Output Prediction Attacks on Block Ciphers using Deep Learning*

Hayato Kimura[¶, §], Keita Emura[§], Takanori Isobe[†, §], Ryoma Ito[§], Kazuto Ogawa[§], and Toshihiro Ohigashi[¶, §]

[¶]Tokai University, Japan.
[§]National Institute of Information and Communications Technology (NICT), Japan.
[†]University of Hyogo, Japan.

June 9, 2021

Abstract

Cryptanalysis of symmetric-key ciphers, e.g., linear/differential cryptanalysis, requires an adversary to know the internal structures of the target ciphers. On the other hand, deep learning-based cryptanalysis has attracted significant attention because the adversary is not assumed to have knowledge of the target ciphers except the algorithm interfaces. Such cryptanalysis in a blackbox setting is extremely strong; thus, we must design symmetric-key ciphers that are secure against deep learning-based cryptanalysis. However, almost previous attacks do not clarify what features or internal structures affect success probabilities. Although Benamira et al. (Eurocrypt 2021) and Chen et al. (ePrint 2021) analyzed Gohr's results (CRYPTO 2019), they did not find any deep learning specific characteristic where it affects the success probabilities of deep learning-based attacks but does not affect those of linear/differential cryptanalysis. Therefore, it is difficult to employ the results of such cryptanalysis to design deep learning-resistant symmetric-key ciphers. In this paper, we focus on two toy SPN block ciphers (small PRESENT and small AES) and one toy Feistel block cipher (small TWINE) and propose deep learning-based output prediction attacks. Due to its small internal structures, we can construct deep learning models by employing the maximum number of plaintext/ciphertext pairs, and we can precisely calculate the rounds in which full diffusion occurs. Specifically for the SPN block ciphers, we demonstrate the following: (1) our attacks work against a similar number of rounds attacked by linear/differential cryptanalysis, (2) our attacks realize output predictions (precisely ciphertext prediction and plaintext recovery) that are much stronger than distinguishing attacks, and (3) swapping the component order or replacement components affects the success probabilities of the proposed attacks. It is particularly worth noting that this is a deep learning specific characteristic because swapping/replacement does not affect the success probabilities of linear/differential cryptanalysis. We also confirm whether the proposed attacks also work on the Feistel block cipher. We expect that our results will be an important stepping stone in the design of deep learning-resistant symmetric key ciphers.

1 Introduction

Unlike public-key cryptography, where security is reduced to mathematically difficult problems, the security of symmetric-key cryptography is evaluated by resistance against classical attacks,

*This work was done when the first author, Hayato Kimura, was a master student at the Tokai University, Japan, and was a research assistant at the National Institute of Information and Communications Technology (NICT), Japan.

e.g., differential, linear, and integral attacks. Specifically, corresponding statistical characteristics, e.g., differential, linear, and integral characteristics, are searched by automatic evaluation programs and tools, e.g., SAT and MILP solvers. If there is a significant security margin against these characteristics, the cipher can be considered secure against these attacks. Generally, these evaluations require deep knowledge of the target algorithms and state-of-the-art cryptanalysis techniques because automatic evaluation programs and tools must be customized for different target algorithms and attacks.

Recently, deep learning-based cryptanalysis has received significant attention in the symmetric-key cryptography field [1, 4, 5, 6, 7, 9, 10, 11, 12, 13, 16, 20, 21, 23, 32, 35, 36]. Remarkably, this type of attack does not require knowledge of the target ciphers, except algorithm interfaces, i.e., the attack is feasible even if the adversary does not know the algorithm of the target ciphers. Such cryptanalysis in a blackbox setting is extremely strong in that the adversary can mount an attack with minimum knowledge of target ciphers and cryptanalysis techniques. In this context, we must consider deep learning-based cryptanalysis when designing symmetric-key ciphers. However, previous studies [1, 4, 5, 6, 7, 9, 10, 11, 12, 13, 16, 20, 21, 23, 32, 35, 36] have not clarified which features or internal structures affect success probabilities. Recently, Benamira et al. [7] and Chen et al. [10] confirmed whether or not characteristics explored by Gohr [16] can actually be employed in the classical distinguishing attacks. These results may be employed to design deep learning-resistant symmetric-key ciphers, however, it does not seem to be enough because they did not find any deep learning specific characteristic in such a way that it affects the success probabilities of deep learning-based attacks but does not affect those of classical attacks such as linear/differential attacks. Such a deep learning specific characteristic is important because it may cause vulnerabilities against deep learning-based cryptanalysis. Thus, it is difficult to employ previous results of these attacks to design such deep learning-resistant symmetric-key ciphers.

1.1 Our Contribution

In this paper, we present new deep learning-based attacks on block ciphers in a *blackbox setting* where the adversary does not know the algorithm of the target ciphers, except algorithm interfaces such as key and block sizes. Deep learning-based cryptanalysis in a blackbox setting allows us to construct deep learning models for our attacks, e.g., ciphertext prediction and plaintext recovery, using pre-obtained input/output pairs, and then we can use these models to evaluate these attacks. However, in such a setting, it is difficult to clarify correlations between the evaluation results obtained by deep learning-based cryptanalysis and the characteristics of the target block ciphers. To solve this problem, we apply a *whitebox analysis* technique to our evaluation phase using deep learning models. The whitebox analysis is to explore the relationship between the ability of deep learning-based attacks and the classical attacks such as linear/differential attacks; therefore, it may be possible to clarify correlations between the evaluation results obtained by deep learning-based cryptanalysis and the characteristics of the target block ciphers.

To obtain highly accurate results from the whitebox analysis in a black box setting, we should perform comprehensive analyses using all input/output pairs. That is, it is not appropriate to target the reduced-round block ciphers, because they have the same block size as the original block ciphers, e.g., 64 or 128 bits, and we cannot use all input/output pairs. For this reason, we design toy block ciphers with a small block size such as 16 bits, and perform the whitebox analysis against these toy block ciphers. Naturally, we have an important task to explore whether the evaluation results obtained by our whitebox analysis against toy block ciphers can be applied to the original block ciphers with a large block size. In addition, we also have an important task to perform the whitebox analysis against the toy block ciphers with the smaller number of input/output pairs.

This study is in the first stage to complete such tasks and we leave them as important future works. The details of our contributions in this study are given as follows.

