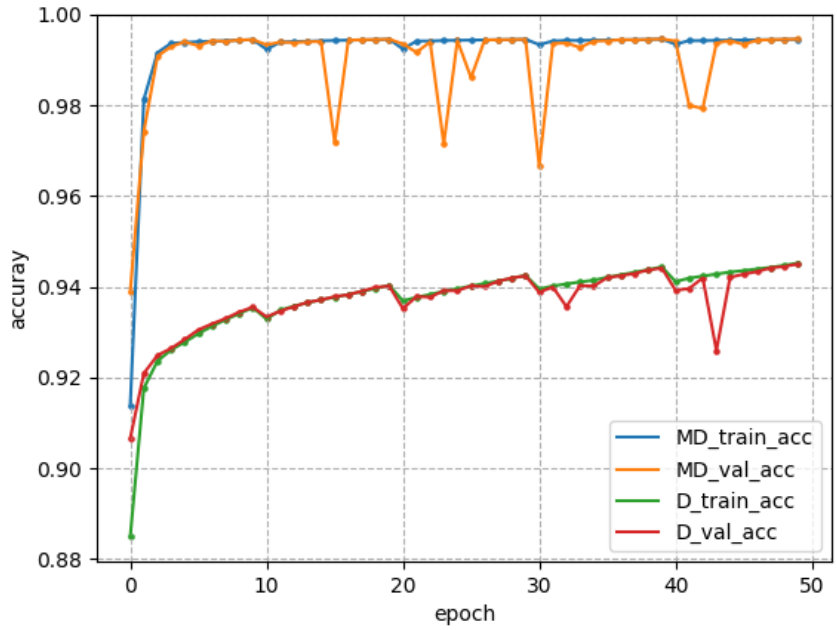

Figure 4. N7 distinguisher accuracy comparison.

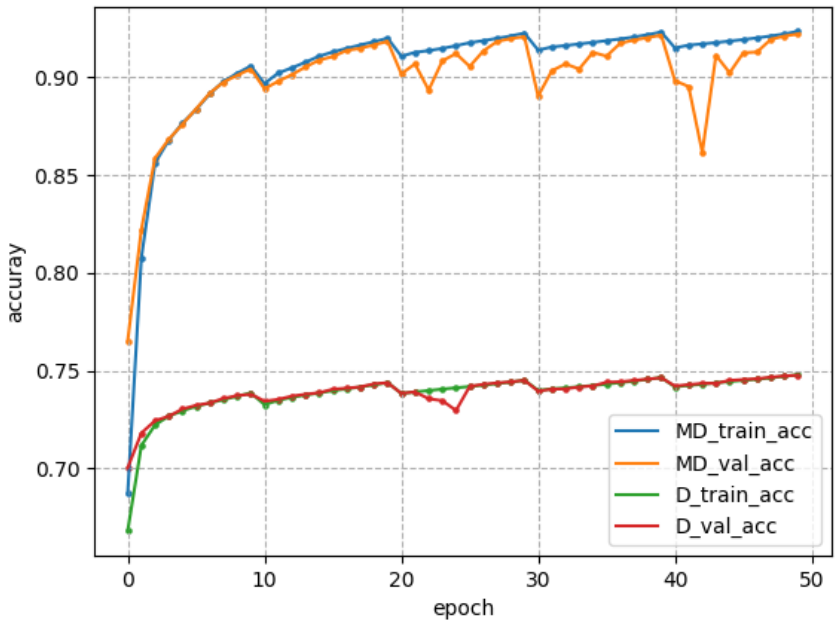

Figure 5. N8 distinguisher accuracy comparison.

An increase in the number of encryption rounds leads to a more thorough obfuscation of the differential features, making it difficult to learn from the output differential features of the ciphertext in relation to the input differential features of the plaintext. Therefore, the prediction accuracy of the model decreases as the number of rounds increases.

4. Key Recovery Attack on 12-Round SIMON32/64

In this section, we detail the basic idea of key recovery attacks using mixture differential neural network distingsuihers. We performed a key recovery attack on 12-round SIMON32/64 using trained 7-round as well as 8-round mixture differential neural network distinguishers. The overall key recovery attack process is shown in Figure 6.

