Output Prediction Attacks on SPN 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.

March 24, 2021

Abstract

Cryptanalysis of symmetric-key ciphers, e.g., linear/differential cryptanalysis, requires an adversary to know the internal structures of the targeted ciphers. On the other hand, deep learning-based cryptanalysis has attracted significant attention because the adversary is not assumed to have knowledge of the targeted ciphers except the interfaces of algorithms. Such a blackbox attack is extremely strong; thus we must design symmetric-key ciphers that are secure against deep learning-based cryptanalysis. However, previous attacks do not clarify what features or internal structures affect success probabilities; therefore it is difficult to employ the results of such attacks to design deep learning-resistant symmetric-key ciphers. In this paper, we focus on toy SPN block ciphers (small PRESENT and small AES) and propose deep learning-based output prediction attacks. Due to its small internal structures, we can build learning models by employing the maximum number of plaintext/ciphertext pairs, and we can precisely calculate the rounds in which full diffusion occurs. 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 plaintext recovery and ciphertext prediction) that are much stronger than distinguishing attacks, and (3) swapping the order of components or replacement components affects the success probabilities of the attacks. It is particularly worth noting that swapping/replacement does not affect the success probabilities of linear/differential cryptanalysis. 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 ciphers is evaluated by resistance against existing attacks, e.g., differential, linear, and integral cryptanalysis. 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

[*]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.

deep knowledge of the target algorithms and state-of-the-art cryptanalysis techniques because automatic 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, 8, 9, 10, 11, 14, 18, 19, 21, 33, 34]. Remarkably, this type of attack does not require knowledge of the targeted ciphers, except algorithm interfaces, i.e., the attack is feasible even if the adversary does not know the algorithm of the target ciphers. Such blackbox attacks are extremely strong in that the adversary can mount an attack with minimum knowledge of cryptanalysis and target ciphers. In this context, we must consider deep learning-based cryptanalysis when designing symmetric-key ciphers. However, previous results [1, 4, 5, 6, 8, 9, 10, 11, 14, 18, 19, 21, 33, 34] have not clarified which features or internal structures affect success probabilities; thus, it is difficult to employ the 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 attacks on SPN 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, and we explore the relationship between the internal structure and success probability of the attack using a white box analysis technique.

1.1.1 New Deep Learning-based Output Prediction Attacks.

We focus on toy SPN block ciphers (16-bit block variants of PRESENT [7] and AES) 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 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, 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, we can mount prediction attacks on 1 round, while differential distinguishing attacks reach 2 rounds. Note that our attacks realize output predictions (i.e., plaintext recovery and ciphertext prediction) that are much stronger than distinguishing attacks even without knowing the algorithm of the target ciphers.

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 replace or swap internal components and evaluate the impact of these modifications relative to the success probability of the prediction attacks. 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 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.

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

	Cipher (Block size)	Structures	Blackbox Model	Target	#Round (#Full)	Whitebox Analysis
AAAA12 [1]	SDES (12 bits)	Feistel	Yes	OP	2 (2)	No
XHY19 [33]	DES (64 bits)	Feistel	Yes	PR	2 (16)	No
BSS08 [4]	Serpent (128 bits)	SPN	No	DD	7 (32)	No
DH14 [11]	SDES (12 bits)	Feistel	Yes	KR	2 (2)	No
Gohr19 [14]	Speck32/64 (32/64 bits)	Feistel	Yes	DD	11 (22)	No
YK20 [34]	Speck 32 (32 bits)	Feistel	No	DD	9 (22)	No
YK20 [34]	Simon 32 (32 bits)	Feistel	No	DD	12 (32)	No
YK20 [34]	GIFT 64 (64 bits)	SPN	No	DD	8 (28)	No
HLZW20 [18]	DES (64 bits)	Feistel	No	LD	4 (16)	No
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)	N/A	No	DD	4 (8)	No
CY20 [8]	Speck 32/64 (32/64 bits)	Feistel	No	DD	13 (22)	No
CY20 [8]	Speck 48/72 (48/72 bits)	Feistel	No	DD	13 (22)	No
CY20 [8]	Speck 48/96 (48/96 bits)	Feistel	No	DD	13 (22)	No
CY20 [8]	DES (64 bits)	Feistel	No	DD	8 (16)	No
CY21 [10]	DES (64 bits)	Feistel	No	DD	6 (16)	No
CY21 [10]	Speck 32 (32 bits)	Feistel	No	DD	7 (22)	No
CY21 [10]	PRESENT (64 bits)	SPN	No	DD	7 (31)	No

