mathematics

MDPI

Article

Comprehensive Neural Cryptanalysis on Block Ciphers Using Different Encryption Methods

Ongee Jeong [1], Ezat Ahmadzadeh [1] and Inkyu Moon [1,2,*]

[1] Department of Robotics and Mechatronics Engineering, Daegu Gyeongbuk Institute of Science & Technology (DGIST), Daegu 42988, Republic of Korea; onji6@dgist.ac.kr (O.J.); ezat.ahmadzadeh@dgist.ac.kr (E.A.)
[2] Department of Artificial Intelligence, Daegu Gyeongbuk Institute of Science & Technology (DGIST), Daegu 42988, Republic of Korea
* Correspondence: inkyu.moon@dgist.ac.kr

Abstract: In this paper, we perform neural cryptanalysis on five block ciphers: Data Encryption Standard (DES), Simplified DES (SDES), Advanced Encryption Standard (AES), Simplified AES (SAES), and SPECK. The block ciphers are investigated on three different deep learning-based attacks, Encryption Emulation (EE), Plaintext Recovery (PR), Key Recovery (KR), and Ciphertext Classification (CC) attacks. The attacks attempt to break the block ciphers in various cases, such as different types of plaintexts (i.e., block-sized bit arrays and texts), different numbers of round functions and quantity of training data, different text encryption methods (i.e., Word-based Text Encryption (WTE) and Sentence-based Text Encryption (STE)), and different deep learning model architectures. As a result, the block ciphers can be vulnerable to EE and PR attacks using a large amount of training data, and STE can improve the strength of the block ciphers, unlike WTE, which shows almost the same classification accuracy as the plaintexts, especially in a CC attack. Moreover, especially in the KR attack, the Recurrent Neural Network (RNN)-based deep learning model shows higher average Bit Accuracy Probability than the fully connected-based deep learning model. Furthermore, the RNN-based deep learning model is more suitable than the transformer-based deep learning model in the CC attack. Besides, when the keys are the same as the plaintexts, the KR attack can perfectly break the block ciphers, even if the plaintexts are randomly generated. Additionally, we identify that DES and SPECK32/64 applying two round functions are more vulnerable than those applying the single round function by performing the KR attack with randomly generated keys and randomly generated single plaintext.

Keywords: artificial intelligence; cryptanalysis; block cipher; data encryption standard (DES); advanced encryption standard (AES); SPECK; deep learning

MSC: 68P25; 68T07

Citation: Jeong, O.; Ahmadzadeh, E.; Moon, I. Comprehensive Neural Cryptanalysis on Block Ciphers Using Different Encryption Methods. *Mathematics* **2024**, *12*, 1936. https://doi.org/10.3390/math12131936

Academic Editor: Antanas Cenys

Received: 29 May 2024
Revised: 19 June 2024
Accepted: 20 June 2024
Published: 22 June 2024

1. Introduction

Due to frequent data breaches, security and privacy concerns have increased. To protect personal information in data, various cryptographic algorithms [1,2] have been utilized in data encryption and also in authentication protocols [3]. Since encrypted data differs from the original data, the sensitive information in data can be concealed and cannot be exposed to anyone who is not authorized. Thus, to ensure data confidentiality, cryptographic algorithms must be secure against attacks. Cryptanalysis can evaluate the strength of the cryptographic algorithms through legitimate attacks. The weaknesses found from the cryptanalysis can help prevent the attacks and construct new cryptographic algorithms that are more resistant to attacks. There are four types of attacks in cryptanalysis, the Ciphertext Only Attack (COA), Known Plaintext Attack (KPA), Chosen Plaintext Attack (CPA), and Chosen Ciphertext Attack (CCA), which are distinguished according to information that

attackers can access. The ciphertext is used in COA, the pair of plaintext and ciphertext is used in KPA, the ciphertext of the plaintext chosen by the attacker is used in CPA, and the plaintext of the ciphertext chosen by the attacker is used in CCA. Furthermore, the brute-force attack [4] tries all possible keys to decrypt ciphertext until the plaintext is correctly recovered. Linear cryptanalysis [5,6] finds a linear equation of relation between plaintext, ciphertext, and key. Differential cryptanalysis [7] analyzes the effect of plaintext changes on ciphertext by comparing ciphertexts of slightly different plaintexts.

Recently, deep learning has been actively applied in information security [8]. It can automatically detect intrusions and malware to protect computing resources, programs, and data from attacks in cyberspace. Moreover, deep learning-based cryptanalysis can enhance the efficiency and effectiveness in finding the weaknesses of cryptographic algorithms. Neural-aided cryptanalysis [9], one of the types of deep learning-based cryptanalysis, uses deep learning models in traditional cryptanalysis, and neural cryptanalysis investigates the vulnerability of cryptographic algorithms and random number generators [10] by using only deep learning models [11]. Unlike the conventional cryptanalysis methods that require mathematical calculations and knowledge of cryptographic algorithms, neural cryptanalysis can automatically identify the vulnerability of cryptographic algorithms by recovering the keys from the pair of the plaintext and the ciphertext, recovering the plaintext from the ciphertext, generating the ciphertext from the plaintext, and analyzing the ciphertext without decryption.

Cyberattacks have been increasing and have caused massive damage, not only for personnel but also for hospitals, banks, and the government. Also, since they are getting sophisticated and complicated by using artificial intelligence, they have become more challenging to detect and defend. Thus, to protect personal information in data from cyberattacks, cryptographic algorithms must be analyzed by using deep learning models to investigate whether they can resist artificial intelligence-based cyberattacks.

Therefore, we perform neural cryptanalysis and comprehensively analyze five block ciphers, such as Data Encryption Standard (DES) [12], Simplified DES (SDES) [2], Advanced Encryption Standard (AES) [13], Simplified AES (SAES) [14], and SPECK [15]. The block ciphers are investigated on deep learning-based Encryption Emulation (EE), Plaintext Recovery (PR), Key Recovery (KR), and Ciphertext Classification (CC) attacks by using randomly generated block-sized bit arrays and texts as plaintexts. The block ciphers apply different numbers of round functions in the encryption of the block-sized bit arrays, and they are investigated by using the deep learning models trained with different numbers of data in EE, PR, and KR attacks. Moreover, the block ciphers use two different text encryption methods, Word-based Text Encryption (WTE) and Sentence-based Text Encryption (STE), to encrypt the texts in various operation modes, and they are analyzed with deep learning models in EE, PR, and CC attacks. To the best of our knowledge, this is the first study that performs the neural cryptanalysis on block ciphers using both block-sized bit arrays and texts as plaintexts and also the EE, PR, KR, and CC attacks in one paper. Furthermore, this paper compares the results in various cases and also discovers a method that can improve the strength of the block ciphers in terms of the encryption process. Consequently, it can help improve the strength of the block ciphers and design more secure block ciphers according to the weaknesses found from the neural cryptanalysis.

The main contributions of this paper are summarized as follows:

- We perform comprehensive neural cryptanalysis to analyze the vulnerability of five block ciphers, Data Encryption Standard (DES), Simplified DES (SDES), Advanced Encryption Standard (AES), Simplified AES (SAES), and SPECK, on Encryption Emulation (EE), Plaintext Recovery (PR), Key Recovery (KR), and Ciphertext Classification (CC) attacks;
- For the block ciphers on randomly generated block-sized bit arrays, different numbers of round functions are applied in the encryption of the block-sized bit arrays, and the deep learning models trained with different numbers of data are used for EE, PR, and KR attacks;

- For the block ciphers on texts, two different text encryption methods, Word-based Text Encryption (WTE) and Sentence-based Text Encryption (STE), are used in various operation modes to encrypt the texts, and the deep learning models are utilized for EE, PR, and CC attacks;
- Experimental results show that the block ciphers can be more vulnerable to deep learning-based EE and PR attacks using more data in model training, and STE can improve the strength of block ciphers compared to WTE, which has almost the same classification accuracy as the plaintexts, especially in CC attacks;
- In KR attacks, the secret keys can be perfectly recovered when the randomly generated keys are the same as the plaintexts;
- For neural cryptanalysis, the Recurrent Neural Network (RNN)-based deep learning model is more suitable than the fully connected-based and transformer-based deep learning models, especially in KR and CC attacks.

The rest of the paper is organized as follows: Section 2 introduces the previous works on deep learning-based cryptanalysis. Section 3 describes the proposed method, such as the block ciphers analyzed in this paper, the methods used for text encryption, and the deep learning model architectures used in each attack. Section 4 provides the results in various cases for each type of plaintext. Section 5 discusses why the results are shown in each case and interprets the results by applying them in real situations. Finally, Section 6 concludes the paper by summarizing the proposed method and the results.

2. Related Works

Deep learning [16,17] has been applied in various fields, such as medical, financial, and other industries. Neural cryptanalysis is one of the important deep learning applications that evaluates the strength of cryptographic algorithms against deep learning-based attacks. Unlike neuro-aide cryptanalysis, which utilizes deep learning to improve the effectiveness of traditional cryptanalysis, neural cryptanalysis breaks the cryptographic algorithms only with deep learning [11].

2.1. Deep Learning-Based Cryptanalysis on Image Data

Computer vision [18] analyzes visual data with deep learning models and uses the analytic information in various tasks, such as in self-driving cars and for medical diagnosis. It can also be utilized in cryptanalysis [19–21] to investigate the weaknesses of cryptographic algorithms used for image encryption. In [19], they identified the vulnerability of optical-based cryptographic algorithms, Double Random Phase Encoding (DRPE) [22] and Triple Random Phase Encoding (TRPE) [23], by recovering the original image from the encrypted image with a deep learning model based on ResNet [24]. The model was trained to output the original image corresponding to the encrypted image fed into the model. Moreover, they evaluated the proposed model by removing some pixel values in the encrypted image and adding noise to the encrypted image. In [20], they used both simple [25] and complex real-world [26] images and classified the encrypted images of DRPE with a DRPE pre-trained deep learning model. In [21], they decrypted the encrypted images of a chaos-based cryptographic algorithm [27] by using a Convolutional Neural Network (CNN) [28] with an encoder–decoder structure.

Although Privacy-preserving Deep Learning (PPDL) schemes [29,30] with image encryption were proposed to protect personal information in image data, they can also be evidence of the weaknesses of the cryptographic algorithms as encrypted images can be analyzed. In [31,32], they classified images encrypted with Homomorphic Encryption (HE) [33,34], which has the same result in calculation regardless of whether the encryption is performed before or after the operations. Considering the properties of HE, which only supports addition and multiplication, activation functions in deep learning models were approximated to low-degree polynomials. In [35], they encrypted images with AES [13], Twofish [36], Serpent [37], and ChaosNet [38] in different operation modes, Electronic Codebook (ECB), Cipher Block Chaining (CBC), and Enhanced Cipher Block Chaining

(ECBC). The encrypted images were decrypted in the last single round and then classified with a CNN-based deep learning model.