1.1.1 New Deep Learning-based Output Prediction Attacks.

We focus on two toy SPN block ciphers (16-bit block variants of PRESENT [8] called small PRESENT-[4] and an AES-like cipher called small AES) and one toy Feistel block cipher (a type-II generalized Feistel structure with 4 branches called small TWINE) because the linear/differential probabilities of 16-bit permutations can be computed. This enables us to compare the power of these classical attacks and the proposed deep learning-based attacks which guess the ciphertext/plaintext from the corresponding plaintext/ciphertext without any knowledge of keys.

Due to its small internal structures, we can construct deep learning models by exploiting the maximum number of plaintext/ciphertext pairs, and we can precisely calculate linear/differential probability in each round. We demonstrate that the proposed attacks work against a similar number of rounds attacked by linear/differential cryptanalysis. For small PRESENT-[4], we successfully mount output prediction attacks on 4 rounds, while the number of rounds that can be attacked by differential distinguisher is also 4. For small AES and small TWINE, we can mount prediction attacks on 1 and 3 rounds, while differential distinguishing attacks reach 2 and 7 rounds, respectively. Note that our attacks realize output predictions (i.e., ciphertext prediction and plaintext recovery) that are much stronger than distinguishing attacks even without knowing the algorithm of the target ciphers. Nevertheless, for small TWINE, the number of rounds attacked by the proposed attack is significantly smaller than that by the linear/differential attacks. It will be a future work to clarify this cause.

In addition, to confirm the accuracy of the success probability obtained from the proposed attacks, we perform the additional experiments with 10000 trials (instead of 100 trials). As a result, we demonstrate that the additional experiments with a small number of secret keys are sufficient to obtain the best success probability, and therefore the proposed attacks lead to reliable results.

1.1.2 Whitebox Analysis for Deep Learning-based Attacks.

To investigate the relationship between the internal components and success probability of our deep learning-based attacks, we swap or replace internal components on the toy SPN block cipher, especially on 4-round small PRESENT-[4], and evaluate the impact of these modifications relative to the success probability of the prediction attacks. The toy Feistel block cipher, i.e., small TWINE, is excluded from this investigation because Feistel block ciphers generally use the same components in both the encryption and decryption algorithms. As a result, we find that swapping the component order or replacement of components significantly affects success probabilities of the proposed attacks. It is particularly worth noting that this is a deep learning specific characteristic because component swapping and replacements that we did in this paper do not affect success probabilities of linear/differential cryptanalysis. We expect that our results will be an important foundation in the design of deep learning-resistant symmetric-key ciphers.

1.2 Comparison with Existing Studies

Table 1 compares existing deep learning-based attacks [1, 4, 5, 6, 7, 9, 10, 11, 12, 13, 16, 20, 21, 23, 32, 35, 36] and our attacks. For comparison, we focus specifically on whether these attacks correspond to a deep learning-based attack in a *blackbox setting* and a deep learning-based attack with the *whitebox analysis*. When an adversary performs a deep learning-based attack in a non-blackbox setting,

Table 1: Comparison of Deep Learning-based Cryptanalysis. OP:=Output Prediction, PR:=Plaintext Recovery, KR:=Key Recovery, DD:=Differential Distinguisher, LD:=Linear Distinguisher, and DLD:=Differential-Linear Distinguisher.

	Cipher (Block size)	Structures	Blackbox Setting	Target	#Round (#Full)	Whitebox Analysis
BSS08 [4]	Serpent (128 bits)	SPN	No	DD	7 (32)	No
AAAA12 [1]	Simplified DES (12 bits)	Feistel	Yes	OP	2	No
DH14 [13]	Simplified DES (12 bits)	Feistel	Yes	KR/DD	2	No
Gohr19 [16]	Speck32/64 (32 bits)	Feistel	No[1]	KR/DD	12 (22)	Yes
XHY19 [35]	DES (64 bits)	Feistel	Yes	PR	2 (16)	No
CY20 [9]	Speck32/64 (32 bits)	Feistel	No	KR/DD	13 (22)	Yes
CY20 [9]	DES (64 bits)	Feistel	No	KR/DD	8 (16)	Yes
HLZW20 [20]	DES (64 bits)	Feistel	No	KR/LD	5 (16)	No
So20 [32]	Simplified DES (8 bits)	Feistel	No	KR/LD	8 (8)	No
So20 [32]	Speck32/64 (32 bits)	Feistel	No	KR/LD	22 (22)	No
So20 [32]	Simon32/64 (32 bits)	Feistel	No	KR/LD	32 (32)	No
YK20 [36]	Speck32/64 (32 bits)	Feistel	No	DD	9 (22)	Yes
YK20 [36]	Simon32/64 (32 bits)	Feistel	No	DD	12 (32)	Yes
YK20 [36]	GIFT 64 (64 bits)	SPN	No	DD	8 (28)	Yes
BBDC21 [5]	Gimli-Perm. (384 bits)	N/A	No	DD	8 (48)	No
BBDC21 [5]	ASCON-Perm. (320 bits)	N/A	No	DD	3 (16)	No
BBDC21 [5]	KNOT-256 (256 bits)	Feistel	No	DD	10 (28)	No
BBDC21 [5]	KNOT-512 (512 bits)	Feistel	No	DD	12 (52)	No
BBDC21 [5]	CHASKEY-Perm. (128 bits)	ARX	No	DD	4 (12)	No
BGMLT21 [6]	Speck32/64 (32 bits)	Feistel	No	KR/DD	13 (22)	Yes
BGMLT21 [6]	Simon32/64 (32 bits)	Feistel	No	KR/DD	16 (32)	Yes
BGPT21 [7]	Speck32/64 (32 bits)	Feistel	No	DD	7 (22)	No
BGPT21 [7]	Simon32/64 (32 bits)	Feistel	No	DD	8 (32)	No
CY21 [10]	CHASKEY-Perm. (128 bits)	ARX	No	DLD	4 (12)	Yes
CY21 [10]	DES (64 bits)	Feistel	No	DLD	6 (16)	Yes
CY21 [10]	Speck32/64 (32 bits)	Feistel	No	DLD	7 (22)	Yes
CY21 [11]	Speck32/64 (32 bits)	Feistel	No	KR/DD	13 (22)	Yes
CY21 [11]	Speck48/72 (48 bits)	Feistel	No	KR/DD	12 (22)	Yes
CY21 [11]	Speck48/96 (48 bits)	Feistel	No	KR/DD	12 (23)	Yes
CY21 [12]	DES (64 bits)	Feistel	No	DD	6 (16)	No
CY21 [12]	Speck32/64 (32 bits)	Feistel	No	KR/DD	11 (22)	No
CY21 [12]	PRESENT (64 bits)	SPN	No	DD	7 (31)	No
HRC21 [21]	Simon32/64 (32 bits)	Feistel	No	KR/DD	13 (32)	No
ITYY21 [23]	TWINE (64 bits)	Feistel	No	DD	8(36)	No
This paper	small PRESENT-[4] (16 bits)	SPN	**Yes**	OP	4	**Yes**
This paper	small AES (16 bits)	SPN	**Yes**	OP	1	**Yes**
This paper	small TWINE (16 bits)	Feistel	**Yes**	OP	3	**Yes**