CY21 [9]	Speck32/64 (32/64 bits)
	48/72 (48/72 bits)
	48/96 (48/96 bits)
		Feistel	No	DD	13(22)
					12(22)
					12(23)
						No

ITYY21 [21]	TWINE (64 bits)	Feistel	No	DD	8(36)	No
Jaewoo20 [30]	Simplified DES (8 bits)	Feistel	No	KR	8 (8)	No
Jaewoo20 [30]	Speck 32/64 (32/64 bits)	Feistel	No	KR	22 (22)	No
Jaewoo20 [30]	Simon 32/64 (32/64 bits)	Feistel	No	KR	32 (32)	No
HRC21 [19]	Simon 32 (32 bits)	Feistel	No	DD	13 (32)	No
BGPT21 [6]	Speck 32/64 (32/64 bits)	Feistel	No	DD	7 (22)	No
BGPT21 [6]	Simon 32/64 (32/64 bits)	Feistel	No	DD	8 (32)	No
This paper	small PRESENT-[4] (16 bits)	SPN	Yes	OP	4	Yes
This paper	small AES (16 bits)	SPN	Yes	OP	1	Yes

1.2 Comparison with Existing Work

Table 1 compares existing deep learning-based attacks and our attacks. Here blackbox attacks mean that the adversary does not have knowledge of the targeted ciphers (except algorithm interfaces), and whitebox analysis explores the relationship between the ability of deep learning-based attacks and the internal components ciphers. As shown in Table 1, the proposed attacks are the first output prediction attacks on SPN structures in a blackbox model.

Regarding whitebox analysisis, 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 [11]. 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 deep learning-based attack [14], Benamira et al. [6] improved success probability of traditional distinguishers using characteristics that are expected to

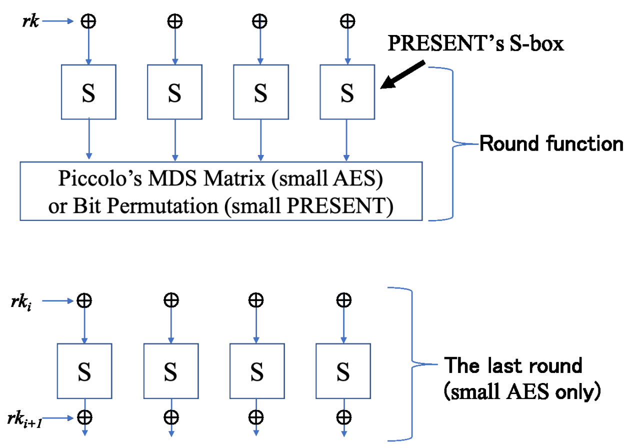

Figure 1: Round Functions (small PRESENT and small AES)

be reacted by the Gohr attack. In other words, their work confirms whether or not characteristics explored by Gohr can actually be employed in the traditional distinguishing attacks. 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. To sum up, to the best of our knowledge, our results are the first ones that perform the whitebox analysis.

1.3 Other Related Work

Alani and Hu presented plaintext recovery attacks on DES, 3-DES and AES [2, 20] 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, 20] because they could not be reproduced. Baek et al. also pointed this out in the literature [28]. 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 [13]. In addition, some results have been obtained classical ciphers such as Caesar cipher, Vigenere cipher and Enigma [12, 15, 16, 26]. Other machine learning-based analyses have also been reported, e.g., [25, 24]. Tan et al. demonstrated that deep learning can be used to distinguish ciphertexts encrypted by AES, Blowfish, DES, 3-DES, and RC5, respectively [31], 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 small PRESENT [23] and small AES.

small PRESENT-[n]: PRESENT [7] 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] [23] 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 $\mathbb{F}_2^4 \to \mathbb{F}_2^4$.

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

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.

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

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

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

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 [29], 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 ${}^t(x_{0(4)}, x_{1(4)}, x_{1(4)}, x_{1(4)}) \leftarrow M \cdot {}^t (x_{0(4)}, x_{1(4)}, x_{1(4)}, x_{1(4)})$.

3 Proposed Attacks

In this section, we present the proposed deep learning-based output prediction attacks under the blackbox model. To realize the proposed attacks, we construct deep learning models for ciphertext prediction and plaintext recovery, respectively. In the following, we first discuss the goals of these attacks and then explain the construction of the deep learning models and their evaluation.

3.1 Goals of Attack

To date, the relationship between the abilities of attackers in previous works and deep learning-based ones has not been clarified; thus, 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 existing attacks and deep learning-based attacks. Specifically, we compare the success probabilities of deep learning-based attacks with those of existing attacks.