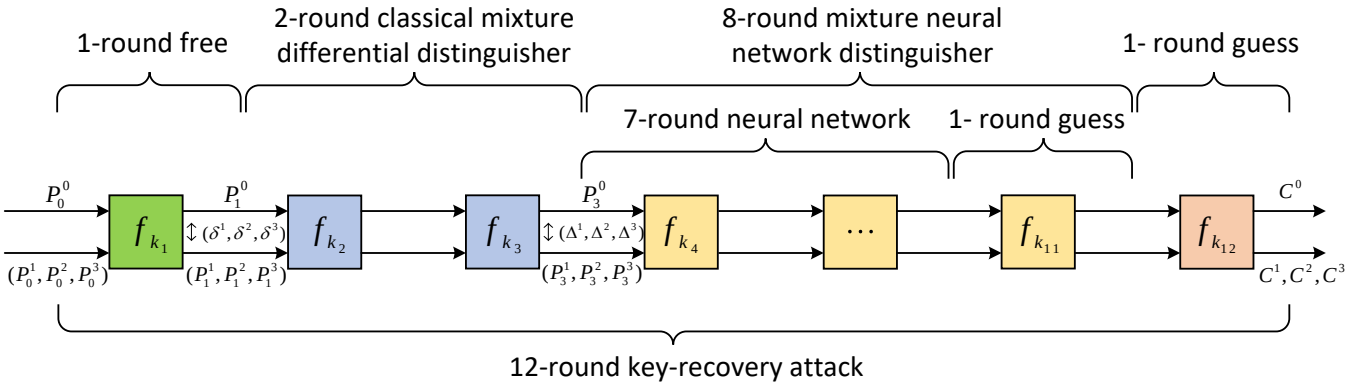

Figure 6. The 12-round key recovery attack using $MDNN$ for SIMON32/64.

4.1. Basic Attack Ideas for MDNN

In Figure 6, we can find that the key recovery attack process consists of three main parts, the classical mixture differential distinguisher, the mixture differential neural network distinguisher and finally the key guessing, where f_{k_i} denotes the round function of each round, $i \in \{1, 2, 3, \ldots 12\}$. $(P_i^0, P_i^1, P_i^2, P_i^3)$ denotes the intermediate state of the quarduples after i rounds of encryption. In the process of data generation, we have generated n pairs of quadruple plaintexts, where the plaintext labeled 1 satisfies the input difference $\Delta = (\Delta^1, \Delta^2, \Delta^3)$. We encrypted these plaintexts for 7 and 8 rounds, using them to train the $MDNN$. In order to extend the number of rounds for the key recovery attack, we added a classical mixture differential distinguisher in front of the neural network distinguisher. The output difference of the classical mixture differential distinguisher is the same as the input difference of the neural network distinguisher. We denote the input difference of the classical mixture differential distinguisher by $\delta = (\delta^1, \delta^2, \delta^3)$. After an additional round of decryption, the plaintext is obtained. When performing key guessing, we assume that n ciphertext quadruples (C^0, C^1, C^2, C^3) are collected, k is the guessed subkey in the appended rounds, and k is used to decrypt n ciphertexts. Z_i^k denotes the predicted value of the i-th ciphertext, $Z_i^k \in (0, 1)$. When we determine the goodness of a guessed key, we can combine the prediction scores of n ciphertexts using Equation (1). When $V(k)$ is greater than a set threshold, we can consider the guessing key to be a good key. The data complexity is related to the number of samples n we selected.

Here, by prepending two rounds of mixture differential transition, we extend the 8 rounds of mixture differential neural network distinguisher to 10 rounds. The specific differential transfer trail is

$$(0{\times}0001/2004, 0 \times 8000/1002, 0 \times 4000/0801) \rightarrow$$
$$(0 \times 0000/0001, 0 \times 0000/0004, 0 \times 0000/0008)$$

where $(0 \times 0001/2004, 0 \times 8000/1002, 0 \times 4000/0801)$ is the mixture difference of the input, and the output mixture difference is $(0 \times 0000/0001, 0 \times 0000/0004, 0 \times 0000/0008)$ with probability $2^{-5} \times 2^{-5} = 2^{-10}$. Furthermore, the output difference is the specified input difference that we use to train the mixture differential neural network distinguisher. We can call this intermediate state data as the available difference structure. The difference in the available difference structure should be consistent with the input difference used to train the neural network. Furthermore, one round of decryption of the inputs of the two-round distinguisher gives the chosen plaintext. We use it as an input for 12 rounds of

SIMON32/64 encryption. In this way, we can guarantee that the mixture difference after three rounds of encryption for $(0 \times 0000/0001, 0 \times 0000/0004, 0 \times 0000/0008)$ is 2^{-10}.

Since the plaintext that satisfies the specified input difference can only obtain the output difference we want with a probability of 2^{-10}, in order to increase the probability as well as reduce the corresponding data complexity. Neutral bits [27] can be utilized to construct the plaintext. When the neutral bits of the plaintext change, its output difference does not change. Therefore, we select a group of plaintexts and change six neutral bits in them to obtain a group of 64 plaintexts with the same output mixture difference. The neutral bits used for the 12 rounds of attack are [2, 3, 4, 6, 8, 9].