2.2. Deep Learning-Based Cryptanalysis on Text Data

Natural Language Processing (NLP) [39], which mainly manipulates human language, has been applied in many tasks, such as in language translation and Chatbots. It can also be utilized in cryptanalysis [11,40–50] to investigate the vulnerability of cryptographic algorithms used for text encryption. In [41], they recovered keys with a deep learning model from the ciphertexts encrypted with classical cryptographic algorithms, Caesar, Vigenère, and substitution cipher. The model was trained by using the relative frequencies of characters in the ciphertexts. Similarly, in [42], they classified the ciphertexts of Caesar, Vigenère, and substitution cipher with a Recurrent Neural Network (RNN)-based deep learning model [51]. The model predicted the class of the ciphertexts fed into the model. In [43], they encrypted plaintexts with Data Encryption Standard (DES), and the ciphertexts were recovered into the plaintexts by using a deep learning model with a Multi-Layer Perceptron (MLP) structure. Plaintexts of 64-bit block size and the keys were generated by using a Pseudo Random Number Generator (PRNG) and used to train the model. In [44], they used round-reduced DES in text encryption and predicted plaintexts from the ciphertexts by using an MLP-based deep learning model. In [45], they recovered plaintexts with a deep learning model, which consisted of only Fully Connected layers (FC), from the ciphertexts encrypted with Advanced Encryption Standard (AES) using 128-bit and 256-bit key sizes. In [46], they used randomly generated block-sized plaintexts and encrypted the plaintexts with lightweight block ciphers, Simplified DES (SDES) [2], SIMON, and SPECK. The keys from the ciphertexts were recovered by using a deep learning model with only FC layers. In [47], they encrypted plaintexts with SDES [2], Simplified AES (SAES), and round-reduced SPECK. The model with skip connection [24] was used for key prediction. In [48], they analyzed SM4 by recovering keys with MLP-based deep learning models, and the plaintexts, ciphertexts, and the keys used in the paper were generated by using a random number generator. In [49], they used an RNN-based deep learning model [51] to generate ciphertexts and recover plaintexts in small-PRESENT. In [50], they encrypted texts as plaintexts with AES-128 and classified them by using an RNN-based deep learning model [51]. However, most previous studies analyzed cryptographic algorithms only on a single attack and used images or block-sized bit arrays as plaintexts. In contrast, this paper analyzes block ciphers from more viewpoints than the previous works by performing more than two types of attacks in various cases. Furthermore, the factors that can affect the strength of the block ciphers are identified. Specifically, the impact of the number of round functions, the quantity of training data, the text encryption method in various operation modes, and the architecture of deep learning models for the attack are investigated. Also, ideas that can improve the security of the block ciphers are suggested.

3. Material and Methods

3.1. Block Ciphers

In this section, we introduce the five block ciphers, Data Encryption Standard (DES), Simplified DES (SDES), Advanced Encryption Standard (AES), Simplified AES (SAES), and SPECK32/64, evaluated in this paper.

3.1.1. Data Encryption Standard (DES)

In the 1970s, Data Encryption Standard (DES) [12], first developed by IBM, was adopted as an encryption standard by the National Institute of Standards and Technology (NIST), known as the National Bureau of Standards (NBS). DES is a block cipher of a Feistel Network (FN) and uses a symmetric key in encryption. It generates 64-bit ciphertext from 64-bit plaintext by using a 64-bit key, including 8-bit parity bits. Initial Permutation (IP) is

first applied to the plaintext P before round functions and then divided into right and left half blocks, R_0 and L_0, as follows:

$$IP(P) = L_0R_0 \tag{1}$$

where $IP(\cdot)$ represents Initial Permutation (IP). In each round function f, expansion P-Box expands the 32-bit right half block R_{r-1} into a 48-bit block, and then an XOR operation is applied to it with a 48-bit key k_r of the round function. After that, S-Box transforms the 48-bit block into a 32-bit block and straight P-Box permutes it. Finally, the right output block R_r of the round function is generated via an XOR operation of the straight P-Box output and the left half block L_{r-1}. The right half block R_{r-1} is the left output block L_r as follows:

$$R_r = f(R_{r-1},\ k_r) \oplus L_{r-1} \tag{2}$$

$$L_r = R_{r-1} \tag{3}$$

where R, L, and k represent the right block, left block, and the key in the round function of the corresponding subscript. After repeating this process 15 more times, Final Permutation (FP), the inverse process of the IP, is applied and generates a 64-bit ciphertext C as follows:

$$C = FP(R_{16}L_{16}) = IP^{-1}(R_{16}L_{16}) \tag{4}$$

where $FP(\cdot)$ represents Final Permutation (FP). The keys for each round function are generated by shifting and compressing the block after removing 8-bit parity bits.

The simplified version of DES, Simplified DES (SDES) [2], has 16-bit plaintext, 16-bit ciphertext, and a 16-bit key in 4 round functions. Straight P-Box, including IP and FP, is not used in SDES.

3.1.2. Advanced Encryption Standard (AES)

As the key space of DES is not large enough to be secure against a brute-force attack, Advanced Encryption Standard (AES) [13], developed by Joan Daemen and Vincent Rijmen, was selected as the new encryption standard in 2001. AES is a block cipher of a Substitution Permutation Network (SPN) and uses a symmetric key in encryption. It encrypts 128-bit plaintext into 128-bit ciphertext. According to the key size, different numbers of round functions are applied, which are 10, 12, and 14 round functions for 128-, 192-, and 256-bit keys, respectively. There are four operations in each round function: SubBytes, ShiftRows, MixColumns, and AddRoundKey. Plaintext is arranged into a 4×4 byte matrix called a state. SubBytes substitutes 16 bytes in the state according to the S-BOX that has an 8-bit input and 8-bit output. ShiftRows shifts the second, third, and fourth rows in a state once, twice, and thrice to the left, respectively. MixColumns multiplies each column in a state with a constant matrix over the finite field $GF(2^8)$ as follows:

$$\begin{bmatrix} d_{1,\,j} \\ d_{2,\,j} \\ d_{3,\,j} \\ d_{4,\,j} \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \cdot \begin{bmatrix} c_{1,\,j} \\ c_{2,\,j} \\ c_{3,\,j} \\ c_{4,\,j} \end{bmatrix} \tag{5}$$

where $c_{i,j}$ and $d_{i,j}$ represent a byte at (i, j) in a state and result of the MixColumns operation, respectively. AddRoundKey is an XOR operation for each byte in the state and corresponding byte in the key of each round function. It is applied to the state of the plaintext before the round functions, and then SubBytes, ShiftRows, MixColumns, and AddRoundKey are applied in sequence for each round function. The final round function applies only three operations, except Mixcolumns, in the same order as the previous round functions. The keys for each round function are generated using key expansion.

The simplified version of AES, Simplified AES (SAES) [14], encrypts 16-bit plaintext into 16-bit ciphertext by using a 16-bit key in 2 round functions. Since the block size of SAES

is smaller than AES, plaintext is arranged into a 2×2 nibble matrix, not a 4×4 byte matrix as in AES. It also uses SubNibbles instead of SubBytes as does AES in each round function.

3.1.3. SPECK

Lightweight ciphers have been actively studied for security in constrained environments like Internet of Things (IoT) devices. SPECK [15] is one of the lightweight block ciphers, which was released by the National Security Agency (NSA) in 2013. It is based on an Addition-Rotation-XOR (ARX) operation and uses various sizes of blocks and keys. The block has two words of 16-, 24-, 32-, 48-, and 64-bit, and the key has two, three, and four words that are the same size as the block. The number of round functions also differs according to the size of the block and key and the number of the words in the key. It can be written as SPECK2n/nm, where 2 represents the number of words in the block, n represents the size of the word in the block and key, and m represents the number of words in the key. For example, SPECK32/64, used in this paper, has two 16-bit words in a block and four 16-bit words in the key, and it applies 22 round functions. Each round function r first applies right rotation to the left-half word L_{r-1} and then adds the right-half word R_{r-1}. It then generates the left output word L_r of the round function via an XOR operation with key k_r of the round function. The right output word R_r of the round function is the result of the XOR operation of the left rotated right half word and left output word L_r of the round function as follows:

$$L_r = ((L_{r-1} \gg r_1) \boxplus R_{r-1}) \oplus k_r \tag{6}$$

$$R_r = (R_{r-1} \ll r_2) \oplus L_r \tag{7}$$

where >> and << represent right and left rotations, $\boxplus$ represents addition modulo word size in the block and key, and $\oplus$ represents bitwise XOR operation, respectively. The rotated bits r_1 and r_2 are $r_1 = 7$ and $r_2 = 2$ for SPECK32/64 and $r_1 = 8$ and $r_2 = 3$ for the other cases. The keys for each round function are generated by the same process as the round function in the encryption.

3.2. Text Encryption Methods

In this section, two different text encryption methods, Word-based Text Encryption (WTE) and Sentence-based Text Encryption (STE), are described, which are used to figure out how to improve the strength of block ciphers in text encryption.

3.2.1. Word-Based Text Encryption (WTE)

Since sentences consist of words, text can be encrypted word by word. In other words, text is divided into word units, and each word is separately encrypted. The Word-based Text Encryption (WTE) consists of five steps, which are the text pre-processing, tokenization, binary encoding, padding, and encryption steps. The text pre-processing step first changes uppercase to lowercase and removes the punctuation and emojis, which are non-alphabetics, in plaintext. The clean plaintext, called the pre-processed plaintext, is then divided into words in the tokenization step, and each word is encoded into a bit array in the binary encoding step, according to the ASCII code. Each bit array is divided into blocks, which are the same size as the block size in the block cipher. If the length of the bit array is not a multiple of the block size and cannot be divided by the block size, zero-padding is applied in the padding step, and then each block is encrypted with a block cipher in the encryption step. The encrypted blocks in each word are combined, and it results in one cipher word for one word. Thus, the number of cipher words CW in ciphertext C_{WTE} encrypted with WTE is the same as the number of words PW in clean plaintext P as follows:

$$N_{CW}(C_{WTE}) = N_{PW}(P) \tag{8}$$

where $N_{CW}(C_{WTE})$ and $N_{PW}(P)$ represent the number of cipher words in the ciphertext encrypted with WTE and the number of words in the clean plaintext, respectively. The process of WTE is depicted in Figure 1.

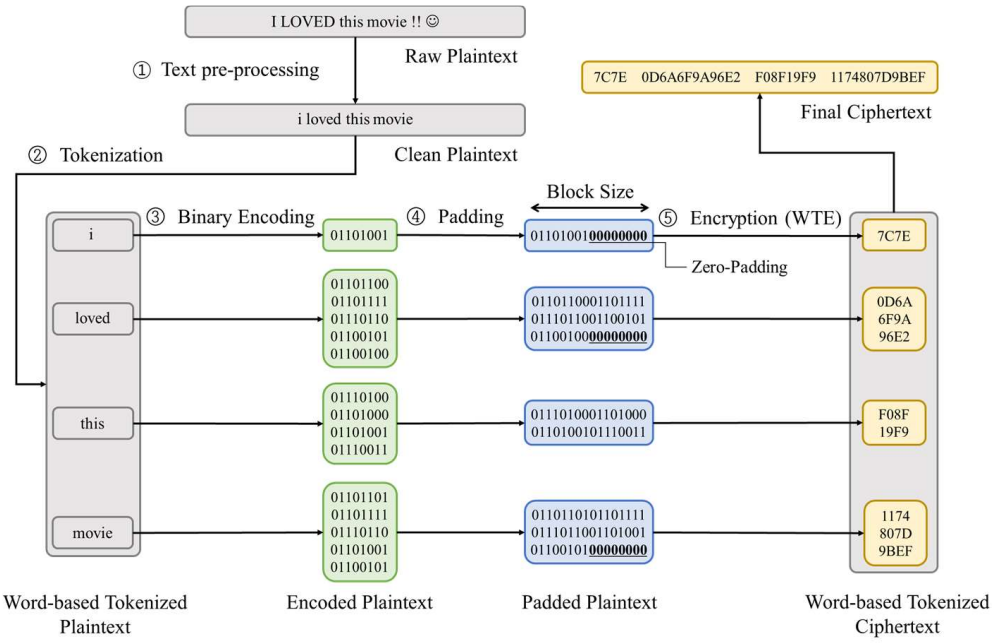

Figure 1. The Word-based Text Encryption (WTE) process. There are five steps, text pre-processing, tokenization, binary encoding, padding, and WTE encryption, in sequence.

