Hindawi
Security and Communication Networks
Volume 2020, Article ID 3701067, 11 pages
https://doi.org/10.1155/2020/3701067

Research Article

Deep Learning-Based Cryptanalysis of Lightweight Block Ciphers

Jaewoo So

Department of Electronic Engineering, Sogang University, Seoul 04107, Republic of Korea

Correspondence should be addressed to Jaewoo So; jwso@sogang.ac.kr

Received 5 February 2020; Revised 21 June 2020; Accepted 26 June 2020; Published 13 July 2020

Academic Editor: Umar M. Khokhar

Most of the traditional cryptanalytic technologies often require a great amount of time, known plaintexts, and memory. This paper proposes a generic cryptanalysis model based on deep learning (DL), where the model tries to find the key of block ciphers from known plaintext-ciphertext pairs. We show the feasibility of the DL-based cryptanalysis by attacking on lightweight block ciphers such as simplified DES, Simon, and Speck. The results show that the DL-based cryptanalysis can successfully recover the key bits when the keyspace is restricted to 64 ASCII characters. The traditional cryptanalysis is generally performed without the keyspace restriction, but only reduced-round variants of Simon and Speck are successfully attacked. Although a text-based key is applied, the proposed DL-based cryptanalysis can successfully break the full rounds of Simon32/64 and Speck32/64. The results indicate that the DL technology can be a useful tool for the cryptanalysis of block ciphers when the keyspace is restricted.

1. Introduction

Cryptanalysis of block ciphers has persistently received great attention. In particular, recently, many cryptanalytic techniques have emerged. The cryptanalysis based on the algorithm of algebraic structures can be categorized as follows: a differential cryptanalysis, a linear cryptanalysis, a differential-linear cryptanalysis, a meet-in-the-middle (MITM) attack, and a related-key attack [1, 2]. Differential cryptanalysis, which is the first general cryptanalytic technique, analyses how differences evolve during encryption and how differences of plaintext pairs evolve to differences of the resultant ciphertext pairs [3]. The differential cryptanalysis has evolved to various types of differential cryptanalysis such as an integral cryptanalysis, which is sometimes known as a multiset attack, a boomerang attack, an impossible differential cryptanalysis, and an improbable differential cryptanalysis [1, 2]. Linear cryptanalysis is also a general cryptanalytic technique, where it analyses linear approximations between plaintexts bits, ciphertexts bits, and key bits. It is a known plaintext attack. The work in [4] showed that the efficiency of the linear cryptanalysis can be improved by use of chosen plaintexts. The authors in [5] proposed a zero-correlation linear cryptanalysis, which is a key recovery technique. The MITM attack, which employs a space-time tradeoff, is a generic attack which weakens the security benefits of using multiple encryptions [6]. The biclique attack, which is a variant of the MITM attack, utilizes a biclique structure to extend the number of possibly attacked rounds by the MITM attack [6]. In a related-key attack, an attacker can observe the operation of a cipher under several different keys whose values are initially unknown, but where some mathematical relationship connecting the keys is known to the attacker [7].

However, the conventional cryptanalysis might be impractical or have limitations to be generalized. First, most of conventional cryptanalytic technologies often require a great amount of time, known plaintexts, and memory. Second, although the traditional cryptanalysis is generally performed without the keyspace restriction, only reduced-round variants are successfully attacked on recent block ciphers. For example, no successful attack on the full-round Simon or the full-round Speck, which is a family of lightweight block ciphers, is known [8–10]. Third, we need an automated and generalized test tool for checking the safety of various lightweight block ciphers for Internet of Things [11]. There are various automated techniques that can be used to build distinguishers against block ciphers [12–14]. Because resistance against differential cryptanalysis is an important design criterion for modern block ciphers, most designs rely on finding some upper bound on probability of differential characteristics [12]. The authors in [13] proposed a truncated searching algorithm which identifies differential characteristics as well as high probability differential paths. The authors in [14] applied a mixed integer linear programming (MILP) to search for differential characteristics and linear approximations in ARX ciphers. However, most automated techniques have endeavoured to search for differential characteristics and linear approximations. Hence, the machine learning- (ML-) based cryptanalysis can be a candidate to solve the above problems.

This paper proposes a generic deep learning- (DL-) based cryptanalysis model that finds the key from known plaintext-ciphertext pairs and shows the feasibility of the DL-based cryptanalysis by applying it to lightweight block ciphers. Specifically, we try to utilize deep neural networks (DNNs) to find the key from known plaintexts. The contribution of this paper is two-fold: first, we develop a generic and automated cryptanalysis model based on the DL. The proposed DL-based cryptanalysis is a promising step towards a more efficient and automated test for checking the safety of emerging lightweight block ciphers. Second, we perform the DL-based attacks on lightweight block ciphers, such as S-DES, Simon, and Speck. In our knowledge, this is the first attempt to successfully break the full rounds of Simon32/64 and Speck32/64 although we apply the text-based key for the block ciphers.

The remainder of this paper is organized as follows: Section 2 presents the related work; Section 3 describes the attack model for cryptanalysis; Section 4 introduces the DL-based approach for the cryptanalysis of lightweight block ciphers and presents the structure of the DNN model; Section 5 describes how to learn and evaluate the model; in Section 6, we apply the DL-based cryptanalysis to lightweight block ciphers and evaluate the performance of the DL-based cryptanalysis; finally, Section 7 concludes this paper.

Notations: we give some notations, which will be used in the rest of this paper. A plaintext and ciphertext are, respectively, denoted by $p = (p_0, p_1, \ldots, p_{n-1})$ and $c = (c_0, c_1, \ldots, c_{n-1})$, where n is the block size, p_i is the ith bit of the plaintext, c_i is the ith bit of the ciphertext, and $\mathbf{p}_i, \mathbf{c}_i \in \{0, 1\}$. A key is denoted by $k = (k_0, k_1, \ldots, k_{m-1})$, where m is the key length and k_i is the ith bit of the key, $k_i \in \{0, 1\}$. Let $k|_i^j$ denote the key bits from the ith bit to the jth bit of the key, that is, $k|_i^j \triangleq (k_i, k_{i+1}, \ldots, k_j)$. A block cipher is specified by an encryption function, $E(p, k)$, that is, $c = E(p, k)$.

2. Related Work

ML has been successfully applied in a wide range of areas with significant performance improvement, including computer vision, natural language processing, speech, and game [15]. The development of ML technologies provides a new development direction for cryptanalysis [16]. The idea of the relationship between the fields of cryptography and ML is introduced in [17] at 1991. After that, many researchers have endeavoured to apply the ML technologies for the cryptanalysis of block ciphers.

The studies on the ML-based cryptanalysis can be classified as follows: first, some studies focused on finding the characteristics of block ciphers by using ML technologies. The authors in [18] used a recurrent neural network to find the differential characteristics of block ciphers, where the recurrent neural network represents the substitution functions of a block cipher. The author in [19] applied an artificial neural network to automate attacks on the classical ciphers of a Caesar cipher, a Vigenère cipher, and a substitution cipher, by exploiting known statistical weakness. They trained a neural network to recover the key by providing the relative frequencies of ciphertext letters. Recent work [20] experimentally showed that a CipherGAN, which is a tool based on a generative adversarial network (GAN), can crack language data enciphered using shift and Vigenère ciphers.

Second, some studies used ML technologies to classify encrypted traffic or to identify the cryptographic algorithm from ciphertexts. In [21], an ML-based traffic classification was introduced to identify SSH and Skype encrypted traffic. The authors in [22] constructed three ML-based classification protocols to classify encrypted data. They showed the three protocols, hyperplane decision, Naïve Bayes, and decision trees, efficiently perform a classification when running on real medication data sets. The authors in [23] used a support vector machine (SVM) technique to identify five block cryptographic algorithms, AES, Blowfish, 3DES, RC5, and DES, from ciphertexts. The authors in [24] proposed an unsupervised learning cost function for a sequence classifier without labelled data, and they showed how it can be applied in order to break the Caesar cipher.

