On the Effects of Neural Network-based Output Prediction Attacks on the Design of Symmetric-key Ciphers*

Hayato Watanabe[1,2], Ryoma Ito[2], and Toshihiro Ohigashi[1,2]

[1] Tokai University, Minato-ku, Japan.
hayato04180625@gmail.com, ohigashi@tokai.ac.jp
[2] NICT, Koganei, Japan.
itorym@nict.go.jp

Abstract. Proving resistance to conventional attacks, e.g., differential, linear, and integral attacks, is essential for designing a secure symmetric-key cipher. Recent advances in automatic search and deep learning-based methods have made this time-consuming task relatively easy, yet concerns persist over expertise requirements and potential oversights. To overcome these concerns, Kimura et al. proposed neural network-based output prediction (NN) attacks, offering simplicity, generality, and reduced coding mistakes. NN attacks could be helpful for designing secure symmetric-key ciphers, especially the S-box-based block ciphers. Inspired by their work, we first apply NN attacks to Simon, one of the AND-Rotation-XOR-based block ciphers, and identify structures susceptible to NN attacks and the vulnerabilities detected thereby. Next, we take a closer look at the vulnerable structures. The most vulnerable structure has the lowest diffusion property compared to others. This fact implies that NN attacks may detect such a property. We then focus on a biased event of the core function in vulnerable Simon-like ciphers and build effective linear approximations caused by such an event. Finally, we use these linear approximations to reveal that the vulnerable structures are more susceptible to a linear key recovery attack than the original one. We conclude that our analysis can be a solid step toward making NN attacks a helpful tool for designing a secure symmetric-key cipher.

Keywords: Block cipher · Simon · Design rationale · Neural network · Output prediction attack

1 Introduction

One of the most essential metrics when designing a new symmetric-key cipher is to prove that the designed cipher is resistant to generic attacks for symmetric-key ciphers, such as differential [8], linear [24], impossible differential [7, 19], zero-correlation linear [9], and integral attacks [20]. A symmetric-key cipher generally consists of a combination of linear and nonlinear operations, and each operation has many design choices (e.g., the rotation parameter). This implies that designers must evaluate the security of all candidate ciphers that can be constructed from these choices; thus, designing a new cipher may be very time-consuming.

Automatic search methods based on the mixed integer linear programming (MILP), Boolean satisfiability problem (SAT), and constraint programming (CP) have rapidly developed over the past ten years as helpful tools that make this time-consuming task relatively easy, e.g., [15, 16, 26–29]. However, the use of these tools has the following concerns: 1) it requires high-level expertise in analysis and modeling methods; 2) modeling methods differ depending on the attack vectors; and 3) unknown vulnerabilities may be overlooked due to coding mistakes. Actually, there have been several cases where unknown vulnerabilities unexpected for the designers were discovered shortly after the new design specification was released. Consequently, the target cipher was subsequently broken. For example, a differential attack on SPEDDY [10], a differential attack on Friet [30], and an algebraic attack on Chaghri [23]. Along with the automatic search methods, deep learning-based analysis methods have rapidly developed over the past five years. To the best of our knowledge, these methods have never been used to design a new symmetric-key cipher. Then, this work examines the possibility of using these methods for such purposes.

* The part of this paper was presented at the 8th International Symposium on Cyber Security, Cryptology and Machine Learning (CSCML 2024).

Since Gohr [14] reported a new differential-neural cryptanalysis method at CRYPTO 2019, deep learning-based analysis methods for symmetric-key ciphers have significantly progressed, such as [2,3,5,6]. However, most existing methods mainly combined differential attacks and deep learning techniques. Thereby, the use of the existing methods may also have the same three concerns regarding the automatic search methods. To overcome these concerns, Kimura et al. [17,18] proposed a new method called neural network-based output prediction (NN) attacks, and the use of this method has the following three properties: 1) it does not require high-level expertise in analysis and modeling methods[1]; 2) it has the potential to deal with all attack vectors; and 3) it is less prone to coding mistakes because it requires only input/output interfaces (e.g., block sizes) as inputs. They applied NN attacks to S-box-based block ciphers and finally concluded that NN attacks have the potential to be a helpful tool for designing a new symmetric-key cipher, especially S-box-based ciphers.

This study aims to propose a step forward toward making NN attacks a helpful tool for designing all types of symmetric-key ciphers. To achieve this goal, we apply NN attacks to SIMON, which is one of the AND-Rotation-XOR-based block ciphers designed by the National Security Agency (NSA) [4]. When designing a SIMON-like cipher, many candidates can be considered depending on the combination of the three rotation parameters, i.e., (a, b, c); then, by varying the rotation parameters, we attempt to gain new insight into considering the design rationale of symmetric-key ciphers, especially SIMON-like ciphers.

Our study is inspired by Kölbl et al.'s and Kondo et al.'s works [21,22], and we aim to answer the following questions from a perspective to their works:

1. Which structures in SIMON-like ciphers are vulnerable to NN attacks?
2. What types of vulnerabilities are detected by NN attacks?
3. If a vulnerability is detected, can we figure out its root cause?
4. How resistant are the vulnerable ciphers to key recovery attacks?

Unraveling these questions will be a solid step toward making NN attacks a helpful tool for designing a new symmetric-key cipher.

Our contributions are summarized as follows:

Identifying Vulnerable Structures. In Sect. 3, we unravel the first question. Specifically, we compare the maximum number of rounds that NN attacks can be successful with the maximum number of rounds for building a differential/linear distinguisher. Then, we clarify which structures in SIMON-like ciphers are more vulnerable to NN attacks than differential and linear attacks. Our experimental results show that SIMON-like ciphers containing one of the following two cases have unknown vulnerabilities: 1) "$a = c$" or "$b = c$" and 2) "$n - a = c$" or "$n - b = c$", where n denotes the block size.

Comprehensive Analysis of the Vulnerable Structures. In Sect. 3.3, we consider the answer to the second question based on the above results. Specifically, we conduct additional experiments on vulnerable SIMON-like ciphers to explore them more deeply and discuss what types of vulnerabilities are detected by NN attacks. These vulnerabilities include not only differential and linear attacks but also impossible differential, zero-correlation linear, and integral attacks. As a result, NN attacks have the possibility of detecting vulnerabilities caused by integral attacks, and SIMON-like ciphers containing "$a = c$" or "$b = c$" and '0' or '$n/2$' have the most vulnerable structure.

Unraveling the Root Cause of Vulnerable Structures. In Sect. 4, we address the third question in the following two aspects. The first one is the well-known diffusion property. We demonstrate that the most vulnerable SIMON-like cipher has the lowest diffusion property compared to others. The second one is the effect of the biased output value of the core function in the target cipher. This biased output event enables us to find effective linear approximations. Based on our findings, the linear distinguisher with up to 30 rounds can be built for the most vulnerable SIMON-like cipher.

Towards Key Recovery on Vulnerable SIMON-like ciphers. In Sect. 5, we provide one possible answer to the fourth question. We use the linear approximation we found to perform a linear key recovery attack based on the well-known Matsui's algorithm 1 [24]. Our experimental results show that the vulnerable SIMON-like ciphers are vulnerable to the key recovery attack compared to the original one. In addition, we gain new insight into the fact that the attack ability does not improve as the number of samples increases. Solving this problem will be our future challenge.

[1] In other words, it enables us to code with basic knowledge of neural networks.

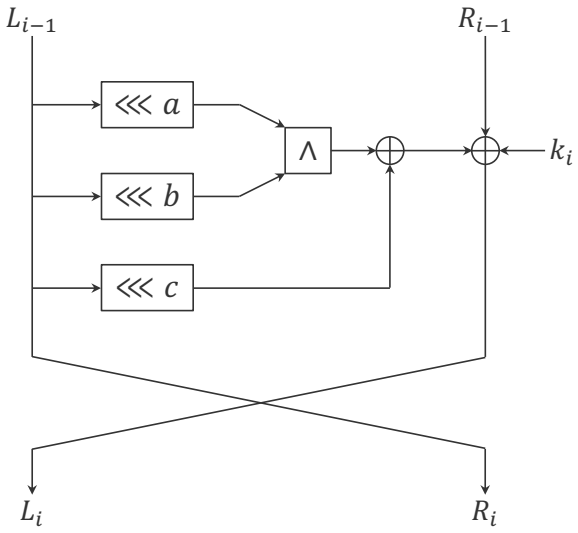


Fig. 1: Round function of SIMON.

2 Preliminaries

2.1 Specification of SIMON

SIMON, designed by NSA in 2013 [4], is a family of lightweight block ciphers based on the Feistel construction. It has ten variants combining a $2n$-bit block and an mn-bit secret key, where $n \in \{16, 24, 32, 48, 64\}$ and $m \in \{2, 3, 4\}$. The SIMON variant is denoted as SIMON$2n/mn$ in general, but for simplicity, we use SIMON$2n$ throughout this paper because the key length is out of scope for our analysis. The main target of this study is SIMON32 with 32 rounds. In addition, to demonstrate the validity of our analysis, we design SIMON16 as a toy model of the SIMON variants.

As illustrated in Fig. 1, the round function of SIMON is composed of three n-bit operations: AND ($\wedge$), left rotation ($\lll$), and XOR ($\oplus$). Let x be an n-bit input of the core function f in the round function; then, f is defined as

$$f(x) = \big((x \lll a) \ \wedge \ (x \lll b)\big) \oplus (x \lll c), \tag{1}$$

where (a, b, c) denotes a rotation parameter, and $(a, b, c) = (1, 8, 2)$ is used for the original SIMON32. For simplicity, we refer to SIMON$2n$ with all possible rotation parameters as SIMON$2n$ variants throughout this paper[2]. Besides, let (L_{i-1}, R_{i-1}) be a $2n$-bit input of the i-th round function; then, its output (L_i, R_i) is computed as

$$L_i = f(L_{i-1}) \oplus R_{i-1} \oplus k_i, \tag{2}$$
$$R_i = L_{i-1}, \tag{3}$$

where k_i denotes the subkey for the i-th round. For the r-round SIMON, the plaintext is (L_0, R_0), and the ciphertext is (L_r, R_r).

2.2 Considerations on the Design Rationale of SIMON Variants

Kölbl, Leander, and Tiessen [21] considered the design rationale of SIMON variants from the following three aspects. First, they measured the number of rounds after which full diffusion is reached for all SIMON variants

[2] Note that this is a different notation from SIMON variants, which include the original versions of SIMON32, SIMON48, SIMON64, SIMON96, and SIMON128.

with all possible rotation parameters. Next, they computed the optimal differential/linear characteristic for 10 rounds of SIMON32, SIMON48, and SIMON64 variants that satisfy $a > b$ and $\gcd(a-b, n) = 1$; then, they listed the optimal 20 parameters with respect to 10-round differential characteristics (see Appendix C in [21] for details). Finally, they conducted further evaluations to obtain differential characteristics for SIMON32, SIMON48, and SIMON64 variants with the following three example parameters: $(12, 5, 3)$, $(7, 0, 2)$, and $(1, 0, 2)$. In conclusion, they clarified that the original rotation parameter may not always be optimal.

Kondo, Sasaki, Todo, and Iwata [22] focused on the design rationale for SIMON32 and extended Kölbl et al.'s work [21] to integral and impossible differential attacks against SIMON32 variants. Regarding integral attacks, they used a supercomputer to evaluate the number of rounds for building integral distinguishers with 2^{31} plaintexts. Regarding impossible differential attacks, they used a general-purpose computer to explore the number of rounds for building impossible differential distinguishers with the miss-in-the-middle approach. Based on these results, they classified SIMON32 variants into 20 groups (see Table 6 in [22] for details) and clarified which parameter is optimal for SIMON32 against integral and impossible differential attacks.

2.3 Neural Network-based Output Prediction Attacks

Kimura et al. [17] proposed neural network-based output prediction (NN) attacks in a black-box setting, targeting ciphertext prediction and plaintext recovery. They used LSTM (long short-term memory) as a neural network model and applied the proposed attacks to the following three toy block ciphers with a 16-bit block size: two toy SPN block ciphers (small PRESENT and small AES) and one toy Feistel block cipher (small TWINE). Then, based on the analysis results for the toy block ciphers, the proposed attacks were applied to the following three block ciphers with 32- and 64-bit block sizes: PRESENT, AES-like, and TWINE-like ciphers. In conclusion, for PRESENT, they demonstrated that the proposed attacks are comparable to building the differential and linear distinguishers. For AES-like and TWINE-like ciphers, they conjectured that the proposed attacks are also comparable to building the differential and linear distinguishers when the amount of training data is increased more.

In their subsequent study [18], they extended their previous work [17] to explore clues for designing symmetric-key ciphers that are secure against NN attacks. To this end, they employed two weak small PRESENT variants by replacing the original S-box with weak S-boxes, which are known to be vulnerable to differential and linear attacks, and applied NN attacks to those variants in the same way as their previous work. In conclusion, they demonstrated that NN attacks can be effective in estimating the number of rounds for building differential and/or linear distinguishers. Still, it remains unclear how to provide feedback on their results for designing NN-resistant symmetric-key ciphers.