[1] Gohr described in his paper [16] that *we consider it interesting that this much knowledge about the differential distribution of round-reduced Speck can be extracted from a few million examples by black-box methods.* However, his *black-box methods* are different from our defined blackbox setting. For this reason, we consider his proposed model as a non-blackbox setting.

he/she must have knowledge of the target ciphers and state-of-the-art cryptanalysis techniques. This impairs the original function of a deep learning-based attack in such a way that it does not require any knowledge of the target ciphers and state-of-the-art cryptanalysis techniques, except algorithm interfaces. In addition, even if an adversary performs a deep learning-based attack with the whitebox analysis in a non-blackbox setting, this should not lead to accurate evaluations of a deep learning-based attack. In summary, it is important to perform a deep learning-based attack with the whitebox analysis in a blackbox setting. As shown in Table 1, the proposed attacks are the first deep learning-based output prediction attacks with *whitebox analysis* on both SPN and Feistel structures in a *blackbox setting*.

Regarding whitebox analysis, Danziger et al. presented deep learning-based attacks that predict key bits of 2-round DES from a plaintext/ciphertext set, and analyze the relationship between these

attacks and the differential probability [13]. They compared variants employing several types of S-boxes with different properties for differential attacks, and they concluded that there is a nontrivial relationship between the differential characteristics and success probability of their deep learning-based attacks. However, their results are very limited because they targeted a two-round Feistel construction, which is quite insecure even if component is ideal function. It is unclear that how much the property of the internal components affects the security of the while construction. In addition to improve the Gohr's deep learning-based attack [16], Benamira et al. [7] and Chen et al. [10] improved success probability of traditional distinguishers using characteristics that are expected to be reacted by the Gohr's attack. In other words, their work confirms whether or not characteristics explored by Gohr can actually be employed in the traditional distinguishing attacks and they did not find any deep learning specific characteristic. On the other hand, we calculated the ability of the traditional distinguisher and our deep learning-based attack, and compared them to investigate a relationship between them. Then, we find a deep learning specific characteristic of small-PRESENT. To sum up, to the best of our knowledge, our results are the first ones that perform the whitebox analysis.

Alani and Hu presented plaintext recovery attacks on DES, 3-DES and AES [2, 22] that guess plaintexts from given ciphertexts. They claimed that attacks on DES, 3-DES and AES are feasible with 2^{11}, 2^{11} and 1741 ($\simeq 2^{10.76}$) plaintext/ciphertext pairs, respectively. However, Xiao et al. doubted the correctness of their results [2, 22] because they could not be reproduced. Baek et al. also pointed this out in the literature [30]. Therefore, we exclude these results in Table 1. Mishra et al. reported that they mounted output prediction attacks on full-round PRESENT; however it did not work well [15]. In addition, some results have been obtained classical ciphers such as Caesar cipher, Vigenere cipher and Enigma [14, 17, 18, 28].

Other machine learning-based analyses have also been reported, e.g., [27, 26]. Tan et al. demonstrated that deep learning can be used to distinguish ciphertexts encrypted by AES, Blowfish, DES, 3-DES, and RC5, respectively [33], for detection of the encryption algorithm that malware utilizes. Alshammari et al. attempted to classify encrypted Skype and SSH traffic [3].

2 Preliminaries

In this section, we introduce two toy SPN block ciphers (small PRESENT-[n] [25] and small AES) and one toy Feistel block cipher (small TWINE).

small PRESENT-[n]: PRESENT [8] is a lightweight SPN block cipher with a 64-bit block size, 31 rounds, and a key length of either 80 or 128 bits. A toy model of PRESENT called small PRESENT-[n] [25] has been proposed to analyze PRESENT. We show the round function of small PRESENT-[n] in Fig. 1. Since the block size is $4n$, small PRESENT-[16] is equivalent to PRESENT-80. The variant n, which specifies the block size and round key length, allows us to control the round of full diffusion. The S-box has 4-bit input and output. We give the correspondence table in Table 2 that maps $\mathbb{F}_2^4 \to \mathbb{F}_2^4$.

The pLayer is described as bit permutation $P(i)$, which is defined as follows. Note that this is a generalization of that of PRESENT, and is equivalent to that of PRESENT when $n = 16$. $P(i)$ is used for encryption and $P^{-1}(i)$ is used for decryption.

$$P(i) = \begin{cases} n \times i \bmod (4n-1) & (0 \le i < 4n-1) \\ 4n-1 & (i = 4n-1) \end{cases}$$

$$P^{-1}(i) = \begin{cases} 4 \times i \bmod (4n-1) & (0 \le i < 4n-1) \\ 4n-1 & (i = 4n-1) \end{cases}$$

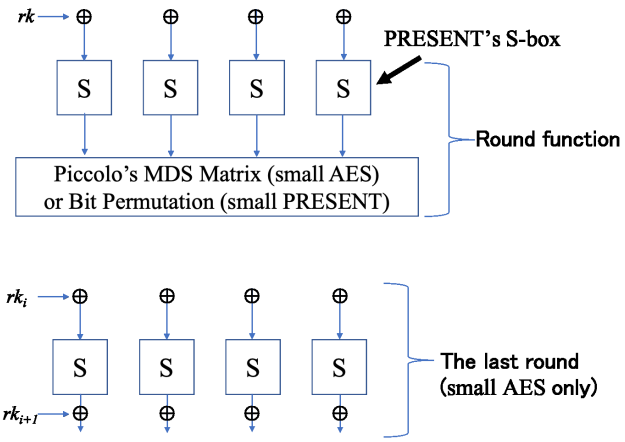

Figure 1: Round Functions (small PRESENT and small AES)

Table 2: S-box (PRESENT and small PRESENT-[n])

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

For key scheduling, the key scheduling algorithm of PRESENT-80 is executed, and the $4n$ rightmost bits are used as round keys rk_i.

small AES: We design small AES in this paper. The round function of small AES is shown in Fig. 1. Here, the S-box and key scheduling are the same as those of small PRESENT-[4]. The maximum distance separable (MDS) matrix (over $GF(2^4)$ defined by the irreducible polynomial $x^4 + x + 1$) is the same as that of Piccolo [31], which is expressed as follows.