We encrypt the corresponding plaintext with a 12-round SIMON32/64 and guess the subkey for the last round. The candidate key is mainly obtained by using the candidate subkey search algorithm (Algorithm 1). The ciphertext is decrypted in one round using the guessed key, and the result is fed into the 8-round mixture differential neural network distinguisher for prediction. Using the prediction output of the distinguisher and Equation (1), a value $V(k)$ for determining a good or bad guessed key is obtained. This value is compared with the boundary value $Cutoff_1$ and if $V(k) > Cutoff_1$, the resulting intermediate state is continued to be decrypted for the penultimate round of key guessing. Again the 7-round mixture differential neural network distinguisher prediction output is combined with Equation (1) to calculate $V(k)$ and compared with the boundary value $Cutoff_2$. If $V(k) > Cufoff_2$, then the two rounds of guessed key are the subkeys we want.

To summarize the basic process of the key recovery attack by the mixture differential neural network distinguisher, we use the guessed subkey to decrypt the ciphertext and observe the output of the distinguisher. After determining the final-round subkeys and then guessing the penultimate round of subkeys, if both rounds of subkeys are greater than the given bounding value, then we obtain the subkeys that we want. Both rounds of key guessing utilize the distributional characteristics of the output mixture difference, and the penultimate round of subkey guessing can play an auxiliary role to the last round of subkey guessing to verify the correctness of the last round of guessing.

4.2. Model Response to Wrong Keys

In order to perform the search for candidate subkeys, we need to obtain the mean as well as the standard deviation of the distinguisher's response to the distribution of wrong keys. We calculated the wrong key response profile for our 7-round as well as 8-round differential and mixture differential neural network distinguishers for SIMON32/64. Here, we take an example of the response of the 8-round mixture difference neural network distinguisher to the wrong key. The experimental key difference is $\Delta \in F_2^{16}$.

For each difference Δ, we generate 1000 sets of plaintexts (P^0, P^1, P^2, P^3) that satisfy the specified input difference used for training and 1000 random keys k. For an eight-round distinguisher, we would encrypt the plaintext for nine rounds with the corresponding keys k, and the encrypted ciphertext would be (C^0, C^1, C^2, C^3). The wrong key is $k' = k \oplus \Delta$, and we use the wrong key k' to decrypt the existing ciphertext for one round to obtain $(E_{k'}^{-1}(C^0), E_{k'}^{-1}(C^1), E_{k'}^{-1}(C^2), E_{k'}^{-1}(C^3))$. Then, the eight rounds of the mixture differential neural network distinguisher are used to make predictions about these ciphertexts. Based on these predictions, $N_8(E_{k'}^{-1}(C^0), E_{k'}^{-1}(C^1), E_{k'}^{-1}(C^2), E_{k'}^{-1}(C^3))$, we can obtain the mean and standard deviation of the difference Δ between the wrong key and the correct key on 1000 trials. The mean μ_Δ and standard deviation σ_Δ depend on random variable Δ. The mean of the wrong key response for seven-round and eight-round differential and mixture differential neural network distinguishers are shown in Figure 7, Figure 8, Figure 9 and Figure 10, respectively.

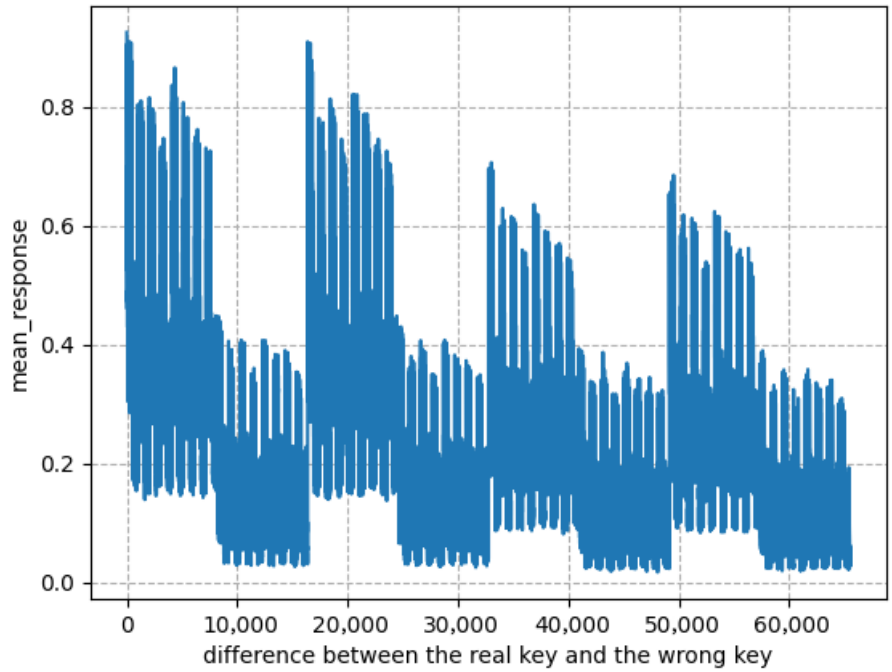

Figure 7. The 7-round differential neural distinguisher wrong key response.