Third, other researchers have endeavoured to find out the mapping relationship between plaintexts, ciphertexts, and the key, but there are few scientific publications. The work in [25] reported astonishing results for attacking the DES and the Triple DES, where a neural network was used to find the plaintexts from the ciphertexts. The authors in [26] used a neural network to find out the mapping relationship between plaintexts, ciphertexts, and the key in simplified DES (S-DES). The author in [27] developed a feedforward neural network that discovers the plaintext from the ciphertext without the key in the AES cipher. The authors in [28] attacked on the round-reduced Speck32/64 by using deep residual neural networks, where they trained the neural networks to distinguish the output of Speck with a given input difference based on the chosen plaintext attack. The attack in [28] is similar to the classical differential cryptanalysis. However, the previous work failed to attack the full rounds of lightweight block ciphers, and moreover, they failed to develop a generic deep learning- (DL-) based cryptanalysis model.

3. System Model

We consider (n, m) lightweight block ciphers such as S-DES, Simon, and Speck, where n is the block size and m is the key length. Our objective is to find the key, $\mathbf{k}$, in which the attacker has access to M pairs, $[\mathbf{p}^{(i)}, \mathbf{c}^{(j)}]$, of known plaintexts, and their resultant ciphertexts encrypted with the same key, that is, $\mathbf{c}^{(j)} = E(\mathbf{p}^{(j)}, \mathbf{k})$, $j = 1, 2, \ldots, M$. Hence, the cryptanalytic model is a known plaintext attack model. Because the algorithms of block ciphers have been publicly released, we assume that the algorithms of block ciphers are known.

4. Deep Learning-Based Approach

4.1. DNN Learning Framework. The modern term "DL" is considered as a better principle of learning multiple levels of composition, which uses multiple layers to progressively extract higher level features from the raw input [29]. In the DL area, a DNN is considered as one of the most popular generative models. As a multilayer processor, the DNN is capable of dealing with many nonconvex and nonlinear problems. The feedforward neural network forms a chain, and thus, the feedforward neural network can be expressed as

$$f(\mathbf{x}; \boldsymbol{\theta}) = f^{(L+1)}\left(f^{(L)}\left(\cdots f^{(1)}(\mathbf{x})\right)\right), \tag{1}$$

where $\mathbf{x}$ is the input, the parameter $\boldsymbol{\theta}$ consists of the weights $\mathbf{W}$ and the biases $\mathbf{b}$, $f^{(l)}$ is called the lth layer of the network, and L is the number of hidden layers. Each layer of the network consists of multiple neurons, each of which has an output that is a nonlinear function of a weighted sum of neurons of its preceding layer. The output of the jth neuron at the lth layer can be expressed as

$$j^{(l)} = f^{(l)}\left(\sum_i w_{ij}^{(l)} u_i^{(l-1)} + b_j^{(l)}\right), \tag{2}$$

where $w_{ij}^{(l)}$ is the weight corresponding to the output of the ith neuron at the preceding layer and $b_j^{(l)}$ is the bias. We apply a DNN to find the key of lightweight block ciphers. The multilayer perception mechanism and special training policy promote the DNN to be a commendable tool to find affine approximations to the action of a cipher algorithm. We train the DNN by using N_r pairs of $(\mathbf{p}, \mathbf{c})$ randomly generated with different keys in order that the system f finds affine approximations to the action of a cipher, as shown in Figure 1. In Figure 1, the loss function can be the mean square error (MSE) between the encryption key, $\mathbf{k}$, and the output of the DNN, $\widehat{\mathbf{k}}$. The performance of the trained DNN is evaluated by using N_t pairs randomly generated with different keys. Finally, given M known plaintexts, we find the key by using the trained DNN and the majority decision.

4.2. DNN Structure for the Cryptanalysis. The structure of a DNN model for the cryptanalysis is shown in Figure 2. We consider a ReLU function, $f_{\text{ReLU}}(x) = \max(0, x)$, as the nonlinear function. The DNN has η_l neurons at the lth hidden layer, where $l = 1, \ldots, L$. Each neuron at the input layer associates each bit of the plaintext and ciphertext; that is, the ith neuron represents $\mathbf{p}_i$, and the $(j+n-1)$th neuron represents $\mathbf{c}_j$, where $i, j = 0, 1, \ldots, n-1$. The number of neurons at the input layer is $2n$. Each neuron at the output layer associates each bit of the key; that is, the output of the ith neuron corresponds to $\mathbf{k}_i$, where $i = 0, 1, \ldots, m-1$. Hence, the number of neurons at the output layer is m. The output of the DNN, $\widehat{\mathbf{k}}$, is a cascade of nonlinear transformation of the input data, $[\mathbf{p}, \mathbf{c}]$, mathematically expressed as

$$\widehat{\mathbf{k}} = f([\mathbf{p}, \mathbf{c}]; \boldsymbol{\theta}) = f^{(L+1)}\left(f^{(L)}\left(\cdots f^{(1)}([\mathbf{p}, \mathbf{c}])\right)\right), \tag{3}$$

where L is the number of hidden layers and $\boldsymbol{\theta}$ is the weights of the DNN.

5. Model Training and Testing

5.1. Data Generation. The ML algorithm learns from data. Hence, we need to generate data set for training and testing the DNN. Because the algorithms of modern block ciphers are publicly released, we can generate N plaintext-ciphertext pairs with different keys, where $N = N_r + N_s$, N_r is used for training the DNN, and N_s is used for testing the DNN. Let the jth sample represent $[\mathbf{p}^{(j)}, \mathbf{c}^{(j)}; \mathbf{k}^{(j)}]$, $j = 1, 2, \ldots, N$, as shown in Figure 3, where $\mathbf{c}^{(j)} = E(\mathbf{p}^{(j)}, \mathbf{k}^{(j)})$ for $i \neq j$, $\mathbf{p}^{(i)} \neq \mathbf{p}^{(j)}$, and $\mathbf{k}^{(i)} \neq \mathbf{k}^{(j)}$.

5.2. Training Phase. The goal of our model is to minimize the difference between the output of the DNN and the key. Let $\mathbf{X}$ represent the training plaintext-ciphertext pairs $[\mathbf{p}^{(j)}, \mathbf{c}^{(j)}]$, and let $\mathbf{K}$ represent the training keys $\mathbf{k}^{(j)}$ corresponding to the jth pair $[\mathbf{p}^{(j)}, \mathbf{c}^{(j)}]$, where $1 \leq j \leq N_r$.

The DNN learns the value of the parameter $\boldsymbol{\theta}$ that minimizes the loss function, from the training samples, as follows:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(f(\mathbf{X}; \boldsymbol{\theta}), \mathbf{K}), \tag{4}$$

where because the samples are i.i.d., the MSE loss function can be expressed as follows:

$$\text{MSE} = \frac{1}{N_r \cdot m} \sum_{j=1}^{N_r} \sum_{i=0}^{m-1} \left(\mathbf{k}_i^{(j)} - \widehat{\mathbf{k}}_i^{(j)}\right)^2, \tag{5}$$

where N_r is noted as the number of training samples, $\mathbf{k}_i^{(j)}$ is the ith bit of the key corresponding to the jth sample, and $\widehat{\mathbf{k}}_i^{(j)}$ is the ith output of the DNN corresponding to the jth sample.