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

When a 16-bit input $X_{(16)}$ is given, the output is computed as ${}^t(x_{0(4)}, x_{1(4)}, x_{1(4)}, x_{1(4)}) \leftarrow M \cdot {}^t(x_{0(4)}, x_{1(4)}, x_{1(4)}, x_{1(4)})$.

small TWINE: We also design small TWINE in this paper. For our design, we adopt the type-II generalized Feistel structure with 4 branches and the similar F function as TWINE, which consists of round key operation and 4-bit S-box, as shown in Fig. 2. The S-box and key scheduling are the same as those of small PRESENT-[4]. Two sub-round keys, rk_i^0 and rk_i^1, are used in each round, which are generated from the round key rk_i as follows:

$$rk_i^0 = (rk_i \gg 12)\ \&\ \mathsf{0xF},$$
$$rk_i^1 = (rk_i \gg 8)\ \&\ \mathsf{0xF},$$

where $\gg$ and $\&$ are bitwise right shift operation and bitwise AND operation, respectively.

3 Proposed Attacks

In this section, we present the proposed deep learning-based output prediction attacks in a blackbox setting. To realize the proposed attacks, we construct deep learning models for ciphertext prediction

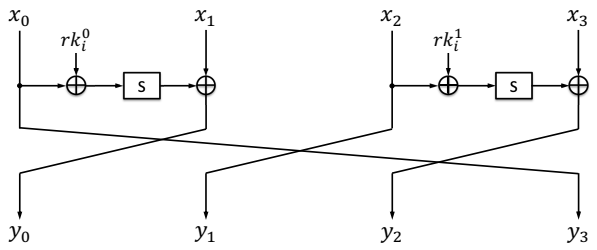

Figure 2: Round Function (small TWINE)

and plaintext recovery, respectively. In the following, we first discuss the goals of these attacks and then explain the construction of deep learning models and their evaluation.

3.1 Goals of Attack

To date, the relationship between the abilities classical attacks and deep learning-based ones has not been clarified. Here, we focus on clarifying this relationship. We then revisit the common sense in previous works using deep learning-based attacks. The targets of this work are summarized as follows:

1. We clarify the difference in capabilities between classical attacks and deep learning-based attacks. Specifically, we compare the success probabilities of deep learning-based attacks with those of classical attacks.

2. Swapping the component order or replacement of components does not affect the success probability of linear/differential cryptanalysis. We clarify how such modifications to cipher's algorithms affect the success probability of deep learning-based attacks.

We evaluate the success probabilities of attacks using the following settings.

Known-plaintext attack setting: In this setting, the adversary is given 2^{n-1} plaintext/ciphertext pairs relating to a single secret key, and the pairs are used as training data to construct a deep learning model.

Blackbox setting: In this setting, the adversary does not have knowledge about the target block ciphers, except algorithm interfaces such as key and block sizes.

In both of these settings, the adversary is a very weak cryptographic attacker.

The blackbox setting assumes that the adversary does not know the internal structures of the cipher. In addition, the adversary does not know the cipher is a permutation. The blackbox setting also assumes that the adversary only knows the input-output format and possesses deep learning knowledge.

Regarding attack settings, a ciphertext-only attack setting, which allows the adversary to obtain only the ciphertext, is the weakest setting. However, information-theoretically no information is provided to the adversary in the setting except for several special cases, e.g., the broadcast setting of RC4 [29]. In fact, the attack in this setting is practically impossible. The known-plaintext attack is the next weakest setting. In this setting, the adversary can obtain some information from the given plaintext/ciphertext pairs and use these pairs for the attacks. The other attack settings, e.g., chosen-plaintext attack setting, require the adversary to possess some knowledge about the

Table 3: Hyperparameters

Hyperparameters	Search ranges
Number of hidden nodes	100, 200, 300, 400, 500
Initial value of learning rates	0.0001, 0.001, 0.01
Number of hidden layers	1, 2, 3, 4, 5, 6, 7
Optimizers	SGD, Adam [24], RMSprop [34]

ciphertext, and the adversary in this setting is stronger than the adversary in the known-plaintext attack setting. Thus, we employ the known-plaintext attack setting.

In these settings, we decide the adversary's goal to ciphertext prediction or plaintext recovery, and we evaluate the success probabilities of these attacks. The ciphertext prediction and plaintext recovery attacks are summarized as follows:

Ciphertext prediction attack: In this attack, the adversary obtains 2^{n-1} plaintext/ciphertext pairs regarding a secret key, where n is the block size. Then, the adversary predicts a ciphertext of a plaintext not included in the previously given pairs.

Plaintext recovery attack: In this attack, the adversary obtains 2^{n-1} plaintext/ciphertext pairs regarding a secret key, and then the adversary recovers a plaintext of a ciphertext that is not included in the pairs given previously.

If the ciphertext prediction attack is possible, forgery of the Cipher-based Message Authentication Code (CMAC) is possible. If the plaintext recovery attack is possible, the adversary can obtain the plaintext of any ciphertext without possessing the secret key used for encryption.

3.2 Neural Network and Hyperparameters

Deep learning allows us to automatically extract features unlike statistical machine learning techniques, e.g., Bayesian inference. Deep learning treats nonlinear separable problems; thus, it appears to work well for simulating cryptographic functions with nonlinearity. Hyperparameters such as the initial learning rate, number of hidden nodes (neurons), and optimizers, are defined prior to the learning phase and are used to construct models. These parameters affect model performance; thus, they are optimized using assessment metrics.

In this paper, we consider ciphertext prediction and plaintext recovery as regression problems with supervised learning where plaintext/ciphertext pairs are used as training data. To solve these problems, we must extract numerous features from the plaintext/ciphertext pairs obtained under the known-plaintext attack; therefore, we employ long short-term memory (LSTM) which is a type of recurrent neural networks (RNN) [19]. By using the LSTM, which enables long-term memory of input sequences, we consider that numerous features can be extracted from plaintext/ciphertext pairs, i.e., the inputs to our deep learning models. We then optimize hyperparameters, e.g., number of hidden nodes, initial learning rates, number of hidden layers, and optimizers. Table 3 shows the search ranges for each hyperparameter. During the hyperparameter optimization, we use different secret keys from those used in the construction of deep learning models because we strictly evaluate the success probabilities of ciphertext prediction and plaintext recovery without depending on secret keys. In the following, the procedure to optimize hyperparameters is similar to constructing deep learning models, with the exception of the number of secret keys.