3.2.2. Sentence-Based Text Encryption (STE)

A sentence can be encrypted as itself without dividing the sentence into words or characters. In Sentence-based Text Encryption (STE), there are four steps, the text pre-processing, binary encoding, padding, and encryption steps. After generating clean plaintext in the text pre-processing step by converting uppercase into lowercase and removing non-alphabetic characters, like emojis, every character in the clean plaintext, including whitespace, is encoded into a bit array B in the binary encoding steps, according to the ASCII code. The bit array is then divided into blocks of the specific size corresponding to the block size bs in the block cipher. To fit the bit array into the multiple of the block size, zero-padding Z is applied in the padding step. The blocks in the bit array with padding are then encrypted with the block cipher in the encryption step. Consequently, the number of cipher words CW is the same as the number of the divided blocks in the bit array with padding as follows:

$$N_{CW}(C_{STE}) = \frac{len(B+Z)}{bs} \tag{9}$$

where $N_{CW}(C_{STE})$ and $len(B + Z)$ represent the number of cipher words in the ciphertext encrypted with STE and the length of bit array with padding, respectively. The process of STE is depicted in Figure 2.

3.3. Deep Learning Model Architectures

We investigate the security of five block ciphers, SDES [2], SAES [14], DES [12], AES-128 [13], and SPECK32/64 [15], by performing deep learning-based attacks, Encryption Emulation (EE), Plaintext Recovery (PR), Key Recovery (KR), and Ciphertext Classification (CC) attacks. EE is an attack that tries to generate the ciphertexts from the plaintexts, and PR is the reverse process of EE, which tries to recover the plaintexts from the ciphertexts without knowledge of the key. KR attempts to recover the keys from the pairs of plaintext and ciphertext, and CC is an attack that attempts to classify the ciphertexts without decryption. The deep learning model architectures used for attacks on block ciphers using block-sized bit arrays and texts as plaintext, respectively, are presented in this section.

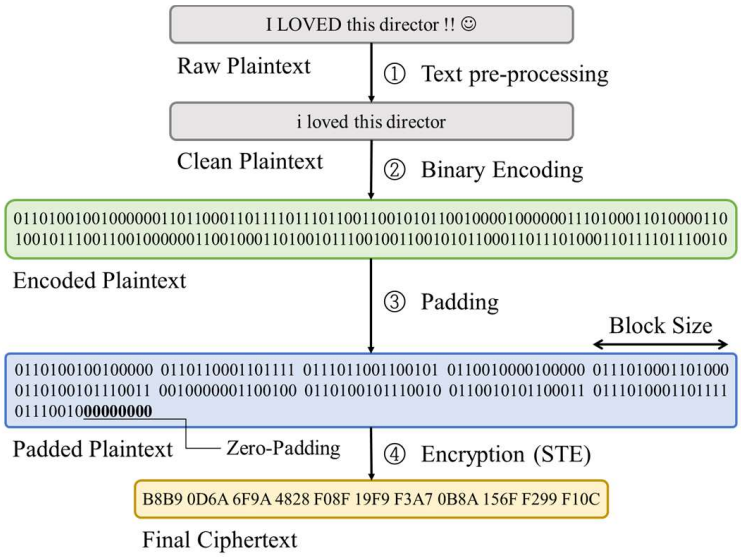

Figure 2. The Sentence-based Text Encryption (STE) process. There are four steps, text pre-processing, binary encoding, padding, and STE encryption, in sequence.

3.3.1. Deep Learning Models for Encryption Emulation (EE), Plaintext Recovery (PR), and Key Recovery (KR) Attacks on Block-Sized Bit Arrays

For Encryption Emulation (EE), Plaintext Recovery (PR), and Key Recovery (KR) attacks on block-sized bit arrays, fully connected-based and Recurrent Neural Network (RNN)-based, bidirectional Long Short-Term Memory (BiLSTM) [51], deep learning models are utilized. Although a fully connected-based deep learning model is the most straightforward neural network, it is still mainly used in neural cryptanalysis. It is because cryptographic algorithms have diffusion and confusion properties, where a single bit in the plaintext or key can affect all of the bits in the ciphertext. Since the fully connected-based deep learning model can train the global features in the data, it has been considered more suitable in neural cryptanalysis. Therefore, we first perform EE, PR, and KR attacks on block-sized bit arrays by using a fully connected-based deep learning model. As shown in Figure 3a, the model comprises four fully connected layers, which are followed by batch normalization and ReLU, but the last layer is applied a sigmoid activation function. Each layer has 512, 1024, and 512 nodes, respectively, and the last layer has the number of nodes that is the same as the block size.

Moreover, since the pairs of the plaintext and the ciphertext are fed into the model in KR attacks on block-sized bit arrays, it can be considered as two sequences. So, the RNN-based deep learning model designed for sequential data can train the relation between the plaintexts and the ciphertexts. Also, to investigate if there are any sequential features in the plaintexts that represent corresponding ciphertexts and vice versa, we additionally perform not only a KR attack but also EE and PR attacks on block-sized bit arrays by using BiLSTM. It is one of the RNN-based deep learning models, which can process the sequences in both forward and backward directions by using input, output, and forget gates that control the information passed, stored, and removed in each step. As shown in Figure 3b, the model consists of three BiLSTMs with 256 hidden state sizes and a fully connected layer followed by a sigmoid. Since DES and SPECK encrypt the plaintexts split into left and right blocks, the plaintexts inputs for the EE attack on block-sized bit arrays, and the ciphertexts inputs for the PR attack on block-sized bit arrays were divided into two parts, and each part was then fed into the model in sequence. Similarly, in the KR attack on block-sized bit arrays, the pair of plaintext and the ciphertext was fed into the model separately. Thus, the models had a sequence length of two, and each input size was half of the block size for EE and PR attacks and the same size as the block size for the KR attack.

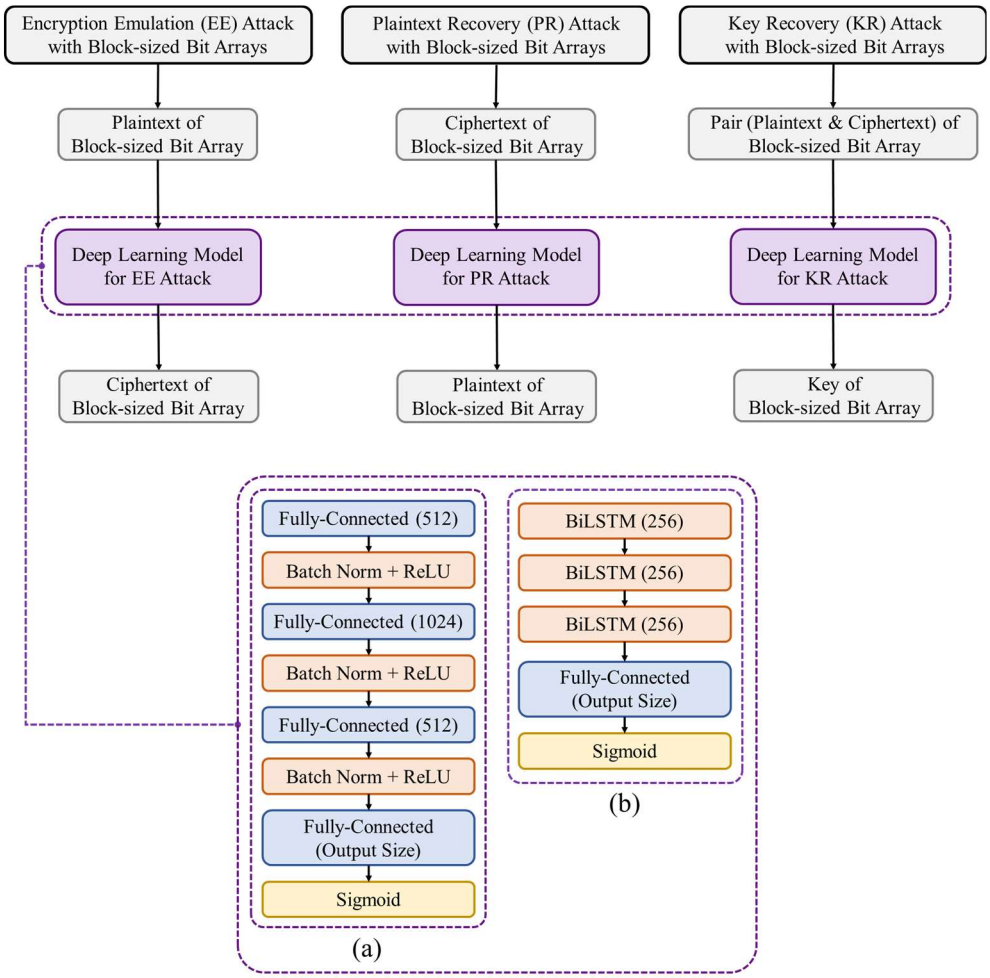

Figure 3. The deep learning models for Encryption Emulation (EE), Plaintext Recovery (PR), and Key Recovery (KR) attacks on block-sized bit arrays. (**a**) Fully connected-based deep learning model for EE, PR, KR attacks on block-sized bit arrays. (**b**) RNN-based deep learning model (BiLSTM) for EE, PR, and KR attacks on block-sized bit arrays.

3.3.2. Deep Learning Models for Encryption Emulation (EE) and Plaintext Recovery (PR) Attacks on Texts

For Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on texts, a transformer-based deep learning model, Text-to-Text Transfer Transformer (T5)-small [52], is utilized. The transformer is a state-of-the-art deep learning model in NLP that uses only attention mechanisms without RNN-based structures. It performs better in capturing the long-term dependencies than Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and even attention mechanisms in RNN-based deep learning models. Since EE and PR attacks on texts use texts as input and output, plaintexts as input, and the ciphertexts as output in EE attacks, and ciphertexts as input and the plaintexts as output in PR attacks, the transformer-based sequence-to-sequence deep learning model is chosen among the transformer-based deep learning models. Unlike the other transformer-based sequence-to-sequence deep learning models that were designed for specific NLP tasks, T5 converts every NLP task into text-to-text problems. Therefore, we perform EE and PR attacks on texts by using T5. As shown in Figure 4, the model comprises a stack of encoder blocks and decoder blocks. Each encoder has a multi-head encoder self-attention and feed-forward layer, and each decoder has a multi-head decoder self-attention, multi-head encoder–decoder attention, and feed-forward layer with ReLU. After each layer, the normalization layer, residual skip connection, and 0.1 dropout is followed, and dropout is also applied to the output of the ReLU. The inputs for the first encoder block and decoder block are applied

to the embedding process before being fed into the first encoder block and decoder block. Each output of the last decoder block is passed into the fully connected layer, which has the same number of nodes as the number of tokens and is followed by softmax. The embedding dimension is 512, the number of self-attention heads is six, the feed-forward has a 2048 output dimensionality, and six encoder blocks and six decoder blocks are stacked.