5.3. Test Phase. After training, the performance of the DNN is evaluated in terms of the bit accuracy probability (BAP) of each key bit. Here, the BAP of the ith key bit is the number of the DNN finding the correct ith key bit, divided by the total number of test samples.

Because the output of the DNN is a real number, $\widehat{\mathbf{k}}_i \in \mathbb{R}$, we quantize the output of the DNN into $\{0, 1\}$. The quantized output of the DNN can then be expressed as

$$\widetilde{\mathbf{k}}_i = \begin{cases} 0, & \text{if } \widehat{\mathbf{k}}_i < 0.5, \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

Then, the BAP of the ith key bit is given as

$$\rho_i = \frac{1}{N_s} \sum_{j=1}^{N_s} \text{XNOR}\left(\mathbf{k}_i^{(j)}, \widetilde{\mathbf{k}}_i^{(j)}\right), \tag{7}$$

where N_s is the number of test samples. XNOR(a, b) has one if two input values, a and b, are identical, and otherwise, it

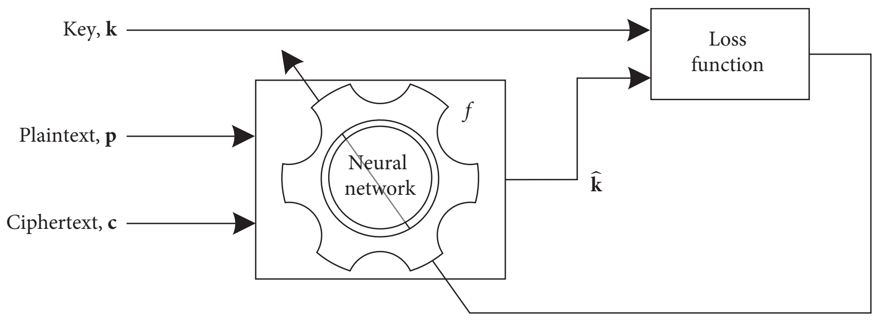


Figure 1: A schematic diagram of the DL-based cryptanalysis.

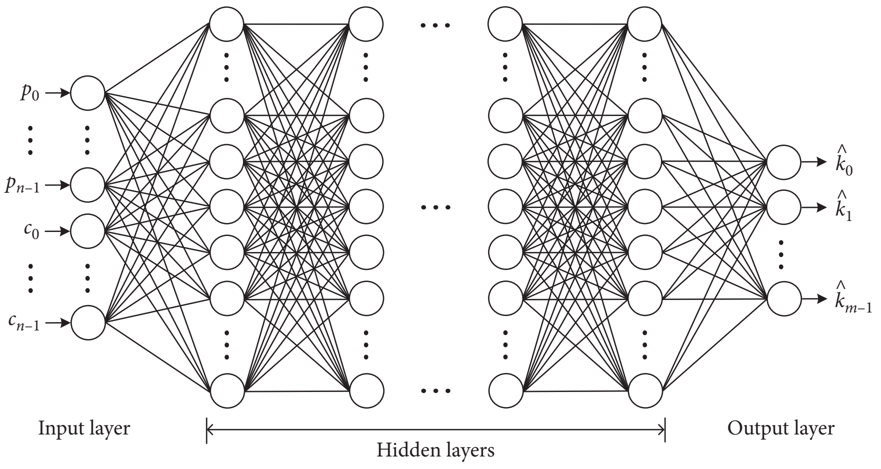


Figure 2: A DNN model.

has zero. $\mathbf{k}_i^{(j)}$ is the ith key bit corresponding to the jth test sample, and $\tilde{\mathbf{k}}_i^{(j)}$ is the quantized output of the DNN with the input of the jth test sample.

5.4. Majority Decision When M Plaintexts Are Known. Assume that we have M plaintext-ciphertext pairs encrypted with the same key. If we have a probability of finding the ith key bit, ρ_i, then the attack success probability of finding the ith key bit, which is the probability of a correct majority decision, is given as

$$\alpha_i(M) = \Pr\left(X \ge \frac{M}{2}+1\right) = 1 - \Pr\left(X \le \frac{M}{2}\right)$$
$$= 1 - \sum_{j=0}^{M/2} \binom{M}{j} \rho_i^j (1-\rho_i)^{M-j}. \tag{8}$$

By using the de Moivre–Laplace theorem, as M grows large, the normal distribution can be used as an approximation to the binomial distribution, as follows:

$$\alpha_i(M) = 1 - \Phi\left(\frac{(M/2) - M\rho_i}{\sqrt{M\rho_i(1-\rho_i)}}\right), \tag{9}$$

where $\Phi(z) = \int_{-\infty}^{z} 1/\sqrt{2\pi} e^{-x^2/2} dz$. Hence, in order to find the ith key bit with a success probability greater than or equal to τ, the number of required known plaintexts is

$$M_i^* = \min\left\{M \mid \alpha_i(M) \ge \tau\right\}. \tag{10}$$

6. Performance Evaluation

6.1. Data Set and Performance Metric. For the data set, we generate the plaintext as any combination of a random binary digit, that is, $\mathbf{p}_i \in \mathrm{rand}\{0,1\}$. However, for the encryption key, we consider two methods. The first method is a "*random key*," where the key has any combination of a random binary digit, that is, $\mathbf{k}_i \in \mathrm{rand}\{0,1\}$, $i = 0, 1, \ldots, m-1$. Hence, the probability that the ith key bit is one is 0.5 for all i. The other method is a "*text key*," where the key has any combination of characters. For the simplicity, as shown in Figure 4, the character is one out of 64 ASCII characters, which consists of lowercase and uppercase alphabet characters, 10 digits, and two special characters: $\mathcal{T} = \{a, b, \ldots, z, A, B, \ldots, Z, 0, 1, \ldots, 9, ?, @\}$ and $|\mathcal{T}| = 64$. Hence, in the text key generation, each eight bits belongs to the set of $\mathcal{T}$, that is, $\mathbf{k}|_{8\cdot i}^{8\cdot i+7} \in \mathrm{rand}(\mathcal{T})$, where $i = 0, 1, \ldots, [m/8] - 1$. For example, for a 64-bit key, the key consists of 8 characters. In the text key, the probability that the ith key bit is one depending on the order in each character. Let the occurrence probability denote $\mu_i = \max(\Pr(\mathbf{k}_i = 1), \Pr(\mathbf{k}_i = 0))$, where $\Pr(\mathbf{k}_i = x)$ is the probability that the ith key bit is x. Figure 5 shows the occurrence probability of the ith key bit μ_i. For example, the

$$\left[\begin{array}{cccc cccc cccc}
p_0^{(1)} & p_1^{(1)} & \cdots & p_{n-1}^{(1)}, & c_0^{(1)} & c_1^{(1)} & \cdots & c_{n-1}^{(1)}; & k_0^{(1)} & k_1^{(1)} & \cdots & k_{m-1}^{(1)} \\
p_0^{(2)} & p_1^{(2)} & \cdots & p_{n-1}^{(2)}, & c_0^{(2)} & c_1^{(2)} & \cdots & c_{n-1}^{(2)}; & k_0^{(2)} & k_1^{(2)} & \cdots & k_{m-1}^{(2)} \\
 & & \vdots & & & & \vdots & & & & \vdots & \\
p_0^{(j)} & p_1^{(j)} & \cdots & p_{n-1}^{(j)}, & c_0^{(j)} & c_1^{(j)} & \cdots & c_{n-1}^{(j)}; & k_0^{(j)} & k_1^{(j)} & \cdots & k_{m-1}^{(j)} \\
 & & \vdots & & & & \vdots & & & & \vdots & \\
p_0^{(N)} & p_1^{(N)} & \cdots & p_{n-1}^{(N)}, & c_0^{(N)} & c_1^{(N)} & \cdots & c_{n-1}^{(N)}; & k_0^{(N)} & k_1^{(N)} & \cdots & k_{m-1}^{(N)}
\end{array}\right]$$

FIGURE 3: Data set.