2. The order of components or component replacement 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 models.

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

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

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

The blackbox model 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 model also assumes that the adversary only knows the input-output format and possesses deep learning knowledge.

Regarding attack models, a ciphertext-only attack model, which allows the adversary to obtain only the ciphertext, is the weakest model. However, information-theoretically no information is provided to the adversary in the model except for several special cases, e.g., the broadcast setting of RC4 [27]. In fact, the attack in this model is practically impossible. The known-plaintext attack model is the next weakest model. Here, the adversary can obtain some information from the given plaintext/ciphertext pairs and use it for the attack. The other attack models, e.g., chosen-plaintext attack models, require the adversary to possess some knowledge about the ciphertext, and the adversary in this model is stronger than the adversary in the known-plaintext attack model. Thus we employ the known-plaintext attack model.

In these models, we set 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 $2^{n/2}$ plaintext/ciphertext pairs regarding a single 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: Here, the adversary obtains $2^{n/2}$ 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.

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 [22], RMSprop [32]

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) [17]. 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. As explained in the following subsection, 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 $2^{n/2}$ plaintext/ciphertext pairs under the known-plaintext attack. In our experiments, we randomly select $2^{n/2}$ 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 $2^{n/2}$ 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 $2^{-n/2}$, 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.

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

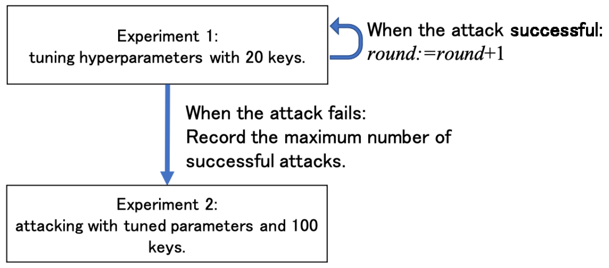
Experiment 1:
tuning hyperparameters with 20 keys.
When the attack **successful**:
round:=*round*+1
When the attack fails:
Record the maximum number of
successful attacks.
Experiment 2:
attacking with tuned parameters and 100
keys.

Figure 2: Experimental flow

4 Comparison with Linear/Differential Cryptanalysis

In this section, we compare the number of rounds attacked by the proposed attack to that of existing classically attacks (linear/differential cryptanalysis) using a whitebox analysis technique.

4.1 Experiments

For comparison, we apply the proposed attack to two toy ciphers, i.e., small PRESENT-[4] and small AES. Here, we attempt ciphertext prediction and plaintext recovery against these toy ciphers using the proposed attack with common experimental hyperparameter values (Table 4). In our experiments, we implement the proposed attack using Keras[1], which is a deep learning library, and we employ TensorFlow as the backend. This experiment involves two sub-experiments (Fig. 2).

In the first sub-experiment (experiment 1), we optimize the hyperparameters for the target ciphers in each round using the proposed attack and procedures described in Section 3.2. Here, for the hyperparameter optimization, we employ Optuna[2], which is an automatic optimization tool. The search algorithm for the hyperparameter optimization is used the default one. We adopt the success probabilities of ciphertext prediction or plaintext recovery as the measure for hyperparameter optimization. In hyperparameter optimization, we obtain 100 hyperparameter candidates using Optuna from the plaintext/ciphertext pairs generated by 20 secret keys, where the next hyperparameter candidate is searched using the information of the previous hyperparameter candidate. Here, the parameter corresponding to the highest average success probabilities of ciphertext prediction/plaintext recovery with 20 secret keys is selected as the next hyperparameter candidate. Each ciphertext prediction/plaintext recovery success probability is obtained from 2^{15} randomly generated plaintext/ciphertext pairs and a deep learning model, where the deep learning model

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

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

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	$2^{-15.00}$

is constructed using the other 2^{15} plaintext/ciphertext pairs. After obtaining 100 candidates, we use the highest average success probabilities of ciphertext prediction/plaintext recovery among all 100 candidates as the optimized hyperparameter. If the average success probabilities of ciphertext prediction/plaintext recovery with the optimized hyperparameter is greater than 2^{-15}, then the round number for searching the optimized hyperparameter is incremented by one; otherwise, the second sub-experiment is executed using all optimized hyperparameters.

In the second sub-experiment (experiment 2), we obtain the average success probabilities of ciphertext prediction/plaintext recovery using the optimized hyperparameters and plaintext/ciphertext pairs generated using 100 secret keys. Here the secret keys differ from those used in the hyperparameter optimization process. We then compare the proposed attack to the linear/differential cryptanalysis using the experimental results and linear/differential probability of the target ciphers.

4.2 Experimental Results and Comparison

Table 5 shows the experimental results of experiment 2 and the hyperparameter obtained in experiment 1.