2.4 SAT/CP-based Automatic Search Methods

We extend the existing works [17, 18, 21, 22] and deeply consider the design rationale for SIMON32 from the perspective of NN attacks. Our analysis aims to clarify what vulnerabilities caused by classical attacks are detected by NN attacks; then, this will help us decide which parameter should not be chosen when designing Simon-like ciphers. To this end, we use existing SAT/CP-based automatic search tools. Fortunately, many automatic tools are available as open-source to search for various types of distinguishers using generic attacks for symmetric-key ciphers, such as differential, linear, impossible differential, zero-correlation linear, and integral attacks. Then, we customize and use the following three tools to analyze SIMON32 variants that satisfy $a > b$ and $\gcd(a-b, n) = 1$[3]:

– To evaluate the maximum number of rounds for building differential and linear distinguishers, we use the SAT-based tool proposed by Sun et al. at IACR ToSC 2021(1) [29]. The source codes are available at GitHub[4].

[3] These conditions must be specified to search for differential characteristics by using Sun et al.'s methods [29].
[4] https://github.com/SunLing134340/Accelerating_Automatic_Search

– To evaluate the maximum number of rounds for building impossible differential and zero-correlation linear distinguishers, we use the CP-based tool proposed by Hadipour et al. at IACR ToSC 2024(1) [16]. The source codes are available at GitHub[5].
– To evaluate the maximum number of rounds for building integral distinguishers, we use the SAT-based tool proposed by Hadipour et al. at IACR ToSC 2022(2) [15]. The source codes are available at GitHub[6].

2.5 Complexity Estimation and Success Probability

Although NN attacks target ciphertext prediction and plaintext recovery, they can also be evaluated as a type of distinguishing attack. In other words, if NN attacks can predict a certain output bit string with a higher probability than a random prediction, we consider the attack to be successful.

To estimate the number of samples and success probability to distinguish two distributions with respect to an output bit string, we use the following theorem provided by Baignères et al. at ASIACRYPT 2004 [1].

Theorem 1 ([1]). *Let $Z_1, \ldots, Z_n$ be independent and identically distributed random variables over the set $\mathcal{Z}$ of distribution D, D_0 and D_1 be two distributions of same support which are close to each other, and n be the number of samples of the best distinguisher between $\mathsf{D} = \mathsf{D}_0$ or $\mathsf{D} = \mathsf{D}_1$. Let d be a real number such that*

$$n = \frac{d}{\sum_{z \in \mathcal{Z}} \frac{\epsilon_z^2}{p_z}}, \tag{4}$$

where p_z and $p_z + \epsilon_z$ are probabilities of a random variable z following D_0 and D_1, respectively. Then, the overall probability of error is $P_e \approx \Phi(-\sqrt{d}/2)$, where $\Phi(\cdot)$ is the distribution function of the standard normal distribution.

Let D_0 and D_1 be a distribution of the result of random prediction and a distribution of the result of NN attacks, respectively. In this case, the target event occurs in D_0 and D_1 with probabilities of $\frac{1}{2}$ and $\frac{1}{2} \cdot (1+\epsilon)$, respectively (i.e., $p_0 = p_1 = \frac{1}{2}$, $|\epsilon_0| = 0$, and $|\epsilon_1| = \frac{\epsilon}{2}$). Based on this, the number of samples of the best distinguisher between $\mathsf{D} = \mathsf{D}_0$ and $\mathsf{D} = \mathsf{D}_1$ can be estimated as $2d\epsilon^{-2}$ with an overall error probability of $P_e \approx \Phi(-\sqrt{d}/2)$; thus, the success probability can be estimated as $1 - P_e$.

3 On the Effects of NN attacks on Simon32 Variants

In this section, we conduct some experiments on NN attacks and five generic attacks for symmetric-key ciphers (e.g., differential, linear, impossible differential, zero-correlation linear, and integral attacks) and discuss the design rationale for Simon-like ciphers. First, we apply SAT/CP-based automatic search methods described in Sect. 2.4 to Simon32 variants and derive the maximum number of rounds that can build a distinguisher for each generic attack. Next, we mainly focus on the experimental results of differential and linear attacks and classify Simon32 variants into four groups for each maximum number of rounds that can build their distinguishers. We randomly pick up 32 Simon32 variants for each group and apply NN attacks described in Sect. 2.3 to these variants. Then, we compare the experimental results of NN attacks with the experimental results of differential and linear attacks and clarify which Simon32 variant is more vulnerable to NN attacks than differential and linear attacks. In other words, this implies that NN attacks may detect vulnerabilities caused by attacks other than differential and linear attacks. Finally, to take a deeper look at Simon32 variants with vulnerable parameters, we conduct additional experiments on these variants and discuss what types of vulnerabilities are detected by NN attacks. These vulnerabilities include not only differential and linear attacks but also impossible differential, zero-correlation linear, and integral attacks. This allows us to clarify that NN attacks can be a helpful tool for designing symmetric-key ciphers, especially Simon-like ciphers.

[5] https://github.com/hadipourh/zeroplus
[6] https://github.com/hadipourh/mpt

3.1 Revisiting Generic Attacks on SIMON32 Variants

As described in Sect. 2.2, Kölbl et al. [21] considered the design rationale of SIMON variants from the perspective of differential and linear attacks. Inspired by their work, Kondo et al. [22] took a deeper look at SIMON32 variants from the perspective of integral and impossible differential attacks.

One of our goals in this study is to clarify which vulnerabilities caused by the classical attacks are detected by NN attacks. To achieve this goal, we compare NN attacks with five generic attacks for symmetric-key ciphers (e.g., differential, linear, impossible differential, zero-correlation linear, and integral attacks) from the perspective of the maximum number of attackable rounds. In preparation for that, we first need to obtain the maximum number of rounds that can build distinguishers based on the generic attacks. We could have used experimental results presented by Kölbl et al. and Kondo et al., but they are not all publicly available; thus, we have to obtain the experimental results ourselves.

We customize and use the existing SAT/CP-based automatic search tools described in Sect. 2.4 to comprehensively analyze the security of SIMON32 variants against five generic attacks for symmetric-key ciphers. The overview of our experimental results is as follows:

- The maximum number of rounds that can build differential and linear distinguishers with 2^{15} data[7] is 15, 11, 8, or 7 rounds.
- The range of the maximum number of rounds that can build impossible differential and zero-correlation linear distinguishers is from 9 to 17 rounds.
- The range of the maximum number of rounds that can build an integral distinguisher with 2^{31} data is from 14 to 32 rounds.

For a fair comparison, we understand that it is desirable to use 2^{15} data to search for valid integral distinguishers because we use 2^{15} training data in all our experiments for NN attacks. However, we estimated that tightly evaluating the maximum number of rounds that can build integral distinguishers with 2^{15} data for all possible SIMON32 variants would be a very time-consuming task. Indeed, this task requires far more than 2^{42} trials[8]. For this reason, we use 2^{31} data instead of 2^{15} data to search for valid integral distinguishers. One of our goals in this study is to capture the relationships between NN and integral attacks. To achieve this goal, we believe that using 2^{31} data is sufficient to search for valid integral distinguishers.

As mentioned at the beginning of this section, we first focus on the experimental results of differential and linear attacks and classify SIMON32 variants into four groups based on the experimental results of differential and linear attacks. More specifically, we assign SIMON32 variants whose maximum number of rounds that can build differential and linear distinguishers is 15, 11, 8, and 7 to the groups $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$, respectively.

3.2 NN attacks on SIMON32 Variants

In this subsection, we randomly pick up 32 SIMON32 variants for each group classified in the previous subsection and apply NN attacks (especially a plaintext prediction attack) to these variants. First, we explain the hyperparameters used for all our experiments, the experimental procedure, the experimental scenario, and the conditions for a successful attack. Next, we show the experimental results of NN attacks on the target variants. Finally, to confirm the correctness of our experimental results, in the same way as above, we conduct experiments on NN attacks for SIMON16 designed as a toy cipher.

Hyperparameters and Experimental Procedure Basically, we follow the hyperparameters and experimental procedure explained in Sect. 3.1 of Kimura et al.'s paper [17].

[7] The reason why the data complexity is limited to 2^{15} is that the number of samples used for NN attacks is 2^{15}, and we mainly compared the ability of differential and linear attacks with that of NN attacks.

[8] The SAT tool we used requires us to specify bit positions with CONSTANT properties in the input. In other words, using 2^{15} data implies that we must specify 17 CONSTANT bit positions out of the 32-bit input; thus, the number of their combinations is approximately 2^{30}. In addition, the total number of possible SIMON32 variants is approximately 2^{12}. Given that trials are performed for each round of the target variants, it is obvious that the total number of trials will be far more than 2^{42}.

Table 1: Hyperparameters used for all our experiments.

Number of training data†	2^{15}
Number of test data†	2^{15}
Method	LSTM (Long short-term memory)
Number of input layer nodes (i.e., block sizes)	16, 32
Number of output layer nodes (i.e., block sizes)	16, 32
Number of hidden nodes	300
Number of hidden layers	1
Loss function	Mean squared error
Optimizers	Adam
Initial value of learning rates	0.01
Batch size	250
Number of epochs	100

† Number of plaintext/ciphertext pairs (no duplication between training and test data).

Hyperparameters Unlike Kimura et al.'s work [17], we do not conduct the hyperparameter optimization (i.e., Experiment 1 explained in Sect. 3.1 of [17]) because it is difficult to determine the optimal hyperparameters for all target variants. Instead, we use the hyperparameters for the plaintext recovery attack against the 4-round small PRESENT listed in Table 3 of [17]. This is because the success rate of NN attacks against small PRESENT is the highest among the target ciphers, and the maximum number of rounds where the plaintext recovery attack against small PRESENT is successful is four rounds. Table 1 lists hyperparameters used for our experiments.

As listed in Table 1, we limit the number of training data to 2^{15} in all our experiments. This may seem like a somewhat small amount of data. Indeed, Kimura et al. demonstrated in Sect. 3.2 of [17] that increasing the number of training data improves the accuracy of NN attacks. However, our analysis aims to create neural network models for many target SIMON32 variants under the same conditions (including the number of training data) and to capture some features of which SIMON32 variants are vulnerable to NN attacks, rather than attempting to improve the accuracy of NN attacks. We have confirmed in preliminary experiments (e.g., A.1) that using 2^{15} training data is sufficient to capture the features. Therefore, we decided to use 2^{15} training data in all our experiments.

Experimental Procedure The NN attack is implemented with Keras and TensorFlow, both of which are open-sourced neural network libraries. Our experimental environment consists of four Linux machines, eight NVIDIA Tesla K40M GPUs, and the Python programming language. The experiment involves the following steps:

Step 1. Generate datasets so that there is no duplication between training and test data (See Listings 1.1 and 1.2 in B for more details).

Step 2. Create an LSTM model (See Listing 1.3 in B for more details).

Step 3. Compile the LSTM model. We specify only the loss function (`loss`) and optimizer (`optimizer`) as the option, but `metrics` is not specified because we evaluate experimental results with the average match rate. The details of the average match rate are explained in Sect. 3.2.

Step 4. Train the LSTM model. We specify only the training data (`x` and `y`), batch size (`batch_size`), and number of epochs (`epochs`) as the option.

Step 5. Use test data (ciphertext) to predict the plaintext bit by bit. Predictions are rounded to the nearest whole number, resulting in 0 or 1.

Step 6. Compare the predicted values with the real plaintext values for all test data, and calculate their average match rates.

Our experiments cover 1 to 16 rounds of SIMON32 variants and are conducted with 10 trials for each round[9]. In other words, we use 10 randomly chosen keys rather than 100 randomly chosen keys used with Kimura et al.'s experiments (i.e., Experiment 2 explained in Sect. 3.1 of [17]) due to the execution time for our experiments. Kimura et al. demonstrated in Sect. 3.3 of [17] that the experiments with a small number of secret keys are sufficient to obtain the best average success probability. More specifically, they compared experimental results using 100 randomly chosen keys with those using 10000 randomly chosen keys and demonstrated that using 100 randomly chosen keys is sufficient to obtain reliable results. Inspired by their work, we have confirmed in preliminary experiments that using even fewer keys (e.g., 10 randomly chosen keys) is also sufficient to obtain reliable results. The detailed results of the preliminary experiments are provided in A.1. To summarize, our preliminary experiments demonstrate that reducing the number of keys will have little impact on our experiments.

Experimental Scenario In the context of neural networks, it is crucial to be clear on how to generate training and test datasets. To clarify this, we discuss the specific secret key setting used in the data generation process. Intuitively, when the test dataset is generated using a distinct secret key than the one used to generate the training dataset, the trained model may not work effectively on the test dataset. Similar results are likely to be obtained when the training dataset is generated using distinct multiple secret keys. We conducted preliminary experiments to validate these hypotheses, and the detailed results are provided in A.2. To summarize the results, when a distinct secret key is employed for generating the training and test datasets, the prediction accuracy of the trained model is observed to degrade significantly, and similar results are obtained when distinct multiple secret keys are used for generating the training dataset. In light of these results, all subsequent experiments are conducted under the single-key setting, which means that both the training and test datasets for each experiment are generated using the same single secret key to ensure consistency and validity.