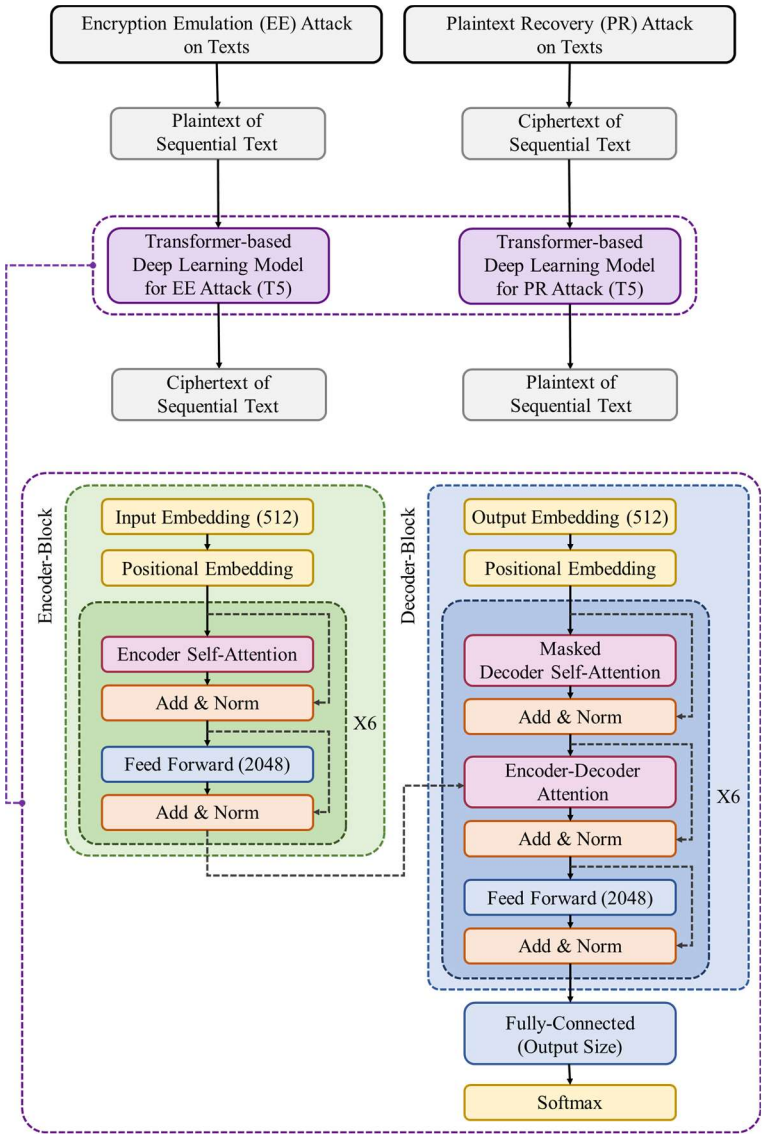

Figure 4. The deep learning models for Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on texts. Transformer-based deep learning model (T5-small) for EE and PR attacks on texts.

3.3.3. Deep Learning Models for Ciphertext Classification (CC) Attack on Texts

For Ciphertext Classification (CC) attacks on texts, RNN-based and transformer-based deep learning models, which are bidirectional GRU (BiGRU) [53] and Bidirectional Encoder Representations from Transformers (BERT)-base [54], respectively, are utilized. Since the plaintexts are sequential data, an RNN-based deep learning model designed for sequential data can be used to train the relation between the tokens in the sentence. Therefore, we perform CC attack on texts by using BiGRU, one of the RNN-based deep learning models. It can process the sequences in the backward direction as well as forward direction and consists of a GRU that uses reset and update gates to determine how much information of the context sequences is to be removed and passed. As shown in Figure 5a, the model

consists of input embedding, three BiGRUs, and a fully connected layer followed by softmax. The embedding dimension is 256, and each BiGRU has a 128 hidden state size.

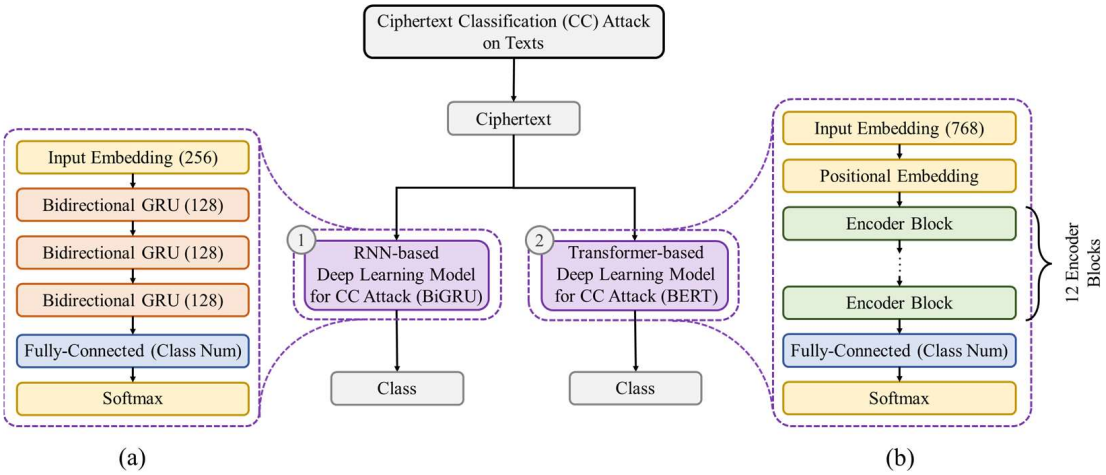

Figure 5. The deep learning models for Ciphertext Classification (CC) attack on texts. (**a**) RNN-based deep learning model (BiGRU) for CC attack on texts. (**b**) Transformer-based deep learning model (BERT-base) for CC attack on texts.

Furthermore, we additionally perform CC attack on texts by using a state-of-the-art deep learning model for text classification, BERT-base [54]. It is a transformer-based deep learning model that uses only the encoder part, a stack of encoder blocks. As shown in Figure 4, each encoder block has the multi-attention encoder self-attention and the feed-forward with GELU, and each layer is followed by a normalization layer, skip connection, and 0.1 dropout in sequence. The inputs are fed into the first encoder block after applying the input embedding and positional embedding. The fully connected layer with the same number of nodes as the number of classes comes after the stack of the encoder blocks, and softmax is applied to its output to classify the ciphertexts. As shown in Figure 5b, the embedding dimension is 768, the number of self-attention heads is 12, the feed-forward has 3072 output dimensionality, 12 encoder blocks and 12 decoder blocks are stacked, and the last fully connected layer has the number of nodes equal to the number of classes.

4. Experiments

4.1. Experimental Setup

All of the models in this paper were constructed by using Python with the PyTorch framework. A server in our laboratory with Intel Xeon Silver 4214 CPU and NVIDIA RTX A5000 GPU was used to train the models and generate the data. The models for Encryption Emulation (EE), Plaintext Recovery (PR), and Key Recovery (KR) attacks on block ciphers using block-sized bit arrays, fully connected-based, and BiLSTM, were trained for 300 epochs and 128 batch sizes by using a binary cross-entropy loss function and AdamW optimizer with a 0.001 learning rate. In EE and PR attacks on block ciphers using texts, the deep learning models of T5-small architecture were trained by using a cross-entropy loss function and Adam optimizer with a 0.005 learning rate, and they used 64 batch sizes and 100 epochs. The models used for Ciphertext Classification (CC) attacks on block ciphers using texts, BiGRU, and BERT-base, were trained for 100 epochs with an Adam optimizer and cross-entropy loss function. The BiGRU used a 0.001 learning rate and 128 batch sizes, and the BERT-base used 0.0001 and 8 for the learning rate and batch sizes, respectively.

4.2. Data Generation

4.2.1. Block-Sized Bit Arrays

For Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on block ciphers using block-sized bit arrays, the integers were first randomly generated by using the method in the 'random' module of Python named the 'getrandbits' with block size as a parameter, which can generate the integers of a specific bit length. Then, the integers were encoded into the binary arrays and then encrypted with the block ciphers.

For Key Recovery (KR) attacks on block ciphers using block-sized bit arrays, a single randomly selected block-sized bit array was used as a plaintext and it was encrypted with keys that were randomly generated by using the same method, with the 'getrandbits' method in the 'random' module in Python, for the plaintexts in EE and PR attacks. The sample plaintexts and the ciphertexts for EE and PR attacks on the five block ciphers using block-sized bit arrays are shown in Table 1.

Table 1. Sample plaintexts (PT) and ciphertexts (CT) for Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on the five block ciphers using block-sized bit arrays.

Block Cipher	Block Size		
SDES	16-bit	PT	11000011 00110011
		CT	10101110 00000011
SAES	16-bit	PT	11000111 01010100
		CT	10100101 11000011
DES	64-bit	PT	10000010 00000001 00001100 01100010 11110101 11110101 10011011 00100010
		CT	10100001 01100101 10110101 10011101 01100011 11100100 01101100 01100000
AES-128	128-bit	PT	11010111 00101011 01100001 00001000 00101010 01000000 01011111 00010010 10111001 01100011 11110011 01111111 01100111 10000001 01001100 00011111
		CT	10100110 01111011 00010010 11001010 00011001 01101110 01110001 11000111 00111010 01010110 10010001 10010111 01101111 11111010 00001110 01010001
SPECK32/64	32-bit	PT	11100100 01100101 11100001 01010000
		CT	10011011 11010000 01110000 11100101

4.2.2. Texts

For Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on block ciphers using texts, randomly generated texts were used as plaintexts. Each sentence has 30 words, and the words are combinations of randomly selected lowercase letters. The length of each word was randomly decided in the range from 1 to 15. The number of sentences is 50,000, and they were divided into 25,000 sentences for training data and 25,000 sentences for the test.

For Ciphertext Classification (CC) attacks on block ciphers using texts, the IMDB dataset [55] was used as plaintexts. The texts in the dataset are the movie reviews that have two classes, positive and negative. The total number of the texts is 50,000, consisting of 25,000 positive reviews and 25,000 negative reviews. They were divided in half and used each for train and test data, respectively, which each have 12,500 positive reviews and 12,500 negative reviews for a total of 25,000 texts. The texts were encrypted with the block ciphers using two different text encryption methods, Word-based Text Encryption (WTE) and Sentence-based Text Encryption (STE). Sample plaintexts and the ciphertexts for CC attacks on five block ciphers using texts are shown in Table 2.

Table 2. Sample plaintexts (PT) and ciphertexts (CT) for Ciphertext Classification (CC) attacks on the five block ciphers using texts.

