mathematics

MDPI

Article

Scalable Neural Cryptanalysis of Block Ciphers in Federated Attack Environments

Ongee Jeong [1], Seonghwan Park [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
[2] Department of Artificial Intelligence, Daegu Gyeongbuk Institute of Science & Technology (DGIST), Daegu 42988, Republic of Korea
* Correspondence: inkyu.moon@dgist.ac.kr

Abstract

This paper presents an extended investigation into deep learning-based cryptanalysis of block ciphers by introducing and evaluating a multi-server attack environment. Building upon our prior work in centralized settings, we explore the practicality and scalability of deploying such attacks across multiple distributed edge servers. We assess the vulnerability of five representative block ciphers—DES, SDES, AES-128, SAES, and SPECK32/64—under two neural attack models: Encryption Emulation (EE) and Plaintext Recovery (PR), using both fully connected neural networks and Recurrent Neural Networks (RNNs) based on bidirectional Long Short-Term Memory (BiLSTM). Our experimental results show that the proposed federated learning-based cryptanalysis framework achieves performance nearly identical to that of centralized attacks, particularly for ciphers with low round complexity. Even as the number of edge servers increases to 32, the attack models maintain high accuracy in reduced-round settings. We validate our security assessments through formal statistical significance testing using two-tailed binomial tests with 99% confidence intervals. Additionally, our scalability analysis demonstrates that aggregation times remain negligible (<0.01% of total training time), confirming the computational efficiency of the federated framework. Overall, this work provides both a scalable cryptanalysis framework and valuable insights into the design of cryptographic algorithms that are resilient to distributed, deep learning-based threats.

Keywords: block ciphers; neural cryptanalysis; distributed learning; encryption emulation (EE); plaintext recovery (PR); data encryption standard (DES); advanced encryption standard (AES); SPECK; deep learning; federated attack environment

MSC: 68P25; 68T07

Received: 13 December 2025
Revised: 15 January 2026
Accepted: 19 January 2026
Published: 22 January 2026

1. Introduction

Due to the increasing deployment of technologies such as the Internet of Things (IoT) and Artificial Intelligence (AI), the significance of data has grown rapidly [1–4]. However, as data are frequently transmitted through wireless and untrusted networks, concerns about data privacy and protection have intensified [5,6]. Moreover, the growing sophistication of cyberattacks—often powered by AI—has made them more difficult to detect and defend. To mitigate these threats, cryptographic algorithms [7,8] are widely used to protect sensitive information by transforming data into forms unintelligible to unauthorized parties.

Nonetheless, several cryptographic algorithms have been shown to be vulnerable to both classical and modern attack strategies. To ensure data confidentiality and trustworthiness, it is crucial to evaluate cryptographic algorithms for potential weaknesses. Cryptanalysis plays this role by testing algorithms through legitimate attack strategies such as Ciphertext Only Attack (COA), Known Plaintext Attack (KPA), Chosen Plaintext Attack (CPA), and Chosen Ciphertext Attack (CCA), depending on what information is available to the attacker. Additionally, methods such as linear [9,10] and differential cryptanalysis [11] have been widely used to evaluate algorithmic resilience.

With recent breakthroughs in deep learning [12,13], neural-based techniques have been successfully adopted in various domains, including image [14] and text analysis [15]. These advances have also enabled a new class of cryptanalytic approaches based on artificial intelligence [16–18]. Compared to traditional cryptanalysis—which often requires extensive mathematical analysis—neural-aided and neural cryptanalysis offer the advantage of automatically learning complex, non-linear patterns in encrypted data. For instance, deep learning models have been used to enhance differential distinguishers for key recovery [19], and they can even capture side-channel leakage without preprocessing steps [20–22]. Moreover, deep models are capable of learning direct mappings between plaintexts and ciphertexts, or even recovering cryptographic keys.

In our prior work [23], we demonstrated that block ciphers such as DES [11], SDES [8], AES [24], SAES [25], and SPECK [26] can be effectively attacked using deep learning-based methods, including Encryption Emulation (EE) and Plaintext Recovery (PR). These attacks were conducted in a centralized single-server environment, where a deep learning model was trained using plaintext–ciphertext pairs collected in one location. While this approach highlighted the vulnerability of reduced-round block ciphers, it also presented a practical limitation: deploying and managing such centralized cryptanalysis in real-world scenarios may not be feasible due to computational constraints or network access limitations.

To address the limitations of centralized cryptanalysis, this study extends our previous framework by introducing a decentralized attack paradigm based on a multi-server architecture. Specifically, we simulate federated deep learning-based cryptanalysis, in which multiple distributed edge servers independently collect plaintext–ciphertext pairs and collaboratively train a global attack model through iterative weight aggregation. This architecture more accurately reflects realistic adversarial scenarios, where attackers may not possess access to a single high-performance server but can instead leverage a network of lower-capacity, geographically distributed devices.

Our empirical results demonstrate that the attack performance in the multi-server environment closely mirrors that of the centralized single-server setting, particularly for lightweight block ciphers such as SDES, SAES, DES, and SPECK, when configured with reduced round complexity. Notably, even as the number of edge servers increases, the system maintains robust performance in both training and inference, validating the scalability and practicality of distributed neural cryptanalysis. In contrast, AES-128—characterized by its higher number of rounds and larger key size—consistently resists both centralized and distributed attacks, underscoring its resilience against data-driven adversaries.

To the best of our knowledge, this is the first comprehensive study to empirically demonstrate the feasibility of deep learning-based attacks on block ciphers within a federated, multi-server environment. These findings open new directions for real-world, automated cryptanalysis and highlight the need for cryptographic algorithm designers to consider federated, AI-powered threat models in future security assessments and standardization efforts.

The remainder of this paper is organized as follows: Section 2 reviews prior work on neural-aided and neural cryptanalysis applied to various cryptographic algorithms.

Section 3 outlines the methodology for implementing Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on the selected block ciphers. Section 4 presents experimental results comparing the performance of block ciphers under single-server and multi-server attack environments. Finally, Section 5 concludes the paper by summarizing the proposed framework and key findings.

2. Related Works

In neural-aided cryptanalysis [27–30], deep learning has been employed to construct differential distinguishers, which are designed to differentiate between randomly generated data and ciphertexts. The initial concept was introduced in [27] to enhance the efficiency and accuracy of key recovery attacks on SPECK32/64 [26]. In [28], ciphertext pairs with multiple differential patterns were input into a neural distinguisher targeting SIMECK32/64 [31]. The work in [29] utilized Convolutional Neural Networks (CNNs) [32] and Recurrent Neural Networks (RNNs) [33] as underlying architectures for neural distinguishers applied to PRIDE [34] and RC5 [35]. More recently, [30] proposed a cost-optimized neural distinguisher with reduced parameter complexity, making it suitable for resource-constrained environments.

In parallel, neural cryptanalysis [23,36–41] has advanced the application of deep learning-based attack models, such as Encryption Emulation (EE), Plaintext Recovery (PR), Key Recovery (KR), and Ciphertext Classification (CC), across a broad spectrum of cryptographic algorithms. In [36], optical cryptographic algorithms such as Double Random Phase Encoding (DRPE) [42] and Triple Random Phase Encoding (TRPE) [43] were shown to be vulnerable to PR attacks, enabling the recovery of original images from their encrypted forms. However, [37] observed that such recovery remained challenging when real-world, complex images were used as inputs. In [38], a KR attack was performed against classical ciphers—Caesar, Vigenère, and substitution—using a deep learning model structured as an encoder–decoder with Long Short-Term Memory (LSTM) [44] and attention mechanisms [45]. The work in [23] provided a comprehensive evaluation of block ciphers including DES [11], AES [24], simplified DES (SDES) [8], simplified AES (SAES) [25], and SPECK32/64 [26], through EE, PR, KR, and CC attacks using various model architectures such as fully connected neural networks, RNNs [33,44,46], and transformer-based models [47,48], with both textual and bit-array plaintexts across different numbers of encryption rounds.