Next, we explain the attack scenario. NN attacks are performed under the known plaintext attack scenario. This attack scenario matches that of linear and zero-correlation linear attacks but does not match that of differential, impossible differential, and integral attacks. We aim to identify whether the neural network model can capture some features from the given training data even when the attack scenario is mismatched.

Then, we present the evaluation metric. The success or failure of the NN attacks is evaluated using the average match rate as a metric, the details of which are explained in Sect. 3.2. Based on the average match rate, we compare NN attacks with classical attacks in terms of the maximum number of attackable rounds to examine whether their relationship can be captured.

Finally, we describe the attack target. The target of NN attacks is plaintext recovery, while the target of classical attacks is distinguishing. This difference implies that NN attacks are performed under stronger assumptions than classical attacks, making it difficult for them to obtain better results. Despite this, if NN attacks outperform classical attacks in terms of the maximum number of attackable rounds, they have likely captured vulnerabilities caused by some classical attacks. Based on this, we identify a deep relationship between the NN and classical attacks.

Conditions for a Successful NN Attack As a condition for determining the success or failure of NN attacks, Kimura et al. [17] defined the exact match rate, which is the probability that the predicted and true values exactly match for a $2n$-bit output string. Specifically, when the block size is $2n$ bits, and the number of training data is 2^x, the attack is considered successful if the exact match rate is higher than $(2^{2n} - 2^x)^{-1}$. This condition is valid when $2^{2n} = 2^x + 2^y$, where 2^y is the number of test data. For example, we can use this condition in our experiments on SIMON16 variants. However, when it's not, since we limit $2^x = 2^y = 2^{15}$, i.e., $2^{2n} > 2^x + 2^y$, we cannot precisely compute the exact match rate; thus, we cannot strictly determine the success or failure of the attack on SIMON32 variants.

To tackle this problem, we define the average match rate. Specifically, the average match rate is derived by computing the predicted probability for each bit and then calculating the average value of all the predicted

[9] If NN attacks are successful for up to 16 rounds, we continue the experiment by increasing the number of rounds until the attack fails.

Table 2: Comparison of the maximum number of rounds that can build differential/linear distinguishers ('D/L' column) and the maximum number of rounds that can be successful for NN attacks ('NN' column) for each SIMON32 variant.

Group	D/L	NN	Rotation parameter
$\mathcal{A}$	15	27	(13,8,13)
		26	(9,8,9),(8,7,8)
		25	(8,1,1)
		13	(4,1,4),(5,4,5),(10,7,10),(12,1,12),(12,3,3),(14,5,14),(15,4,4),(15,10,10),(15,12,12)
		12	(4,3,3),(5,2,2),(5,2,5),(9,2,2),(10,1,1),(10,9,10),(11,2,2),(13,10,10)
		11	(2,1,1),(2,1,2),(3,2,2),(3,2,3),(4,3,4),(6,1,1),(6,3,6),(6,5,6),(7,2,2),(7,6,6),(7,6,7)
$\mathcal{B}$	11	7	(2,1,8),(3,2,8),(4,1,8),(4,3,8),(5,2,8),(5,4,8),(6,1,8),(6,3,8),(6,5,8),(7,2,8),(7,4,8), (7,6,8),(9,2,8),(9,4,8),(9,6,8),(10,1,8),(10,3,8),(10,5,8),(10,7,8),(12,5,8),(12,11,8), (13,4,8),(14,1,2),(14,11,8),(14,13,8),(15,6,8),(15,10,8)
		4	(11,6,8),(11,10,8),(13,6,8),(14,7,8),(15,4,8)
$\mathcal{C}$	8	9	(12,5,4)
		8	(4,1,12),(5,4,12),(12,1,4)
		5	(2,1,10),(4,3,11),(4,3,12),(9,2,1),(9,8,1)
		4	(4,1,9)
		2	(3,2,4),(3,2,11),(3,2,13),(3,2,14),(4,1,15),(8,7,9),(9,2,12),(9,4,1),(9,6,1),(10,1,12), (10,7,2),(11,8,12),(11,10,3),(13,2,5),(14,1,15),(14,5,13),(15,8,7),(15,10,12)
		1	(2,1,9),(3,2,12),(6,1,4),(7,6,12)
$\mathcal{D}$	7	3	(3,2,5)
		2	(2,1,6),(2,1,11),(3,2,1),(3,2,7),(4,1,2),(4,1,11),(4,1,14),(5,4,7),(5,4,14),(6,1,5), (6,5,7),(7,2,6),(7,2,13),(7,4,13),(8,1,11),(8,5,7),(8,5,15),(9,2,3),(9,4,2),(9,4,3), (9,8,13),(10,1,5),(10,3,5),(10,9,5),(11,2,1),(11,2,9),(11,8,15),(15,12,9),(15,12,10), (15,14,10),(15,14,13)

probabilities for $2n$ bits. For example, if the average match rate is 2^{-1}, we can use 2^{-32} as the approximated value of the exact match rate, especially for NN attacks on SIMON32 variants.

Here, we follow Theorem 1 and define a condition for determining the success or failure of our experiments using the average match rate. Assuming that the success probability of distinguishing attacks is 0.7, we can get $d \approx 1.12$ from the theorem. Then, since the number of test data is $2^{15} = 2d\epsilon^{-2}$, we can get $\epsilon \approx 0.00826$; thus, we can also get $2^{-1} \cdot (1 + \epsilon) \approx 0.504$, which can be considered as a distribution of the result of NN attacks. This means that NN attacks can be successful with a probability of 0.7 when the average match rate is higher than 0.504. To summarize, we consider a slight margin and define a successful attack when the average match rate is 0.505 or higher.

Experimental Results We conduct experiments on NN attacks for SIMON32 variants and classify our experimental results into four groups, as listed in Table 2. Specifically, this table shows a comparison of the maximum number of rounds that can build differential/linear distinguishers ('D/L' column) and the maximum number of rounds that can be successful for NN attacks ('NN' column) for each SIMON32 variant, where 'Group' column is as defined in Sect. 3.1. As a concrete example, the details of our experimental results for a SIMON32 variant with $(4, 1, 12)$ in Table 3. It can be seen from this table that the attack can be successful up to eight rounds from the perspective of both differential and linear characteristic probabilities (see 'DCP' and 'LCP' columns)[10], indicating that the abilities of differential, linear, and NN attacks are

[10] Strictly speaking, differential (resp. linear) probabilities should be considered here, but since DL-based analysis is expected to detect single differential (resp. linear) characteristic rather than the clustering effect of differential (resp. linear) characteristics, we consider differential (resp. linear) characteristic probabilities in this study.

Table 3: Experimental results of NN attacks for a SIMON32 variant with $(4, 1, 12)$. ''DCP‘ and LCP' stand for differential and linear characteristic probabilities, respectively.

Round	DCP	LCP	Average match rate	Exact match rate	success (✓) or failure (–)
1	1	1	0.99047	$2^{-0.44}$	✓
2	2^{-2}	2^{-2}	0.57558	$2^{-25.5}$	✓
3	2^{-4}	2^{-4}	0.55783	$2^{-26.9}$	✓
4	2^{-6}	2^{-6}	0.54309	$2^{-28.2}$	✓
5	2^{-8}	2^{-8}	0.52027	$2^{-30.2}$	✓
6	2^{-10}	2^{-10}	0.51634	$2^{-30.5}$	✓
7	2^{-12}	2^{-12}	0.51215	$2^{-30.9}$	✓
8	2^{-14}	2^{-14}	0.50619	$2^{-31.4}$	✓
9	2^{-18}	2^{-16}	0.50413	$2^{-31.6}$	–

consistent. We have obtained similar results to this example for all target variants and have reflected those results in Table 2.

It can be seen from Table 2 that NN attacks on SIMON32 variants have the following features:

- For the groups $\mathcal{A}$ and $\mathcal{C}$, we have identified several cases where NN attacks outperform or are comparable to differential and linear attacks in terms of the maximum number of attackable rounds. More specifically, SIMON32 variants with $\{(8, 1, 1), (8, 7, 8), (9, 8, 9), (13, 8, 13)\}$ in the group $\mathcal{A}$ and with $\{(12, 5, 4), (4, 1, 12), (5, 4, 12), (12, 1, 4)\}$ in the group $\mathcal{C}$ fall into that case. This result implies that NN attacks might be able to capture vulnerabilities caused by some classical attacks, excluding differential and linear attacks. Therefore, we conclude that SIMON32 variants in the groups $\mathcal{A}$ and $\mathcal{C}$ may contain vulnerable structures.
- For the groups $\mathcal{B}$ and $\mathcal{D}$, we have not identified any cases where NN attacks outperform differential and linear attacks in terms of the maximum number of attackable rounds. More specifically, for the group $\mathcal{B}$ (resp. $\mathcal{D}$), the maximum number of rounds for building differential/linear distinguishers is 11 (resp. 7), whereas the maximum number of attackable rounds for NN attacks is 7 or 4 (resp. 3 or 2). This result implies that NN attacks have not captured any vulnerabilities caused by classical attacks, including differential and linear attacks. Therefore, we conclude that all SIMON32 variants in the groups $\mathcal{B}$ and $\mathcal{D}$ do not have a vulnerable structure.

We will take a deeper look at these features, especially for SIMON32 variants in the groups $\mathcal{A}$ and $\mathcal{C}$, in Sect. 3.3.

Experimental Verification In our experiments for SIMON32 variants, we have determined the success or failure of the attacks using the average match rate. To verify the correctness of this condition, we conduct experiments on NN attacks for all SIMON16 variants in the same way as above. We can use the exact match rate as a condition for determining the success or failure of the attack in the experiments for SIMON16 variants; thus, our analysis of SIMON32 variants can be considered valid if the experimental results of SIMON16 variants are similar to those of SIMON32 variants.

Our experimental verifications are classified into four groups in the same manner as the case of SIMON32 variants, as listed in Table 4. It can be seen from this table that NN attacks on all SIMON16 variants have the following features:

- For the groups $\mathcal{A}$ and $\mathcal{C}$, we have identified several cases where NN attacks outperform or are comparable to differential and linear attacks in terms of the maximum number of attackable rounds. More specifically, 18 SIMON32 variants in the group $\mathcal{A}$ (i.e., $\{(4, 1, 4), \ldots, (7, 6, 7)\}$) and 8 SIMON32 variants in the group $\mathcal{C}$ (e.g., $\{(3, 2, 6), \ldots, (7, 6, 1)\}$) fall into that case.
- For the groups $\mathcal{B}$ and $\mathcal{D}$, we have not identified any cases where NN attacks outperform differential and linear attacks in terms of the maximum number of attackable rounds. More specifically, for the group $\mathcal{B}$

Table 4: Comparison of the maximum number of rounds that can build differential/linear distinguishers ('D/L' column) and the maximum number of rounds that can be successful for NN attacks ('NN' column) for each SIMON16 variant.

Group	D/L	NN	Rotation parameter
$\mathcal{A}$	15	29	(4,1,1),(5,4,4),(5,4,5),(7,0,0)
		28	(1,0,0),(3,0,0),(5,0,5),(7,0,7)
		27	(3,0,3),(4,1,4),(5,0,0)
		26	(1,0,1),(4,3,3),(4,3,4),(7,4,7)
		24	(7,4,4)
		15	(6,1,1),(7,6,7)
		14	(2,1,2),(6,5,6)
		13	(2,1,1),(3,2,2),(5,2,2),(5,2,5),(6,3,6),(6,5,5),(7,2,2),(7,2,7),(7,6,6)
		12	(3,2,3),(6,1,6),(6,3,3)
		7	(2,1,0)
$\mathcal{B}$	11	10	(4,3,0),(5,0,4)
		9	(1,0,4),(3,0,4),(3,2,0),(4,1,0),(5,2,4),(5,4,0),(6,5,4),(7,0,4),(7,4,0)
		8	(2,1,4),(3,2,4),(6,1,0),(6,5,0),(7,6,4)
		7	(5,2,0),(6,1,4),(6,3,0),(6,3,4),(7,2,0),(7,2,4),(7,6,0)
$\mathcal{C}$	8	10	(3,2,6)
		9	(6,3,2),(7,2,6)
		8	(2,1,6),(2,1,7),(5,2,6),(6,5,2),(7,6,1)
		7	(6,1,2)
		5	(3,0,7),(4,1,5),(5,2,3),(5,4,1),(6,1,7),(6,3,5),(7,0,3),(7,2,1)
		4	(1,0,3),(1,0,5),(1,0,7),(2,1,5),(3,0,1),(3,0,5),(3,2,5),(3,2,7),(4,1,7),(4,3,7),(5,0,1), (5,2,1),(5,4,3),(6,1,5),(6,3,7),(6,5,1),(6,5,3),(7,2,3),(7,4,3),(7,6,3)
		3	(3,0,2),(4,3,5),(5,0,3),(5,4,6),(7,0,1),(7,4,1),(7,6,1)
		2	(1,0,2),(1,0,6),(3,0,6),(4,1,2),(4,1,6),(4,3,2),(4,3,6),(5,0,2),(5,0,6),(5,4,2),(7,0,2), (7,0,6),(7,4,2),(7,4,6)
$\mathcal{D}$	7	4	(6,1,3),(6,3,1),(6,5,7),(7,0,5),(7,2,5),(7,4,5),(7,6,5),(2,1,3),(3,2,1),(4,1,3),(4,3,1), (5,0,7),(5,2,7),(5,4,7)

(resp. $\mathcal{D}$), the maximum number of rounds for building differential/linear distinguishers is 11 (resp. 7), whereas the maximum number of attackable rounds for NN attacks is in the range of 7 to 10 (resp. 4).