3.3 Deep Learning Models and Their Evaluation

We construct and evaluate deep learning models for ciphertext prediction according to the following procedure. Note that we show the plaintext recovery case in parentheses.

Step 1. The adversary obtains 2^{n-1} plaintext/ciphertext pairs under the known-plaintext attack. In our experiments, we randomly select 2^{n-1} plaintexts and generate ciphertexts corresponding to the selected plaintexts.

Step 2. The adversary uses the obtained plaintext/ciphertext pairs as training data to construct deep learning models. Then, the adversary constructs a deep learning model for ciphertext prediction (plaintext recovery) using the plaintexts (ciphertexts) as inputs and the ciphertexts (plaintexts) as the correct outputs.

Step 3. The adversary uses the remaining 2^{n-1} plaintexts (ciphertexts), which were not used as training data, to evaluate the constructed deep learning models. The adversary uses these plaintexts (ciphertexts) as the input to the constructed deep learning models. Then, the adversary predicts the unknown ciphertext (plaintext) corresponding to each plaintext (ciphertext).

Step 4. The adversary calculates the percentage of exact match between the predicted ciphertext (plaintext) and the correct ciphertext (plaintext) as the predicted probability.

If the predicted probability is greater than 2^{1-n}, we consider the proposed attacks to be successful. This means that an attacker without knowledge of the target algorithms can predict the output value with a higher probability than a random probability.

4 Comparison with Linear/Differential Cryptanalysis

In this section, we apply the proposed attacks to three toy block ciphers, i.e., small PRESENT-[4], small AES, and small TWINE, and compare the number of rounds attacked by the proposed attack to that of existing classically attacks such as linear/differential attacks.

4.1 Experiments

In our experiments, we implement the proposed attack using Keras[1], which is a deep learning library, and we employ TensorFlow as the backend. As an initial setting, we use common experimental hyperparameter values (see Table 4). Our experiments involve the following two sub-experiments (see Fig. 3).

In the first sub-experiment, called Experiment 1, we optimize the hyperparameters for the target block ciphers in each round using the proposed attack, as described in Section 3.2. For our hyperparameter optimization, we employ Optuna[2], which is an automatic optimization tool, and use its default search algorithm. The indication for our hyperparameter optimization is the success probability of ciphertext prediction or plaintext recovery. In our hyperparameter optimization, we obtain 100 hyperparameter candidates from the plaintext/ciphertext pairs generated by 20 secret keys. From these candidates, we select the optimized hyperparameter with the highest average success probabilities of ciphertext prediction or plaintext recovery. Each average success probability

[1]https://github.com/keras-team/keras
[2]https://github.com/optuna/optuna

Table 4: Experimental hyperparameters

Hyperparameters	Values
Number of plaintext/ciphertext pairs for learning phase	2^{15}
Number of plaintext/ciphertext pairs for prediction phase	2^{15}
Number of input layer nodes	250
Number of output layer nodes	1
Batch size	100
Number of epochs	100

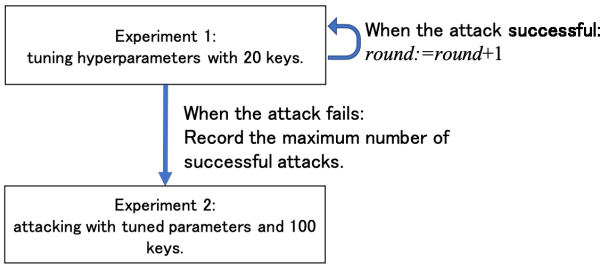

Figure 3: Experimental flow

is calculated from 2^{15} randomly generated plaintext/ciphertext pairs. If the average success probabilities of ciphertext prediction or plaintext recovery with the optimized hyperparameter is greater than 2^{-15}, then the number of rounds for finding the optimized hyperparameter is incremented by one; otherwise, the second sub-experiment is executed using the optimized hyperparameter.

In the second sub-experiment, called Experiment 2, we use the optimized hyperparameters obtained in Experiment 1, execute the proposed attacks for ciphertext prediction or plaintext recovery with randomly generated 100 secret keys, and calculate the average success probabilities of ciphertext prediction or plaintext recovery. It should be noted here that the secret keys used in Experiment 2 differ from those used in Experiment 1. After clarifying the number of attacked rounds for the target block ciphers by Experiment 2, we compare the proposed attacks to the classical linear/differential attacks using the experimental results and linear/differential probability of the target block ciphers.

4.2 Experimental Results and Comparison

Table 5 shows the experimental results of Experiment 2 using the optimized hyperparameter obtained in Experiment 1.

First, we compare the proposed attacks and the classical linear/differential attacks for small PRESENT-[4]. From the experimental results, the proposed attacks succeed up to 5 rounds for ciphertext prediction and up to 4 rounds for plaintext recovery against small PRESENT-[4]. Although the average success probability of ciphertext prediction for the 5-round small PRESENT-[4] is nearly 2^{-15}, the average success probability of plaintext recovery for the 4-round small PRESENT-[4] is sufficiently greater than 2^{-15}. That is, we consider that the number of rounds attacked by the proposed attacks is at least 4. On the other hand, from the precisely calculated differential probability of small PRESENT-[4] (see Table 6), the largest number of rounds attacked by the differential attack is 4. Similarly, from the precisely calculated linear probability, the largest number

Table 5: Average success probabilities of ciphertext prediction/plaintext recovery by the proposed attacks