BIN	HEX	Char.	BIN	HEX	Char.	BIN	HEX	Char.
00000000	0	NUL	00101011	2B	+	01010110	56	V
00000001	1	SOH	00101100	2C	,	01010111	57	W
00000010	2	STX	00101101	2D	-	01011000	58	X
00000011	3	ETX	00101110	2E	.	01011001	59	Y
00000100	4	EOT	00101111	2F	/	01011010	5A	Z
00000101	5	ENQ	00110000	30	0	01011011	5B	[
00000110	6	ACK	00000110	31	1	01011100	5C	₩
00000111	7	BEL	00110010	23	2	01011101	5D	]
00001000	8	BS	00110011	33	3	01011110	5E	^
00001001	9	HT	00110100	34	4	01011111	5F	–
00001010	0A	LF	00110101	35	5	01100000	60	`
00001011	0B	VT	00110110	36	6	01100001	61	a
00001100	0C	FF	00001100	37	7	01100010	62	b
00001101	0D	CR	00111000	38	8	01100011	63	c
00001110	0E	SO	00111000	39	9	01100100	64	d
00001111	0F	SI	00111010	3A	:	01100101	65	e
00010000	10	DLE	00111011	3B	;	01100110	66	f
00010001	11	DC1	00111100	3C	<	01100111	67	g
00010010	12	DC2	00111101	3D	=	01101000	68	h
00010011	3	DC3	00111110	3E	>	01101001	69	i
00010100	14	DC4	00111111	3F	?	01101010	6A	j
00010101	15	NAK	01000000	40	@	01101011	6B	k
00010110	16	SYN	01000001	41	A	01101100	6C	l
00010111	17	ETB	01000010	42	B	01101101	6D	m
00011000	18	CAN	01000011	43	C	01101110	6E	n
00011001	19	EM	01000100	44	D	01101111	6F	o
00011010	1A	SUB	01000101	45	E	01110000	70	p
00011011	1B	ESC	01000110	46	F	01110001	71	q
00011100	1C	FS	01000111	47C	G	01110010	72	r
00011101	1D	GS	01001000	48	H	01110011	73	s
00011110	1E	RS	01001001	49	I	01110100	74	t
00011111	1F	US	01001010	4A	J	01110101	75	u
00100000	20	SPACE	01001011	4B	K	01110110	76	v
00100001	21	!	01001100	4C	L	01110111	77	w
00100010	22	″	01001101	4D	M	01111000	78	x
00100011	23	#	01001110	4E	N	00100011	79	y
00100100	24	$	01001111	4F	O	01111010	7A	z
00100101	25	%	01010000	50	P	01111011	7B	{
00100110	26	&	01010001	51	Q	01111100	7C	|
00100111	27	′	01010010	52	R	01111101	7D	}
00101000	28	(	01010011	53	S	01111110	7E	~
00101001	29	)	01010100	54	)	01111111	7F	DEL
00001010	2A	∗	01010101	55	T			

FIGURE 4: Characters used in the text key generation.

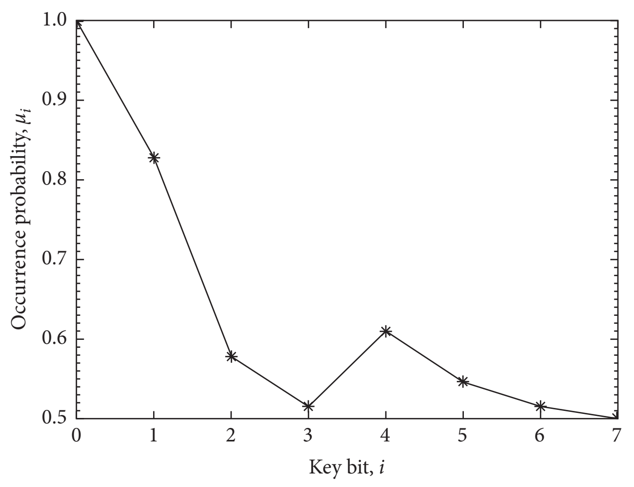


Figure 5: Occurrence probability in the text key generation.

first bit of the key character is always 0, and the second bit is one with the probability of 0.828.

Taking the occurrence probability of each key bit into consideration, the performance of finding the ith key bit can be expressed as the deviation as follows:

$$\varepsilon_i = \rho_i - \mu_i, \quad (11)$$

where ρ_i is the BAP and μ_i is the occurrence probability of the ith key bit. If M known plaintexts is given, the performance of finding the ith key bit is given by $\varepsilon_i(M) = \alpha_i(M) - \mu_i$, where $\alpha_i(M)$, which is the probability of a correct majority decision, is obtained from equation (9).

6.2. Simulation Environment. The performance of the DL-based cryptanalysis is evaluated for the lightweight block ciphers: S-DES, Simon32/64, and Speck32/64, as shown in Table 1.

In order to train the DNN with an acceptable loss rate, it is necessary to expand the network size. Hyperparameters, such as the number of hidden layers, the number of neurons per hidden layer, and the number of epochs, should be tuned in order to minimize a predefined loss function. The traditional way of performing hyperparameter optimization has been grid search or random search. Other hyperparameter optimizations are Bayesian optimization, gradient-based optimization, evolutionary optimization, and population-based training [30, 31]. Moreover, automated ML (AutoML) has been proposed to design and train neural networks automatically [30]. In our simulation, by using the data set of Simon32/64 and Speck32/64 ciphers, we simply perform an exhaustive searching to set the number of hidden layers, L, and the number of neurons per hidden layer, η_l, through a manually specified subset of the hyperparameter space, $L \in \{3, 5, 7\}$ and $\eta_l \in \{128, 256, 512\}$. Additionally, to reduce the complexity, we choose a smaller number of hidden layers if the performance difference is not greater than 10^{-5}. If the number of epochs is greater than 3000, the error becomes small, and when it reaches 5000, it is sufficiently minimized, so we set the number of epochs is fixed to 5000. Consequently, the parameters used for training the DNN models are as follows: the number of hidden layers is 5, the number of neurons at each hidden layer is 512, and the number of epochs is 5000. We use the adaptive moment (Adam) algorithm for the learning rate optimization of the DNN.

Table 1: Block ciphers used in case studies.

Item	S-DES	Simon	Speck
Block size (bits), n	8	32	32
Key size (bits), m	10	64	64
Round, R	2	32	22

The powerful "*Tensorflow*" is introduced to design and process the DNN. Also, we deploy a GPU-based server, which is equipped with Nvidia GeForce RTX 2080 Ti and its CPU is Intel Core i9-9900K. The implemented DL-based cryptanalysis tool is shown in Figure 6. The GUI was implemented by using PyQt over Python 3.7. The implemented tool provides various combinations of ML architectures, hyperparameters, and training/test samples.