In other studies, Harrison et al. [39] classified laptop keystrokes recorded via mobile microphones using CNNs [32] combined with attention mechanisms [45], demonstrating the potential of deep learning in side-channel attack scenarios. In [40], AES keys [24] were recovered from electromagnetic traces captured at varying distances from Bluetooth devices. Similarly, Alemerien et al. [41] detected side-channel attacks from execution traces using CNN-based models [45].

While deep learning-based cryptanalysis has demonstrated effectiveness across optical, classical, and symmetric block ciphers, its application to public-key cryptographic systems has thus far relied primarily on traditional analytical techniques. In this study, we focus on block ciphers and extend the scope of neural cryptanalysis by assessing vulnerabilities under both single-server and multi-server attack environments. To the best of our knowledge, this work represents the first systematic evaluation of block cipher security against deep learning-based attacks conducted in a federated, multi-server cryptanalytic framework.

3. Methods

3.1. Cryptographic Algorithms

Cryptographic algorithms are broadly classified into two categories—symmetric-key cryptography and asymmetric-key cryptography—based on the nature of the keys used for encryption and decryption [7,8]. In symmetric-key cryptography, the same key is shared between the sender and the receiver for both encryption and decryption processes. While this approach is computationally efficient, it presents significant challenges in secure key distribution and management, as both parties must agree upon and protect the shared secret key.

In contrast, asymmetric-key cryptography, also known as public-key cryptography, employs a pair of mathematically related keys: a public key for encryption and a private key for decryption. The public key can be openly distributed, allowing any sender to encrypt messages, while only the intended receiver—who holds the private key—can decrypt the ciphertext. This eliminates the need for secure key exchange, thereby simplifying key management in distributed environments.

In this study, we focus on symmetric block ciphers and evaluate the security of several representative algorithms, including the Data Encryption Standard (DES) [11], Simplified DES (SDES) [8], Advanced Encryption Standard (AES) [24], Simplified AES (SAES) [25], and SPECK [26]. These algorithms are analyzed under deep learning-based attacks, with the goal of comparing their relative vulnerabilities in both single-server and multi-server cryptanalytic environments.

3.1.1. Data Encryption Standard (DES)

The Data Encryption Standard (DES) [11] was created by IBM in the early 1970s as part of a research project on data security and was subsequently standardized in 1977 through collaboration with the U.S. National Bureau of Standards (now NIST) and the National Security Agency (NSA). It emerged as one of the most significant encryption algorithms and has underpinned numerous contemporary cryptographic systems. DES is a block cipher utilizing a 16-round Feistel network that processes 64-bit plaintext blocks with a 64-bit key, which includes 8 parity bits. The encryption starts with an Initial Permutation (IP), then divides the plaintext into two 32-bit halves, L_0 and R_0:

$$IP(P) = L_0 R_0 \tag{1}$$

During each round of DES, the right half of the data block, which is 32 bits, is expanded to 48 bits using an expansion permutation box. The expanded block is then XORed with the round's 48-bit subkey. The result is XORed with the left half to create the new right half, and the left half is copied from the previous right half:

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

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

After 16 rounds, the final output is generated by concatenating R_{16} and L_{16} and then using the inverse permutation:

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

Round keys are created by shifting and compressing the main key, removing the parity bits. Simplified DES (SDES) [8] is a simplified variant that retains the Feistel structure but uses 16-bit plaintext and 16-bit key with only four rounds. It lacks the initial and final permutations and is primarily intended for educational or experimental use [8].

3.1.2. Advanced Encryption Standard (AES)

The US National Institute of Standards and Technology (NIST) established the Advanced Encryption Standard (AES) [24] in 2001 following an open competition to find a successor to DES. The chosen algorithm, Rijndael, was created by Joan Daemen and Vincent Rijmen and has since become the global standard for symmetric encryption. AES is a block cipher which performs on 128-bit plaintext blocks and is based on a Substitution-Permutation Network (SPN). There are 10, 12, or 14 rounds applied, depending on the size of the key, which can be either 128, 192, or 256 bits. Four transformations are included in each round: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The plaintext is encrypted using a 4×4 byte matrix called the state.

The SubBytes step conducts non-linear substitution using an S-box that maps each byte individually. ShiftRows cyclically shifts the state's rows to the left: the second, third, and fourth rows are shifted one, two, and three positions, respectively. MixColumns performs a linear transformation by multiplying each column in the state with a fixed 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 $c_{i,j}$ is the input byte at position (i, j) of the state, and $d_{i,j}$ is the output byte following MixColumns. Add RoundKey bitwise XORs the current state with a round-specific key obtained from the primary key. The first round contains only AddRoundKey, while the final round skips the MixColumns stage. Each round generates keys by means of a key expansion process comprising XOR operations, rotation, and substitution.

Operating with a 16-bit key across two rounds, a compact form of Simplified AES (SAES) [25] represents 16-bit plaintext using a 2×2 matrix of 4-bit nibbles. Although the architecture reflects AES, its smaller block size and simplified operations make it appropriate for lightweight testing or teaching.

3.1.3. SPECK

SPECK [26] is a light block cipher that was invented by the U.S. National Security Agency (NSA) in 2013. It is part of a family of algorithms designed for embedded systems and the Internet of Things (IoT). Its design prioritizes software simplicity and efficiency by utilizing basic operations such as modular addition, bitwise XOR, and rotation—collectively known as ARX. SPECK operates on two-word blocks and supports a variety of block/key size configurations. The SPECK32/64 version used in this study encrypts 32-bit plaintext blocks (two 16-bit words) using a 64-bit key (four 16-bit words) in 22 rounds. The cipher structure is based on a Feistel-style arrangement, with ARX operations performed in each round. For a given round r, the left and right input words L_{r-1} and R_{r-1} are converted 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 << denote right and left circular rotations, $\oplus$ is bitwise XOR, and $\boxplus$ is addition modulo 2^{16}. The SPECK32/64 rotation constants are $r_1 = 7$ and $r_2 = 2$. Each round employs a subkey k_r, which is derived from the original key via a key scheduling process that also employs the ARX structure. The combination of minimal logic and flexible parameterization makes SPECK appealing for cryptographic applications in low-power and memory-constrained devices.

3.2. Deep Learning-Based Attacks

Deep learning-based neural cryptanalysis enables the evaluation of cryptographic algorithm strength by leveraging data-driven attack models. In this study, we perform two representative forms of such attacks—Encryption Emulation (EE) and Plaintext Recovery (PR)—using both fully connected neural networks and Recurrent Neural Networks (RNNs) as attack models. These attacks are conducted under two distinct operational settings: a centralized single-server environment and a distributed multi-server environment, in order to assess the scalability and practicality of deep learning-based cryptanalysis.

3.2.1. Attack Environments

In the single-server attack environment, all stages of the attack process—data collection, model training, and inference—are executed on a single centralized server. Specifically, the attacker collects plaintext–ciphertext pairs within the same session, where a fixed encryption key is used, and subsequently trains a deep learning model through centralized learning. As illustrated in Figure 1, this environment follows a straightforward three-step procedure:

(1) Data Collection: The central server gathers plaintext and corresponding ciphertext pairs;
(2) Attack Model Training: A deep learning model is trained to predict either ciphertexts (in EE attacks) or plaintexts (in PR attacks);
(3) Attack Execution: The trained model is applied to new inputs to carry out the designated attack.

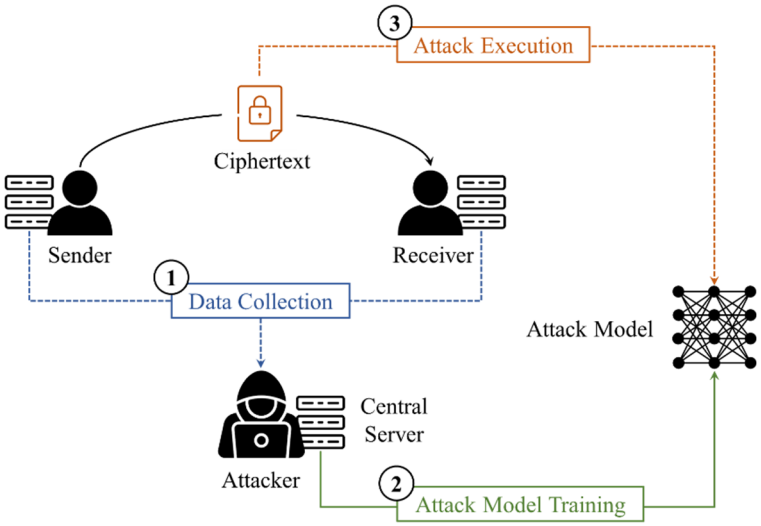

Figure 1. Overview of the attack process in the single-server environment, where data collection, model training, and attack execution are performed on a centralized server.