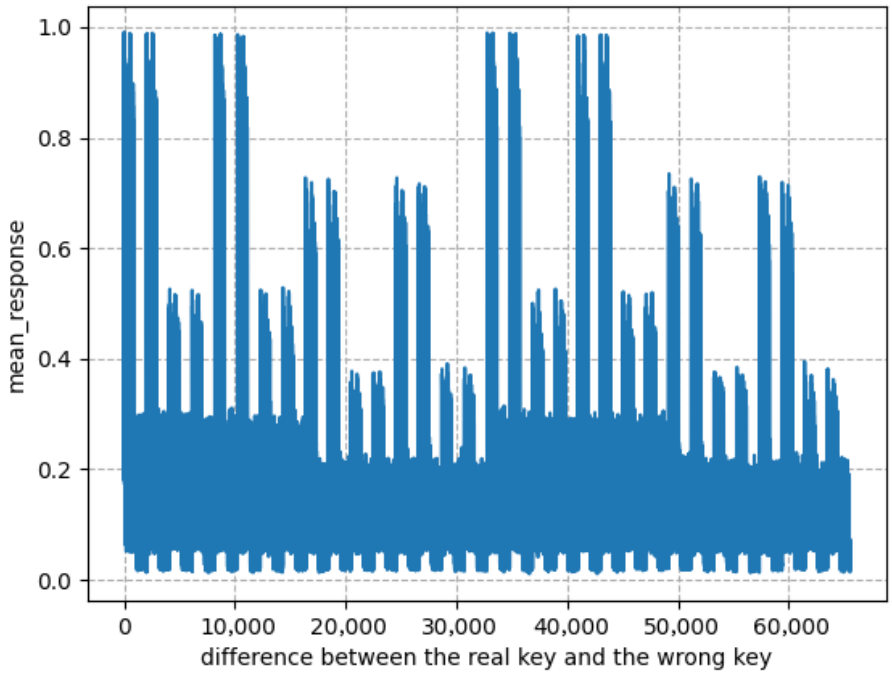

Figure 8. The 7-round *MDNN* wrong key response.

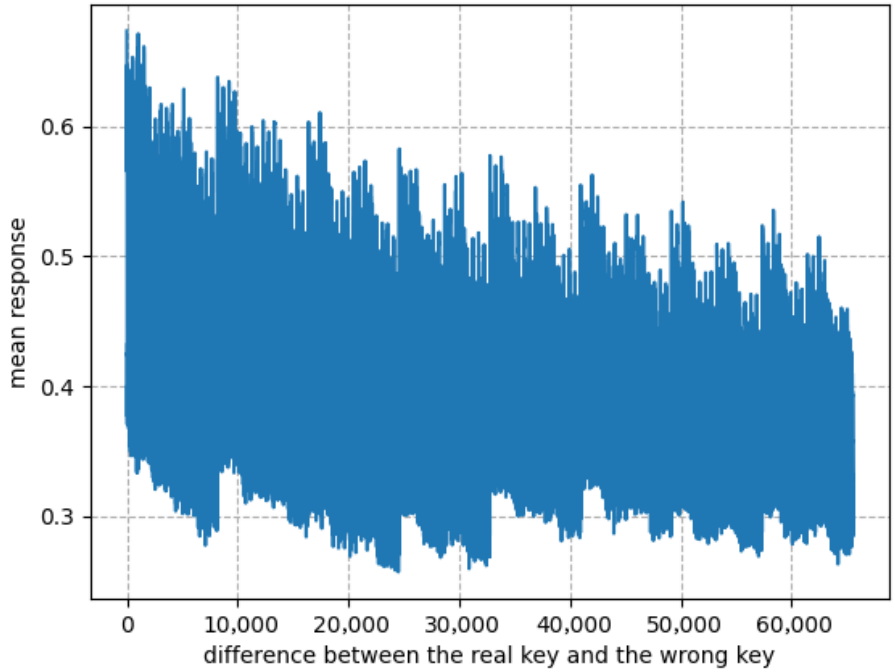

Figure 9. The 8-round differential neural distinguisher wrong key response.