To summarize, the experimental verifications of SIMON16 variants are similar to those of SIMON32 variants; therefore, it can be concluded that the experimental results of SIMON32 variants can be considered valid.

3.3 Discussions

In this subsection, based on the experimental results presented in Sect. 3.2, we discuss which structures of SIMON32 variants are vulnerable to NN attacks and then which types of vulnerabilities are detected by NN attacks.

Identifying Vulnerable Structures of SIMON32 Variants In Sect. 3.2, we have clarified that NN attacks outperform or are comparable to differential and linear attacks against SIMON32 variants with $\{(8,1,1), (8,7,8), (9,8,9), (13,8,13)\}$ in the group $\mathcal{A}$ and with $\{(12,5,4), (4,1,12), (5,4,12), (12,1,4)\}$ in the group $\mathcal{C}$. In addition, we have obtained similar results for SIMON16 variants. These results show that SIMON32 variants (and SIMON16 variants) which make them more vulnerable to NN attacks than differential and linear attacks have the following features:

Table 5: Comparison of the maximum number of rounds that can build differential/linear ('D/L' column), impossible differential/zero-correlation linear ('ID/ZC' column), and integral ('I' column) distinguishers and the maximum number of rounds that can be successful for NN attacks ('NN' column) for each SIMON32 variants with rotation parameters that satisfy "$a = c$" or "$b = c$".

Group	D/L	ID/ZC	I	NN	Rotation parameter
$\mathcal{A}$	15	17	32	29	(1,0,0)
				28	(1,0,1),(8,7,7),(11,8,11),(13,8,8),(15,0,0),(15,0,15)
				27	(3,0,0),(3,0,3),(5,0,0),(5,0,5),(7,0,0),(7,0,7),(8,1,8),(8,3,3),(8,3,8),(8,5,5),(8,5,8),(9,0,0),(9,0,9),(9,8,8),(11,0,11),(11,8,8),(13,0,0),(13,8,13),(15,8,8),(15,8,15)
				26	(8,7,8),(9,8,9),(11,0,0),(13,0,13)
				25	(8,1,1)
			20	13	(2,1,1),(2,1,2),(5,2,5),(6,3,3),(10,5,5),(13,2,13),(14,7,14),(15,14,14),(15,14,15)
				12	(5,2,2),(9,2,2),(9,2,9),(10,1,1),(10,1,10),(10,5,10),(10,9,9),(10,9,10),(11,6,11),(13,10,10),(13,10,13),(14,11,11),(14,11,14),(15,6,15)
				11	(6,3,6),(7,6,6),(7,6,7),(11,6,11),(13,2,2),(14,3,3),(14,3,14),(14,7,7),(15,6,6)
		13	20	13	(4,1,1),(4,1,4),(5,4,4),(5,4,5),(7,4,4),(7,4,7),(9,4,4),(9,4,9),(11,4,4),(11,4,11),(12,1,1),(12,1,12),(12,3,3),(12,3,12),(12,5,5),(12,5,12),(12,7,7),(12,7,12),(12,9,9),(12,9,12),(12,11,11),(12,11,12),(13,4,4),(13,4,13),(15,4,4),(15,4,15),(15,12,12),(15,12,15)
				12	(13,12,13)
				11	(4,3,3),(4,3,4),(13,12,12)
			18	13	(6,5,5),(9,6,6),(9,6,9),(10,3,3),(10,3,10),(10,7,7),(10,7,10),(11,2,11),(11,10,10),(11,10,11),(13,6,13),(14,5,14),(14,13,13),(14,13,14),(15,2,2),(15,10,10)
				12	(7,2,7),(11,2,2),(13,6,6),(14,1,14),(14,9,9),(14,9,14),(15,10,15)
				11	(3,2,2),(3,2,3),(6,1,1),(6,1,6),(6,5,6),(7,2,2),(14,1,1),(14,5,5),(15,2,15)

– Focusing on the rotation parameters in the group $\mathcal{A}$, SIMON32 variants with a rotation parameter containing "$a = c$" or "$b = c$" are more vulnerable to NN attacks than differential and linear attacks. For example, three SIMON32 variants with $\{(8, 7, 8), (9, 8, 9), (13, 8, 13)\}$ are the case of "$a = c$", and a SIMON32 variant with $(8, 1, 1)$ is the case of "$b = c$". Similar cases can be seen for SIMON16 variants.
– Focusing on the rotation parameters in the group $\mathcal{C}$, SIMON32 variants with a rotation parameter containing "$n - a = c$" or "$n - b = c$" are more vulnerable to NN attacks than differential and linear attacks. For example, three SIMON32 variants with $\{(12, 5, 4), (4, 1, 12), (12, 1, 4)\}$ are the case of "$n - a = c$", and a SIMON32 variant with $(5, 4, 12)$ is the case of "$n - b = c$". Similar cases can be seen for SIMON16 variants.

It can be seen that SIMON32 variants with such a rotation parameter in the group $\mathcal{A}$ are most likely to cause vulnerabilities against NN attacks; then, we comprehensively analyze these SIMON32 variants and clarify which types of vulnerabilities are detected by NN attacks. Nevertheless, the above rotation parameters regarding the group $\mathcal{C}$ are also interesting; thus, the detailed analysis of these will be our future work.

A Comprehensive Analysis of the Vulnerable Structure We comprehensively analyze SIMON32 variants with the vulnerable structure (i.e., its rotation parameter contains "$a = c$" or "$b = c$") from the following two aspects:

– We have conducted experiments on NN attacks for only 32 SIMON32 variants with the rotation parameter in the group $\mathcal{A}$, as shown in Sect. 3.2. Then, we conduct additional experiments on NN attacks for all

the remaining SIMON32 variants with the rotation parameter in the group $\mathcal{A}$ in the same way as the previous experiments.
– We compare NN attacks with differential and linear attacks as well as impossible differential, zero-correlation linear, and integral attacks in terms of their maximum number of attackable rounds.

Table 5 shows a comparison of the maximum number of rounds that can build differential/linear ('D/L' column), impossible differential/zero-correlation linear ('ID/ZC' column), and integral ('I' column) distinguishers and the maximum number of rounds that can be successful for NN attacks ('NN' column) for each SIMON32 variant, where 'Group' column is as defined in Sect. 3.1. It can be seen from this table that NN attacks on all SIMON16 variants in the group $\mathcal{A}$ have the following features:

– It is difficult to consider that NN attacks capture vulnerabilities caused by impossible differential and zero-correlation linear attacks. This insight is derived by focusing on the case where the maximum number of rounds for building integral distinguishers is 20. While the maximum number of rounds for building impossible differential and zero-correlation linear distinguishers is 13 or 17, the maximum number of attackable rounds for NN attacks is in the range of 11 to 13 in both cases. This implies that NN attacks have not captured the changes in the maximum number of attackable rounds for impossible differential and zero-correlation linear attacks.
– NN attacks have the possibility of capturing vulnerabilities caused by integral attacks. This insight is derived from the fact that the maximum number of attackable rounds for NN attacks varies in accordance with the changes in the maximum number of rounds for building integral distinguishers. More specifically, when the maximum number of rounds for building integral distinguishers is 20 or less, the maximum number of attackable rounds for NN attacks is in the range of 11 to 13. However, when the maximum number of rounds for building integral distinguishers is 32, the maximum number of attackable rounds for NN attacks is in the range of 25 to 29. A notable point is that the rotation parameters in such a case all contain '0' or '8 ($= n/2$)'. Although this insight alone does not conclusively explain how the neural network models have learned the ALL or BALANCE in the integral properties, the experimental results provide a clue to show its explainability. More in-depth analysis will be a future challenge.

It should be noted that the maximum number of rounds for building integral distinguishers has been evaluated with 2^{31} data. The maximum number of rounds for building integral distinguishers with 2^{15} data should be even lower. This suggests that in the cases where the maximum number of attackable rounds for NN attacks is 25 or more, NN attacks may outperform integral attacks in a fair comparison. Therefore, such cases may capture vulnerabilities caused by other than the five generic attacks. Taking this into consideration, we will conduct a more detailed analysis of these SIMON32 variants in the next section.

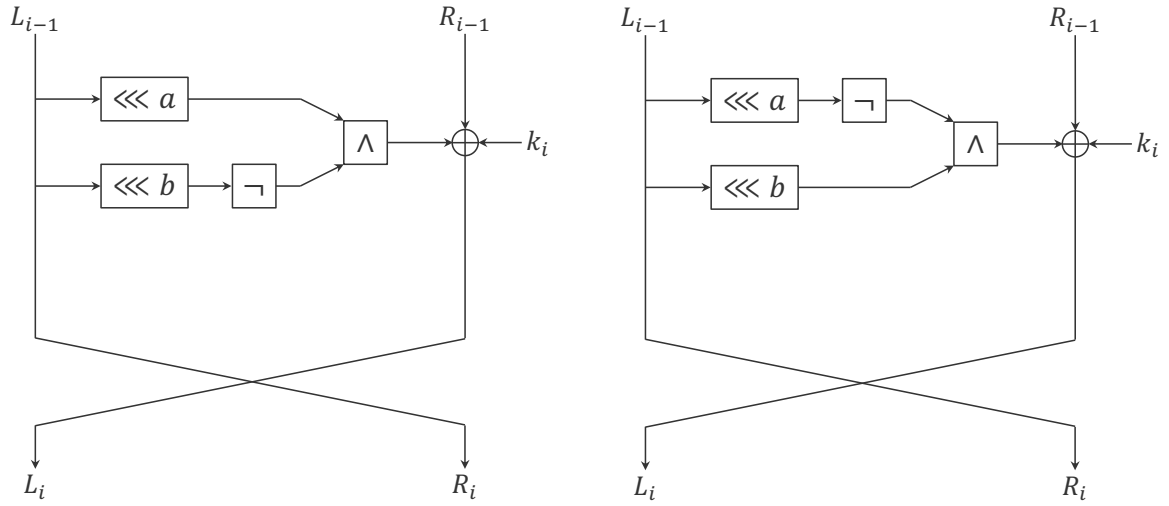
To summarize, we have revealed through NN attacks that SIMON32 variants containing "$a = c$" or "$b = c$" and '0' or '8 ($= n/2$)' become the most vulnerable structure. NN attacks are somewhat simple, but the attacks help identify vulnerable structures from the perspectives of five generic attacks for symmetric-key ciphers (i.e., differential, linear, impossible differential, zero-correlation linear, and integral attacks); thus, this fact should suggest that NN attacks can be a helpful tool for designing secure symmetric-key ciphers, especially SIMON-like ciphers.

4 Closer Look at the Vulnerable Structure

In this section, we take a closer look at SIMON32 variants with the vulnerable structure, containing "$a = c$" or "$b = c$" and '0' or '8 ($= n/2$)'.

4.1 Effect of a Rotation Parameter Containing "$a = c$" or "$b = c$"

We first consider how the structure of the f function changes when the rotation parameter in SIMON32 variants contains "$a = c$" or "$b = c$". For example, when the rotation parameter in SIMON32 variants



(a) Rotation parameter: "$a = c$".

(b) Rotation parameter: "$b = c$".

Fig. 2: Round function when the rotation parameter in SIMON32 variants contains "$a = c$" or "$b = c$".

contains "$a = c$", the f function expressed by Eq. (1) can be converted into

$$\begin{aligned}\hat{f}(x) &= \big((x \lll a) \wedge (x \lll b)\big) \oplus (x \lll a) \\ &= (x \lll a) \wedge \big((x \lll b) \oplus (11\ldots11)\big) \\ &= (x \lll a) \wedge \neg(x \lll b),\end{aligned} \tag{5}$$

where '$\neg$' denotes the logical negation operator. The $\hat{f}$ function of SIMON32 variants containing "$b = c$" can also be converted in the same way as Eq. (5). A notable point here is that when the rotation parameter in SIMON32 variants contains "$a = c$" or "$b = c$", the $\hat{f}$ function can be expressed only by AND operation, as illustrated in Fig. 2. Based on this, we further analyze the effects on rotation parameters containing not only "$a = c$" or "$b = c$" but also '0' or '$n/2$' in Sect. 4.2 and the biased output of the $\hat{f}$ function in the subsequent subsections in Sect. 4.3.