In contrast, the multi-server attack environment reflects a more decentralized and scalable setting. Here, multiple edge servers simultaneously collect plaintext–ciphertext pairs within independent sessions. These local models are trained using federated learning, allowing collaborative model improvement without raw data sharing. As shown in Figure 2, the multi-server framework consists of six key steps:

(1) Data Collection: Each edge server gathers its own local dataset;
(2) Global Model Distribution: The central coordinating server distributes the current global model weights to all edge servers;
(3) Local Model Training: Each server independently trains a local model using its local data and the received initial weights;

(4) Model Aggregation: Trained local weights are transmitted back to the central server;
(5) Global Model Update: The central server aggregates the received local weights to update the global model;
(6) Attack Execution: After sufficient global training rounds, the finalized global model is used to perform the designated attack.

This federated setup simulates real-world adversarial scenarios where attackers may not have access to a powerful centralized resource but can exploit a network of weaker distributed devices. The full training procedure, including weight updates and aggregation logic, is provided in Algorithm 1.

In both attack environments, we assume that all plaintext-ciphertext pairs are encrypted using a fixed secret key throughout the data collection and training phases. This assumption aligns with the 'Known Plaintext Attack (KPA)' scenario in classical cryptanalysis, where the adversary has access to plaintext-ciphertext pairs encrypted under the same unknown key. While this may appear restrictive in some distributed settings, it reflects realistic attack scenarios where

(1) Session-based encryption protocols maintain a consistent key for the duration of a communication session (e.g., TLS session keys, VPN tunnels);
(2) IoT and edge computing deployments often employ shared group keys among multiple devices for efficiency and scalability;
(3) Security evaluation standards require assessing algorithm vulnerability under consistent keying to isolate structural weaknesses from key management issues.

Moreover, the primary objective of this study is to evaluate the intrinsic cryptographic strength of block cipher algorithms against deep learning-based attacks, rather than to address key distribution or multi-key scenarios, which constitute orthogonal research problems in key management and access control. By fixing the encryption key, we can systematically measure how round complexity and federated training dynamics affect the attack model's ability to learn the cipher's transformation function. This controlled setting enables direct performance comparison between centralized and distributed attack environments and provides insights into the fundamental resilience of cipher structures against neural cryptanalysis.

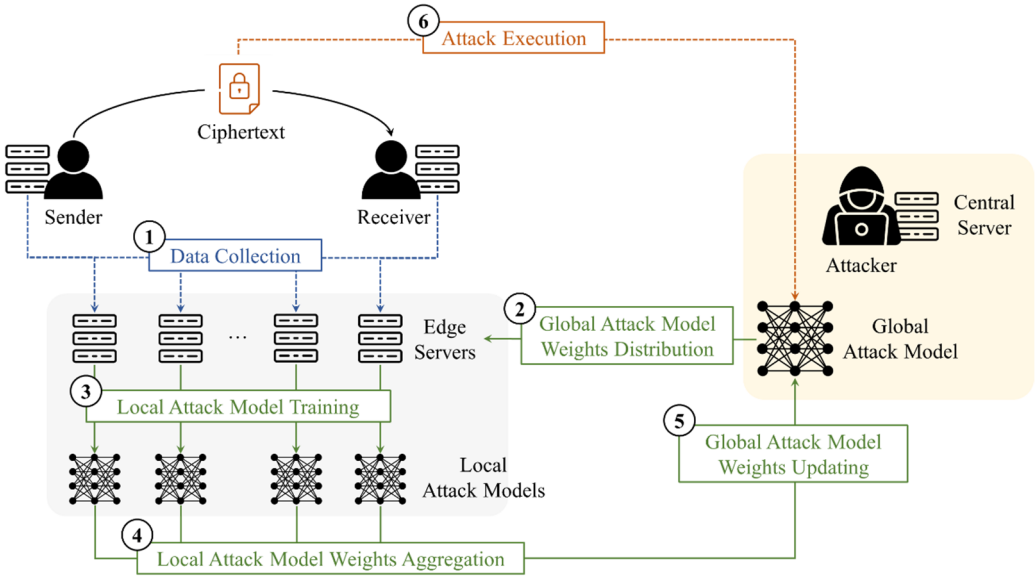

Figure 2. Overview of the federated attack process in the multi-server environment, where distributed edge servers collaboratively train a global model for cryptanalysis.

Algorithm 1. Process of Training the Attack Model in the Multi-Server Environment

Inputs:
Initial weights of global attack model w_g^0;
Pairs of plaintext and corresponding ciphertext in edge servers
$D = \{D_1, D_2, \ldots, D_{K-1}, D_K\}$.
Parameters:
Total number of global rounds T;
Total number of edge servers K;
Weights of global attack model for t^{th} global round w_g^t;
Weights of local attack model for t^{th} global round w_k^t;
Total number of pairs of plaintext and corresponding ciphertext for all edge servers N;
Total number of pairs of plaintext and corresponding ciphertext for k edge server n_k;
Total number of local epochs E, Mini-batch for k edge server b_k;
Learning rate η.
Outputs:
Final weights of global attack model w_g^T.

for global round $t = 1, 2, \ldots, T$ **do**
  for edge server $k = 1, 2, \ldots, K$ **do**
    $w_k^t \leftarrow$ **LocalAttackModelTraining** (w_g^{t-1})
  end
  $w_g^t \leftarrow \sum_{k=1}^{K} \frac{n_k}{N} w_k^t$
end
return w_g^T

Function LocalAttackModelTraining (w_g^{t-1}):
$w_k^t \leftarrow w_g^{t-1}$
for local epoch $e = 1, 2, \ldots, E$ **do**
  for batch $b_k \in D_k$ **do**
    Compute CrossEntropyLoss L
    $w_k^t \leftarrow w_k^t - \eta \, \Delta L$
  end
end
return w_k^t

3.2.2. Attack Models

In this study, we investigate two types of deep learning-based cryptanalytic attacks: Encryption Emulation (EE) and Plaintext Recovery (PR). The EE attack aims to generate ciphertexts directly from given plaintexts, while the PR attack attempts to recover plaintexts from observed ciphertexts. Accordingly, the deep learning model for EE takes plaintexts as input and is trained to produce the corresponding ciphertexts. In contrast, the model for PR uses ciphertexts as input and is trained to reconstruct the original plaintexts.

While EE does not directly compromise confidentiality in the traditional sense—as it requires plaintext input and does not recover keys or decrypt messages—successful emulation indicates that the cipher's transformation can be reverse-engineered and reproduced without knowledge of the secret key. This capability poses security concerns in scenarios where attackers can observe plaintext-ciphertext pairs and subsequently generate valid ciphertexts for arbitrary plaintexts, potentially enabling forgery attacks or facilitating chosen-plaintext attack strategies. In contrast, the PR approach attempts to recover plain-

texts from observed ciphertexts, representing a direct confidentiality breach that aligns with classical cryptanalytic threat models.

To implement these attacks, we employ two neural architectures: the fully connected neural network (FCNN) and the Recurrent Neural Network (RNN) [33], as illustrated in Figure 3. Due to the inherent confusion and diffusion properties of modern block ciphers, ciphertexts generally lack straightforward bitwise correlations. Consequently, many prior studies in deep learning-based cryptanalysis have relied on FCNNs to capture global, non-linear mappings between plaintext and ciphertext.