Cipher	Round	Category of Attack	# nodes of hidden layer	# layers of hidden layer	Initial learning rate	Optimizer	Succ. prob.
small PRESENT-[4]	1	Ciphertext Prediction	400	5	0.001	Adam	1
		Plaintext Recovery	100	2	0.001	RMSprop	1
	2	Ciphertext Prediction	400	4	0.001	RMSprop	1
		Plaintext Recovery	400	1	0.001	Adam	1
	3	Ciphertext Prediction	300	6	0.001	RMSprop	1
		Plaintext Recovery	300	5	0.001	RMSprop	1
	4	Ciphertext Prediction	300	4	0.01	Adam	$2^{-5.63}$
		Plaintext Recovery	300	1	0.01	Adam	$2^{-14.50}$
	5	Ciphertext Prediction	200	7	0.001	Adam	$2^{-14.08}$
		Plaintext Recovery	300	6	0.001	Adam	$2^{-15.73}$
small AES	1	Ciphertext Prediction	300	4	0.001	RMSprop	1
		Plaintext Recovery	200	4	0.001	RMSprop	1
	2	Ciphertext Prediction	300	1	0.01	Adam	$2^{-16.02}$
		Plaintext Recovery	200	2	0.01	Adam	$2^{-15.00}$
small TWINE	1	Ciphertext Prediction	300	3	0.001	RMSprop	1
		Plaintext Recovery	300	2	0.001	Adam	$2^{-0.01}$
	2	Ciphertext Prediction	400	4	0.001	RMSprop	$2^{-0.01}$
		Plaintext Recovery	500	3	0.001	RMSprop	$2^{-0.01}$
	3	Ciphertext Prediction	300	2	0.001	RMSprop	$2^{-10.46}$
		Plaintext Recovery	400	2	0.001	RMSprop	$2^{-9.72}$
	4	Ciphertext Prediction	200	4	0.001	RMSprop	$2^{-14.61}$
		Plaintext Recovery	100	1	0.01	RMSprop	$2^{-15.49}$
	5	Ciphertext Prediction	300	5	0.01	RMSprop	$2^{-15.64}$
		Plaintext Recovery	500	4	0.001	RMSprop	$2^{-15.16}$

of rounds attacked by the linear cryptanalysis is also 4. Therefore, the largest number of rounds attacked by the proposed attacks and that of the linear/differential attacks are equivalent on small PRESENT-[4].

Next, we compare the proposed attacks and the classical linear/differential attacks for small AES. From Table 5, we evaluate the largest number of rounds attacked by the proposed attacks is 1. From the precisely calculated linear/differential probabilities, the largest number of rounds attacked by the differential attack is 2, and that of the linear attack is 3. Similarly, we compare the proposed attacks and the classical linear/differential attacks for small TWINE. We obtain that the largest

Table 6: Maximum differential probabilities of small PRESENT-[4], small AES, and small TWINE

Cipher	Round	Maximum differential probability
small PRESENT-[4]	1	2^{-2}
	2	2^{-4}
	3	2^{-7}
	4	2^{-9}
	5	2^{-14}
	6	2^{-15}
	7	2^{-15}
	8	2^{-15}
small AES	1	2^{-2}
	2	2^{-9}
	3	2^{-11}
	4	2^{-11}
	5	2^{-11}
	6	2^{-11}
	7	2^{-11}
	8	2^{-12}
small TWINE	1	2^{0}
	2	2^{-2}
	3	2^{-4}
	4	2^{-6}
	5	2^{-7}
	6	2^{-9}
	7	2^{-9}
	8	2^{-11}
	9	2^{-11}
	10	2^{-11}
	11	2^{-11}
	12	2^{-11}

number of rounds attacked by the proposed attack is 3, that of the differential attack is 7, and that of the linear attack is 9. In summary, the largest number of rounds attacked by the proposed attacks is less than that of linear/differential attack for small AES and small TWINE. It should be noted here that the proposed attacks realize ciphertext prediction and plaintext recovery that are much stronger than the distinguishing attacks of linear/differential cryptanalysis. Nevertheless, for small TWINE, the number of rounds attacked by the proposed attacks is significantly smaller than that by the linear/differential attacks. It will be a future work to clarify this cause.

5 Accuracy of the Experimental Results

In Section 4, we have presented the experimental results of Experiment 1 with 20 secret keys and Experiment 2 with 100 secret keys. These experimental results may seem correct. However, due to the small number of secret keys used in these experiments, we should have further discussion to confirm the accuracy of the experimental results. To do this, this section shows two additional experimental results on the 3-round small TWINE with 100 secret keys for Experiment 1 and 10000 secret keys for Experiment 2, respectively. The reason why we choose the 3-round small TWINE for confirming the accuracy is explained as follows. If we choose a target with the probability of 1 or 2^{-15}, then it seems difficult to observe how the number of secret keys affect the accuracy. As shown in Table 5, the average success probabilities of ciphertext prediction and plaintext recovery

Table 7: Experimental results of Experiment 1' for the 3-round small TWINE.

Category of Attack	# keys	# trials	# nodes of hidden layer	# layers of hidden layer	Initial learning rate	Optimizer	Succ. prob.
Ciphertext Prediction	20	100	300	2	0.001	RMSprop	$2^{-11.42}$
	100	40	300	7	0.001	RMSprop	$2^{-11.26}$
Plaintext Recovery	20	100	400	2	0.001	RMSprop	$2^{-7.80}$
	100	40	100	4	0.001	RMSprop	$2^{-12.82}$

by the proposed attacks in the 3-round small TWINE are approximately $2^{-10.46}$ and approximately $2^{-9.72}$, respectively. Since these probabilities are more likely to vary significantly if the number of keys affects the accuracy, we choose the 3-round small TWINE to be the best target for additional experiments.

5.1 Experiments

We explain the following two additional sub-experiments.

In the first additional sub-experiment, called Experiment 1', we optimize the hyperparameters for the 3-round small TWINE in the same procedures of Experiment 1. Unlike Experiment 1, however, we use the plaintext/ciphertext pairs generated by 100 secret keys instead of 20 secret keys. In the hyperparameter optimization, we examine the impact of the number of secret keys used in Experiment 1' on the experimental results.

In the second additional sub-experiment, called Experiment 2', we obtain the average success probabilities of ciphertext prediction/plaintext recovery for the 3-round small TWINE in the same procedures of Experiment 2 using the hyperparameters optimized by Experiment 1 (see Table 5). Unlike Experiment 2, however, we use the plaintext/ciphertext pairs generated by 10000 secret keys instead of 100 secret keys. In the ciphertext prediction/plaintext recovery, we explore the influence of the number of secret keys used in Experiment 2' on the experimentel results.

5.2 Experimental Results

Table 7 shows the experimental results of Experiment 1' for the 3-round small TWINE. From the table, we can see that the highest average success probabilities obtained from Experiment 1 and Experiment 1' are approximately equal in the hyperparameter optimization for the ciphertext prediction, such as $2^{-11.42}$ and $2^{-11.26}$. Conversely, in the hyperparameter optimization for the plaintext recovery, we can also see that the highest average success probability obtained from Experiment 1 is much higher than that obtained from Experiment 1', such as $2^{-7.80}$ and $2^{-12.82}$. These experimental results imply that the additional hyperparameter optimization with a small number of secret keys is sufficient to obtain the hyperparameters with the best average success probability; therefore, we consider that the hyperparameter optimization presented in Section 4 has led to reliable results.