4.2 Effect of a Rotation Parameter Containing '0' or '$n/2$'

We focus here on the diffusion property, which is the ease of spreading the effect of a plaintext bit to the internal state of the intermediate rounds. If the diffusion property is low, the correlation occurs between the plaintext bits and the ciphertext bits in the large rounds. We speculate that NN attacks may recover the plaintext bits from the ciphertext bits with a higher probability than a random search by using the correlation in the large round.

In this section, we verify the ease of spreading the effect of the least significant bit (LSB) of L_0 for several rotation parameters that satisfy "$a = c$" or "$b = c$" and include '0' or '$n/2$'[11].

Local Observation We first focus on the rotation parameter of $(1, 0, 1)$, which contains "$a = c$" and '0', and consider the ease of spreading the effect of the LSB of L_0 to the 2-round internal state. Fig. 3 illustrates how the effect of the LSB of L_0 is spread to (L_2, R_2). In the first round, the effect of the LSB of L_0 propagates

[11] To simplify the explanation, we considered the effect of the LSB of L_0 as an example, but it is obvious that by considering the effect of a certain bit in R_0, the number of rounds that satisfies the full diffusion property can be extended by one round.

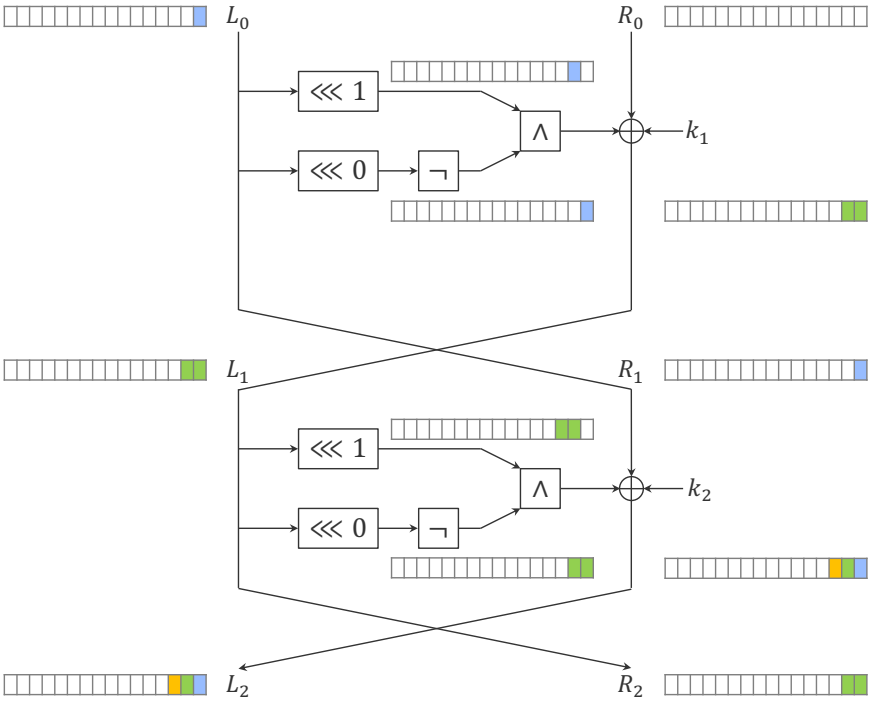


Fig. 3: Diffusion property for the rotation parameter of $(1, 0, 1)$.

into the least significant two bits of L_1 and the LSB of R_1. Then, the effect of the LSB of L_0 included in the LSB of L_1 is merged into the LSB of R_1 from the output of 0-bit left rotation of L_1. This means that the spread of the effect of a plaintext bit decreases by one bit after two rounds. Next, we focus on the rotation parameter of $(8, 1, 1)$, which contains "$b = c$" and '$n/2$'. Similar to the former case, the spread of the effect of a plaintext bit decreases one bit after two rounds. More precisely, the effect of the LSB of L_0 propagates into the lower $(n/2)$-th bit of R_0 from the output of $(n/2)$-bit left rotation of L_0. Then, the effect of the LSB of L_0 included in the lower $(n/2)$-th bit of L_1 is merged into the LSB of R_1 from an output of $n/2$-bit left rotation of L_1.

Long-term Round Observation Finally, we conduct experiments targeting the rotation parameters of $(1, 0, 1)$, $(5, 2, 2)$, and $(1, 8, 2)$ to evaluate the ease of spreading the effect of a plaintext bit to the internal state of the long-term rounds. Note that the rotation parameter of $(5, 2, 2)$ contains "$b = c$" but not '0' or '$n/2$', and the rotation parameter of $(1, 8, 2)$ is for the original Simon32.

Before going into the details of our experimental results, we explain the procedure of our computer simulation. In this simulation, we compute the probability that the effect of a plaintext bit is included for each bit position in the internal state of each round. To this end, we define the following five propagation rules to evaluate the spread of the effect of the certain bit probabilistically:

Rule 1 (Initialize). The target plaintext bits are set to 1.0, and the remaining bits are set to zero.

Rule 2 (Rotation/Swap). The probability, including the effect of the plaintext bits in the rotation and swap operations, is propagated according to the target operation.

Rule 3 (AND). In the AND operation, only when one input bit value is 1, whose probability is assumed to be $\frac{1}{2}$, the effect of the plaintext bits in the corresponding another input bit propagates to the corresponding output bit. Then, the probability, including the effect of the plaintext bits in two inputs of the AND operation, is multiplied by $\frac{1}{2}$ and propagates to the corresponding output bit.

Table 6: Simulation result for the rotation parameter of $(1, 0, 1)$. We set only the LSB of L_0 to 1.0 and check its effect to the r-round internal state.

Bit position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L_0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
R_0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
L_1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.50
R_1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
L_2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.50	1.00
R_2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.50
⋮	⋮															
L_{22}	0.65	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
R_{22}	0.27	0.84	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
L_{23}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
R_{23}	0.65	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
L_{24}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
R_{24}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Rule 4 (XOR). In the XOR operation, there is a possibility that the effect of the plaintext bits is propagated to the output bit without depending on the corresponding two input bits. Then, the probability, including the effect of the plaintext bits in two inputs of the XOR operation, is added and propagates to the corresponding output bit. If the resulting probability exceeds 1.0, it is corrected to 1.0.

Rule 5 (Stop). The simulation stops when all the probabilities, including the effect of the plaintext bits, become 1.0.

This evaluation aims to visualize the spread of the effect of the plaintext bits as much as possible and to focus on the changes in the probabilities affecting the output. Therefore, the cancellation of the effect is not considered; thus, our simulation results are considered to show the minimum number of rounds needed to satisfy the full diffusion property of a single plaintext bit.

Table 6 lists our simulation result for the SIMON32 variant with $(1, 0, 1)$. This table shows that our simulation stopped after 24 rounds; thus, the target variant is considered to satisfy the full diffusion property in 24 rounds. We also evaluate the same experiments for the rotation parameters of $(5, 2, 2)$ and $(1, 8, 2)$, and their results are listed in Tables 7 and 8. These tables show that the number of rounds that satisfy the full diffusion property is 13 and 7 for the rotation parameters of $(5, 2, 2)$ and $(1, 8, 2)$, respectively. To summarize, our experimental results show the full diffusion property of SIMON32 variants is reduced when a rotation parameter containing '0' or '$n/2$' and "$a = c$" or "$b = c$" holds.

We do not aim to directly compare the maximum number of attackable rounds for NN attacks with the number of rounds to satisfy the full diffusion property based on the simulation. Our goal here is to capture the relationship between NN attacks and diffusion properties. We believe that our simulation has successfully captured such a relationship. Further improving this simulation will lead to the possibility of explaining in more details the deep relationship between NN attacks and diffusion properties, which remains our future challenge.

4.3 Effect of a Biased Output Value of the $\hat{f}$ Function

This subsection clarifies the effect of a biased output value of the $\hat{f}$ function. To this end, we mainly focus on the case of SIMON32 variants with "$a = c$" and first consider the following bit-wise representation of Eq. (5):

$$\hat{f}_j(x) = (x \lll a)_j \wedge \neg(x \lll b)_j, \tag{6}$$

Table 7: Simulation result for the rotation parameter of $(5, 2, 2)$. We set only the LSB of L_0 to 1.0 and check its effect to the r-round internal state.

Bit position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L_0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
R_0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
L_1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.50	0.00	0.00
R_1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
L_2	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.50	0.00	0.00	0.25	0.00	0.00	0.00	1.00
R_2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.50	0.00	0.00
$\vdots$								$\vdots$								
L_{11}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.79	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
R_{11}	1.00	1.00	0.12	1.00	1.00	1.00	1.00	1.00	1.00	0.93	1.00	1.00	0.51	1.00	1.00	1.00
L_{12}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
R_{12}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.79	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
L_{13}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
R_{13}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 8: Simulation result for the rotation parameter of $(1, 8, 2)$. We set only the LSB of L_0 to 1.0 and check its effect to the r-round internal state.

Bit position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L_0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
R_0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
L_1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	1.00	0.50	0.00
R_1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
L_2	0.00	0.00	0.00	0.00	0.00	1.00	0.50	0.00	0.00	0.00	0.00	1.00	1.00	0.25	0.00	1.00
R_2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	1.00	0.50	0.00
L_3	0.00	0.00	0.00	1.00	1.00	0.38	0.00	1.00	0.00	1.00	1.00	0.75	0.13	1.00	1.00	0.00
R_3	0.00	0.00	0.00	0.00	0.00	1.00	0.50	0.00	0.00	0.00	0.00	1.00	1.00	0.25	0.00	1.00
L_4	0.00	1.00	1.00	1.00	0.25	1.00	1.00	1.00	1.00	1.00	0.50	1.00	1.00	0.94	0.00	1.00
R_4	0.00	0.00	0.00	1.00	1.00	0.38	0.00	1.00	0.00	1.00	1.00	0.75	0.13	1.00	1.00	0.00
L_5	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.72	1.00	1.00	1.00
R_5	0.00	1.00	1.00	1.00	0.25	1.00	1.00	1.00	1.00	1.00	0.50	1.00	1.00	0.94	0.00	1.00
L_6	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
R_6	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.72	1.00	1.00	1.00
L_7	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
R_7	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

where x_j denotes the j-th bit of the value of x for $j \in \{0, 1, \ldots, n-1\}$[12]. From Eq. (6), the necessary and sufficient condition under which $\hat{f}_j(x) = 1$ holds is

$$\big((x \lll a)_j = 1\big) \wedge \big((x \lll b)_j = 0\big).$$

[12] It is important to note that the indices of L and R represent the number of rounds.

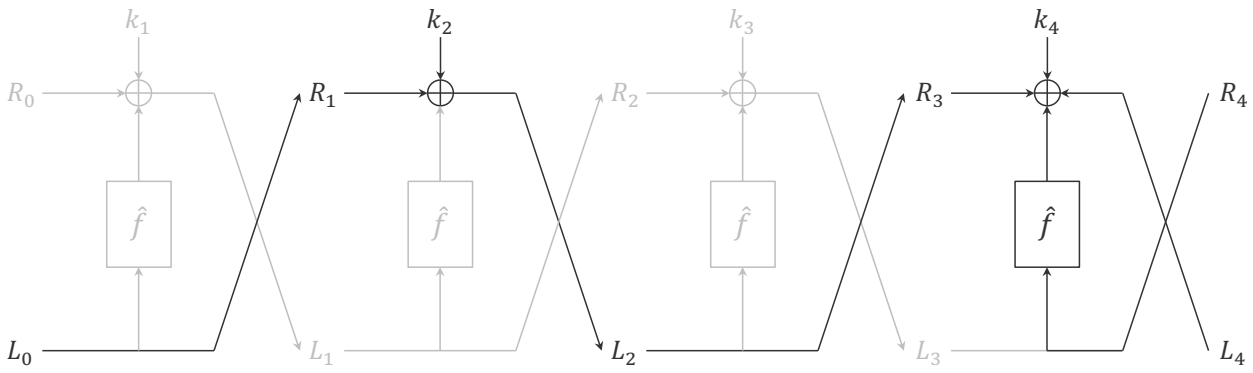


Fig. 4: Schematic diagram for visualizing the 4-round linear approximation expressed by Eq. (11) as an example. We assume that $\hat{f}_j(L_{i-1}) = 0$ holds in all the even-numbered rounds excluding the last round. For this, our linear approximation does not consider the gray-colored lines.

Let p be the ratio of $x_j = 1$ for the input x to the $\hat{f}$ function. Then, assuming that the probability of $x_j = 1$ at all bit positions for $j \in \{0, 1, \ldots, n-1\}$ is unbiased, we can derive the following equation:

$$\Pr\big((x \lll a)_j = 1\big) = \Pr\big((x \lll b)_j = 1\big) = p. \tag{7}$$

According to Eq. (7), the probability that $\hat{f}_j(x) = 1$ holds is as follows:

$$\begin{aligned}\Pr\big((x \lll a)_j \wedge \neg(x \lll b)_j = 1\big) &= \Pr\big(\big((x \lll a)_j = 1\big) \wedge \big((x \lll b)_j = 0\big)\big) \\ &= \Pr\big((x \lll a)_j = 1\big) \cdot \Pr\big((x \lll b)_j = 0\big) \\ &= p \cdot (1-p). \end{aligned} \tag{8}$$

Since $0 \leq p \leq 1$, it is obvious that the probability expressed by Eq. (8) always remains below $\frac{1}{4}$.