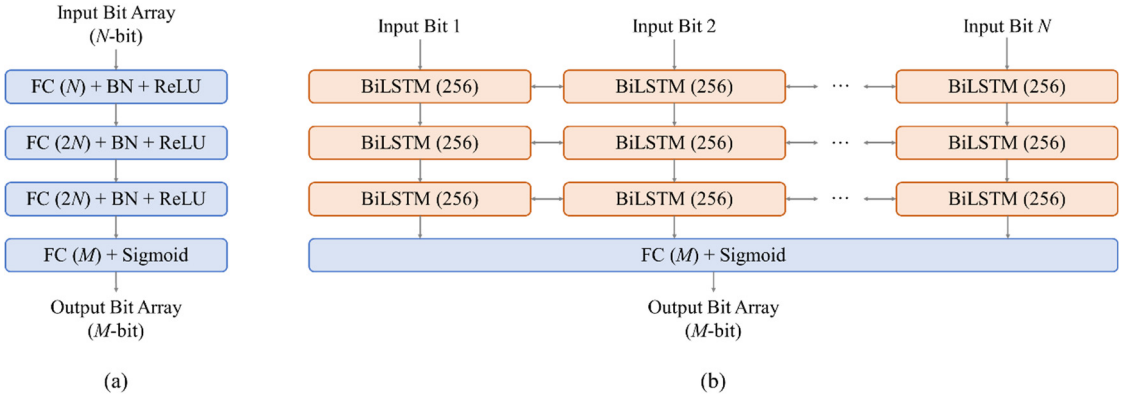

Figure 3. Architectures of the attack models used in this study: (**a**) fully connected neural network (FCNN); (**b**) recurrent neural network (RNN) based on bidirectional Long Short-Term Memory (BiLSTM).

We implement an RNN-based model using Bidirectional Long Short-Term Memory (BiLSTM) units. This model architecture includes three stacked BiLSTM layers, each with 256 hidden units, followed by a single fully connected output layer with sigmoid activation. Unlike fully connected networks that process all input bits simultaneously through parallel transformations, the BiLSTM processes the bit array sequentially, allowing the model to learn position-specific mappings between plaintext and ciphertext bits. The bidirectional architecture further enables the model to incorporate information from both preceding and succeeding bit positions when computing transformations for each bit, potentially enhancing its ability to approximate the complex, interdependent bit-level operations performed by block cipher round functions.

The FCNN-based model used in this work comprises four fully connected layers. Given an input vector of size N, the first three layers have N, $2N$, and $2N$ neurons, respectively, followed by a final output layer matching the required output size (depending on EE or PR). Each layer is followed by batch normalization and ReLU activation, except for the final layer, which uses a sigmoid activation to enable bitwise prediction.

3.2.3. Statistical Significance Testing for Attack Performance

To rigorously determine whether observed BAP_{avg} values represent genuine cryptanalytic success or merely random guessing, we employ formal statistical hypothesis testing. Under the null hypothesis, the attack model performs no better than random bit prediction, yielding an expected BAP_{avg} of 0.5.

We use a two-tailed binomial test to evaluate whether the observed accuracy significantly deviates from this baseline. For each cipher configuration, we calculate the 99% confidence interval for BAP_{avg} based on the number of test samples (N = 32,768) and individual bit predictions. An attack is considered "statistically successful" if the confidence interval does not contain 0.5 and the p-value is below 0.01. Conversely, when $BAP_{avg} \approx 0.5$

and falls within the confidence interval around random guessing, we conclude that the cipher resists the attack under the tested configuration.

4. Experiments

4.1. Dataset and Setup

The plaintexts used in this study were randomly generated as bit arrays, with lengths matched to the block sizes defined by each cryptographic algorithm. These plaintexts were produced using a custom random bit generator that accepts the desired bit length as a parameter and outputs uniformly distributed integers of that length. Each plaintext was then encrypted using one of the selected block cipher algorithms—Data Encryption Standard (DES), Simplified DES (SDES), Advanced Encryption Standard (AES-128), Simplified AES (SAES), or SPECK32/64—with varying numbers of round iterations (1, 2, 3, or 4). Representative samples of plaintext–ciphertext pairs for each cipher are summarized in Table 1.

Table 1. Representative samples of plaintext–ciphertext pairs generated by each cryptographic algorithm used in the experiments.

Cryptographic Algorithms	Key Size	Plaintext Size	Plaintext	Ciphertext
SDES	16-bit	16-bit	10110010 11101110	11101110 00110011
SAES	16-bit	16-bit	00100111 10001110	10110110 01001011
DES	64-bit	64-bit	11111101 00100011 11010010 10001010 01101101 11010111 10010011 11100100	11111000 00100011 10000110 10011110 01101101 10000111 11000111 11110000
AES-128	128-bit	128-bit	10111101 11010110 01000000 11111011 00000110 01100111 00011010 11010001 00011100 10000000 00110001 01111111 10100011 10110001 01111001 10011101	00111110 00110010 10101001 00010010 10110000 01011101 10111010 01100011 10011011 01110100 00011011 10101000 11010111 11101100 00101110 01111101
SPECK32/64	64-bit	32-bit	11010110 00001100 10000100 00111110	11001110 10011011 11011110 01100001

Random plaintexts represent a conservative and rigorous testing scenario for several reasons. First, block ciphers are explicitly designed to produce uniformly random-looking outputs regardless of input structure, meaning that any weakness exploitable with random inputs would equally apply to structured data. Second, random plaintexts eliminate potential biases that could arise from structured data patterns, ensuring that observed attack performance reflects genuine cipher weaknesses rather than artifacts of input regularity. Third, in real-world deployments, plaintexts are often compressed, encrypted in counter mode, or otherwise preprocessed in ways that eliminate apparent structure before encryption, making random bit sequences a reasonable approximation of actual operational conditions. The use of random inputs in this study provides a robust foundation for evaluating cipher security under worst-case assumptions aligned with cryptographic best practices.

To ensure consistency and fairness across experiments, the total number of plaintext–ciphertext pairs used for training and testing was fixed at $N = 2^{20}$ (=1,048,576) and 2^{15} (=32,768), respectively, for both the single-server and multi-server attack environments. In

the multi-server setup, these training pairs were distributed across edge servers depending on the total number of participating nodes K. Specifically, for $K = 2^1$ (=2), 2^2 (=4), 2^3 (=8), 2^4 (=16), and 2^5 (=32) edge servers, each server received $n_k = 2^{19}$ (=524,288), 2^{18} (=262,144), 2^{17} (=131,072), 2^{16} (=65,536), and 2^{15} (=32,768) training pairs, respectively, while maintaining the overall training size constant. These configurations are summarized in Table 2.

Table 2. Summary of training and testing data volumes and the number of edge servers used in the multi-server attack environment.

Attack Environment	The Number of Training Pairs	The Number of Testing Pairs	The Number of Edge Servers	The Number of Training Pairs for Each Edge Server
Single-Server	2^{20} (=1,048,576)	2^{15} (=32,768)	-	-
Multi-Server	2^{20} (=1,048,576)	2^{15} (=32,768)	2^1 (=2)	2^{19} (=524,288)
	2^{20} (=1,048,576)	2^{15} (=32,768)	2^2 (=4)	2^{18} (=262,144)
	2^{20} (=1,048,576)	2^{15} (=32,768)	2^3 (=8)	2^{17} (=131,072)
	2^{20} (=1,048,576)	2^{15} (=32,768)	2^4 (=16)	2^{16} (=65,536)
	2^{20} (=1,048,576)	2^{15} (=32,768)	2^5 (=32)	2^{15} (=32,768)

In the federated learning implementation, we employed a full client participation strategy where all K edge servers participate in every global training round, ensuring stable convergence and reproducibility across experiments. The model aggregation follows the standard FedAvg algorithm as described in Algorithm 1. Since all clients possess equal-sized local datasets ($n_1 = n_2 = \ldots = n_k = N/K$, see Table 2), the weighted aggregation reduces to uniform averaging across all participating edge servers.

Each global round consists of three phases: (1) broadcasting the current global model w_g^{t-1} to all clients, (2) local training for $E = 10$ epochs on each client's dataset using mini-batch size $b_k = 4096$ and learning rate $\eta = 0.0001$, and (3) aggregating the updated local weights to form the new global model. The total number of global rounds was set to $T = 10$, balancing convergence quality with computational cost. Client training was performed sequentially rather than in parallel to ensure deterministic behavior and facilitate reproducibility, as simultaneous multi-process encryption operations introduced process management complexity without affecting the final model performance. This synchronous, full-participation approach with uniform weighting represents the most straightforward federated learning scenario and serves as a baseline for evaluating the scalability of neural cryptanalysis in distributed environments.