Table 8 shows the experimental results of Experiment 2' for the 3-round small TWINE. From the table, we can see that the average success probabilities obtained from Experiment 2 and Experiment 2' are approximately equal in both ciphertext prediction and plaintext recovery, such as $2^{-10.46}$ and $2^{-10.64}$ in the ciphertext prediction and $2^{-9.72}$ and $2^{-9.22}$ in the plaintext recovery. These experimental results also imply that the additional experiments with a small number of secret keys are sufficient to obtain the best average success probability; therefore, we consider that the ciphertext prediction/plaintext recovery presented in Section 4 has led to reliable results.

Table 8: Experimental results of Experiment 2' for the 3-round small TWINE.

Category of Attack	# keys	# nodes of hidden layer	# layers of hidden layer	Initial learning rate	Optimizer	Succ. prob.
Ciphertext Prediction	100	300	2	0.001	RMSprop	$2^{-10.46}$
	10000					$2^{-10.64}$
Plaintext Recovery	100	400	2	0.001	RMSprop	$2^{-9.72}$
	10000					$2^{-9.22}$

6 Extended Whitebox Analysis on Small PRESENT-[4]

As shown in Table 5, the average success probability of ciphertext prediction by the proposed attack on the 4-round small PRESENT-[4] is approximately 2^9 times greater than that of plaintext recovery. Conversely, it is considered that the security of the encryption and decryption are equivalent in terms of the linear/differential probabilities on small PRESENT-[4]; thus, the experimental result of the proposed attacks on the 4-round small PRESENT-[4] seems contrary to intuition. In this section, we redesign the 4-round small PRESENT-[4] by swapping or replacing the internal components, e.g., S-box and bit permutation, and execute Experiment 2 against the new designs of the 4-round small PRESENT-[4] to reveal the relationship between the designs of block ciphers and the average success probability of the proposed attacks.

6.1 Experiments

We discuss two types of experiments to investigate the average success probabilities of ciphertext prediction and plaintext recovery by the proposed attacks under the conditions that (1) the substitution layer (sLayer) and inverse function (sLayer-inv) are replaced, and (2) the order of the sLayer and permutation layer (pLayer) is swapped in the encryption and decryption algorithms. The target toy block ciphers are the 4-round small PRESENT-[4] and the 2-round small AES, and small TWINE is excluded from the target of these experiments. This is because the Feistel block ciphers generally use the same components in both the encryption and decryption algorithms. The order of the sLayer and pLayer is the same in both the encryption and decryption algorithms, and sLayer-inv is not used in neither the encryption nor decryption algorithms. Instead of the experiments described in this section, we should compare the number of rounds attacked by the proposed attacks on small TWINE (a type-II Feistel cipher) to that on the other types of the Feistel block ciphers, such as classical, unbalanced, alternating, type-I and type-III Feistel ciphers. This will be our future work.

6.2 Experimental Results

Table 9 shows the experimental results for the new designs of the 4-round small PRESENT-[4]. When the sLayer is replaced with the sLayer-inv, it can be seen from the table that the average success probability of ciphertext prediction is greater than that of the original small PRESENT-[4]. When the order of the sLayer and pLayer is swapped, however, the average success probability of ciphertext prediction is less than that of the original small PRESENT-[4]. The difference in these average success probabilities is relatively large, and therefore, we think that swapping the component order affects the average success probabilities of the proposed attacks. Given that swapping or replacing components does not affect the linear/differential probabilities, we expect that our results can be an important stepping stone for designing deep learning-resistant symmetric-key ciphers.

Table 9: Average of success probabilities when swapping the order of components or replacing components on 4-round small PRESENT-[4]

	Category of Attack	# nodes of hidden layer	# layers of hidden layer	Initial learning rate	Optimizer	Succ. prob.
Original small PRESENT-[4]	Ciphertext Prediction	300	4	0.01	Adam	$2^{-5.63}$
	Plaintext Recovery	300	1	0.01	Adam	$2^{-14.50}$
Replacement of components (Enc: sLayer-inv→pLayer) (Dec: pLayer→sLayer)	Ciphertext Prediction	200	4	0.01	Adam	$2^{-3.75}$
	Plaintext Recovery	500	1	0.001	Adam	$2^{-12.13}$
Swapping the order of components (Enc: pLayer→sLayer) (Dec: sLayer-inv→pLayer)	Ciphertext Prediction	500	1	0.001	Adam	$2^{-12.21}$
	Plaintext Recovery	400	7	0.001	Adam	$2^{-13.74}$

On the other hand, the averages success probability of plaintext recovery for both cases are greater than that of the original small PRESENT-[4], and this result tends to differ from ciphertext prediction. Since the probabilities are nearly 2^{-15}, the results require more detailed analyses to increase reliability, which we leave as a future work.

In the experimental results of the 2-round small AES, all average success probabilities for ciphertext prediction/plaintext recovery by the proposed attacks are less than 2^{-15}. Therefore, these results do not demonstrate whether swapping the component order or replacing the components has any effect on the average success probabilities of the proposed attacks in the 2-round small AES.

7 Conclusion

In this paper, we have presented deep learning-based attacks on three toy block ciphers in a blackbox setting, where the adversary does not know the algorithm of the target ciphers, with the exception of the algorithm interface such as key and block sizes. In addition, we have redesigned the 4-round small PRESENT-[4] by swapping or replacing the internal components, and have investigated the relationship between the new designs of the target cipher and the success probability of the proposed attacks using a whitebox analysis technique. We expect that the obtained results will be a foundation for designing deep learning-resistant symmetric-key ciphers.

As discussed in Section 1.1, this study is in the first stage to complete the following tasks:

- We explore whether the evaluation results obtained by our whitebox analysis against toy block ciphers can be applied to the original block ciphers with a large block size.
- We perform the whitebox analysis against the toy block ciphers with the smaller number of input/output pairs.

In addition, we have the following future works:

- Clarify why the number of rounds attacked by the proposed attacks is significantly smaller than that by the linear/differential attacks for small TWINE.
- Clarify why our deep learning-based attack affects component swapping and replacements that we did although it does not affect success probabilities of linear/differential cryptanalysis.

- Compare the number of rounds attacked by the proposed attacks on small TWINE (a type-II Feistel cipher) to that on the other types of the Feistel block ciphers, such as classical, unbalanced, alternating, type-I and type-III Feistel ciphers.
- Clarify why differences are observed in the average success probabilities regarding the underlying optimizer.
- Clarify how to feedback our results for designing deep learning-resistant symmetric-key ciphers.