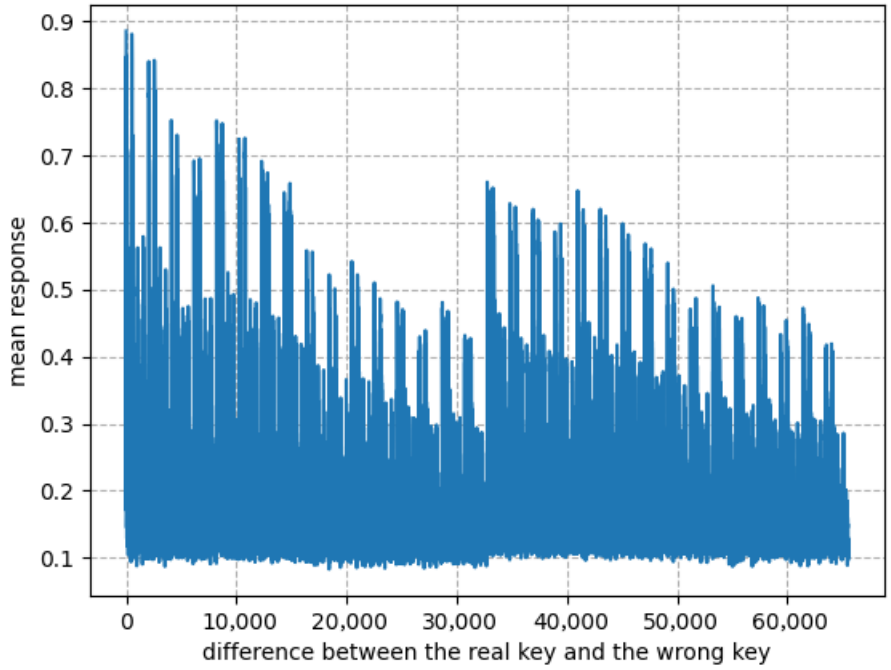

Figure 10. The 8-round *MDNN* wrong key response.

Taking the 8-round wrong key response profile as an example, we can see a lot of obvious non-random structures in Figures 9 and 10. Figure 9 shows the mean value of the wrong key response for the eight-round differential neural network distinguisher, where the input difference of the plaintext used is $(0 \times 000/0001)$. The input difference of the plaintext used in Figure 10 is $(0 \times 0000/0001, 0 \times 0000/0004, 0 \times 0000/0008)$. The x-axis represents the difference between the real key and the wrong key, and the y-axis represents the mean value of the distinguisher prediction on 1000 trials. From the figure, it can be seen that the response of the differential neural differentiator has the same trend as the response of the *MDNN*, as the larger the difference in bits between the wrong key and the real key exists, the prediction of the distinguisher is lower. The reason is that when the number of

bits in which there is a difference between the wrong key and the real key goes up, the correct ciphertext output differential features obtained from the decrypted ciphertext are less, which affects the prediction accuracy of the distinguisher. By comparison, we can see that there is a more significant feature structure in Figures 7 and 8, indicating that the information in the output differential features is weakening as the number of rounds increases. It can also be noticed that the structure of the wrong key response is more significant for the mixture differential, suggesting that the mixture differential can capture more output difference features than the differential in neural network training. The wrong key response profile can be used to perform Bayesian inference to obtain candidate keys.

4.3. Candidate Subkey Search

While performing the search for candidate subkeys, we have used the Bayesian optimization method [28]. Bayesian optimization methods can be used to find the maximum value of the function and the corresponding parameters based on the values of the sampling points that have been obtained without knowing the specific function equation, which means that this method is mainly used for black box functions. The fine-tuning of hyperparameters in machine learning is a very common example of its use. Bayesian optimization consists of two main core processes, a prior function, which is used to construct a probabilistic model of a function that can be easily optimized in place of the black box function. The other one is the sampling function, which is used to decide how to select new sampling points and contains the commonly used standard exploration–exploitation technique, namely Upper Confidence Bounds (UCBs) [29].

In the scenario of a key recovery attack using the mixture differential neural network distinguisher, we can treat the $MDNN$ model as a black box function. Based on the wrong key response, we know that the output of the $MDNN$ model is correlated with the difference between the real key and the wrong key, so the process of finding the maximum value of the model is actually the process of searching for candidate subkeys. Given a subkey and trial decryption, the Bayesian optimization derives a series of new key assumptions based on the pre-computed wrong key response profile. The output prediction of the model is closely associated with the difference of the key, mainly the mean μ_Δ and the standard deviation $\sigma_\Delta(\Delta = k \oplus k'$, k: realkey, k': guesskey). The specific key search strategy is described in Algorithm 1:

Algorithm 1 Candidate Key Search Implemented by Bayesian Optimization

Input: Ciphertext $C = \{C_1, , \ldots, C_m\}, m$: number of ciphertext, $MDNN$, n: number of candidate keys, it: number of iterations, wrong key response profile: $\mu_\Delta, \sigma_\Delta$.
Output: $S = \{k_1, \ldots, k_n\}, L$