In other words, by exploiting this biased event in the $\hat{f}$ function, the probability that $\hat{f}_j(x) = 0$ holds for $j \in \{0, 1, \ldots, n-1\}$ is at least $\frac{3}{4}$. Based on these considerations, we can assume that $\hat{f}_j(L_{i-1}) = 0$ holds with a relatively high probability, and then the i-th round function is expressed as

$$L_i = \hat{f}(L_{i-1}) \oplus R_{i-1} \oplus k_i = R_{i-1} \oplus k_i, \tag{9}$$

$$R_i = L_{i-1}. \tag{10}$$

Finding an Effective Linear Approximation We assume that $\hat{f}_j(L_{i-1}) = 0$ holds for $i \in \{2, 4, \ldots, 2(r-1)\}$, i.e., all the even-numbered rounds excluding the last round of the $2r$-round SIMON32 variant with the vulnerable structure. Under the known plaintext attack scenario, an attacker can get multiple known plaintext/ciphertext pairs. Then, if the above assumption holds, the attacker can use the following equation for the j-th bit position:

$$(L_0 \oplus L_{2r} \oplus \hat{f}(R_{2r}))_j = \bigoplus_{x \in \{1,2,\ldots,r-1\}} (k_{2x})_j, \tag{11}$$

where L_0 denotes a left side of the plaintext, and (L_{2r}, R_{2r}) denotes the ciphertext. Therefore, Eq. (11) indicates a linear approximation expressed by a plaintext bit, a ciphertext bit, and a linear sum of the subkey bits. For a better understanding, we provide a schematic diagram for visualizing the 4-round linear approximation expressed by Eq. (11), as illustrated in Fig. 4. Here, we provide the following theorem.

Theorem 2. *For the $2r$-round* SIMON32 *variants with the vulnerable structure, the probability that the linear approximation expressed by Eq. (11) holds is given by*

$$\Pr(X = Y) = \begin{cases} 1 & \text{if } r = 1; \\ \sum_{i=0}^{\lfloor \frac{r-1}{2} \rfloor} \binom{r-1}{2i} \left(\frac{1}{4}\right)^{2i} \left(\frac{3}{4}\right)^{r-2i-1} & \text{otherwise,} \end{cases} \tag{12}$$

where $X = (L_0 \oplus L_{2r} \oplus \hat{f}(R_{2r}))_j$ *and* $Y = \bigoplus_{x \in \{1,2,\ldots,r-1\}} (k_{2x})_j$.

Proof. We first target the case where $r = 1$. In this case, we can derive

$$\left(L_0 \oplus L_2 \oplus \hat{f}(R_2)\right)_j = (k_2)_j \tag{13}$$

from Eq. (11), which obviously holds with a probability of 1.

Next, we target the case where $r = 2$. To compute the probability of the target event, we must consider the unknown output value of the $\hat{f}$ function in the second round, i.e., $\hat{f}_j(L_1)$. If $\hat{f}_j(L_1) = 0$, which holds with a probability of approximately $\frac{3}{4}$, the target event occurs with a probability of 1; otherwise, it never occurs. Then, we can get the probability of approximately $\frac{3}{4}$ when $r = 2$.

Finally, we target the case where $r > 2$, but it can be generalized to include the case where $r = 2$. Similar to the case where $r = 2$, we must consider the unknown output values of the $\hat{f}$ function in the even-numbered rounds excluding the last round, i.e., $\hat{f}_j(L_1), \hat{f}_j(L_3), \ldots, \hat{f}_j(L_{2r-3})$. In other words, the number of $\hat{f}$ functions that must be considered is $r - 1$. Then, the condition that the target event occurs with a probability of 1 is given by

$$\hat{f}_j(L_1) \oplus \hat{f}_j(L_3) \oplus \cdots \oplus \hat{f}_j(L_{2r-3}) = 0, \tag{14}$$

and this condition holds in the following two cases:

Case 1. The output value of the target $\hat{f}$ functions are all zero.
Case 2. The number for which the output value of the target $\hat{f}$ function is 1 is an even number.

Considering these two cases, we can get

$$\begin{aligned} &\binom{r-1}{0}\left(\frac{1}{4}\right)^{0}\left(\frac{3}{4}\right)^{r-1} + \binom{r-1}{2}\left(\frac{1}{4}\right)^{2}\left(\frac{3}{4}\right)^{r-2-1} + \cdots \\ &= \sum_{i=0}^{\lfloor \frac{r-1}{2} \rfloor} \binom{r-1}{2i}\left(\frac{1}{4}\right)^{2i}\left(\frac{3}{4}\right)^{r-2i-1} \end{aligned} \tag{15}$$

when $r \geq 2$. This concludes the proof.

Experimental Verification for Theorem 2 We verify the accuracy of the theoretical values in Theorem 2. To this end, we conducted experiments with 2^8 trials (i.e., randomly generated 2^8 secret keys) using 2^{28} samples for each trial.

Fig. 5 shows a comparison between the theoretical and experimental probabilities. The vertical axis represents the probability, and the horizontal axis represents the number of rounds for the target cipher. The blue line represents the theoretical probabilities provided in Theorem 2. The purple, green, and orange dot lines represent the experimental probabilities for SIMON32 variants with a rotation parameter of $(1, 8, 2)$, $(1, 8, 1)$[13], and $(5, 2, 2)$, respectively.

It can be seen from this graph that the theoretical values almost match the experimental values for SIMON32 variants with the vulnerable structure (i.e., a rotation parameter of $(1, 8, 1)$ and $(5, 2, 2)$). For this reason, it can be considered that the theoretical values provided in Theorem 2 are correct. In addition, it can also be seen from the graph that the theoretical values for the original SIMON32 (i.e., the rotation of $(1, 8, 2)$) are converged to the probability of random. This fact demonstrates that there is a big difference in randomness between the original SIMON32 and SIMON32 variants with the vulnerable structure.

[13] A rotation parameter of $(1, 8, 1)$ is equal to that of $(8, 1, 1)$. This is very similar to the original parameter of $(1, 8, 2)$.

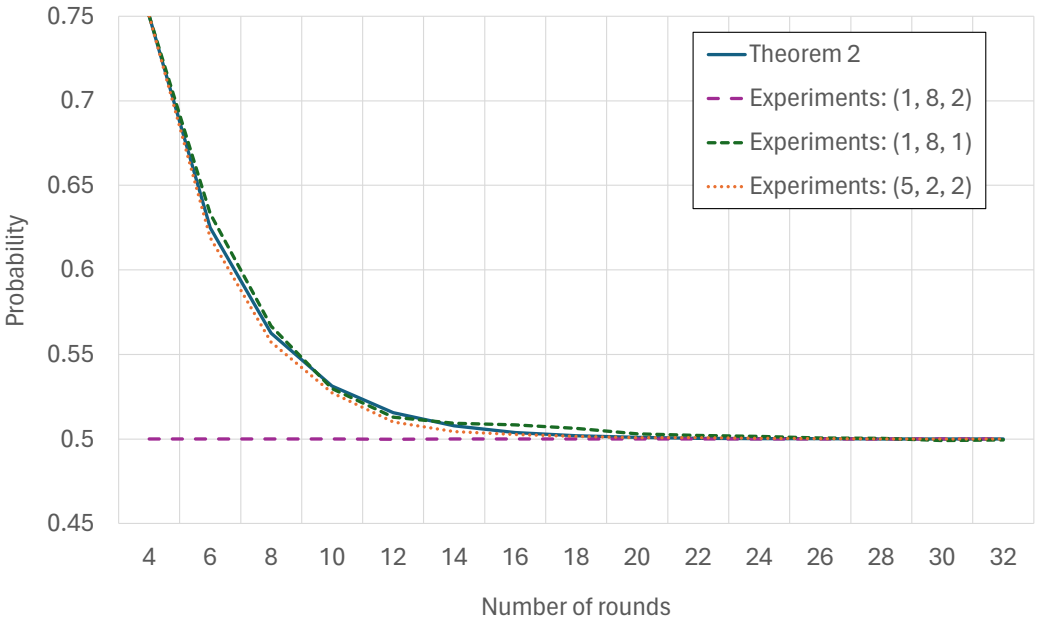


Fig. 5: Comparison between the theoretical and experimental probabilities.

Building a Linear Distinguisher The linear approximation expressed by Eq. (11) enables the attacker to build an effective linear distinguisher. Table 9 lists the theoretical probabilities of the linear approximation and the data complexity for building linear distinguishers with a success probability of 0.7 for each even-numbered round, which are derived from based on Theoerms 1 and 2. It can be seen from this table that the linear distinguisher with up to 30 rounds can be built for SIMON32 variants with the vulnerable structure. Given that NN attacks have been successful up to 29 rounds, as listed in Table 5, this suggests that NN attacks may detect these types of linear distinguishers. The detailed analysis will be our future challenge.

Other Linear Approximations We briefly introduce three linear approximations that are very similar to the linear approximation expressed by Eq. (11).

The first linear approximation is given by

$$(R_0 \oplus \hat{f}(L_0) \oplus R_{2r})_j = \bigoplus_{x \in \{1,2,\ldots,r\}} (k_{2x-1})_j. \tag{16}$$

This can be useful for computing a linear sum of all the *odd*-numbered rounds of subkey bits (i.e., $k_1 \oplus k_3 \oplus \cdots \oplus k_{2r-1}$) for the *even*-numbered rounds (i.e., $2r$ rounds) of the target SIMON32 variant.

The next linear approximation is given by

$$(L_0 \oplus R_{2r+1})_j = \bigoplus_{x \in \{1,2,\ldots,r\}} (k_{2x})_j. \tag{17}$$

This can be useful for computing a linear sum of all the *even*-numbered rounds of subkey bits (i.e., $k_2 \oplus k_4 \oplus \cdots \oplus k_{2r}$) for the *odd*-numbered (i.e., $2r+1$ rounds) rounds of the target SIMON32 variant.

The last linear approximation is given by

$$(R_0 \oplus \hat{f}(L_0) \oplus L_{2r+1})_j = \bigoplus_{x \in \{1,2,\ldots,r,r+1\}} (k_{2x-1})_j. \tag{18}$$

Table 9: Theoretical probabilities of the linear approximation expressed by Eq. (11) and the data complexity for building linear distinguishers with a success probability of 0.7 for each even-numbered round.

Round	Probability	Data ($\log_2$)	Success rate
4	0.750000	5.164	0.7
6	0.625000	7.164	0.7
8	0.562500	9.164	0.7
10	0.531250	11.164	0.7
12	0.515625	13.164	0.7
14	0.507812	15.164	0.7
16	0.503906	17.164	0.7
18	0.501953	19.164	0.7
20	0.500976	21.164	0.7
22	0.500488	23.164	0.7
24	0.500244	25.164	0.7
26	0.500122	27.164	0.7
28	0.500061	29.164	0.7
30	0.500030	31.164	0.7
32	0.500015	33.164	0.7

This can be useful for computing a linear sum of all the *odd*-numbered rounds of subkey bits (i.e., $k_1 \oplus k_3 \oplus \cdots \oplus k_{2r+1}$) for the *odd*-numbered rounds (i.e., $2r+1$ rounds) of the target SIMON32 variant.

The probability that each linear approximation holds can also be derived in a similar way as Theorem 2, but this is beyond the scope of the paper.

5 Towards Key Recovery on SIMON32 Variants with the Vulnerable Structure

We mainly use the linear approximation expressed by Eq. (11) in this section, but it is expected that the other linear approximations expressed by Eqs. (16)–(18) can also be used for our analysis.

Focusing on Eq. (11), this is expressed by a plaintext bit, a ciphertext bit, and a linear sum of subkey bits. This means that we have found an effective linear approximation, which is the primary purpose of linear cryptanalysis [24]. In other words, this linear approximation can be used to develop a key recovery attack based on linear cryptanalysis.

Linear cryptanalysis was first reported by Matsui at EUROCRYPT 1993 [24]. He proposed two simple but powerful algorithms, famously known as Matsui's algorithm 1 and algorithm 2. His subsequent work [25] provided an improved version of linear cryptanalysis and applied it to the first successful computer experiment in breaking the full 16-round DES. After that, key recovery attacks based on linear cryptanalysis have been improved by using several techniques, such as Walsh transformation (or fast Fourier transformation) [11], affine pruned Walsh transformation [12], Walsh spectrum puncturing [13], and more.

Here, we use simple but powerful Matsui's algorithm 1 to show that a linear sum of subkey bits for SIMON32 variants with the vulnerable structure can be recovered with a relatively high probability. Using other techniques [11–13, 25] may lead to further improvements, but this is beyond the scope of this paper.