	Block Cipher	Word-Based Text Encryption (WTE)	Sentence-Based Text Encryption (STE)
Raw PT		Long, boring, blasphemous. Never have I been so glad to see ending credits roll.	
Clean PT		<sos> long boring <unk> never have i been so glad to see ending credits roll	
CT	SDES	F898FCF0 0D6AE0E4 E49D609FE0E4 63147C7C E8746F9AF10C E3F76F9A 7C7E 0E05778D F898 65628A18 F299 E48E9BEF 778D728CE0E4 0A687F707660FCF0 F4947585	F898 CD2F 0D6A E0E4 13A4 E0E1 7AE0 252C 6314 B8BB E874 6F9A 3151 E3F7 6F9A 7A5B 13A4 E2E4 28A3 F898 154D 1E6D 4828 F299 E5B3 E2E4 9D5C 836B 7AE0 252C 0A68 7F70 7660 CD2F F494 7585
	SAES	99C6AA77 D589FBB8 DD698D6CFBB8 BF396E2B 5BB17E13AC87 A3F47E13 60EB 5D618F36 99C6 C2AE919C 9A56 79C361FB 8F36BA52FBB8 E7D79F3C99FCAA77 9D66C58E	99C6 ABF7 D589 FBB8 A008 E4B7 8726 6D9B BF39 63BB 5BB1 7E13 A207 A3F4 7E13 D00A A008 5F31 69EB 99C6 0007 A584 654B 9A56 2A4B 5F31 C006 9BBC 8726 6D9B E7D7 9F3C 99FC ABF7 9D66 C58E
	DES	CD4566317E56A93A B08A99398DBA92F9 4EEB5E9AFEA28938 AF17B02BA5C5338C 9DFC755044A4DCBA D6C05F3006D20C3D A7143EBC9CAE9204 769946F99C485DEE 5610353D62C49911 8D74B609B71E5152 3E45D02D673D4408 CB329810D3237036 62D20D09B363E0E8 3B3D46C16952F5E1 80DF7F1D6EDA029D	77237D3087C01421 1C625E2ACFA91E43 1798A6F4C1728A3 441CB1F82DE88B91 D115601D4635B08B 17E83944B92A2C3 6EDEE356D92CFB57 9C307FC6A8AB315C DA6307B428EFA210
	AES-128	4E868947AE87032CB4AB3AB1259FD2CC 70F97BAD90E32C3DC2BA64F333C8FC34 79782DCA09F8F78E79D155EF275766AB 7D380847DE6926FEE197829340375F85 DB4A3E7157DCCD0DE4532E6D513CE3E3 FED33C19304B8BFAE39F312D59B8679F 7A3DBE11315B8BD7F325BA9F0A560E42 85672272B1FD784F4FCE630D87CECF45 37A8F73D65868E4D2AD78E3817611106 F2440CE66FB9A30D2F3496AAFFE8D40B EEB323DDA10073F6D56137102684D6D9 44A2565D57F17525BE9FEBE534550C61 1FECEA93FC81AE8541C24FB09BA36BAF CD66B413450473A63082B3CBCC2673B7 F5171781FD2E56C8575C778F38F16402	606DD27DB193B86D47E752D9C414C902 58965476EDC7D79BCB9F010B90913C18 CE5AFF1E3E248FBB30A80BD0850C65E0 83BBE3B98C3FA6EEDB22DF9A2FD954F0 A1CF4930A7119C65861CB17D8A1A7D40
	SPECK32/64	763B3E6C 9A74EB3F F7E740D152C9C786 46D232A0 A9D02DB1A873A100 B1C1106D 59C3F40D B6D58649 28D6EEC1 A47F9C19 64625A3A 4FA8C3B4 D691084652C9C786 33CBDB0F0E5F734E 765A23F2	994E22A7 9A74EB3F F8727415 6CA852E0 EF876AE3 A9D02DB1 15112268 8C2B4007 EF230235 73914DFB F9692D63 9663649E 53C4D879 112B4EAD 6CA852E0 33CBDB0F 61BE74EB 765A23F2

4.3. Neural Cryptanalysis on Block-Sized Data

4.3.1. Results on Different Numbers of Training Data and Round Functions

To investigate the impact of the data amount used in training the models for Encryption Emulation (EE) and Plaintext Recovery (PR) attacks, we trained the fully connected-based deep learning model while increasing the quantity of training data. Since SDES and SAES use 16-bit block size, we used only half of the data, 2^{15} (=32,768), among the possible block-sized bit arrays, 2^{16} (=65,536), as the training data. In the other block ciphers, DES, AES-128, and SPECK32/64, the quantity of training data was increased by 4-times from

2^{16} (=65,536) and these data were used to train the models. We compared the results in different numbers of round functions by calculating the average of Bit Accuracy Possibility (BAP_{avg}) [46,47] in 2^{15} (=32,768) of test data as follows:

$$pred_i^n = \begin{cases} 0, \ if \ \ pred_i^n \ \leq \ 0.5 \\ 1, \ otherwise. \end{cases} \quad BAP_i = \frac{\sum_{n=1}^{N} XNOR(real_i^n, \ pred_i^n)}{N} \tag{10}$$

$$BAP_{avg} = \frac{\sum_{i=1}^{bs} BAP_i}{bs} \tag{11}$$

where $real_i^n$ and $pred_i^n$ represent the ith bit of the nth real and predicted outputs, respectively. BAP_i is the BAP of the ith bit, N is the total number of test data, BAP_{avg} is the average of BAP in total data, and bs is the block size of the block cipher. As shown in Figures 6 and 7, the more training data used in training, the higher the accuracy in every block cipher. In SPECK32/64, 2^{16} of training data was enough to break the single round function, and the BAP_{avg} reached 1.0 when the training data were increased to 2^{22} in two round functions. Furthermore, for the three round functions in SPECK32/64, BAP_{avg} increased to 0.6~0.7 in 2^{22} of training data. Also, DES with a single and two round functions reached 1.0 and 0.8~0.9 BAP_{avg}, respectively, on 2^{22} of training data. However, every BAP_{avg} of AES-128 was 0.5, even with a single round function. As a result, the possibility for EE and PR attacks on block ciphers using a block-sized bit array was higher when we trained the deep learning models with as much data as possible.

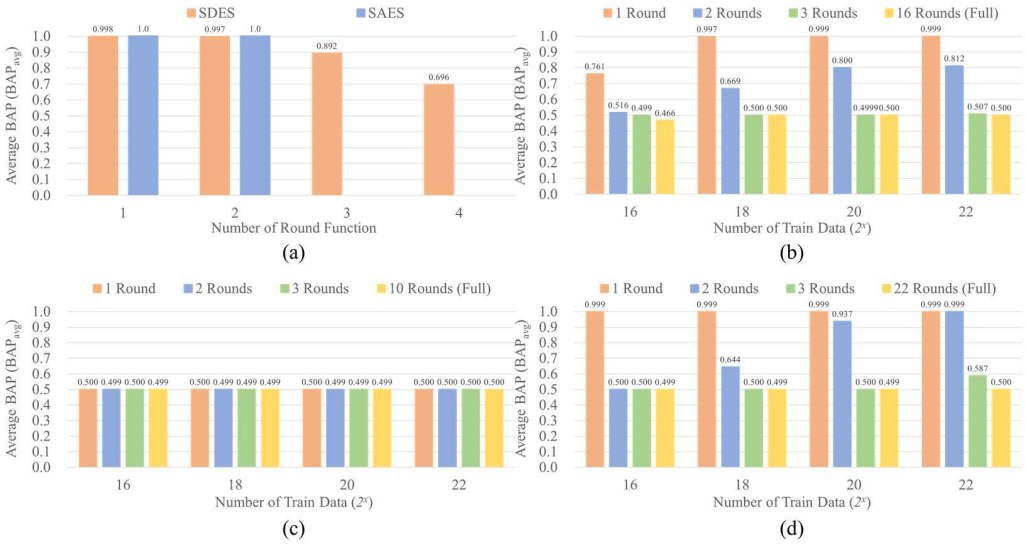

Figure 6. Average Bit Accuracy Probability (BAP_{avg}) of Encryption Emulation (EE) attack with different quantities of training data and different numbers of round functions. (**a**) SDES and SAES with 2^{15} training data. (**b**) DES. (**c**) AES-128. (**d**) SPECK32/64.

4.3.2. Results on Different Deep Learning Models

To identify the impact of the deep learning model architecture in neural cryptanalysis, we additionally performed the Encryption Emulation (EE), Plaintext Recovery (PR), and Key Recovery (KR) attacks on block ciphers using block-sized bit arrays by using an RNN-based deep learning model, BiLSTM [51]. The result was compared with that of the fully connected-based deep learning model. The model used 2^{22} (4,194,304) training data, which was the maximum number of training data in the fully connected-based deep learning model. The BAP_{avg} in the RNN-based deep learning model was slightly increased compared to the fully connected-based deep learning model as shown in Table 3. Consequently, it was more efficient to extract the features by dividing inputs into meaningful parts with

the RNN-based deep learning model, BiLSTM, rather than using them all at once with the fully connected-based deep learning model.

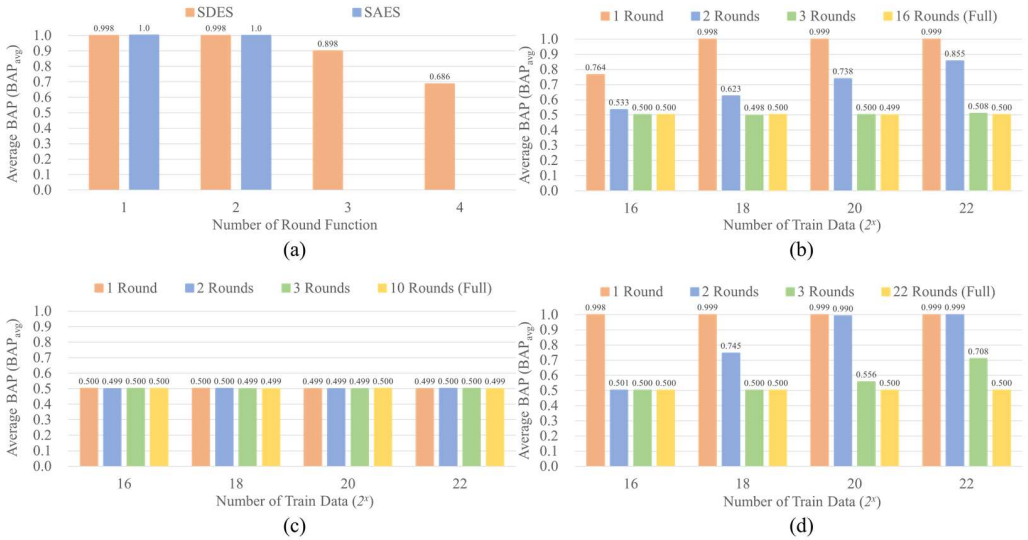

Figure 7. Average Bit Accuracy Probability (BAP_{avg}) of Plaintext Recovery (PR) attack with different quantities of training data and different numbers of round functions. (**a**) SDES and SAES with 2^{15} training data. (**b**) DES. (**c**) AES-128. (**d**) SPECK32/64.

Table 3. Average Bit Accuracy Probability (BAP_{avg}) of Encryption Emulation (EE), Plaintext Recovery (PR), and Key Recovery (KR) attacks using different deep learning models on the five block ciphers with different numbers of round functions for block-sized bit array encryption.