$S = \{k_1, \ldots, k_n\}$(random generated), $L = \{\}$
for $iteration \in [1, it]$ **do**
  $P_{i,k_j} = \text{dec\_one\_round}(C_i, k_j), i \in \{1, \ldots, m\}, k_j \in S$
  $v_{i,k_j} = MDNN(P_{i,k_j}), i \in \{1, \ldots, m\}, k_j \in S$
  $w_{k_j} = \sum_{i=1}^{m} \log_2(v_{i,k_j}/(1 - v_{i,k_j})), k_j \in S$
  $L = L \cup (k_j, w_{k_j}), k_j \in S$
  $mean_{k_j} = \sum_{i=1}^{m} v_{i,k_j}/m, k_j \in S$
  for $k \in \{0, 1, \ldots, 2^{16} - 1\}$ **do**
    $\lambda_k = \sum_{j=1}^{n} (means_{k_j} - \mu_{k_j \oplus k})^2 / (\sigma^2_{k_j \oplus k}), k_j \in S$
  end for
  $keys = sort\_by\_\lambda_k(k)$(from smallest to largest)
  $S = keys$
end for
return S, L

For Algorithm 1, we provide a brief explanation. Firstly, the algorithm exists as a randomly generated set of candidate keys S and an empty set L. The subkey search process in the algorithm loops it times, and in each loop, we decrypt each ciphertext quadruple in the ciphertext set C by using the candidate keys k_j in S for one round and obtain the intermediate state quadruple after one round of decryption, which here we denote as P_{i,k_j}. P_{i,k_j} is put into the trained $MDNN$ for prediction, and an 8-round $MDNN$ is used for a 12-round key recovery attack. For each key k_j in S, we have the prediction scores v_{i,k_j} under $MDNN$ of the intermediate states P_{i,k_j} obtained by decrypting one round using these keys, and using Equation (1), we synthesize the prediction scores v_{i,k_j} under $MDNN$ of the m ciphertext quadruples decrypted under each key k_j, and we obtain a measure of how good or bad each key k_j is, w_{k_j}. We put all of the candidate keys k_j in the set S with their corresponding w_{k_j} into the set L. In the Model Response to Wrong Keys section, we derive the relationship between the mean and standard deviation of the model response and the key difference. Here, we compute the mean of the predicted scores v_{i,k_j} of the m ciphertext quadruples under each key k_j in the set S by $MDNN$, denoted by $mean_{k_j}$. Let k be the possible correct key, traverse it from 0 to $2^{16}-1$ and use the mean μ_Δ and standard deviation σ_Δ of the model response to wrong keys obtained previously, combined with the mean $mean_{k_j}$ of the predicted scores of the m ciphertexts decrypted by key k_j to compute the additive-full Euclidean distance λ_k of the guessing key k_j from the possible true key k:

$$\lambda_k = \sum_{j=1}^{n} (means_{k_j} - \mu_{k_j \oplus k})^2 / (\sigma^2_{k_j \oplus k})$$

After finishing the traversal, the n k with the smallest λ_k are put into the candidate subkey set S for the next round of the outer loop. After all the loops are finished, the set L of candidate subkey sets S with all the keys k stored with their w_{k_j} during the loop can be obtained.

In the algorithm, L is used to store the keys and their corresponding w_k. The number of keys to be stored is related to the number of iterations it as well as the size of the set of candidate keys S, which is $it \times n$. If there exists a key whose w_k is greater than the boundary $cutoff_1$, it will be extended to the penultimate round, and the process will be repeated. If a candidate key in the penultimate round has an w_k greater than the boundary $cutoff_2$, the key search will be terminated.

4.4. Results

In a 12-round key recovery attack experiment against SIMON32/64, we used 1024 ciphertext structures of 64 chosen plaintext pair encryptions each. The bounding value of the judgment $Cutoff_1 = 25$, $Cutoff_2 = 100$. The number of calls to the Bayesian key search algorithm is 512. Furthermore, the number of loop iterations within the Bayesian algorithm it is five. The size n of the candidate key set S is 32. So the number of keys kept in L is $32 \times 5 = 160$. During the overall experiment, we attempted to recover 100 keys, in a total of 21,090 s. The average time taken to recover a key was around 210.9 s. The probability that the last round of subkeys is completely correct and the error of the penultimate round of recovered subkeys is controlled to be between 2 bits during key recovery is 55%.