All experiments were conducted on a server equipped with an Intel Xeon Silver 4214 processor and an NVIDIA RTX A5000 GPU. For both single-server and multi-server environments, the attack models were trained using binary cross-entropy as the loss function and the Adam optimizer.

4.2. Results and Analysis

The vulnerabilities of the cryptographic algorithms against deep learning-based attacks, especially Encryption Emulation (EE) and Plaintext Recovery (PR) attacks, was evaluated by using Bit Accuracy Probability (BAP) [49] and average BAP (BAP_{avg}), which can be calculated as follows:

$$o_i^n = \begin{cases} 0, & if\ \ o_i^n \leq 0.5 \\ 1, & otherwise \end{cases}, \ BAP_i = \frac{\sum_{n=1}^{N} XNOR\left(o_i^n, g_i^n\right)}{N} \tag{8}$$

$$BAP_{avg} = \frac{\sum_{i=1}^{L} BAP_i}{L} \quad (9)$$

where BAP_i and BAP_{avg} represent BAP for the i^{th} bit in predicted bit arrays and average BAP for predicted bit arrays, respectively. And o_i^n and g_i^n are the i^{th} bit in the n^{th} predicted bit array and the i^{th} bit in the n^{th} ground truth bit array, respectively, and N is the total number of testing pairs. And L is the output length according to the type of attack.

4.2.1. Results in Single-Server Attack Environment

Our previous study [23] evaluated the vulnerability of five representative block ciphers—SDES, SAES, DES, AES-128, and SPECK32/64—under Encryption Emulation (EE) and Plaintext Recovery (PR) attacks in a centralized single-server environment. The analysis focused on two key factors: the number of round functions and the choice of attack model architecture.

The results showed that increasing the number of rounds generally led to a decrease in average bit-wise accuracy (BAP_{avg}), indicating stronger resistance against deep learning-based attacks. For instance, in both DES and SPECK32/64, BAP_{avg} values dropped significantly from nearly 1.0 with a single round to approximately 0.5 when full-round encryption was applied—across both EE and PR attack settings. Notably, AES-128 exhibited consistently low BAP_{avg} values close to 0.5 regardless of the number of rounds, suggesting high resilience even in reduced-round configurations.

Regarding model architectures, the BiLSTM-based recurrent neural network slightly outperformed the fully connected model in both attack scenarios. While the improvements in BAP_{avg} were modest, the RNN-based model demonstrated superior performance across several block ciphers---particularly SDES, SAES, DES, and SPECK32/64. The performance advantage of BiLSTM can be attributed to its sequential processing mechanism, which enables the model to learn position-dependent transformations for each bit location in the plaintext and ciphertext arrays. By processing bits sequentially and incorporating bidirectional context, the BiLSTM architecture may better approximate the position-sensitive substitution and permutation operations inherent in block cipher designs, particularly in reduced-round configurations where such structural patterns remain more discernible. This trend held even for AES-128, where BAP_{avg} values consistently remained low for both model types, thereby confirming its robustness against neural cryptanalysis regardless of the attack model architecture employed.

4.2.2. Result Comparison in Single-Server and Multi-Server Attack Environments

To assess the feasibility and effectiveness of distributed cryptanalysis, we compared the average bit-wise accuracy (BAP_{avg}) of Encryption Emulation (EE) and Plaintext Recovery (PR) attacks using a Recurrent Neural Network (RNN) model across both single-server and multi-server environments. The evaluation was conducted on five block cipher algorithms—SDES, SAES, DES, AES-128, and SPECK32/64—using 1 to 4 rounds of encryption functions. In the multi-server setting, the number of edge servers was varied as 2^1 (=2), 2^2 (=4), 2^3 (=8), 2^4 (=16), and 2^5 (=32), with each server receiving an equal fraction of the total training data (2^{20} pairs), as detailed in Section 4.1.

For SDES and SAES, both algorithms maintained high BAP_{avg} values across all conditions. As shown in Tables 3 and 4, the performance remained nearly perfect in EE and PR attacks, even as the number of edge servers increased. Notably, in the case of SDES with 4 rounds and 32 edge servers, there was a slight performance decline (BAP_{avg} = 0.9336 for EE and 0.9261 for PR), yet the model still achieved high prediction accuracy. SAES maintained an average BAP_{avg} of 1.0 under 2-round settings, regardless of the attack environment.

Table 3. Comparison of average Bit Accuracy Probability (BAP_{avg}) for Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on Simplified Data Encryption Standard (SDES) across different round configurations and attack environments. * $p < 0.01$ (statistically significant attack success).

Number of Edge Server		Encryption Emulation (EE)				Plaintext Recovery (PR)			
		1 Round	2R	3R	4R (Full)	1 Round	2R	3R	4R (Full)
Single-Server		1.0 *	1.0 *	1.0 *	1.0 *	1.0 *	1.0 *	1.0 *	1.0 *
Multi-Server	2^1	1.0 *	1.0 *	1.0 *	1.0 *	1.0 *	1.0 *	1.0 *	1.0 *
	2^2	1.0 *	1.0 *	1.0 *	1.0 *	1.0 *	1.0 *	1.0 *	0.9999 *
	2^3	1.0 *	1.0 *	1.0 *	0.9996 *	1.0 *	1.0 *	1.0 *	0.9993 *
	2^4	1.0 *	1.0 *	1.0 *	0.9851 *	1.0 *	1.0 *	1.0 *	0.9803 *
	2^5	1.0 *	1.0 *	0.9999 *	0.9336 *	1.0 *	1.0 *	0.9999 *	0.9261 *

Table 4. Comparison of average Bit Accuracy Probability (BAP_{avg}) for Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on Simplified Advanced Encryption Standard (SAES) across different round configurations and attack environments. * $p < 0.01$ (statistically significant attack success).

Number of Edge Server		Encryption Emulation (EE)		Plaintext Recovery (PR)	
		1 Round	2R (Full)	1 Round	2R (Full)
Single-Server		1.0 *	1.0 *	1.0 *	1.0 *
Multi-Server	2^1	1.0 *	1.0 *	1.0 *	1.0 *
	2^2	1.0 *	1.0 *	1.0 *	1.0 *
	2^3	1.0 *	1.0 *	1.0 *	1.0 *
	2^4	1.0 *	1.0 *	1.0 *	1.0 *
	2^5	1.0 *	1.0 *	1.0 *	1.0 *

DES exhibited a more pronounced sensitivity to both round complexity and distribution scale. As shown in Table 5, when configured with only one round, DES remained vulnerable across both environments. However, as the number of edge servers increased, a slight degradation in performance was observed in the multi-server setting (e.g., BAP_{avg} dropped to 0.9219 in EE and 0.8622 in PR with 32 servers). Once the number of rounds reached two or more, the BAP_{avg} consistently converged toward 0.5, indicating that the attack model failed to extract meaningful patterns—effectively reducing its predictive capability to that of random guessing. These findings confirm that increasing round complexity significantly enhances the robustness of DES, particularly under distributed attack scenarios.

Table 5. Comparison of average Bit Accuracy Probability (BAP_{avg}) for Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on the Data Encryption Standard (DES) across different round configurations and attack environments. * $p < 0.01$ (statistically significant attack success), † $p > 0.05$ (statistically indistinguishable from random guessing, indicating cipher security).