5.1 Revisiting Matsui's Algorithm 1

This subsection briefly reviews Matsui's algorithm 1 [24]. The fundamental idea of linear cryptanalysis is to find the following effective linear approximation for a given cipher:

$$P_{i_1} \oplus \cdots \oplus P_{i_a} \oplus C_{j_1} \oplus \cdots \oplus C_{j_b} = K_{k_1} \oplus \cdots \oplus K_{k_c}, \tag{19}$$

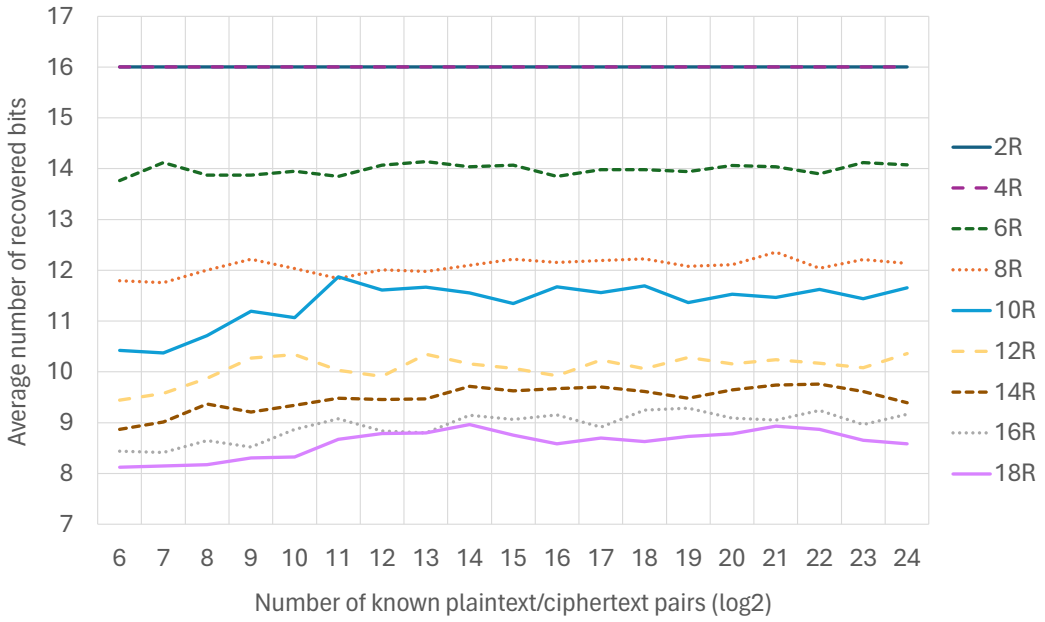


Fig. 6: Experimental results of the key recovery attack on SIMON32 variants with a rotation parameter of $(5, 2, 2)$.

where P_i, C_j, and K_k denote the i-th plaintext bit, the j-th ciphertext bit, and the k-th secret key (or subkey) bits, respectively. Also, $i_1, \ldots, i_a, j_1, \ldots, j_b, k_1, \ldots, k_c$ denote the fixed bit position, and Eq. (19) holds with a probability of p ($\neq \frac{1}{2}$) for randomly generated plaintext and the corresponding ciphertext.

We can recover a linear sum of subkey bits (i.e., $\hat{K} = K_{k_1} \oplus \cdots \oplus K_{k_c}$) using the following algorithm:

Step 1. Let T be the number of plaintexts such that the left side of Eq. (19) is equal to zero, and let N be the number of plaintext/ciphertext pairs obtained under the known plaintext attack scenario.

Step 2. If $T > \frac{N}{2}$, then we guess $\hat{K} = 0$ when $p > \frac{1}{2}$ or $\hat{K} = 1$ when $p < \frac{1}{2}$. Otherwise, we guess $\hat{K} = 1$ when $p > \frac{1}{2}$ or $\hat{K} = 0$ when $p < \frac{1}{2}$.

We follow Theorem 1 to estimate the number of plaintext/ciphertext pairs and the success probability for the attack based on Matsui's algorithm 1.

5.2 Experimental Verifications

We verify the validity of the key recovery attack based on Matsui's algorithm 1 against SIMON32 variants with the vulnerable structure. To this end, we conduct experiments with 2^8 trials (i.e., 2^8 secret keys) using 2^d samples for each trial, where $d \in \{6, 7, \ldots, 24\}$. The following two $2r$-round variants for $r \in \{1, 2, \ldots, 9\}$ are targeted for our analysis: the first one has a rotation parameter of $(5, 2, 2)$, which contains "$b = c$" but not '0' or '$n/2$'; and the second one has a rotation parameter of $(8, 1, 1)$, which contains both "$b = c$" and '$n/2$'.

We provide the details of our experimental results on Figs. 6 and 7. In their figures show the experimental results of the key recovery attack on Simon32 variants with a rotation parameter of (5, 2, 2) and (8, 1, 1), respectively. More precisely, these figures show the average number of recovered bits of a linear sum of 16-bit subkeys for each number of known plaintext/ciphertext pairs. The following can be seen from these figures:

– We can recover the full 16 bits of the linear sum of subkeys with a probability of 1 for the target variants with up to four rounds.

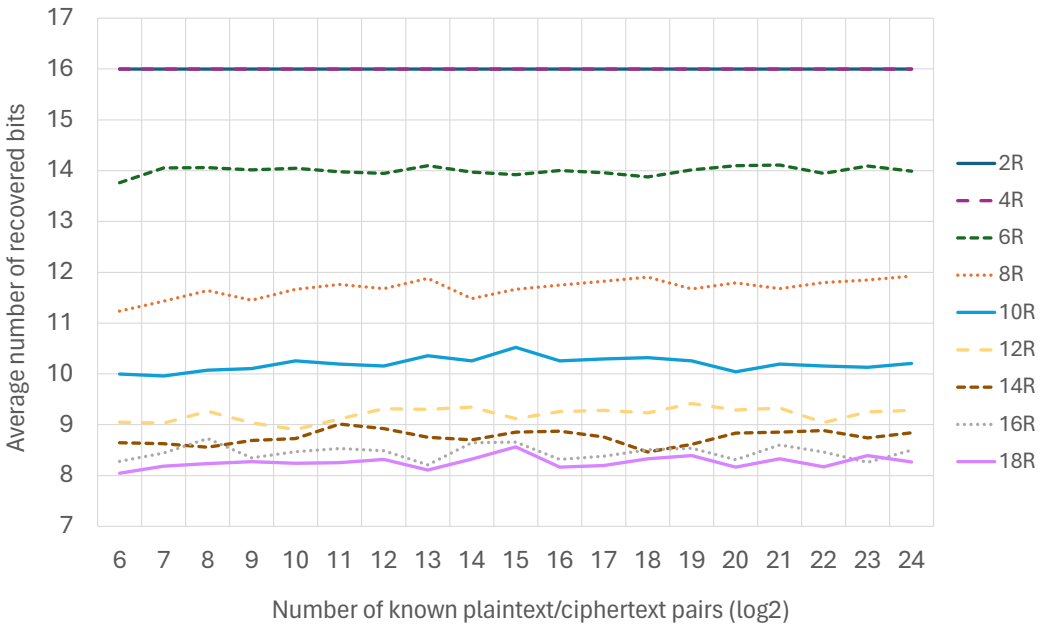


Fig. 7: Experimental results of the key recovery attack on SIMON32 variants with a rotation parameter of $(8, 1, 1)$.

- For the target variants with more than six rounds, the average number of recovered bits gradually decreases, and the number of rounds that reaches the same level of attack ability with random guessing (i.e., 8 out of 16 bits can be recovered) is almost 18.
- Surprisingly, increasing the number of known plaintext/ciphertext pairs only slightly improves attack ability. We speculate that this factor is due to the diffusion property of SIMON32 variants with the vulnerable structure discussed in Sect. 4.2, but the detailed analysis is our future challenge.

Additionally, we conduct the same experiments against the original SIMON32 and have verified that the key recovery attack is valid for up to two rounds, which is a trivial attack. To summarize, the above type of attack on SIMON32 variants with the vulnerable structure compared to the original SIMON32 is relatively valid, and we gain new insight into the fact that the attack ability does not improve as the number of known plaintext/ciphertext pairs increases.

6 Conclusion and Future Challenges

We have proposed a step forward toward making neural network-based output prediction (NN) attacks a helpful tool for designing a secure symmetric-key cipher, especially a SIMON-like cipher. Our analysis has provided new insight into evaluating the security of the target cipher using NN attacks. A notable point is that NN attacks can identify a vulnerable structure of the target cipher. This has the advantage that similar ciphers with a vulnerable structure can be excluded from all design candidates.

NN attacks can be a helpful tool for detecting vulnerabilities, but unfortunately, the use of this tool alone cannot determine the attack vectors that cause such vulnerabilities. For this reason, we recommend effectively combining NN attacks and automatic search methods to maximize the advantages of both approaches. This would enable more reliable and faster security analysis when designing a secure symmetric-key cipher.

We hope that addressing the following five challenges will be a further step toward making NN attacks a helpful tool for designing a secure symmetric-key cipher: 1) We mainly focused on the case of "$a = c$" or "$b = c$" to take a closer look at vulnerable structures. Similarly, analyzing the case of "$n - a = c$" or

"$n - b = c$" may provide new insight into the applications of NN attacks. 2) Interestingly, NN attacks may detect a vulnerability due to integral attacks. In our experiments, we limited the number of training data to 2^{15}, and our observation may seem intuitively suspicious from the perspective of the integral attack methodology. However, if we can prove that our observation is correct, the applicability of NN attacks will further expand. 3) The maximum number of rounds that NN attacks can be successful is very similar to that for building a linear distinguisher based on the linear approximation expressed by Eq. (11). This may suggest that NN attacks can detect a vulnerability due to linear attacks. 4) Surprisingly, increasing the number of known plaintext/ciphertext pairs only slightly improves the ability of key recovery attacks. Identifying its root cause may be able to contribute to developing linear key recovery attacks. 5) This study targeted only the AND-Rotation-XOR-based cipher. To achieve our primary goal, we must consider the effectiveness of NN attacks against more types of symmetric-key ciphers, such as the Addition-Rotation-XOR-based and arithmetization-oriented ciphers.

References

1. Baignères, T., Junod, P., Vaudenay, S.: How far can we go beyond linear cryptanalysis? In: ASIACRYPT. Lecture Notes in Computer Science, vol. 3329, pp. 432–450. Springer (2004)
2. Bao, Z., Guo, J., Liu, M., Ma, L., Tu, Y.: Enhancing differential-neural cryptanalysis. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 13791, pp. 318–347. Springer (2022)
3. Bao, Z., Lu, J., Yao, Y., Zhang, L.: More insight on deep learning-aided cryptanalysis. In: ASIACRYPT (3). Lecture Notes in Computer Science, vol. 14440, pp. 436–467. Springer (2023)
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. p. 404 (2013)
5. Bellini, E., Gérault, D., Hambitzer, A., Rossi, M.: A cipher-agnostic neural training pipeline with automated finding of good input differences. IACR Trans. Symmetric Cryptol. **2023**(3), 184–212 (2023)
6. Benamira, A., Gérault, D., Peyrin, T., Tan, Q.Q.: A deeper look at machine learning-based cryptanalysis. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12696, pp. 805–835. Springer (2021)
7. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 1592, pp. 12–23. Springer (1999)
8. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: CRYPTO. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1990)
9. Bogdanov, A., Rijmen, V.: Linear hulls with correlation zero and linear cryptanalysis of block ciphers. Des. Codes Cryptogr. **70**(3), 369–383 (2014)
10. Boura, C., David, N., Boissier, R.H., Naya-Plasencia, M.: Better steady than speedy: Full break of SPEEDY-7-192. In: EUROCRYPT (4). Lecture Notes in Computer Science, vol. 14007, pp. 36–66. Springer (2023)
11. Collard, B., Standaert, F., Quisquater, J.: Improving the time complexity of matsui's linear cryptanalysis. In: ICISC. Lecture Notes in Computer Science, vol. 4817, pp. 77–88. Springer (2007)
12. Flórez-Gutiérrez, A.: Optimising linear key recovery attacks with affine walsh transform pruning. In: ASIACRYPT (4). Lecture Notes in Computer Science, vol. 13794, pp. 447–476. Springer (2022)
13. Flórez-Gutiérrez, A., Todo, Y.: Improving linear key recovery attacks using walsh spectrum puncturing. IACR Cryptol. ePrint Arch. p. 151 (2024)
14. Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 11693, pp. 150–179. Springer (2019)
15. Hadipour, H., Eichlseder, M.: Integral cryptanalysis of WARP based on monomial prediction. IACR Trans. Symmetric Cryptol. **2022**(2), 92–112 (2022)
16. Hadipour, H., Gerhalter, S., Sadeghi, S., Eichlseder, M.: Improved search for integral, impossible differential and zero-correlation attacks application to ascon, forkskinny, skinny, mantis, PRESENT and qarmav2. IACR Trans. Symmetric Cryptol. **2024**(1), 234–325 (2024)
17. Kimura, H., Emura, K., Isobe, T., Ito, R., Ogawa, K., Ohigashi, T.: Output prediction attacks on block ciphers using deep learning. In: ACNS Workshops. Lecture Notes in Computer Science, vol. 13285, pp. 248–276. Springer (2022)
18. Kimura, H., Emura, K., Isobe, T., Ito, R., Ogawa, K., Ohigashi, T.: A deeper look into deep learning-based output prediction attacks using weak SPN block ciphers. J. Inf. Process. **31**, 550–561 (2023)
19. Knudsen, L.: Deal-a 128-bit block cipher. complexity **258**(2), 216 (1998)