6.3. Simplified DES

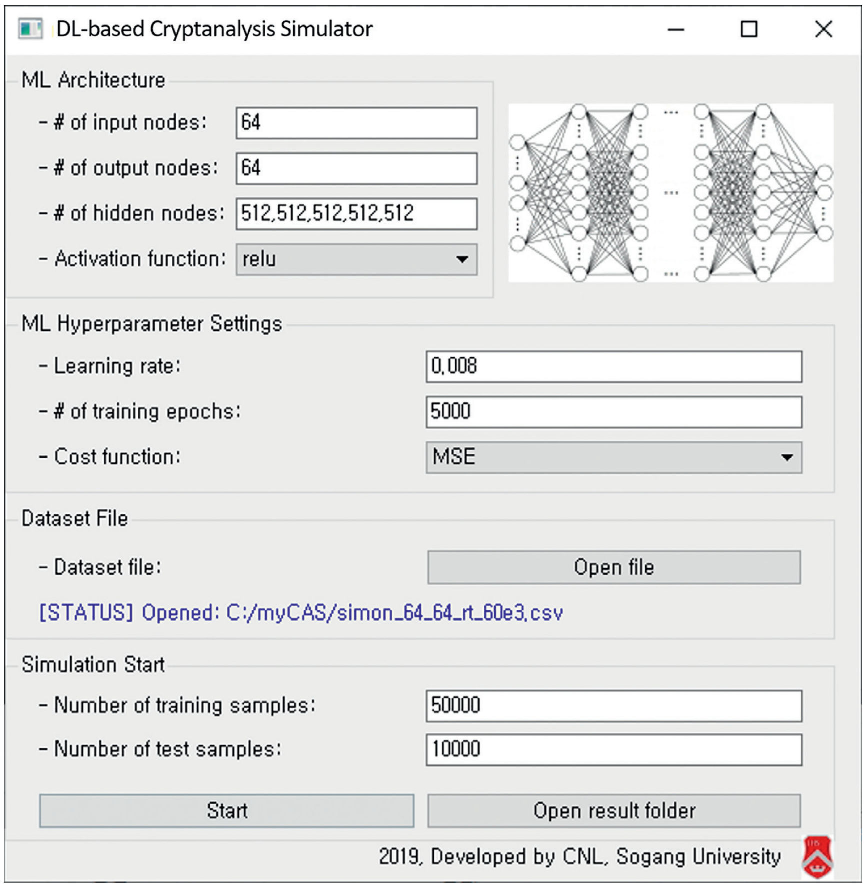
6.3.1. Overview of S-DES. S-DES, designed for education purposes at 1996, has similar properties and structure as DES but has been simplified to make it easier to perform encryption and decryption [32]. The S-DES has an 8-bit block size and a 10-bit key size. The encryption algorithm involves five functions: an initial permutation (IP); a complex function labelled f_K, which involves both permutation and substitution operations and depends on a key input; a simple permutation function that switches the two halves of the data; the function f_K again; and finally a permutation function that is the inverse of the initial permutation (IP^{-1}). S-DES may be said to have two rounds of the function f_K.

Because the length of the key is limited, the brute-force attack, which is known as an exhaustive key search, is available. Some previous work presented an approach for breaking the key using genetic algorithm and particle swarm optimization [33, 34], which is concluded that the genetic algorithm is a better approach than the brute force for analysing S-DES.

6.3.2. Test Results. For training and testing the DNN, we generate N plaintext-ciphertext pairs with different keys, as follows:

$$\mathbf{c}^{(j)} = S - DES\left(\mathbf{p}^{(j)}, \mathbf{k}^{(j)}\right), \quad j = 0, 1, \ldots, N, \quad (12)$$

where $\mathbf{k}^{(i)} \neq \mathbf{k}^{(j)}$ for $i \neq j$ and $N = N_r + N_s$. Here, N_r is the number of samples for training and N_s is the number of samples for testing. In the simulation, we use $N_r = 50000$ and $N_s = 10000$. The plaintext is any combination of a random binary digit, that is, $p_i \in \mathrm{rand}\{0, 1\}$. We generate the encryption key by using two methods: a random key and a text


DL-based Cryptanalysis Simulator

ML Architecture

- # of input nodes: 64
- # of output nodes: 64
- # of hidden nodes: 512,512,512,512,512
- Activation function: relu

ML Hyperparameter Settings

- Learning rate: 0.008
- # of training epochs: 5000
- Cost function: MSE

Dataset File

- Dataset file: Open file

[STATUS] Opened: C:/myCAS/simon_64_64_rt_60e3.csv

Simulation Start

- Number of training samples: 50000
- Number of test samples: 10000

2019, Developed by CNL, Sogang University

Figure 6: Implemented DL-based cryptanalysis simulator.

key. In the S-DES with a 10-bit key, the text key has any combination of one character and two random binary bits.

Figure 7 shows the BAP of the DNN when we apply a random key and a text key. The results show the DL-based cryptanalysis can break the S-DES cipher. When we apply a random key, the key bits, $\mathbf{k}_1$, $\mathbf{k}_5$, and $\mathbf{k}_8$, are quite vulnerable to the attack and the key bit of $\mathbf{k}_6$ is the safest. Because the minimum value of the BAP is $\rho_{\text{min}} = 0.5389$ at the 6th key bit, from equation (10), we need $M = 271$ known plaintexts to find all the key bits with a probability of 0.9 and we need $M = 891$ known plaintexts to find all the key bits with a probability of 0.99. When we apply a text key, the BAP becomes high, thanks to the bias of the occurrence probability of each key bit, μ_i, as shown in Figure 5. Because the minimum value of the BAP is $\rho_{\text{min}} = 0.6484$ at the 6th key bit, from equation (10), we need $M = 19$ known plaintexts to find all the key bits with a probability of 0.9 and we need $M = 59$ known plaintexts to find all the key bits with a probability of 0.99.

Figure 8 shows the deviation between the BAP and the occurrence probability of each key bit. Because of the bias of the occurrence probability of each key bit in the text key, we need to eliminate the bias characteristics of each key bit. The DNN shows that the key bits, which are quite vulnerable to the attack, are ($\mathbf{k}_2$, $\mathbf{k}_5$, $\mathbf{k}_8$) in the text key and ($\mathbf{k}_1$, $\mathbf{k}_5$, $\mathbf{k}_8$) in the random key. The key bit of $\mathbf{k}_6$ is the safest both in the text key and in the random key.

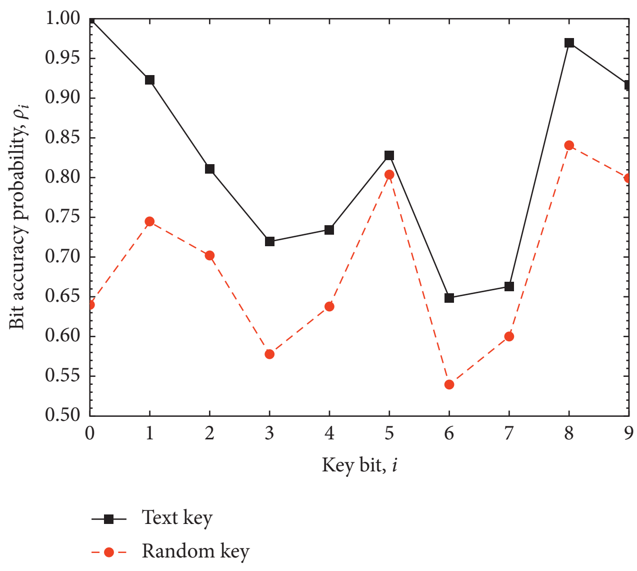


Figure 7: Bit accuracy in the S-DES with a random key.