As for the complexity of the key recovery attack using $MDNN$, the data complexity is $2^{10} \times 4 \times 64 = 2^{18}$. This is because we use 2^{10} ciphertext structures, each of which is composed of 64 corresponding quadruple ciphertext groups. Time complexity is calculated in terms of SIMON32/64 encryption; our device can execute approximately $2^{25.06}$ 1-round encryption per second, so the final time complexity is $4 \times \frac{210.9}{12} \times 2^{25.06} = 2^{31.19}$. The complexity of the differential neural network distinguisher is calculated similarly.

In Table 3, we show the results of 12-round SIMON32/64 key recovery attacks using the differential neural network distinguisher and mixture differential neural network distinguisher, respectively. From the table, we can find that the key recovery attack using

the mixture differential neural network distinguisher with higher accuracy has higher data and time complexity but higher key recovery accuracy than the differential neural network distinguisher with lower accuracy.

Table 3. Key recovery accuracy.

Distinguishers	Data Complexity	Time Complexity	Recovery Acu.
Differential	2^{14}	$2^{29.45}$	52%
Mixture differential	2^{18}	$2^{31.19}$	55%

5. Conclusions

How to combine machine learning with cryptanalysis has been an important and much needed research topic. In this paper, we combine mixture differential analysis with machine learning to train eight-round $MDNN$. Compared with the differential neural network distinguisher, $MDNN$ can acquire more output differential features and therefore has higher prediction accuracy. We also find that the prediction accuracy of the differential neural network distinguisher is easily affected by the specified input difference chosen, while the mixture differential neural network distinguisher is less affected by the input difference and has stronger robustness. The basic principle of the attack using $MDNN$ and the specific effect are also illustrated by a 12-round key recovery attack on SIMON32/64. In the key recovery attack, the computation method of the wrong key response and the Bayesian optimization candidate subkey search strategy based on it are given. Comparing the attacks using the differential neural network distinguisher and $MDNN$, we find that $MDNN$ can achieve a higher key recovery accuracy; although, the data and time complexity are increased. When using the mixture differential neural network distinguisher for key recovery attacks, the data and time complexity is still a bit high, and the number of rounds covered is not long enough at the same time. Whether the architecture of the model can be optimized in order to increase the number of rounds covered by the distinguisher for more complex key recovery attacks will be the goal of the subsequent work.

Author Contributions: Conceptualization, K.Q. and Z.W. (Zehan Wu); methodology, K.Q. and Z.W. (Zehan Wu); software, Z.W. (Zehan Wu); visualization, Z.W. (Zhaoyang Wang); writing—original draft preparation, Z.W. (Zehan Wu); writing—review and editing, K.Q. and J.C.; supervision, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China (2022YFB3103800, 2023 YFB2704000), the National Natural Science Foundation of China (62102025), the Beijing Natural Science Foundation (4222035), the Open Project Fund of State Key Laboratory of Cryptology (MMKFKT202212) and Beijing Institute of Technology Research Fund Program for Young Scholars (XSQD-202024003).

Data Availability Statement: Data are contained within the article and the repository.

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

Abbreviations

The following abbreviations are used in this manuscript:

MILP	Mixed Integer Linear Programming
SAT	boolean SATisfiability problem
MDNN	Mixture Differential Neural Network distinguisher