20. Knudsen, L.R., Wagner, D.A.: Integral cryptanalysis. In: FSE. Lecture Notes in Computer Science, vol. 2365, pp. 112–127. Springer (2002)
21. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 9215, pp. 161–185. Springer (2015)
22. Kondo, K., Sasaki, Y., Todo, Y., Iwata, T.: On the design rationale of SIMON block cipher: Integral attacks and impossible differential attacks against SIMON variants. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **101-A**(1), 88–98 (2018)
23. Liu, F., Anand, R., Wang, L., Meier, W., Isobe, T.: Coefficient grouping: Breaking chaghri and more. In: EUROCRYPT (4). Lecture Notes in Computer Science, vol. 14007, pp. 287–317. Springer (2023)
24. Matsui, M.: Linear cryptanalysis method for DES cipher. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer (1993)
25. Matsui, M.: The first experimental cryptanalysis of the data encryption standard. In: CRYPTO. Lecture Notes in Computer Science, vol. 839, pp. 1–11. Springer (1994)
26. Mouha, N., Preneel, B.: Towards finding optimal differential characteristics for arx: Application to salsa20. Cryptology ePrint Archive, Paper 2013/328 (2013), https://eprint.iacr.org/2013/328, https://eprint.iacr.org/2013/328
27. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Inscrypt. Lecture Notes in Computer Science, vol. 7537, pp. 57–76. Springer (2011)
28. Sakamoto, K., Ito, R., Isobe, T.: Parallel SAT framework to find clustering of differential characteristics and its applications. In: SAC. Lecture Notes in Computer Science, vol. 14201, pp. 409–428. Springer (2023)
29. Sun, L., Wang, W., Wang, M.: Accelerating the search of differential and linear characteristics with the SAT method. IACR Trans. Symmetric Cryptol. **2021**(1), 269–315 (2021)
30. Wang, S., Feng, D., Hu, B., Guan, J., Shi, T.: Practical attacks on full-round FRIET. IACR Trans. Symmetric Cryptol. **2022**(4), 105–119 (2022)

A Preliminary Experiments

A.1 Impact of Reducing the Number of Keys

We have conducted preliminary experiments to streamline our analysis. Specifically, we randomly pick up one SIMON32 variant for each group (i.e., $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$), follow the experimental procedure described in Sect. 3.2 to conduct the experiments, compare experimental results using 10 randomly chosen keys with those using 100 randomly chosen keys, and clarify whether using 10 randomly chosen keys is sufficient to obtain reliable results.

Tables 10–13 show experimental results of NN attacks for SIMON32 variants with $\{(4,1,4), (2,1,8), (5,4,12), (7,2,6)\}$, respectively. It can be seen from these tables that comparing the experimental results using 10 randomly chosen keys with those using 100 keys shows that the number of attackable rounds is the same in all cases. Considering that using 100 randomly chosen keys is sufficient to obtain reliable results, which was presented by Kimura et al. [17], our preliminary experiments demonstrate that using 10 randomly chosen keys is sufficient to obtain reliable results.

Table 10: Experimental results of NN attacks for a SIMON32 variant with $(4,1,4)$, which is in group $\mathcal{A}$. ''DCP' and LCP' stand for differential and linear characteristic probabilities, respectively.

Round	DCP	LCP	Average match rate (10 keys)	Average match rate (100 keys)	success (✓) or failure (–)
1	1	1	0.97993	0.98466	✓
2	2^{-2}	2^{-2}	0.88790	0.96798	✓
3	2^{-2}	2^{-2}	0.85479	0.91612	✓
4	2^{-4}	2^{-4}	0.94194	0.84523	✓
5	2^{-4}	2^{-4}	0.66882	0.67818	✓
6	2^{-6}	2^{-6}	0.61709	0.60212	✓
7	2^{-6}	2^{-6}	0.57227	0.56613	✓
8	2^{-8}	2^{-8}	0.54541	0.54117	✓
9	2^{-8}	2^{-8}	0.52793	0.52784	✓
10	2^{-10}	2^{-10}	0.51958	0.51941	✓
11	2^{-10}	2^{-10}	0.51603	0.51390	✓
12	2^{-12}	2^{-12}	0.50720	0.50891	✓
13	2^{-12}	2^{-12}	0.50566	0.50739	✓
14	2^{-14}	2^{-14}	0.50266	0.50370	–

Table 11: Experimental results of NN attacks for a SIMON32 variant with $(2, 1, 8)$, which is in group $\mathcal{B}$. ''DCP‘ and LCP' stand for differential and linear characteristic probabilities, respectively.

Round	DCP	LCP	Average match rate (10 keys)	Average match rate (100 keys)	success (✓) or failure (–)
1	1	1	0.96954	0.98885	✓
2	2^{-2}	2^{-2}	0.77169	0.73522	✓
3	2^{-4}	2^{-4}	0.80161	0.79571	✓
4	2^{-4}	2^{-4}	0.57116	0.57451	✓
5	2^{-6}	2^{-6}	0.52340	0.50639	✓
6	2^{-8}	2^{-8}	0.51333	0.51542	✓
7	2^{-8}	2^{-8}	0.50714	0.50861	✓
8	2^{-10}	2^{-10}	0.50012	0.50016	–

Table 12: Experimental results of NN attacks for a SIMON32 variant with $(5, 4, 12)$, which is in group $\mathcal{C}$. ''DCP‘ and LCP' stand for differential and linear characteristic probabilities, respectively.

Round	DCP	LCP	Average match rate (10 keys)	Average match rate (100 keys)	success (✓) or failure (–)
1	1	1	0.95938	0.97338	✓
2	2^{-2}	2^{-2}	0.55664	0.58083	✓
3	2^{-4}	2^{-4}	0.51445	0.57755	✓
4	2^{-6}	2^{-6}	0.57596	0.58728	✓
5	2^{-8}	2^{-8}	0.53381	0.54645	✓
6	2^{-10}	2^{-10}	0.52744	0.52847	✓
7	2^{-12}	2^{-12}	0.51530	0.51611	✓
8	2^{-14}	2^{-14}	0.50878	0.50954	✓
9	2^{-16}	2^{-16}	0.50440	0.50487	–

Table 13: Experimental results of NN attacks for a SIMON32 variant with $(7, 2, 6)$, which is in group $\mathcal{D}$. ''DCP‘ and LCP' stand for differential and linear characteristic probabilities, respectively.

Round	DCP	LCP	Average match rate (10 keys)	Average match rate (100 keys)	success (✓) or failure (–)
1	1	1	0.98752	0.97929	✓
2	2^{-2}	2^{-2}	0.51807	0.57053	✓
3	2^{-4}	2^{-4}	0.49999	0.49993	–

A.2 Impact of the Number of Keys for Generating Datasets

We have conducted preliminary experiments to validate the effectiveness of the single-key setting. Specifically, we generate the training and test datasets using 1, 2^3, 2^6, or 2^9 secret keys, follow the experimental procedure described in Sect. 3.2 to conduct experiments for all these combinations (i.e., 16 combinations), compare these results with those under the single-key setting, and clarify that NN attacks work effectively under the single-key setting.

Table 14 shows experimental results of NN attacks for SIMON32 variants with $\{(13, 8, 13), (9, 8, 9), (8, 7, 8), (8, 1, 1)\}$, respectively. These variants have a large number of attackable rounds for NN attacks under the single-key setting, making them good targets for comparing differences in attack capabilities. As can be seen from the table, comparing the experimental results using multiple keys with those using a single key clearly shows that NN attacks work effectively under the single-key setting.

Table 14: Experimental results of NN attacks for SIMON32 variants with $\{(13, 8, 13), (9, 8, 9), (8, 7, 8), (8, 1, 1)\}$, respectively. †: single-key setting.

# secret keys		# attackable rounds			
training dataset	test dataset	(13,8,13)	(9,8,9)	(8,7,8)	(8,1,1)
$1^\dagger$		**27**	**26**	**26**	**25**
1	1	4	2	4	2
	2^3	1	1	1	1
	2^6	1	1	1	1
	2^9	1	1	1	1
2^3	1	1	1	1	1
	2^3	1	1	1	2
	2^6	1	1	1	1
	2^9	1	1	1	1
2^6	1	2	1	1	1
	2^3	1	1	1	1
	2^6	1	1	1	1
	2^9	1	1	1	1
2^9	1	1	1	1	1
	2^3	1	1	1	1
	2^6	1	1	1	1
	2^9	1	1	1	1

B Part of the Source Code for Our Experiments

We provide here part of the source code for our experiments: `gen_dataset`, `get_data`, and `create_model` functions.

Listing 1.1: `gen_dataset` function

```
import numpy as np
import secrets
from simon import Simon as CipherClass # self-made function

## k_list_: a list of keys, n_round_: number of rounds
def gen_dataset(k_list_, n_round_):
    # generate arrays to split training and test data
    ## n_train: number of training data, n_test: number of test data
    ## b_size: block size, n_train_keys: number of keys for training data
    ## n_test_keys: number of keys for test data
    trainX_ = np.zeros((n_train, b_size))
    trainY = np.zeros((n_train, b_size))
    testX_ = np.zeros((n_test, b_size))
    testY = np.zeros((n_test, b_size))
    k_list_train = k_list_[:n_train_keys]
    if n_test_keys == 0:
        # The keys for test data are the same as those for training data.
        k_list_test = k_list_train
    else:
        # The keys for test data are different from those for training data.
        k_list_test = k_list_[n_train_keys:]
    # generate ciphertexts without duplicates
    ## n_all: number of all data (i.e., n_train + n_test)
    c_list = []
    while True:
        i = secrets.randbits(32)
        if i not in c_list:
            c_list.append(i)
        if len(c_list) == 2**n_all:
            break
    # generate training data
    trainY, trainX_ = get_data(k_list_train, n_round_, c_list[:n_train], n_train)
    # generate test data
    testY, testX_ = get_data(k_list_test, n_round_, c_list[n_train:], n_test)
    # reshape 2d arrays to 3d arrays for the input of our LSTM model
    trainX = trainX_.reshape((n_train, blocksize, 1))
    testX = testX_.reshape((n_test, blocksize, 1))
    return trainX, trainY, testX, testY, index
```

Listing 1.2: `get_data` function

```
import numpy as np
from simon import Simon as CipherClass # self-made function

# k_list_: a list of keys, n_round_: number of rounds,
# c_list_: a list of ciphertexts, n_data_: number of training and test data
def get_data(k_list_, n_round_, c_list_, n_data_):
    f_bits = '0%db' % blocksize
    # generate arrays for plaintext-ciphertext (pc) pairs
    p = np.zeros((2**n_data_, blocksize))
    c = np.zeros((2**n_data_, blocksize))
    # generate plaintexts without duplicates
    k_list_len = len(k_list_)
    dvide_n_data = n_data_/k_list_len
    for k in range(k_list_len):
        # generate cipher instance
        # a, b, c: rotation parameter, k_size: key size, w_size: word size
        cInstance = CipherClass(k_list_[k], a, b, c, k_size, w_size, n_round_)
        for i in range(k*divide_n_data, (k+1)*divide_n_data):
            # store the i-th ciphertext (binary format) in the array
            c[i,] = np.array([int(j) for j in list(format(c_list_[i], f_bits))])
            # generate the plaintext corresponding the i-th ciphertext
            pt = cInstance_.dec(c_list_[i].to_bytes(b_size//8, byteorder='big'))
            # store the i-th ciphertext (binary format) in the array
            p[i,] = np.array([int(j) for j in list(format(int.from_bytes(pt, 'big'), f_bits)
                )])
    return p, c
```

Listing 1.3: `create_model` function

```
import secrets

# n_hidden_units_: number of hidden units, n_layers_: number of hidden layers
def create_model(n_hidden_units_, n_hidden_layers_):
    # generate return_sequences_list for stacked LSTM model
    return_sequences_list = [True]*n_hidden_layers_
    return_sequences_list[-1] = False
    # define neural network model: Sequential
    model = Sequential()
    # add an input layer
    model.add(LSTM(
        n_hidden_units_,
        input_shape=(blocksize, 1),
        return_sequences=return_sequences_list[0]))
    # add hidden layers
    for i in range(1, n_hidden_layers_):
        model.add(LSTM(
            n_hidden_units_,
            return_sequences=return_sequences_list[I]))
    # add an output layer:
    model.add(Dense(blocksize))
    return model
```