Number of Edge Server		Encryption Emulation (EE)				Plaintext Recovery (PR)			
		1 Round	2R	3R	4R	1 Round	2R	3R	4R
Single-Server		0.9998 *	0.4998 †	0.5000 †	0.5000 †	0.9997 *	0.5001 †	0.4998 †	0.5000 †
Multi-Server	2^1	0.9532 *	0.5244 *	0.4998 †	0.5001 †	0.9531 *	0.5001 †	0.5004 †	0.5001 †
	2^2	0.9529 *	0.4996 †	0.4999 †	0.5001 †	0.9531 *	0.5417 *	0.4997 †	0.4997 †
	2^3	0.9549 *	0.4999 †	0.4993 †	0.5001 †	0.9532 *	0.5000 †	0.5000 †	0.5000 †
	2^4	0.9219 *	0.4996 †	0.5007 †	0.4998 †	0.9295 *	0.4995 †	0.5001 †	0.4997 †
	2^5	0.8622 *	0.5000 †	0.5000 †	0.4998 †	0.8301 *	0.4998 †	0.4999 †	0.5002 †

Among all ciphers, AES-128 demonstrated the highest level of resilience. As shown in Table 6, AES-128 consistently yielded BAP_{avg} values near 0.5 across all conditions—irrespective of the number of rounds, attack type, or deployment environment. This suggests that the attack model was unable to learn any exploitable structure in the ciphertexts, even with powerful recurrent architecture and federated training. Compared to SAES, which shares a similar structure but with fewer rounds and a smaller key size, AES-128's strong resistance is attributed to its greater algorithmic complexity and cryptographic strength. These results reinforce the status of AES-128 as a highly secure block cipher, even in the face of distributed, data-driven attacks.

Table 6. Comparison of average Bit Accuracy Probability (BAP_{avg}) for Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on Advanced Encryption Standard (AES-128) across different round configurations and attack environments. † $p > 0.05$ (statistically indistinguishable from random guessing, indicating cipher security).

Number of Edge Server		Encryption Emulation (EE)				Plaintext Recovery (PR)			
		1 Round	2R	3R	4R	1 Round	2R	3R	4R
Single-Server		0.5000 †	0.5004 †	0.5003 †	0.5000 †	0.4998 †	0.4998 †	0.5002 †	0.5004 †
Multi-Server	2^1	0.4999 †	0.4997 †	0.5003 †	0.4996 †	0.4994 †	0.4999 †	0.5001 †	0.4999 †
	2^2	0.4998 †	0.4997 †	0.5005 †	0.5003 †	0.4997 †	0.4996 †	0.4997 †	0.4999 †
	2^3	0.4998 †	0.4999 †	0.4997 †	0.4998 †	0.4998 †	0.4999 †	0.5001 †	0.4996 †
	2^4	0.5000 †	0.5001 †	0.5000 †	0.4998 †	0.4999 †	0.5000 †	0.4999 †	0.4999 †
	2^5	0.4995 †	0.4998 †	0.4997 †	0.5002 †	0.4999 †	0.4999 †	0.4997 †	0.4999 †

SPECK32/64 showed patterns similar to DES. As indicated in Table 7, when using only one round, attack models achieved relatively high BAP_{avg} values, particularly in the centralized setting. However, performance declined as the number of edge servers increased—dropping to 0.6312 (EE) and 0.7783 (PR) with 32 servers. After two or more rounds, BAP_{avg} values across all settings approached 0.5, indicating minimal predictability. These results suggest that SPECK is vulnerable under shallow configurations but becomes increasingly robust with higher round complexity, particularly when model coordination becomes more challenging in distributed training.

Table 7. Comparison of average Bit Accuracy Probability (BAP_{avg}) for Encryption Emulation (EE) and Plaintext Recovery (PR) attacks on SPECK32/64 across different round configurations and attack environments. * $p < 0.01$ (statistically significant attack success), † $p > 0.05$ (statistically indistinguishable from random guessing, indicating cipher security).

Number of Edge Server		Encryption Emulation (EE)				Plaintext Recovery (PR)			
		1 Round	2R	3R	4R	1 Round	2R	3R	4R
Single-Server		1.0 *	0.4997 †	0.4999 †	0.5008 †	0.9999 *	0.4993 †	0.4999 †	0.4991 †
Multi-Server	2^1	1.0 *	0.4999 †	0.4999 †	0.5012 †	0.9999 *	0.5000 †	0.4997 †	0.4995 †
	2^2	1.0 *	0.4996 †	0.5002 †	0.4995 †	1.0 *	0.4992 †	0.4997 †	0.4994 †
	2^3	0.9688 *	0.4998 †	0.4999 †	0.5001 †	0.9999 *	0.5007 †	0.5000 †	0.4999 †
	2^4	0.8001 *	0.4995 †	0.4998 †	0.5001 †	1.0 *	0.5004 †	0.4999 †	0.5002 †
	2^5	0.6312 *	0.4996 †	0.5007 †	0.5005 †	0.7783 *	0.4997 †	0.5005 †	0.5004 †

Collectively, these findings demonstrate that distributed neural cryptanalysis can achieve performance comparable to centralized attacks, even when model training is partitioned across multiple servers. This validates the scalability and practicality of the proposed multi-server attack framework and confirms that federated, deep learning-based

attacks represent a realistic threat, particularly for lightweight ciphers with reduced-round configurations. Moreover, the use of local edge servers enables attackers to collect and share a larger volume of training data from distributed environments, potentially increasing the reach and efficiency of cryptanalysis in practice. While a slight decline in attack performance was observed as the number of edge servers increased, this effect is likely due to challenges in model synchronization and data heterogeneity under federated learning. Future work will focus on mitigating these limitations through adaptive aggregation, client selection, and communication-efficient federated strategies, further enhancing the robustness and efficiency of distributed neural cryptanalysis.

4.2.3. Scalability Analysis: Training Efficiency and Computational Overhead

To comprehensively evaluate the scalability of the proposed federated cryptanalysis framework, we analyzed the computational efficiency of the distributed training process by measuring two key timing metrics: aggregation time and global round training time. The aggregation time represents the duration required by the central server to compute the weighted average of local model parameters received from all edge servers, as described in Algorithm 1 (line 5). This metric reflects the computational cost of the FedAvg aggregation step, which is a fundamental operation in federated learning. The global round training time measures the total wall-clock time required to complete one full round of federated training, encompassing local model training across all edge servers (executed sequentially in our simulation), weight aggregation at the central server, and subsequent model evaluation on the test dataset. These metrics provide critical insights into the practical feasibility of deploying distributed neural cryptanalysis in real-world adversarial scenarios, particularly in resource-constrained environments where computational efficiency is paramount.

Table 8 presents the aggregation time and global round training time for 1-round Encryption Emulation (EE) attacks across five block ciphers—TDES, SAES, DES, AES, and SPECK—using the BiLSTM model architecture with varying numbers of edge servers (2^1 to 2^5). The aggregation time increases slightly as the number of edge servers grows, ranging from approximately 0.0013–0.0015 s with 2 servers to 0.0051–0.0087 s with 32 servers. This modest increase is attributed to the linear scaling of the weighted averaging operation over a larger number of local weight sets. However, these aggregation times remain negligible compared to the overall training duration, accounting for less than 0.01% of the total round time in all configurations. Notably, the global round training time varies substantially across different ciphers, reflecting differences in model convergence behavior and the inherent complexity of learning cryptographic transformations. Lightweight ciphers such as TDES, SAES, and SPECK require approximately 290–320 s per round, whereas more complex ciphers like DES and AES demand 740–750 s and 870–890 s, respectively. These findings collectively demonstrate that the proposed federated cryptanalysis framework achieves efficient scalability, maintaining consistent training performance even as the number of distributed edge servers increases.

Table 8. Comparison of aggregation time (s) and global round training time (s) for 1-round Encryption Emulation (EE) attacks across different edge server configurations in the multi-server environment.

Number of Edge Server		SDES	SAES	DES	AES-128	SPECK32/64
Multi-Server	2^1	0.0013/301.35	0.0013/288.81	0.0015/741.84	0.0014/880.12	0.0014/294.42
	2^2	0.0016/302.62	0.0021/296.61	0.0018/744.42	0.0016/874.43	0.0017/295.57
	2^3	0.0027/310.65	0.0027/301.01	0.0023/745.33	0.0021/886.80	0.0027/307.34
	2^4	0.0034/325.96	0.0040/304.37	0.0031/752.41	0.0031/888.10	0.0034/310.97
	2^5	0.0063/319.47	0.0063/298.44	0.0087/745.94	0.0051/869.66	0.0055/312.42

4.2.4. Comparison of Attack Model Architectures

To further investigate the impact of model architecture on cryptanalytic effectiveness, we conducted a direct comparison between FCNN and BiLSTM models under identical experimental conditions. Table 9 presents the attack performance of both architectures across five block ciphers with varying round configurations, evaluated in a federated environment with eight (2^3) edge servers. From a statistical significance perspective, both models achieve comparable results: when either architecture attains statistically significant attack success ($p < 0.01$), the alternative architecture similarly demonstrates significant performance, and conversely, when one model fails to exceed random guessing thresholds, the other exhibits equivalent limitations. This statistical parity suggests that both architectures possess sufficient capacity to exploit vulnerabilities in reduced-round ciphers. However, a closer examination of the raw BAP_avg values reveals a consistent performance advantage for the BiLSTM architecture, particularly in cases where cryptanalytic attacks succeed.