References

1. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349. [CrossRef] [PubMed]
2. Abou Elassad, Z.E.; Mousannif, H.; Al Moatassime, H.; Karkouch, A. The application of machine learning techniques for driving behavior analysis: A conceptual framework and a systematic literature review. *Eng. Appl. Artif. Intell.* **2020**, *87*, 103312. [CrossRef]
3. Singh, S.P.; Kumar, A.; Darbari, H.; Singh, L.; Rastogi, A.; Jain, S. Machine translation using deep learning: An overview. In Proceedings of the IEEE 2017 International Conference on Computer, Communications Furthermore, Electronics (Comptelix), Jaipur, India, 1–2 July 2017; pp. 162–167.
4. Floridi, L.; Chiriatti, M. GPT-3: Its nature, scope, limits, and consequences. *Minds Mach.* **2020**, *30*, 681–694. [CrossRef]
5. Maghrebi, H.; Portigliatti, T.; Prouff, E. Breaking cryptographic implementations using deep learning techniques. In Proceedings of the Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, 14–18 December 2016; Proceedings 6; Springer: Berlin/Heidelberg, Germany, 2016; pp. 3–26.
6. Picek, S.; Samiotis, I.P.; Kim, J.; Heuser, A.; Bhasin, S.; Legay, A. On the performance of convolutional neural networks for side-channel analysis. In Proceedings of the Security, Privacy, and Applied Cryptography Engineering: 8th International Conference, SPACE 2018, Kanpur, India, 15–19 December 2018; Proceedings 8; Springer: Berlin/Heidelberg, Germany, 2018; pp. 157–176.
7. Gohr, A. Improving attacks on round-reduced speck32/64 using deep learning. In Proceedings of the Advances in Cryptology—CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2019; Proceedings, Part II 39; Springer: Berlin/Heidelberg, Germany, 2019; pp. 150–179.
8. Bao, Z.; Guo, J.; Liu, M.; Ma, L.; Tu, Y. Enhancing Differential-Neural Cryptanalysis. In *Proceedings of the Advances in Cryptology—ASIACRYPT 2022*; Agrawal, S., Lin, D., Eds.; Springer: Cham, Switzerland, 2022; pp. 318–347.
9. Lai, X. On the Design and Security of Block Ciphers. Ph.D. Thesis, ETH Zurich, Zürich, Switzerland, 1992.
10. Hatzivasilis, G.; Fysarakis, K.; Papaefstathiou, I.; Manifavas, C. A review of lightweight block ciphers. *J. Cryptogr. Eng.* **2018**, *8*, 141–184. [CrossRef]
11. Iqbal, W.; Abbas, H.; Daneshmand, M.; Rauf, B.; Bangash, Y.A. An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security. *IEEE Internet Things J.* **2020**, *7*, 10250–10276. [CrossRef]
12. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. Simon and Speck: Block Ciphers for the Internet of Things. In Proceedings of the NIST Lightweight Cryptography Workshop,Gaithersburg, MD, USA, 20–21 July 2015.
13. Grassi, L.; Rechberger, C.; Rønjom, S. A new structural-differential property of 5-round AES. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, 30 April–4 May 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 289–317.
14. Grassi, L. Mixture differential cryptanalysis: A new approach to distinguishers and attacks on round-reduced AES. *IACR Trans. Symmetric Cryptol.* **2018**, *2018*, 133–160. [CrossRef]
15. Grassi, L. Probabilistic mixture differential cryptanalysis on round-reduced AES. In Proceedings of the International Conference on Selected Areas in Cryptography, Waterloo, ON, Canada, 12–16 August 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 53–84.
16. Qiao, K.; Zhang, Z.; Niu, Z.; Zhu, L.; Ma, J. The Exchange Attack and the Mixture Differential Attack Revisited: From the Perspective of Automatic Evaluation. *Chin. J. Electron.* **2024**, *33*, 19–29. [CrossRef]
17. Bardeh, N.G.; Rønjom, S. The Exchange Attack: How to Distinguish Six Rounds of AES with Chosen Plaintexts. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, 8–12 December 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 347–370.
18. Sun, S.; Hu, L.; Wang, P.; Qiao, K.; Ma, X.; Song, L. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES (L) and other bit-oriented block ciphers. In Proceedings of the Advances in Cryptology—ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, 7–11 December 2014; Proceedings, Part I 20; Springer: Berlin/Heidelberg, Germany, 2014; pp. 158–178.
19. Chao-Hui, F.; Ming, D.; Qiang, W.; Qian-Qiong, W.; Rui, Z.; Heng-Chuan, S. Polytopic differential attack based on deep learning and its application. *J. Cryptol. Res.* **2021**, *8*, 591–600.
20. Tiessen, T. Polytopic cryptanalysis. In Proceedings of the Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, 8–12 May 2016; Proceedings, Part I 35; Springer: Berlin/Heidelberg, Germany, 2016; pp. 214–239.
21. Bao, Z.; Guo, J.; Liu, M.; Ma, L.; Tu, Y. Conditional Differential-Neural Cryptanalysis. *IACR Cryptol. ePrint Arch.* **2021**, *2021*, 719.
22. Baksi, A.; Baksi, A. Machine learning-assisted differential distinguishers for lightweight ciphers. In *Classical and Physical Security of Symmetric Key Cryptographic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 141–162.
23. Bao, Z.; Lu, J.; Yao, Y.; Zhang, L. More insight on deep learning-aided cryptanalysis. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, 4–8 December 2023; Springer: Berlin/Heidelberg, Germany, 2023; pp. 436–467.

24. 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; Proceedings, Part I 40; Springer: Berlin/Heidelberg, Germany, 2021; pp. 805–835.
25. Biham, E.; Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.* **1991**, *4*, 3–72. [CrossRef]
26. 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, 27–30 June 2016; pp. 770–778.
27. Biham, E.; Chen, R. Near-collisions of SHA-0. In Proceedings of the Advances in Cryptology–CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2004; Proceedings 24; Springer: Berlin/Heidelberg, Germany, 2004; pp. 290–305.
28. Pelikan, M.; Pelikan, M. Bayesian optimization algorithm. In *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 31–48.
29. Kaufmann, E.; Cappé, O.; Garivier, A. On Bayesian upper confidence bounds for bandit problems. In Proceedings of the Artificial Intelligence and Statistics, PMLR, La Palma, Spain, 21–23 April 2012; pp. 592–600.