Block Cipher	Round Number	EE Attack		PR Attack		KR Attack	
		Fully Connected-Based	RNN-Based (BiLSTM)	Fully Connected-Based	RNN-Based (BiLSTM)	Fully Connected-Based	RNN-Based (BiLSTM)
SDES	1	0.998	1.0	0.998	1.0	0.601	0.601
	2	0.997	1.0	0.998	0.999	0.826	0.851
	3	0.892	0.817	0.898	0.828	0.593	0.651
	4 (Full Rounds)	0.696	0.691	0.686	0.689	0.450	0.607
SAES	1	1.0	1.0	1.0	0.998	0.573	0.621
	2 (Full Rounds)	0.998	0.999	1.0	0.999	0.528	0.610
DES	1	1.0	1.0	0.999	1.0	0.589	0.589
	2	0.812	0.875	0.855	0.961	0.666	0.825
	3	0.507	0.510	0.508	0.513	0.517	0.510
	16 (Full Rounds)	0.500	0.500	0.500	0.500	0.500	0.499
AES-128	1	0.500	0.499	0.500	0.499	0.500	0.500
	2	0.499	0.499	0.500	0.500	0.499	0.499
	3	0.500	0.500	0.499	0.500	0.500	0.499
	10 (Full Rounds)	0.499	0.499	0.499	0.499	0.499	0.499
SPECK32/64	1	1.0	1.0	0.999	0.999	0.624	0.624
	2	1.0	0.999	0.999	0.999	0.698	0.749
	3	0.587	0.883	0.708	0.925	0.499	0.499
	22 (Full Rounds)	0.500	0.500	0.500	0.500	0.500	0.500

Interestingly, the single round function showed lower BAP_{avg} than the two round functions in KR attacks on SDES, DES, and SPECK32/64. Since SDES and DES split the plaintexts into two blocks, left and right, and then applied the XOR operation with the key only to the right block of the input for the round function, only half of the ciphertexts for the single round function have information about the keys. Because the left block of the first round function is the same as the right block of the input for the first round function, the ciphertexts for the single round function in SDES and DES have less information of the key than the ciphertexts with two round functions. Similarly, SPECK32/64 splits the plaintexts and the keys into two blocks, respectively. However, the inputs for the first round function are applied to the operations with the right block of the keys, which are not applied to any operations in the key schedule. Therefore, the ciphertexts encrypted with a single round function in SPECK32/64 only have information about the half key, the right part of the key. In other words, the deep learning model could extract more features related to the keys from the ciphertexts with two round functions than those with a single round function in SDES, DES, and SPECK32/64.

Furthermore, we performed the KR attack on DES using randomly generated block-sized bit arrays of 2^{22} (=4,194,304), not only for a randomly generated single block-sized bit array, by using the RNN-based deep learning model, BiLSTM. The keys were generated for two cases, the same keys as the plaintexts and randomly generated keys. When the keys were the same as the plaintexts, DES was vulnerable and perfectly broken. In contrast, it was difficult to recover the keys, which were randomly generated regardless of the plaintexts, and showed 0.5 BAP_{avg}. Consequently, using the same key as the plaintexts can make the block ciphers vulnerable even if the plaintexts are randomly generated.

4.4. Neural Cryptanalysis on Text Data

4.4.1. Results on Different Text Encryption Methods

To figure out how to improve the strength of block ciphers using texts, we encrypted plaintexts with two different text encryption methods, Word-based Text Encryption (WTE) and Sentence-based Text Encryption (STE). In Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on block ciphers using texts, a transformer-based deep learning model, T5-small [52], was used to generate the ciphertexts and recover the plaintexts. It tokenizes the plaintexts and the ciphertexts by using the SentencePiece [56] and considers context information of the sequences in word embedding. We used the ratio of the correctly predicted tokens over the total tokens as metrics, which can be calculated as follows:

$$\text{CorrrectlyPredictedTokenRatio} = \frac{100}{M} \times \sum_{n=1}^{M} \frac{N(predtok^{n}_{cor})}{N(predtok^{n}_{cor}) + N(predtok^{n}_{incor})} \tag{12}$$

where M represents the total number of test data, and $N(predtok^{n}_{cor})$ and $N(predtok^{n}_{incor})$ are the number of correctly and incorrectly predicted tokens in n^{th} test data, respectively.

The correctly predicted token ratio in WTE was higher than that in STE, as shown in Figure 8, which means STE can make the block ciphers more secure compared to the WTE in EE and PR attacks.

In Ciphertext Classification (CC) attacks on block ciphers using texts, the ciphertexts for each text encryption method were classified by using the RNN-based deep learning model that consists of a BiGRU [53], which utilized Word2Vec [57,58] to represent the words as embedding vectors that a deep learning model can train. The result was compared using classification accuracy, which is the percentage of correctly classified ciphertexts out of the total ciphertexts as follows:

$$Classification\ Acc = \frac{N(predC_{cor})}{N(predC_{cor}) + (predC_{incor})} \times \frac{100}{M} \tag{13}$$

where $N(predC_{cor})$ and $N(predC_{incor})$ represent the number of correctly and incorrectly classified test ciphertexts, respectively, and M is the total number of test data.

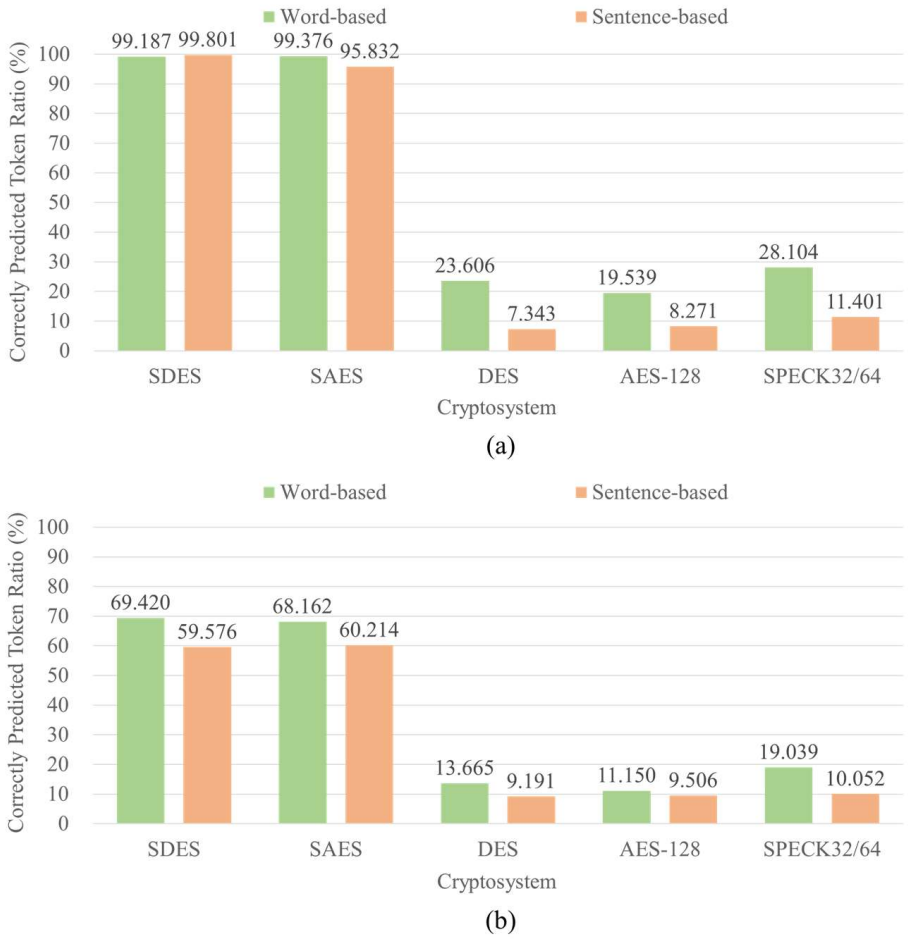

Figure 8. Correctly predicted token ratio of Encryption Emulation (EE) and Plaintext Recovery (PR) attacks with different text encryption methods. (**a**) EE attack. (**b**) PR attack.

The classification accuracy of the ciphertexts encrypted with STE was lower than that of ciphertexts encrypted with WTE in every block cipher as shown in Figure 9a. Furthermore, the accuracy of ciphertext classification on WTE was almost the same as the accuracy of plaintext classification. Therefore, ciphertexts encrypted with WTE were more vulnerable to CC attack than STE in every block cipher.

4.4.2. Results on Different Deep Learning Models

To identify the impact of the deep learning model architecture in the Ciphertext Classification (CC) attacks on block ciphers using texts, we additionally classified the ciphertexts with a transformer-based deep learning model, BERT-base [54], which represents the words into the embedding vectors according to the context information in the sentences. It also tokenizes the words by using the subword-based tokenizer WordPiece. Since transformer-based deep learning model uses a subword tokenizer, it can divide a single block in the ciphertexts into several tokens. Thus, the classification accuracy of WTE in the transformer-based deep learning model was lower than that in the RNN-based deep learning model with Word2Vec as shown in Figure 9b. In other words, it was challenging to capture the features from the separated blocks because the block ciphers encrypt the texts block-wise. Moreover, unlike in WTE, contexts in the ciphertexts were removed in STE, and the transformer-based deep learning model that uses only attention mechanisms to extract context information and relations between the tokens was not proper in neural cryptanalysis on block ciphers. Thus, the transformer-based deep learning model could not train the ciphertexts encrypted with STE and showed 50% classification accuracy in every block cipher. Consequently, word-based tokenization and static word embedding

were more efficient in the CC attacks on block ciphers using texts than the subword-based tokenization and contextualized word embedding.

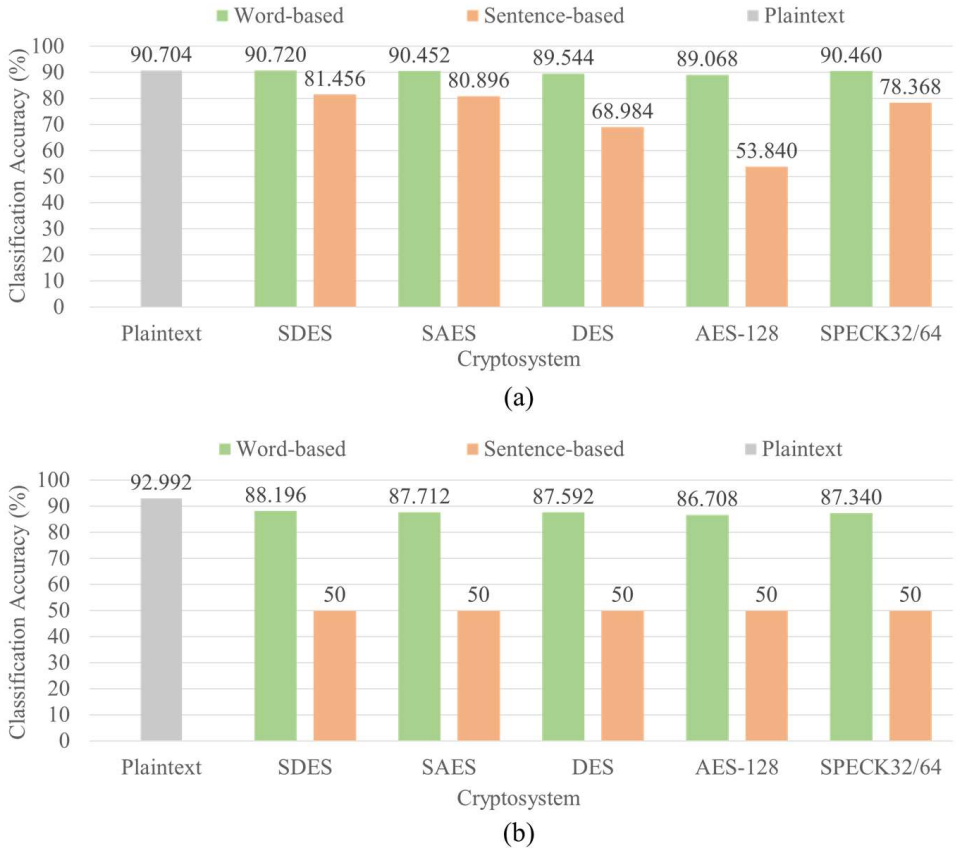

Figure 9. Classification accuracy of Ciphertext Classification (CC) attacks in different deep learning models with different text encryption methods. (**a**) RNN-based (BiGRU) CC attack. (**b**) Transformer-based (BERT-base) CC attack.