Table 9. Comparison of average Bit Accuracy Probability (BAP_{avg}) between FCNN and BiLSTM architectures for Encryption Emulation (EE) and Plaintext Recovery (PR) attacks in the multi-server environment with 2^3 edge servers. * $p < 0.01$ (statistically significant attack success), † $p > 0.05$ (statistically indistinguishable from random guessing, indicating cipher security).

Number of Edge Server		Encryption Emulation (EE)		Plaintext Recovery (PR)	
		FCNN	**BiLSTM**	**FCNN**	**BiLSTM**
SDES	1R	0.9687 *	1.0 *	0.9483 *	1.0 *
	2R	0.7478 *	1.0 *	0.7478 *	1.0 *
	3R	0.6253 *	1.0 *	0.6217 *	1.0 *
	4R	0.5940 *	0.9996 *	0.5919 *	0.9993 *
SAES	1R	0.8331 *	1.0 *	0.8644 *	1.0 *
	2R	0.8433 *	1.0 *	0.7497 *	1.0 *
DES	1R	0.8273 *	0.9549 *	0.8126 *	0.9532
	2R	0.5002 †	0.4999 †	0.5000 †	0.5000 †
	3R	0.4999 †	0.4993 †	0.5000 †	0.5000 †
	4R	0.4995 †	0.5001 †	0.5006 †	0.5000 †
AES-128	1R	0.4997 †	0.4998 †	0.5000 †	0.4998 †
	2R	0.4998 †	0.4999 †	0.5003 †	0.4999 †
	3R	0.5001 †	0.4997 †	0.5002 †	0.5001 †
	4R	0.5000 †	0.4998 †	0.4995 †	0.4996 †
SPECK32/64	1R	0.8094 *	0.9688 *	0.9239 *	0.9999 *
	2R	0.4999 †	0.4998 †	0.5001 †	0.5007 †
	3R	0.5000 †	0.4999 †	0.4997 †	0.5000 †
	4R	0.5000 †	0.5001 †	0.5000 †	0.4999 †

The performance differences become evident when examining specific cipher configurations. For instance, in 1-round attacks on SDES, the BiLSTM model achieves perfect accuracy (BAP_avg = 1.0) for both EE and PR attacks, whereas the FCNN model, while still demonstrating strong performance, attains lower accuracy values (BAP_avg = 0.9687 for EE and 0.9483 for PR). Similar patterns emerge across other vulnerable configurations: for 1-round SPECK32/64 EE attacks, BiLSTM reaches BAP_avg = 0.9688 compared to FCNN's 0.8094, and for 2-round DES EE attacks, BiLSTM achieves 0.4999 while FCNN obtains 0.8273. These consistent improvements demonstrate that the BiLSTM's sequential processing mechanism and bidirectional architecture enable more effective learning of the position-dependent transformations and interdependent bit-level operations that characterize block cipher encryption. By processing plaintext and ciphertext bit arrays sequentially while incorporating contextual information from both directions, the BiLSTM

architecture more accurately approximates the complex mappings between input and output, particularly in reduced-round scenarios where the cipher's structural properties remain partially intact.

5. Conclusions

In this study, we conducted a comprehensive analysis of the vulnerability of block cipher algorithms—SDES, SAES, DES, AES-128, and SPECK32/64—against deep learning-based cryptanalysis in both single-server and multi-server environments. Specifically, we implemented two neural attack strategies: Encryption Emulation (EE) and Plaintext Recovery (PR). The EE attack models were trained to generate ciphertexts from input plaintexts, whereas the PR models learned to recover plaintexts from ciphertexts. For model architectures, we employed both fully connected neural networks and Recurrent Neural Networks (RNNs) with bidirectional Long Short-Term Memory (BiLSTM) layers to evaluate performance across varying cipher structures.

In the single-server attack environment, all data collection and model training were centralized. In contrast, the multi-server attack framework simulated a distributed adversarial scenario, where multiple edge servers collaboratively trained models using federated learning. We further varied the number of edge servers from 2 to 32 to assess the scalability and limitations of this distributed cryptanalysis setting.

Our experimental results revealed several key findings. First, block ciphers with only one round function, such as SDES, SAES, DES, and SPECK32/64, exhibited high BAP_{avg} values, indicating susceptibility to both EE and PR attacks—even in a distributed environment. However, AES-128 consistently maintained low BAP_{avg} values (~0.5) across all configurations, including reduced-round settings and multi-server deployments, thereby demonstrating strong resilience to neural cryptanalysis.

Second, for DES and SPECK, the attack performance degraded substantially once the number of encryption rounds increased to two or more, with BAP_{avg} values converging toward 0.5. This suggests that even lightweight block ciphers can achieve practical resistance to deep learning-based attacks through sufficient round complexity.

Third, and most notably, the cryptanalysis conducted in the multi-server environment achieved nearly identical performance to the single-server case, especially for ciphers with lower round complexity. This result validates the feasibility and practicality of performing deep learning-based cryptanalysis in distributed environments—an important consideration in real-world attack scenarios where access to centralized computing resources may be limited.

Fourth, we established the statistical rigor of our security assessments by employing formal hypothesis testing with 99% confidence intervals. This approach enables us to distinguish genuine cryptanalytic vulnerabilities ($p < 0.01$) from random performance, providing a principled basis for security claims. Furthermore, our scalability analysis demonstrates that the federated framework maintains computational efficiency across all tested configurations, with aggregation overhead remaining consistently below 0.01% of total training time even when scaling to 32 edge servers.

However, we also observed a slight decline in attack performance as the number of edge servers increased, particularly for shallow ciphers like SDES and DES. This reduction is likely due to the growing difficulty of model synchronization and generalization under highly partitioned training data. As a promising direction for future work, we plan to explore adaptive federated learning techniques, dynamic client weighting, and communication-efficient model aggregation to mitigate this performance degradation. Such approaches could further enhance the robustness and effectiveness of distributed cryptanalysis frameworks.

In summary, this work demonstrates that deep learning-based attacks on block ciphers can be effectively scaled to multi-server environments, offering critical insights into both the capabilities of modern neural cryptanalysis and the design of resilient cryptographic primitives.

Author Contributions: Conceptualization, I.M.; methodology, O.J. and S.P.; software, O.J. and S.P.; validation, O.J. and I.M.; writing—original draft preparation, O.J.; 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).

Data Availability Statement: The data presented in this study are available upon request from the corresponding author due to privacy and ethical restrictions related to the experimental data.