6.4. Lightweight Block Ciphers

6.4.1. Overview of Simon and Speck. Lightweight cryptography is a rapidly evolving and active area, which is driven by the need to provide security or cryptographic measures to resource-constrained devices such as mobile phones, smart

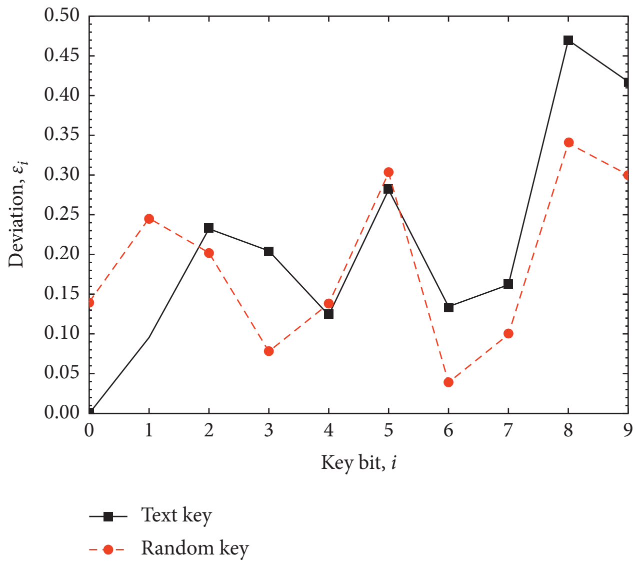


FIGURE 8: Deviation in the S-DES.

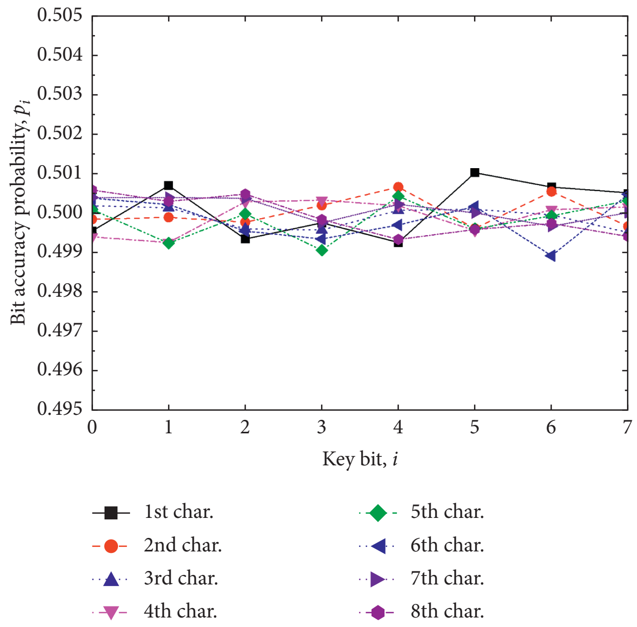


FIGURE 9: Bit accuracy probability of the Simon32/64 with a random key.

cards, RFID tags, and sensor networks. Simon and Speck is a family of lightweight block ciphers publicly released in 2013 [35, 36]. Simon has been optimized for performance in hardware implementations, while Speck has been optimized for software implementations. The Simon block cipher is a balanced Feistel cipher with a u-bit word, and therefore, the block length is $n = 2u$. The key length, m, is a multiple of u by 2, 3, or 4. Simon supports various combinations of block sizes, key sizes, and number of rounds [35]. In this paper, we consider a Simon32/64 which refers to the cipher operating on a 32-bit plaintext block that uses a 64-bit key. The Speck is an add-rotate-xor (ARX) cipher. The block of the Speck is always two words, but the words may be 16, 24, 32, 48, or 64 bits in size. The corresponding key is 2, 3, or 4 words. Speck also supports various combinations of block sizes, key sizes, and number of rounds [35].

As of 2018, no successful attack on full-round Simon or full-round Speck of any variant is known. The authors in [37] showed differential attacks of up to slightly more than half of the number of rounds for Simon and Speck families of block ciphers. The authors in [38] showed an integral attack on 24-round Simon32/64 with time complexity of 2^{63} and the data complexity of 2^{32}. The work in [39] showed an improved differential attack on 14-round Speck32/64 with time complexity of 2^{63} and the data complexity of 2^{31}.

6.4.2. Data Generation.

For training and testing the DNN, we generate N plaintext-ciphertext pairs with different keys, as follows:

$$\begin{aligned}\mathbf{c}^{(j)} &= \text{Simon}\frac{32}{64}\left(\mathbf{p}^{(j)}, \mathbf{k}^{(j)}\right),\\ \mathbf{c}^{(j)} &= \text{Speck}\frac{32}{64}\left(\mathbf{p}^{(j)}, \mathbf{k}^{(j)}\right),\end{aligned} \tag{13}$$

where $j = 0, 1, \ldots, N$ and $N = N_r + N_s$. Here, N_r is the number of samples for training and N_s is the number of samples for testing. The plaintext is any combination of a random binary digit, that is, $\mathbf{p}_i \in \text{rand}\{0, 1\}$. We generated the encryption key by using two methods: a random key and a text key. In the text key, the 64-bit key consists of 8 characters, where each character is one of 64-character set, $\mathcal{T}$. Hence, although the total keyspace is 2^{64}, the actual keyspace is reduced to 2^{48}. For training, we use $N_r = 5 \times 10^5$ samples, and for the test, we use $N_s = 10^6$ samples.

6.4.3. Test Results.

Figure 9 shows the BAP of the Simon32/64 with a random key in unit of character. The DNN shows that the BAP of each key bit varies randomly with an average of almost 0.5. Moreover, the results vary with each simulation with different hyperparameters. That is, the DNN failed to attack the Simon32/64 with a random key.

Figure 10 shows the BAP and the deviation of the Simon32/64 with a text key in unit of character. The BAP of each key bit is almost identical to the occurrence probability of the text key because the DNN learns the characteristics of the training data. However, when we eliminate the bias characteristics of the text key, the DNN shows the positive deviations, which means the DNN can break a Simon32/64 with a text key. For example, from equation (10), we need just $M = 215$ known plaintexts in order to find the key bit of $\mathbf{k}_2$ with a probability of 0.99. The minimum value of BAPs is 0.51603 at $\mathbf{k}_3$, which is greater than μ_3 by about $\varepsilon_3 = 0.00040$, except the last bits of each character. Hence, we can find the encryption key with a probability of 0.9 given $M \approx 2^{10.58}$ known plaintexts, and we can find the encryption key with a probability of 0.99 given $M \approx 2^{12.34}$ known plaintexts.

Figure 11 shows the BAP of the Speck32/64 with a random key in unit of character. The BAP of each key bit varies randomly with an average of almost 0.5, similar to the results of the Simon32/64. Moreover, the results vary with