4.4.3. Results on Different Operation Modes

To confirm that the block cipher using WTE is more vulnerable than that using STE against Encryption Emulation (EE), Plaintext Recovery (PR), and Ciphertext Classification (CC) attacks, we also performed neural cryptanalysis on the block ciphers using texts in the other operation modes. The attacks on DES [12] and AES-128 [13] in CBC and CFB modes were compared with the attacks in ECB mode. The models and hyper-parameters for EE, PR, and CC attacks in CBC and CFB modes were the same as the model and hyper-parameters previously used in ECB mode. The correctly predicted token ratio for EE attacks was higher than that in PR attacks. However, both DES and AES-128 were more difficult to break when the texts were encrypted with STE than WTE as shown in Figure 10. The average correctly predicted token ratio was around 10% in STE but 15% and 25% for EE and PR attacks, respectively, in WTE. Although the classification accuracy decreased in CBC and CFB modes compared to that in ECB mode, the classification accuracy of the ciphertexts encrypted with WTE still showed over 80% as shown in Figure 11. In contrast, deep learning models could not train and classify the ciphertexts encrypted with STE, and the classification accuracy was almost 50% in every operation mode. As a result, STE could improve the strength of the block ciphers against the EE, PR, and CC attacks on block ciphers using texts, and STE must be used for text encryption instead of WTE.

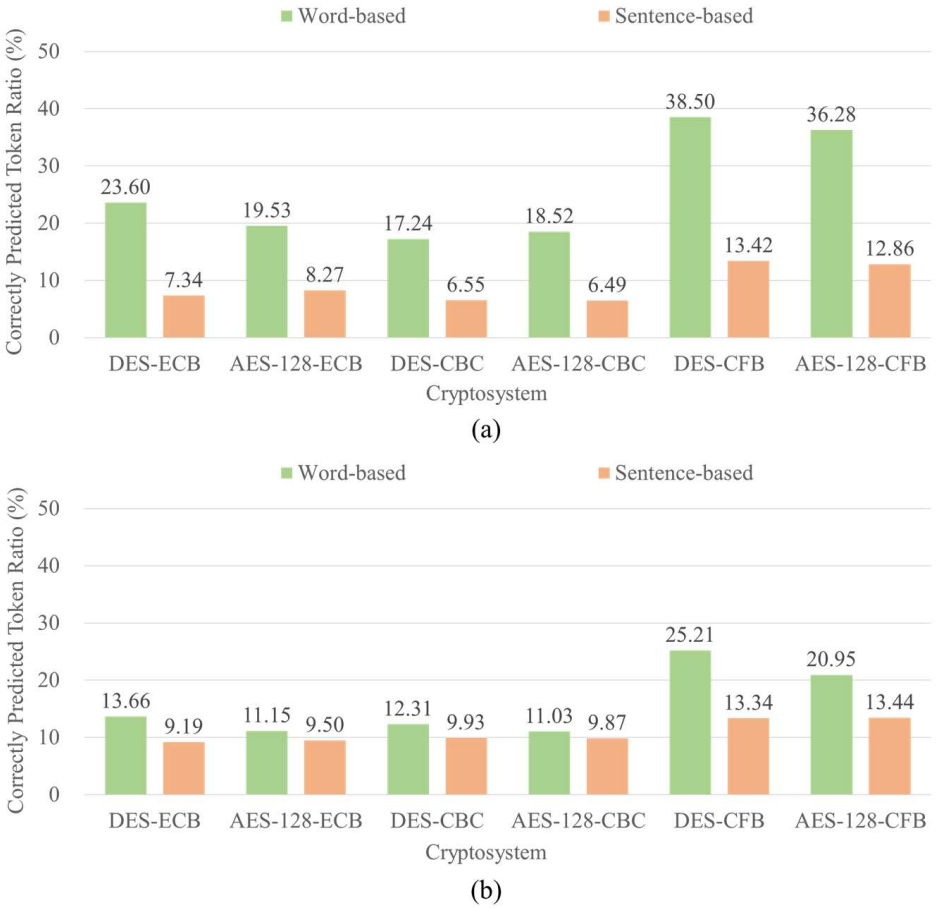

Figure 10. Correctly predicted token ratio of Encryption Emulation (EE) and Plaintext Recovery (PR) attacks in different operation modes with different text encryption methods. (**a**) EE attack. (**b**) PR attack.

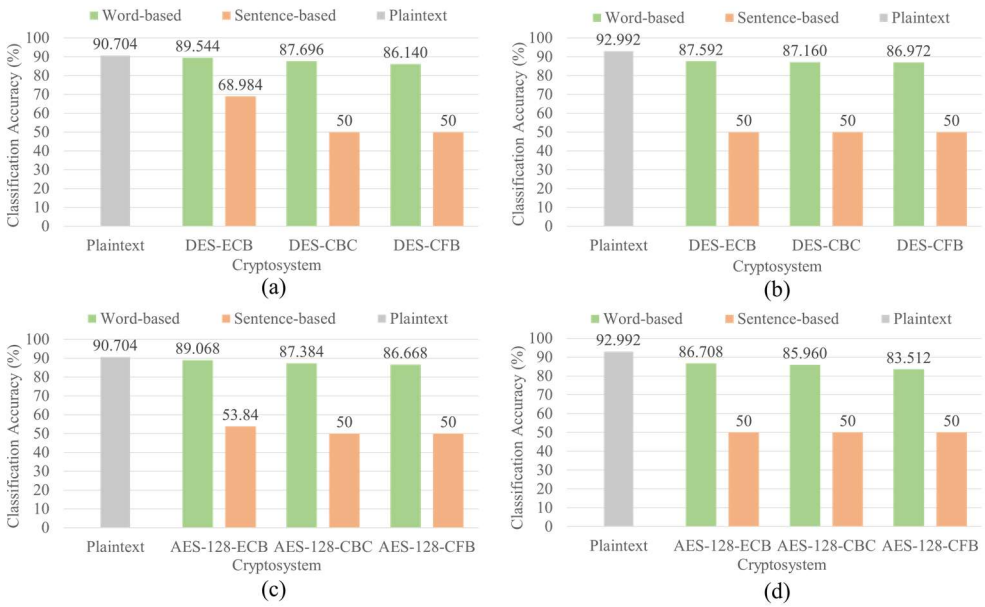

Figure 11. Classification accuracy of Ciphertext Classification (CC) attack in different operation modes with different text encryption methods. (**a**) RNN-based (BiGRU) CC attack on DES. (**b**) Transformer-based (BERT-base) CC attack on DES. (**c**) RNN-based (BiGRU) CC attack on AES-128. (**d**) Transformer-based (BERT-base) CC attack on AES-128.

5. Discussion

Block ciphers using block-sized bit arrays and texts as plaintexts were analyzed by performing Encryption Emulation (EE), Plaintext Recovery (PR), Key Recovery (KR), and Ciphertext Classification (CC) attacks based on different deep learning models. Since the performance of the deep learning model is typically dependent on the amount of training data, BAP_{avg} increased when more data were used for model training in EE and PR attacks on block-sized bit arrays. Moreover, applying two round functions was more vulnerable than using a single round function in KR attacks on DES and SPECK32/64 using block-sized bit arrays, because of the encryption process that splits the plaintexts into two blocks. Specifically, DES applies the XOR operation with a key only to the right block, and SPECK32/64 applies the XOR operation with only the right part of the key to the left block. Therefore, key information is contained in the part of the ciphertexts of DES to which a single round function is applied, and part of the key information is included in the ciphertexts of SPECK32/64 in which a single round function is applied. In addition, even if texts are encrypted, WTE generates the same cipher word for the same plain word, while whitespace position information is removed in STE. So, WTE showed almost the same classification accuracy as plaintexts in CC attacks on texts. Moreover, since block ciphers like Feistel Network and SPECK32/64 encrypt the plaintexts after splitting them, the RNN-based deep learning model outperformed the fully connected-based deep learning model in capturing the relation between the split blocks and showed better BAP_{avg}. Also, the inputs for the RNN-based deep learning model in CC attacks on texts were the tokens generated by using a word-based tokenizer, which are the cipher words and cipher blocks in the ciphertexts. However, the transformer-based deep learning model separated the ciphertexts into the tokens with a subword-based tokenizer, and the tokens are converted into embedding vectors by using contextualized word embedding, which is not meaningful in the ciphertexts removing the context information. Thus, the classification accuracy was higher in the RNN-based deep learning model than the transformer-based deep learning model in CC attacks on texts.

In addition, we could identify that the key size is significant in determining the strength of the block ciphers. The simplified version of the block ciphers, SDES and SAES, using a small key size and fewer round functions, were easily broken compared to the other block ciphers. In contrast, AES was challenging to break, even with a single round function, as it uses the largest key size. And DES and SPECK32/64 showed almost the same vulnerability on deep learning-based attacks, but SPECK32/64, using a smaller key size than DES, was slightly more vulnerable.

The results of the neural cryptanalysis in this paper can be interpreted from two points of view, the person who wants to protect personal information by using block ciphers in the system and the person who wants to break the block ciphers and take the personal information. First, from the perspective of the protector, the block ciphers must be designed with a large key size and sufficient round functions, and the key must not be the same as the plaintext. Also, texts must be encrypted with STE, not WTE. From the perspective of the attacker, RNN-based deep learning models are more appropriate in deep learning-based attacks on the block ciphers, and as much data for training the model should be collected as possible.