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

References

1. Duan, Y.; Edwards, J.S.; Dwivedi, Y.K. Artificial intelligence for decision making in the era of big data–evolution, challenges and research agenda. *Int. J. Inf. Manag.* **2019**, *48*, 63–71. [CrossRef]
2. Allam, Z.; Dhunny, Z.A. On big data, artificial intelligence and smart cities. *Cities* **2019**, *89*, 80–91. [CrossRef]
3. Alshehri, F.; Muhammad, G. A comprehensive survey of the Internet of Things (IoT) and AI-based smart healthcare. *IEEE Access* **2020**, *9*, 3660–3678. [CrossRef]
4. Misra, N.; Dixit, Y.; Al-Mallahi, A.; Bhullar, M.S.; Upadhyay, R.; Martynenko, A. IoT, big data, and artificial intelligence in agriculture and food industry. *IEEE Internet Things J.* **2020**, *9*, 6305–6324. [CrossRef]
5. Domingo-Ferrer, J.; Farras, O.; Ribes-González, J.; Sánchez, D. Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges. *Comput. Commun.* **2019**, *140*, 38–60. [CrossRef]
6. Yang, P.; Xiong, N.; Ren, J. Data security and privacy protection for cloud storage: A survey. *IEEE Access* **2020**, *8*, 131723–131740. [CrossRef]
7. Paar, C.; Pelzl, J. *Understanding Cryptography*; Springer: Berlin, Germany, 2010. [CrossRef]
8. Stamp, M. *Information Security: Principles and Practice*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
9. 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*; Springer: Berlin/Heidelberg, Germany, 1993; pp. 81–91. [CrossRef]
10. Matsui, M. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology—EUROCRYPT'93: Workshop on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 386–397.
11. National Institute of Standards and Technology (NIST). *Data Encryption Standard (DES)*; Federal Information Processing Standards Publication 46-3; U.S. Department of Commerce: Washington, DC, USA, 1999. Available online: https://csrc.nist.gov/pubs/fips/46-3/final (accessed on 3 July 2025).
12. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
13. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
14. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349. [CrossRef]
15. 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]
16. Mahdavifar, S.; Ghorbani, A.A. Application of deep learning to cybersecurity: A survey. *Neurocomputing* **2019**, *347*, 149–176. [CrossRef]
17. Ferrag, M.A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *J. Inf. Secur. Appl.* **2020**, *50*, 102419. [CrossRef]
18. Dixit, P.; Silakari, S. Deep learning algorithms for cybersecurity applications: A technological and status review. *Comput. Sci. Rev.* **2021**, *39*, 100317. [CrossRef]
19. Bellini, E.; Rossi, M. Performance comparison between deep learning-based and conventional cryptographic distinguishers. In *Intelligent Computing, Proceedings of the 2021 Computing Conference, Virtual, 15–16 July 2021*; Springer: Cham, Switzerland, 2021; Volume 3, pp. 681–701. [CrossRef]

20. Hettwer, B.; Gehrer, S.; Güneysu, T. Applications of machine learning techniques in side-channel attacks: A survey. *J. Cryptogr. Eng.* **2020**, *10*, 135–162. [CrossRef]
21. Zhang, L.; Xing, X.; Fan, J.; Wang, Z.; Wang, S. Multilabel deep learning-based side-channel attack. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *40*, 1207–1216. [CrossRef]
22. Kubota, T.; Yoshida, K.; Shiozaki, M.; Fujino, T. Deep learning side-channel attack against hardware implementations of AES. *Microprocess. Microsyst.* **2021**, *87*, 103383. [CrossRef]
23. Jeong, O.; Ahmadzadeh, E.; Moon, I. Comprehensive neural cryptanalysis on block ciphers using different encryption methods. *Mathematics* **2024**, *12*, 1936. [CrossRef]
24. National Institute of Standards and Technology (NIST). *Advanced Encryption Standard (AES)*; Federal Information Processing Standards Publication 197; U.S. Department of Commerce: Washington, DC, USA, 2001. Available online: https://csrc.nist.gov/pubs/fips/197/final (accessed on 3 July 2025).
25. Musa, M.A.; Schaefer, E.F.; Wedig, S. A simplified AES algorithm and its linear and differential cryptanalyses. *Cryptologia* **2003**, *27*, 148–177. [CrossRef]
26. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference, San Franicsco, CA, USA, 7–11 June 2015*; ACM: New York, NY, USA, 2015; pp. 1–6. [CrossRef]
27. Gohr, A. Improving attacks on round-reduced SPECK32/64 using deep learning. In *Advances in Cryptology–CRYPTO 2019, Proceedings of the 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2019*; Proceedings, Part II; Boldyreva, A., Micciancio, D., Eds.; Springer: Cham, Swizterland, 2019; pp. 150–179. [CrossRef]
28. Wang, H.; Tian, J.; Zhang, X.; Wei, Y.; Jiang, H. Multiple differential distinguisher of SIMECK32/64 based on deep learning. *Secur. Commun. Netw.* **2022**, *2022*, 7564678. [CrossRef]
29. Pal, D.; Mandal, U.; Das, A.; Chowdhury, D.R. Deep learning based differential classifier of PRIDE and RC5. In *Proceedings of the International Conference on Applications and Techniques in Information Security, Manipal, India, 30–31 December 2022*; Springer: Cham, Switzerland, 2022; pp. 46–58. [CrossRef]
30. Liu, J.; Ren, J.; Chen, S. A deep learning aided differential distinguisher improvement framework with more lightweight and universality. *Cybersecurity* **2023**, *6*, 47. [CrossRef]
31. Yang, G.; Zhu, B.; Suder, V.; Aagaard, M.D.; Gong, G. The SIMECK family of lightweight block ciphers. In *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems–CHES 2015, Saint-Malo, France, 13–16 September 2015*; Güneysu, T., Handschuh, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; pp. 307–329. [CrossRef]
32. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*; Arbib, M.A., Ed.; MIT Press: Cambridge, MA, USA, 1998; pp. 255–258.
33. 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 (ICASSP), Vancouver, BC, Canada, 26–31 May 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 6645–6649. [CrossRef]
34. Dai, Y.; Chen, S. Cryptanalysis of Full PRIDE Block Cipher. *Cryptology ePrint Arch.* 2014. Available online: https://eprint.iacr.org/2014/987 (accessed on 3 July 2025).
35. Rivest, R.L. The RC5 encryption algorithm. In *Fast Software Encryption, Proceedings of the Second International Workshop, FSE 1994, Leuven, Belgium, 14–16 December 1994*; Proceedings; Preneel, B., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1995; Volume 1008, pp. 86–96. [CrossRef]
36. 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]
37. Jeong, O.; Moon, I. Adaptive transfer learning-based cryptanalysis on double random phase encoding. *Opt. Laser Technol.* **2024**, *168*, 109916. [CrossRef]
38. Ahmadzadeh, E.; Kim, H.; Jeong, O.; Moon, I. A novel dynamic attack on classical ciphers using an attention-based LSTM encoder-decoder model. *IEEE Access* **2021**, *9*, 60960–60970. [CrossRef]
39. Harrison, J.; Toreini, E.; Mehrnezhad, M. A practical deep learning-based acoustic side channel attack on keyboards. In *Proceedings of the 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Delft, The Netherlands, 3–7 July 2023*; IEEE: New York, NY, USA, 2023; pp. 270–280. [CrossRef]
40. Wang, R.; Wang, H.; Dubrova, E. Far field EM side-channel attack on AES using deep learning. In *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security (ASHES'20), Online, 13 November 2020*; ACM: New York, NY, USA, 2020; pp. 35–44. [CrossRef]
41. Alemerien, K.; Al-Suhemat, S.; Alsuhimat, F.; Altarawneh, E. Enhancing side-channel attacks prediction using convolutional neural networks. In *Proceedings of the 2024 14th International Conference on Advanced Computer Information Technologies (ACIT 2024), Ceske Budejovice, Czech Republic, 19–21 September 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 505–510. [CrossRef]

42. Refregier, P.; Javidi, B. Optical image encryption based on input plane and Fourier plane random encoding. *Opt. Lett.* **1995**, *20*, 767–769. [CrossRef] [PubMed]
43. Ahouzi, E.; Zamrani, W.; Azami, N.; Lizana, A.; Campos, J.; Yzuel, M.J. Optical triple random-phase encryption. *Opt. Eng.* **2017**, *56*, 113114. [CrossRef]
44. Kumar, K.; Tanwar, S.; Kumar, S. Deep-Learning-based Cryptanalysis through Topic Modeling. *Eng. Technol. Appl. Sci. Res.* **2024**, *14*, 12524–12529. [CrossRef]
45. Niu, Z.; Zhong, G.; Yu, H. A review on the attention mechanism of deep learning. *Neurocomputing* **2021**, *452*, 48–62. [CrossRef]
46. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014*; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1724–1734. [CrossRef]
47. 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.
48. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 4171–4186. [CrossRef]
49. So, J. Deep learning-based cryptanalysis of lightweight block ciphers. *Secur. Commun. Netw.* **2020**, *2020*, 3701067. [CrossRef]