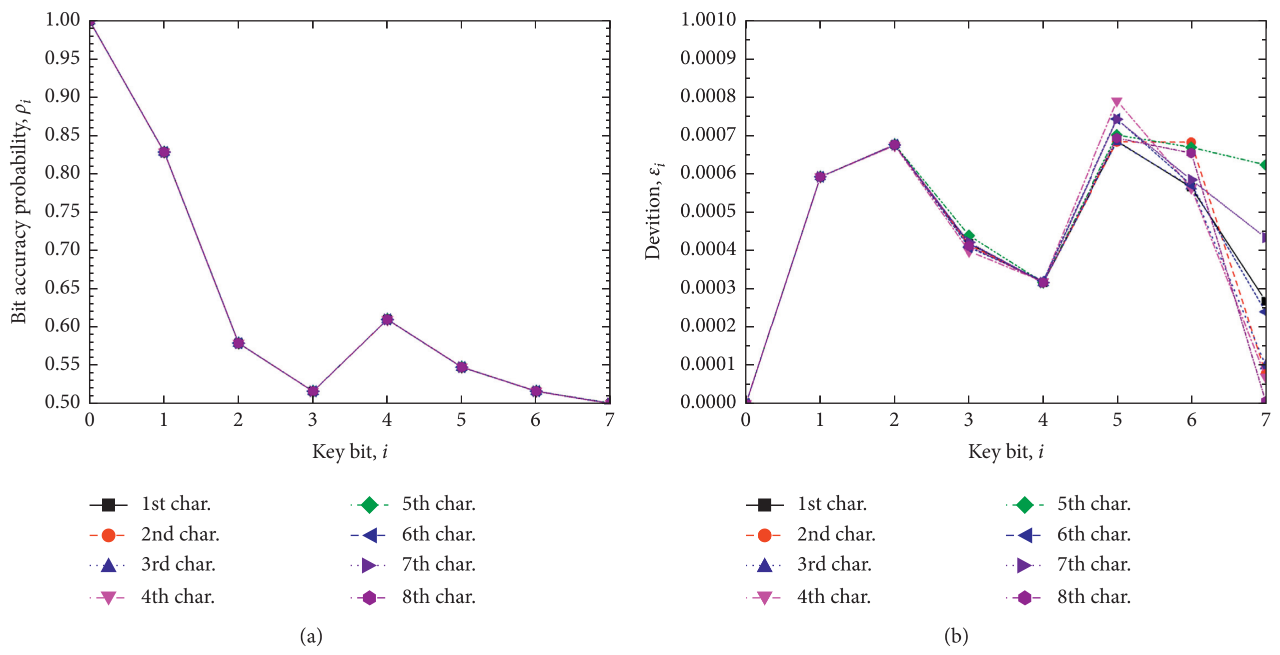


Figure 10: Bit accuracy probability and deviation of the Simon32/64 with a text key.

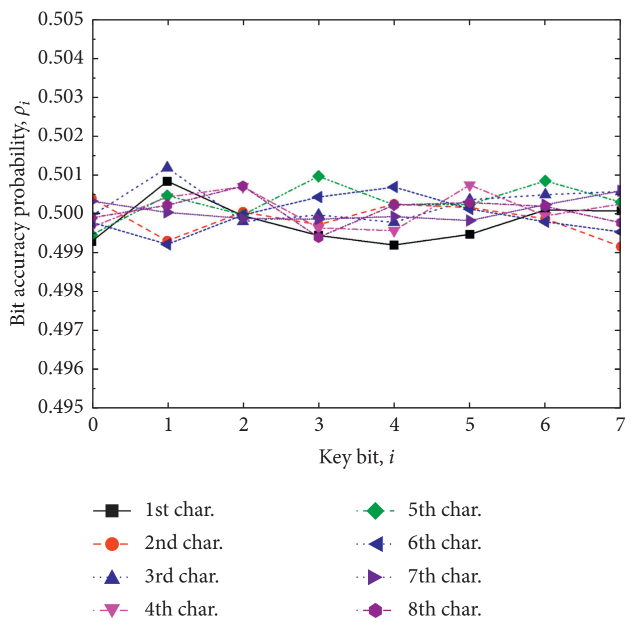


Figure 11: Bit accuracy probability of the Speck32/64 with a random key.

different hyperparameters. That is, the DL-based attacks against the Speck32/64 with a random key have been failed.

Figure 12 shows the BAP and the deviation of the Speck32/64 with a text key in unit of character. The DNN shows the positive deviations. That is, the DNN shows the possibility of breaking a Speck32/64 with a text key. The minimum value of BAPs is 0.51607 at $\mathbf{k}_3$, which is greater than μ_3 by about $\varepsilon_3 = 0.00044$, except the last bits of each character. Hence, we can find the encryption key with a probability of 0.9 given $M \approx 2^{10.57}$ known plaintexts, and we can find the encryption key with a probability of 0.99 given $M \approx 2^{12.33}$ known plaintexts.

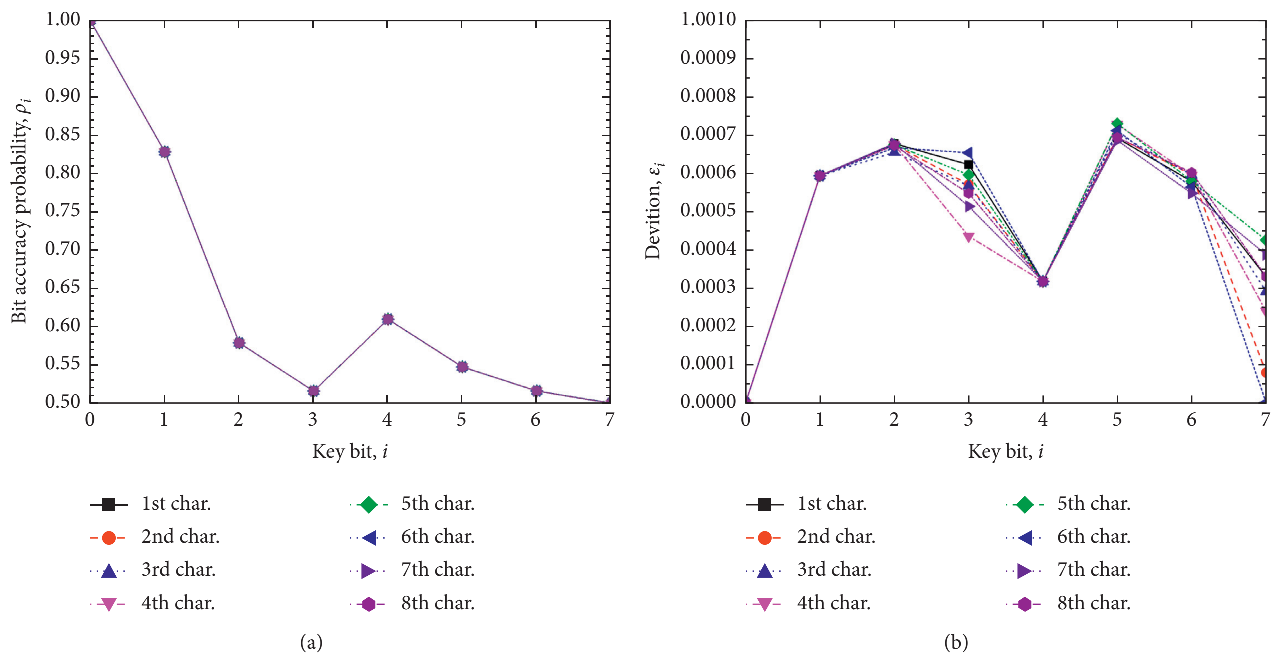


FIGURE 12: Bit accuracy probability and deviation of the Speck32/64 with a text key.

7. Conclusions

We developed a DL-based cryptanalysis model and evaluated the performance of the DL-based attack on the S-DES, Simon32/64, and Speck32/64 ciphers. The DL-based cryptanalysis may successfully find the text-based encryption key of the block ciphers. When a text key is applied, the DL-based attack broke the S-DES cipher with a success probability of 0.9 given $2^{8.08}$ known plaintexts. That is, the DL-based cryptanalysis reduces the search space nearly by a factor of 8. Moreover, when a text key is applied to the block ciphers, the DL-based cryptanalysis finds the linear approximations between the plaintext-ciphertext pairs and the key, and therefore, it successfully broke the full rounds of Simon32/64 and Speck32/64. When a text key is applied, with a success probability of 0.99, the DL-based cryptanalysis finds 56 bits of Simon32/64 with $2^{12.34}$ known plaintexts and 56 bits of Speck32/64 with $2^{12.33}$ known plaintexts, respectively. Because the developed DL-based cryptanalysis framework is generic, it can be applied to attacks on other block ciphers without change.

The drawback of our proposed DL-based cryptanalysis is that the keyspace is restricted to the text-based key. However, although uncommon, a text-based key can be used to encrypt. For example, the login password entered with the keyboard can be text based if the input data are not hashed. Modern cryptographic functions are designed to be very random looking and to be very complex, and therefore, ML can be difficult to find meaningful relationships between the inputs and the outputs if the keyspace is not restricted. Hence, our approach limited the keyspace to only text-based keys, and the proposed DL-based cryptanalysis could successfully break the 32 bit variants of Simon and Speck ciphers. If the keyspace is not limited, the DL-based cryptanalysis failed to attack the block ciphers. In the future, the accuracy of ML will be improved, and the accuracy becomes more precise, thanks to the development of algorithms and hardware. Moreover, advanced data transformation that efficiently maps cryptographic data onto ML data will help the DL-based cryptanalysis to be performed without the keyspace restriction.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (nos. 2019R1F1A1058716 and 2020R1F1A1065109).