6. Conclusions

We comprehensively analyze five block ciphers, DES, SDES, AES-128, SAES, and SPECK32/64, on deep learning-based Encryption Emulation (EE), Plaintext Recovery (PR), Key Recovery (KR), and Ciphertext Classification (CC) attacks. The block ciphers using different numbers of round functions in a block-sized bit array encryption are investigated in EE, PR, and KR attacks using deep learning models trained with different numbers of data. Also, the block ciphers with two different text encryption methods, Word-based Text Encryption (WTE) and Sentence Text Encryption (STE), for text encryption are analyzed in three operation modes, ECB, CBC, and CFB, on EE, PR, and CC attacks using the deep

learning models. As a result, more data for training the models can increase the possibility of successful attacks, and STE can improve security, even in the CBC and CFB modes, unlike WTE, which shows almost the same classification accuracy as the plaintexts, especially in CC attacks. Moreover, using the same key as the plaintext is vulnerable against KR attacks, and applying two round functions in the encryption of SDES, DES, and SPECK32/64 provides a better KR-attack performance than applying a single round function. Also, the RNN-based deep learning model is more suitable in neural cryptanalysis than the fully connected-based and transformer-based deep learning models, especially in KR and CC attacks, and shows higher BAP_{avg} and classification accuracy. From the experiments, we could investigate the weaknesses of the block ciphers, which can help prevent attacks and design new cryptographic algorithms. However, although we found that using more training data can increase the likelihood of attacks, it is challenging to develop efficient learning algorithms for the huge amount of data. In the future, we will design new deep learning architectures based on distributed machine learning or federated learning [59], which can be more efficient in neural cryptanalysis. Moreover, since recent neural cryptanalysis has used deep learning architectures designed for the data, not encrypted, we will explore new architectures of deep learning models that are more suitable for neural cryptanalysis and show better performance than traditional cryptanalysis.

Author Contributions: Conceptualization, I.M.; methodology, O.J. and E.A.; software, O.J. and E.A.; validation, O.J. and I.M.; writing—original draft preparation, O.J. and E.A; writing—review and editing, I.M.; visualization, O.J.; supervision, I.M.; funding acquisition, I.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00126, Research on AI-based Cryptanalysis and Security Evaluation). This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2024-00400368, Development of Image Integrity Verification Technology for Forgery Prevention and Original Proof).

Data Availability Statement: The dataset used in this paper is a public dataset which has been referenced in the manuscript.

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

References

1. Paar, C.; Pelzl, J. *Understanding Cryptography: A Textbook for Students and Practitioners*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2009.
2. Stamp, M. *Information Security: Principles and Practice*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
3. Gupta, A.; Barthwal, A.; Vardhan, H.; Kakria, S.; Kumar, S.; Parihar, A.S. Evolutionary study of distributed authentication protocols and its integration to UAV-assisted FANET. *Multimed. Tools Appl.* **2023**, *82*, 42311–42330. [CrossRef]
4. Adleman, L.M.; Rothemund, P.W.; Roweis, S.; Winfree, E. On applying molecular computation to the data encryption standard. *J. Comput. Biol.* **1999**, *6*, 53–63. [CrossRef] [PubMed]
5. Matsui, M.; Yamagishi, A. A new method for known plaintext attack of FEAL cipher. In *Advances in Cryptology—EUROCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques, Balatonfüred, Hungary, 24–28 May 1992*; Proceedings 11; Springer: Berlin/Heidelberg, Germany, 1993; pp. 81–91.
6. Matsui, M. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology–EUROCRYPT'93: Workshop on the Theory and Application of Cryptographic Techniques Lofthus, Norway, 23–27 May 1993*; Springer: Berlin/Heidelberg, Germany, 2003; Volume 765, p. 386.
7. Biham, E.; Shamir, A. *Differential Cryptanalysis of the Data Encryption Standard*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
8. Berman, D.S.; Buczak, A.L.; Chavis, J.S.; Corbett, C.L. A survey of deep learning methods for cyber security. *Information* **2019**, *10*, 122. [CrossRef]
9. Chen, Y.; Shen, Y.; Yu, H. Neural-Aided Statistical Attack for Cryptanalysis. *Comput. J.* **2023**, *66*, 2480–2498. [CrossRef]
10. Truong, N.D.; Haw, J.Y.; Assad, S.M.; Lam, P.K.; Kavehei, O. Machine learning cryptanalysis of a quantum random number generator. *IEEE Trans. Inf. Forensics Secur.* **2018**, *14*, 403–414. [CrossRef]

11. Baek, S.; Kim, K. Recent advances of neural attacks against block ciphers. In Proceedings of the 2020 Symposium on Cryptography and Information Security (SCIS 2020), Kochi, Japan, 28–31 January 2020; IEICE Technical Committee on Information Security: Tokyo, Japan, 2020.
12. *FIPS-Pub.46*; Data Encryption Standard. Federal Information Processing Standards Publication. National Institute of Standards and Technology: Gaithersburg, MD, USA, 1999.
13. Rijmen, V.; Daemen, J. Advanced Encryption Standard. In *Proceedings of Federal Information Processing Standards Publications*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001; Volume 19, p. 22.
14. Musa, M.A.; Schaefer, E.F.; Wedig, S. A simplified AES algorithm and its linear and differential cryptanalyses. *Cryptologia* **2003**, *27*, 148–177. [CrossRef]
15. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK families of lightweight block ciphers. *Cryptol. Eprint Arch.* **2013**, 404.
16. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
17. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
18. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349. [CrossRef]
19. Hai, H.; Pan, S.; Liao, M.; Lu, D.; He, W.; Peng, X. Cryptanalysis of random-phase-encoding-based optical cryptosystem via deep learning. *Opt. Express* **2019**, *27*, 21204–21213. [CrossRef]
20. Jeong, O.; Moon, I. Adaptive transfer learning-based cryptanalysis on double random phase encoding. *Opt. Laser Technol.* **2024**, *168*, 109916. [CrossRef]
21. He, C.; Ming, K.; Wang, Y.; Wang, Z.J. A deep learning based attack for the chaos-based image encryption. *arXiv* **2019**, arXiv:1907.12245.
22. Refregier, P.; Javidi, B. Optical image encryption based on input plane and Fourier plane random encoding. *Opt. Lett.* **1995**, *20*, 767–769. [CrossRef] [PubMed]
23. Ahouzi, E.; Zamrani, W.; Azami, N.; Lizana, A.; Campos, J.; Yzuel, M.J.; Engineering, O. Optical triple random-phase encryption. *Opt. Eng.* **2017**, *56*, 113114. [CrossRef]
24. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
25. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
26. Liu, X.; Liu, W.; Mei, T.; Ma, H. Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance. *IEEE Trans. Multimed.* **2017**, *20*, 645–658. [CrossRef]
27. Guan, Z.-H.; Huang, F.; Guan, W. Chaos-based image encryption algorithm. *Phys. Lett. A* **2005**, *346*, 153–157. [CrossRef]
28. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*; MIT Press: Cambridge, MA, USA, 1995; Volume 3361, pp. 255–258.
29. Tanuwidjaja, H.C.; Choi, R.; Baek, S.; Kim, K. Privacy-preserving deep learning on machine learning as a service—A comprehensive survey. *IEEE Access* **2020**, *8*, 167425–167447. [CrossRef]
30. Boulemtafes, A.; Derhab, A.; Challal, Y. A review of privacy-preserving techniques for deep learning. *Neurocomputing* **2020**, *384*, 21–45. [CrossRef]
31. Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 201–210.
32. Hesamifard, E.; Takabi, H.; Ghasemi, M. Cryptodl: Deep neural networks over encrypted data. *arXiv* **2017**, arXiv:1711.05189.
33. Rivest, R.L.; Adleman, L.; Dertouzos, M.L. On data banks and privacy homomorphisms. *Found. Secur. Comput.* **1978**, *4*, 169–180.
34. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
35. Lidkea, V.M.; Muresan, R.; Al-Dweik, A. Convolutional neural network framework for encrypted image classification in cloud-based ITS. *IEEE Open J. Intell. Transp. Syst.* **2020**, *1*, 35–50. [CrossRef]
36. Ferguson, N. *Impossible Differentials in Twofish*; Counterpane Systems: Minneapolis, MN, USA, 1999.
37. Biham, E.; Dunkelman, O.; Keller, N. Linear cryptanalysis of reduced round Serpent. In Proceedings of the International Workshop on Fast Software Encryption, Yokohama, Japan, 2–4 April 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 16–27.
38. Thoms, G.R.; Muresan, R.; Al-Dweik, A. Chaotic encryption algorithm with key controlled neural networks for intelligent transportation systems. *IEEE Access* **2019**, *7*, 158697–158709. [CrossRef]
39. Otter, D.W.; Medina, J.R.; Kalita, J.K. A survey of the usages of deep learning for natural language processing. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 604–624. [CrossRef] [PubMed]
40. Sikdar, S.; Kule, M. Recent Trends in Cryptanalysis Techniques: A Review. In Proceedings of the International Conference on Frontiers in Computing and Systems, Punjab, India, 19–21 December 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 209–222.
41. Focardi, R.; Luccio, F.L. Neural Cryptanalysis of Classical Ciphers. In Proceedings of the ICTCS, Urbino, Italy, 18–20 September 2018; pp. 104–115.

42. Ahmadzadeh, E.; Kim, H.; Jeong, O.; Kim, N.; Moon, I. A deep bidirectional LSTM-GRU network model for automated ciphertext classification. *IEEE Access* **2022**, *10*, 3228–3237. [CrossRef]
43. Alani, M.M. Neuro-cryptanalysis of DES and triple-DES. In Proceedings of the Neural Information Processing: 19th International Conference, ICONIP 2012, Doha, Qatar, 12–15 November 2012; Proceedings, Part V 19. Springer: Berlin/Heidelberg, Germany, 2012; pp. 637–646.
44. Xiao, Y.; Hao, Q.; Yao, D.D. Neural cryptanalysis: Metrics, methodology, and applications in CPS ciphers. In Proceedings of the 2019 IEEE Conference on Dependable and Secure Computing (DSC), Hangzhou, China, 8–20 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–8.
45. Hu, X.; Zhao, Y. Research on plaintext restoration of AES based on neural network. *Secur. Commun. Netw.* **2018**, *2018*, 6868506. [CrossRef]
46. So, J. Deep-learning-based cryptanalysis of lightweight block ciphers. *Secur. Commun. Netw.* **2020**, *2020*, 3701067. [CrossRef]
47. Kim, H.; Lim, S.; Kang, Y.; Kim, W.; Kim, D.; Yoon, S.; Seo, H. Deep-learning-based cryptanalysis of lightweight block ciphers revisited. *Entropy* **2023**, *25*, 986. [CrossRef] [PubMed]
48. Abdurakhimov, B.; Abdurazzokov, J.; Lingyun, L. Analysis of the use of artificial neural networks in the cryptanalysis of the SM4 block encryption algorithm. *AIP Conf. Proc.* **2023**, *2812*, 020048.
49. Kimura, H.; Emura, K.; Isobe, T.; Ito, R.; Ogawa, K.; Ohigashi, T. A Deeper Look into Deep Learning-based Output Prediction Attacks Using Weak SPN Block Ciphers. *J. Inf. Process.* **2023**, *31*, 550–561. [CrossRef]
50. Kumar, K.; Tanwar, S.; Kumar, S. Deep-Learning-based Cryptanalysis through Topic Modeling. *Eng. Technol. Appl. Sci. Res.* **2024**, *14*, 12524–12529. [CrossRef]
51. Graves, A.; Mohamed, A.-R.; Hinton, G. Speech recognition with deep recurrent neural networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 6645–6649.
52. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
53. Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
54. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
55. Maas, A.; Daly, R.E.; Pham, P.T.; Huang, D.; Ng, A.Y.; Potts, C. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 142–150.
56. Kudo, T.; Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv* **2018**, arXiv:1808.06226.
57. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
58. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems 26 (NIPS 2013), Lake Tahoe, NV, USA, 5–10 December 2013; Volume 26, pp. 1–9.
59. Konečný, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv* **2016**, arXiv:1610.05492.