First, we compare the proposed attack and the linear/differential cryptanalysis for small PRESENT-[4]. From the experimental result, the proposed attack succeeds 5 rounds of ciphertext prediction and 4 rounds of plaintext recovery against small PRESENT-[4]. Although the success probability of ciphertext prediction for 5-round small PRESENT-[4] is nearly 2^{-15}, the success probability of plaintext recovery for 4-round small PRESENT-[4] is sufficiently greater than 2^{-15}.

Here, we consider that the number of rounds attacked by the proposed attack is at least 4. On the other hand, from the precisely calculated differential probability of small PRESENT-[4]

Table 6: Maximum differential probability of small PRESENT-[4] and small AES

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}

(Table 6), the largest number of rounds attacked by the differential cryptanalysis is 4. Similarly, from the precisely calculated linear probability, the largest number of rounds attacked by the linear cryptanalysis is 4. Therefore, the largest number of rounds attacked by the proposed attack and that of the linear/differential cryptanalysis are equivalent on small PRESENT-[4].

Next, we compare the proposed attack and the linear/differential cryptanalysis for small AES. From Table 5, we evaluate the largest number of rounds attacked by the proposed attack is 1. From the precisely calculated linear/differential probabilities, the largest number of rounds attacked by the differential cryptanalysis is 2, and that of linear cryptanalysis is 3. Therefore, the largest number of rounds attacked by the proposed attack is less than that of linear/differential cryptanalysis for small AES. However, the proposed attack realizes output ciphertext prediction and plaintext recovery that are much stronger than the distinguishing attacks of linear/differential cryptanalysis.

5 Additional Whitebox Analysis

As shown in Table 5, the average success probability of ciphertext prediction by the proposed attack in 4-round small PRESENT-[4] is approximately 2^9 times greater than that of plaintext recovery. In the success probability of linear/differential cryptanalysis on small PRESENT-[4], the security of the encryption and decryption are equivalent. Therefore, the result of the proposed attack in 4-round small PRESENT-[4] seems contrary to intuition. In this section, we replace or swap internal components in order to reveal the relationship between internal components and the success probability of deep learning-based attacks.

5.1 Experiments

Here, we discuss two kinds of experiments, i.e., we investigate the average success probabilities of ciphertext prediction/plaintext recovery by the proposed attack 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) are swapped in the encryption/decryption algorithm. The target toy ciphers are 4-round small PRESENT-[4] and 2-round small AES.

Table 7: Average of success probabilities when swapping the order of components or replacing components on 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	$2^{-13.74}$

5.2 Experimental Results

The results for 4-round small PRESENT-[4] are shown in Table 7. In ciphertext prediction, the average of success probability when the sLayer is replaced with sLayer-inv is greater than that of the original small PRESENT-[4]. However, the average success probability when the order of the sLayer and pLayer is replaced is less than that of the original small PRESENT-[4]. Thus, the differences in success probabilities are relatively large, and we think that swapping the component order of replacing components affects the success probabilities of the proposed attack. It is particularly worth noting that swapping and replacing components does not affect the success probabilities of linear/differential cryptanalysis. Therefore, we expect that our results can be an important stepping stone for designing deep learning-resistant symmetric-key ciphers.

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

6 Conclusion

In this paper, we have presented deep learning attacks on SPN block ciphers in a blackbox setting, where the adversary does not know the algorithm of the target ciphers, with the exception of the interface such as key size and block size. In addition, we have investigated the relationship between the internal structure and the success probability of the attack using a whitebox analysis technique. We expect that the obtained results will be a foundation for designing deep learning-resistant symmetric-key ciphers.

As future works, it is interested in whether the proposed attacks also work on Feistel block ciphers, and it is also desirable to clarify why differences are observed in the success probabilities regarding the underlying optimizer. Moreover, how to feedback our results for designing deep learning-resistant symmetric-key ciphers is also regarded important future work.

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] Adrien Benamira, David Gerault, Thomas Peyrin, and Quan Quan Tan. A deeper look at machine learning-based cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2021:287, 2021.

[7] 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.

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

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

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

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

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

[13] 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.

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

[15] 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.

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

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

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

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

[20] 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.

[21] 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.

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

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

[24] 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.

[25] 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.

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

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

[28] Kwangjo Kim Seunggeun Baek. Recent advances of neural attacks against block ciphers. *2020 Symposium on Cryptography and Information Security (SCIS)*, 2020. available at https://caislab.kaist.ac.kr/publication/paper_files/2020/scis2020_SG.pdf.

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

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

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

[32] 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.

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

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