References

[1] R. Avanzi, *A Salad of Block Ciphers-The State of the Art in Block Ciphers and Their Analysis*, IACR, Lyon, France, 2017.
[2] X. Dewu and C. Wei, “A survey on cryptanalysis of block ciphers,” in *Proceedings of the International Conference on Computer Application and System Modeling (ICCASM)*, pp. 1–6, Taiyuan, China, October 2010.
[3] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer, Berlin, Germany, 1993.
[4] L. R. Knudsen and J. E. Mathiassen, “A chosen-plaintext linear attack on DES,” in *Proceedings of the International Workshop*

on Fast Software Encryption (FSE), pp. 262–272, New York, NY, USA, April 2000.

[5] A. Bogdanov and V. Rijmen, "Zero-correlation linear cryptanalysis of block ciphers," Report 2011/123, IACR, Lyon, France, 2011.

[6] S. Zhao, X. Duan, Y. Deng, Z. Peng, and J. Zhu, "Improved meet-in-the middle attacks on generic Feistel constructions," *IEEE Access*, vol. 7, pp. 34416–34424, 2019.

[7] C. Guo, "Understanding the related-key security of Feistel ciphers from a provable perspective," *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 5260–5280, 2019.

[8] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, *The SIMON and SPECK Lightweight Block Ciphers*, pp. 1–23, IACR, Lyon, France, 2018.

[9] K. Fu, L. Sun, and M. Wang, "New integral attacks on SIMON," *IET Information Security*, vol. 11, no. 5, pp. 277–286, 2017.

[10] A. D. Dwivedt, P. Morawiecki, and G. Spivastava, "Differential cryptanalysis of round-reduced SPECK suitable for Internet of Things devices," *IEEE Access*, vol. 7, pp. 16476–16486, 2019.

[11] X. Guo, J. Hua, Y. Zhang, and D. Wang, "A complexity-reduced block encryption algorithm suitable for Internet of Things," *IEEE Access*, vol. 7, pp. 54760–54769, 2019.

[12] R. Ankele and S. Kölbl, "Mind the gap-a closer look at the security of block ciphers against differential Cryptanalysis," in *Proceedings of the Selected Areas in Cryptography (SAC)*, pp. 163–190, Alberta, Canada, August 2018.

[13] J. Chen, J. Teh, Z. Liu, C. Su, A. Samsudin, and Y. Xiang, "Towards accurate statistical analysis of security margins: new searching strategies for differential attacks," *IEEE Transactions on Computers*, vol. 66, no. 10, pp. 1763–1777, 2017.

[14] K. Fu, M. Wang, Y. Guo, S. Sun, and L. Hu, "MILP-based automatic search algorithms for differential and linear trails for Speck," Report 2016/407, pp. 1–20, IACR, Lyon, France, 2016.

[15] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[16] J. Blackledge, S. Bezobrazov, and P. Tobin, "Cryptography using artificial intelligence," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, Killarney, Ireland, July 2015.

[17] R. L. Rivest, "Cryptography and machine learning," in *Proceedings of the Advances in Cryptology (ASIACRYPT)*, pp. 427–439, Fujiyoshida, Japan, November 1991.

[18] A. G. Bafghi, R. Safabakhsh, and B. Sadeghiyan, "Finding the differential characteristics of block ciphers with neural networks," *Information Sciences*, vol. 178, no. 15, pp. 3118–3132, 2008.

[19] R. Focardi and F. L. Luccio, "Neural cryptanalysis of classical ciphers," in *Proceedings of the Italian Conference on Theoretical Computer Science (ICTCS)*, pp. 104–115, Urbino, Italy, September 2018.

[20] A. N. Gomez, S. Huang, I. Zhang, B. M. Li, M. Osama, and L. Kaiser, "Unsupervised cipher cracking using discrete GANs," in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–6, Vancouver, Canada, January 2018.

[21] R. Alshammari and A. N. Zincir-Heywood, "Machine learning based encrypted traffic classification: identifying SSH and Skype," in *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–8, Ottawa, Canada, July 2009.

[22] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pp. 1–34, San Diego, CA, USA, February 2015.

[23] C. Tan and Q. Ji, "An approach to identifying cryptographic algorithm from ciphertext," in *Proceedingsof the IEEE International Conference on Communication Software and Networks (ICCSN)*, pp. 19–23, Beijing, China, Junuary 2016.

[24] Y. Liu, J. Chen, and L. Deng, "Unsupervised sequence classification using sequential output statistics," in *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pp. 1–10, Long Beach, CA, USA, December 2017.

[25] M. M. Alani, "Neuro-cryptanalysis of DES and triple-DES," in *Proceedingsof the International Conference on Neural Information Processing (ICONIP)*, pp. 637–646, Doha, Qatar, November 2012.

[26] M. Danziger and M. A. A. Henriques, "Improved cryptanalysis combining differential and artificial neural network schemes," in *Proceedings of the International Telecommunications Symposium (ITS)*, pp. 1–5, Vienna, Austria, August 2014.

[27] X. Hu and Y. Zhao, "Research on plaintext restoration of AES based on neural network," *Security and Communication Networks*, vol. 2018, Article ID 6868506, 9 pages, 2018.

[28] A. Gohr, "Improving attacks on round-reduced speck32/64 using deep learning," *Advances in Cryptology-CRYPTO 2019*, Springer, Berlin, Germany, pp. 150–179, 2019.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, USA, 2016.

[30] K. Pešková and R. Neruda, "Hyperparameters search methods for machine learning linear workflows," in *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1205–1210, Boca Raton, Florida, USA, December 2019.

[31] T. Yu and H. Zhu, "Hyper-parameter optimization: a review of algorithms and applications," pp. 1–56, 2020, http://arxiv.org/abs/2003.05689.

[32] E. F. Schaefer, "A simplified data encryption standard algorithm," *Cryptologia*, vol. 20, no. 1, pp. 77–84, 1996.

[33] R. Vimalathithan and M. L. Valarmathi, "Cryptanalysis of simplified-DES using computational intelligence," *WSEAS Transactions on Computers*, vol. 10, pp. 77–84, 2011.

[34] L. Sharma, B. K. Pathak, and N. Sharma, "Breaking of simplified data encryption standard using binary particle swarm optimization," *International Journal of Computer Science Issues*, vol. 9, no. 3, pp. 307–313, 2012.

[35] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," Report 2013/404, IACR, Lyon, France, 2013.

[36] S. Dehnavi, "Further observations on SIMON and SPECK block cipher families," *Cryptography*, vol. 3, no. 1, pp. 1–12, 2018.

[37] F. Abed, E. List, S. Lucks, and J. Wenzel, "Differential cryptanalysis of round-reduced Simon and speck," in *Proceedings of the International Conference on Fast Software Encryption (FSE)*, pp. 525–545, London, UK, March 2014.

[38] Z. Chu, H. Chen, X. Wang, X. Dong, and L. Li, "Improved integral attacks on SIMON32 and SIMON48 with dynamic key-guessing techniques," *Security and Communication Networks*, vol. 2018, Article ID 5160237, 12 pages, 2018.

[39] I. Dinur, "Improved differential cryptanalysis of round-reduced Speck," in *Proceedings of the International Workshop on Selected Areas in Cryptography*, pp. 147–164, Montreal, Canada, August 2014.