Acknowledgments

This work was supported in part by the JSPS KAKENHI Grant Number 19K11971.

References

[1] Khaled M. Alallayah, Alaa H. Alhamami, Waiel AbdElwahed, and Mohamed Amin. Applying neural networks for simplified data encryption standard (SDES) cipher system cryptanalysis. *Int. Arab J. Inf. Technol.*, 9(2):163–169, 2012.

[2] Mohammed M. Alani. Neuro-cryptanalysis of DES and triple-des. In *ICONIP*, pages 637–646, 2012.

[3] Riyad Alshammari and A. Nur Zincir-Heywood. Machine learning based encrypted traffic classification: Identifying SSH and skype. In *IEEE CISDA*, pages 1–8, 2009.

[4] Abbas Ghaemi Bafghi, Reza Safabakhsh, and Babak Sadeghiyan. Finding the differential characteristics of block ciphers with neural networks. *Information Sciences*, 178(15):3118 – 3132, 2008. Nature Inspired Problem-Solving.

[5] Anubhab Baksi, Jakub Breier, Xiaoyang Dong, and Chen Yi. Machine learning assisted differential distinguishers for lightweight ciphers. *2021 Design, Automation & Test in Europe Conference & Exhibition, DATE 2021*, 2021. to appear, available at `https://eprint.iacr.org/2020/571`.

[6] Zhenzhen Bao, Jian Guo, Meicheng Liu, Li Ma, and Yi Tu. Conditional differential-neural cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2021:719, 2021.

[7] Adrien Benamira, David Gerault, Thomas Peyrin, and Quan Quan Tan. A deeper look at machine learning-based cryptanalysis. In *EUROCRYPT*, 2021. to appear, available at `https://eprint.iacr.org/2021/287`.

[8] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In *CHES*, pages 450–466, 2007.

[9] Yi Chen and Hongbo Yu. Neural aided statistical attack for cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2020:1620, 2020.

[10] Yi Chen and Hongbo Yu. Bridging machine learning and cryptanalysis via edlct. *IACR Cryptol. ePrint Arch.*, 2021:705, 2021.

[11] Yi Chen and Hongbo Yu. Improved neural aided statistical attack for cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2021:311, 2021.

[12] Yi Chen and Hongbo Yu. A new neural distinguisher model considering derived features from multiple ciphertext pairs. *IACR Cryptol. ePrint Arch.*, 2021:310, 2021.

[13] M. Danziger and M. A. Amaral Henriques. Improved cryptanalysis combining differential and artificial neural network schemes. In *ITS*, pages 1–5, 2014.

[14] Riccardo Focardi and Flaminia L. Luccio. Neural cryptanalysis of classical ciphers. In *Italian Conference on Theoretical Computer Science*, pages 104–115, 2018.

[15] SK Pal Girish Mishra, SVSSNVG Krishna Murthy. Neural network based analysis of lightweight block cipher PRESENT. In Harmony Search and Nature Inspired Optimization Algorithms, 2019.

[16] Aron Gohr. Improving attacks on round-reduced speck32/64 using deep learning. In *CRYPTO*, pages 150–179, 2019.

[17] Aidan N. Gomez, Sicong Huang, Ivan Zhang, Bryan M. Li, Muhammad Osama, and Lukasz Kaiser. Unsupervised cipher cracking using discrete gans. *CoRR*, abs/1801.04883, 2018.

[18] Sam Greydanus. Learning the enigma with recurrent neural networks. *CoRR*, abs/1708.07576, 2017.

[19] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. In *Neural Computation Volume 9 Issue 8*, pages 1735–1780, 1997.

[20] Botao Hou, Yongqiang Li, Haoyue Zhao, and Bin Wu. Linear attack on round-reduced DES using deep learning. In *ESORICS*, pages 131–145, 2020.

[21] Zezhou Hou, Jiongjiong Ren, and Shaozhen Chen. Cryptanalysis of round-reduced SIMON32 based on deep learning. *IACR Cryptol. ePrint Arch.*, 2021:362, 2021.

[22] Xinyi Hu and Yaqun Zhao. Research on plaintext restoration of AES based on neural network. *Security and Communication Networks*, 2018:6868506:1–6868506:9, 2018.

[23] Mohamed Fadl Idris, Je Sen Teh, Jasy Liew Suet Yan, and Wei-Zhu Yeoh. A deep learning approach for active S-box prediction of lightweight block ciphers. *IACR Cryptol. ePrint Arch.*, 2021:066, 2021.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[25] Gregor Leander. Small scale variants of the block cipher PRESENT. *IACR Cryptol. ePrint Arch.*, 2010:143, 2010.

[26] Ting Lee, Je Sen Teh, Jasy Liew, Suet Yan, Norziana Jamil, and Wei-Zhu Yeoh. A machine learning approach to predicting block cipher security. In *CRYPTOLOGY*, 2020.

[27] Ting Rong Lee, Je Sen Teh, Jasy Suet Yan Liew, Norziana Jamil, and Jiageng Chen. Assessing block cipher security using linear and nonlinear machine learning models. *IACR Cryptol. ePrint Arch.*, 2020:1235, 2020.

[28] Yu Liu, Jianshu Chen, and Li Deng. Unsupervised sequence classification using sequential output statistics. In *NIPS*, pages 3550–3559, 2017.

[29] Itsik Mantin and Adi Shamir. A practical attack on broadcast RC4. In *Fast Software Encryption*, pages 152–164, 2001.

[30] Kwangjo Kim Seunggeun Baek. Recent advances of neural attacks against block ciphers. *2020 Symposium on Cryptography and Information Security (SCIS)*, 2020.

[31] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An ultra-lightweight blockcipher. In *CHES*, pages 342–357, 2011.

[32] Jaewoo So. Deep learning-based cryptanalysis of lightweight block ciphers. *Security and Communication Networks*, 2020:3701067, 2020.

[33] C. Tan and Q. Ji. An approach to identifying cryptographic algorithm from ciphertext. In *ICCSN*, pages 19–23, 2016.

[34] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[35] Ya Xiao, Qingying Hao, and Danfeng Daphne Yao. Neural cryptanalysis: Metrics, methodology, and applications in CPS ciphers. In *IEEE DSC*, pages 1–8, 2019.

[36] Tarun Yadav and Manoj Kumar. Differential-ML distinguisher: Machine learning based generic extension for differential cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2020:913, 2020.