

Deep learning-based cryptanalysis in recovering the secret key and plaintext on lightweight cryptography

Yulia Fatma^{1,2}, Muhammad Akmal Remli^{2,3}, Mohd Saberi Mohamad⁴, Januar Al Amien^{1,2}

¹Department of Informatics Engineering, Faculty of Computer Science, Universitas Muhammadiyah Riau, Riau, Indonesia

²Faculty of Data Science and Computing, Universiti Malaysia Kelantan, Kelantan, Malaysia

³Institute for Artificial Intelligence and Big Data, Universiti Malaysia Kelantan, Kelantan, Malaysia

⁴Health Data Science Lab, Department of Genetics and Genomics, College of Medicine and Health Sciences, United Arab Emirates University, Abu Dhabi, United Arab Emirates

Article Info

Article history:

Received May 22, 2024

Revised Nov 6, 2024

Accepted Nov 11, 2024

Keywords:

ADAM

Cryptanalysis

Deep learning

Multi-layer perceptron

S-DES

ABSTRACT

The development of machine learning (ML) technologies provide a new development direction for cryptanalysis. Several ML research in the field of cryptanalysis was carried out to identify the cryptographic algorithm used, find out the secret key, and even recover the secret message. The first objective of this study is to see how much influence optimization and activation function have on the multi-layer perceptron (MLP) model in performing cryptanalysis. The second research objective, which is to compare the performance of cryptanalysis in recovering keys and the plaintext. Several experiments have been carried out, the observed parameters found that the use of the rectified linear unit (ReLU) activation function and the ADAM optimizer improves the performance of deep learning (DL)-based cryptanalysis as evidenced by a significantly smaller error rate. DL-based cryptanalysis works more effectively in recovering keys than recovering plaintext. DL-based cryptanalysis managed to recover the keys with an average loss of 0.007, an average of 49 epochs, and an average time of 0.178 minutes.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Muhammad Akmal Remli

Faculty of Data Science and Computing, Universiti Malaysia Kelantan

Kelantan, Malaysia

Email: akmal@umk.edu.my

1. INTRODUCTION

Cryptography is the study of techniques to ascertain confidentiality and or authenticity of information. Cryptography research primarily focuses on two areas: cryptographic design and cryptanalysis [1]. Cryptanalysis, commonly known as “code-breaking,” involves techniques to decipher encrypted information. Cryptanalysis is the techniques used for deciphering a message without any knowledge of the enciphering [2]. Existing cryptanalytic techniques no longer work on new algorithms [3]. Traditional cryptanalysis techniques tend to require a large amount of time and resources so that they are not compatible with new algorithms, so a new cryptanalysis technique is needed [4]. Advancements in machine learning (ML) technology offer a new direction for cryptography and cryptanalysis [5]. The connection between cryptography and ML was initially introduced in 1991 [6]. Since then, numerous researchers have explored using ML techniques to conduct cryptanalysis on block ciphers. Several ML research in the field of cryptanalysis was carried out to guess or identify the cryptographic algorithm used, guess the S-Box, find out the secret key and even recover the secret message [4], [7], [8]. Based on previous studies, AI capabilities,

especially in the field of deep learning (DL) for cryptanalysis, are considered quite effective against existing algorithms [4], [3], [9].

Research utilizing DL for cryptanalysis is often found in the context of known-plaintext attack (KPA), where the attacker has access to pairs of plaintext and ciphertext, and then uses this information to recover the key. [4], [9], [10]–[16]. Previous research in recover plaintext [1] the experimental results show that the neural network model developed in this paper achieves excellent results in restoring plaintext, with a fitting accuracy exceeding 90% compared to the actual plaintext. Future work will involve further refinement in selecting and adapting neural network weights during training. There may be alternative types of neural networks for cryptanalysis that could yield surprising outcomes. A lot of experiments [17] using varied datasets, keys, and neural network types are planned, with the aim of performing known ciphertext attacks through ML algorithms, specifically leveraging neural networks. Research focused on recovering keys on the S-DES algorithm was conducted [9], [16] where the accuracy achieved 80%.

Based on the results of the author's search, many cryptanalyses of plaintext and key recovery were found to be almost balanced, where the purpose of both cryptanalyses is the same to recover the secret key. This research is to conduct on what cryptanalysis method is most effective by comparing the two types of cryptanalyses. In this study, the author uses the multi-layer perceptron (MLP) neural network model which has higher accuracy compared to convolutional neural network (CNN) and long short-term memory (LSTM) [9]. Training neural network architectures such as MLP can be formulated as optimization problems; hence, they can be solved by some optimization methods [18]. ADAM represents the latest trends in DL optimization [19], [20]. More memory efficient and less computational power are two advantages of ADAM. ADAM is used to modify the weights to minimize losses on the network. In this study, we focus on the S-DES cipher, which, despite its simplicity, effectively encapsulates the core principles of its more complex predecessors. The first objective of this study is to see how much influence optimization and activation function have on the MLP model in performing cryptanalysis. The second research objective, which is to compare the performance of cryptanalysis in recovering keys and the plaintext. Some of the parameters that will be observed are loss, epoch and the time take for the cryptanalysis process to run. The contributions of this research are: i) to see how much influence optimization and activation functions have on the MLP model in performing cryptanalysis; ii) to compare cryptanalysis in terms of recovering keys and plaintext in terms of time, epoch and accuracy

This research offers valuable insights into the application of DL models for cryptanalysis, showcasing their potential in breaking down complex encryption schemes. This research discusses cryptanalysis for recovering plaintext and keys. Although there have been studies on this topic, they have typically been conducted separately and none have explicitly addressed a comparison of the analyzed objects. By demonstrating how these advanced algorithms can be leveraged to analyze and potentially compromise cryptographic systems, the study paves the way for applying similar techniques to other types of ciphers. Furthermore, it establishes a solid foundation for future research in both cybersecurity and cryptography, highlighting the need for continued exploration of how artificial intelligence and ML can enhance our understanding and capabilities in these critical fields. This work not only contributes to the advancement of cryptanalytic methods but also encourages the development of more robust security measures against emerging threats.

2. LITERATURE REVIEW

2.1. Cryptanalysis

Cryptography is a way to ensure the security of information between communicating parties [21]. Cryptography includes cryptography and cryptanalysis [22]. Analytical criticism is known as “breaking the code”. Cryptanalysis is a method used by attackers to access information without knowing the secret key used [23]. Cryptanalysis with a positive direction can be used as a method to evaluate the security level of cryptographic algorithms so as to find weaknesses in order to improve future development directions. Critical analysis can also prevent the use of insecure algorithms for real communication.

2.2. MLP

MPL a type of feedforward artificial neural network (ANN), is an algorithm for supervised learning. It is often regarded as the foundational architecture for DL or deep neural networks (DNN). Typically, an MLP is a fully connected network comprising an input layer that handles the incoming data, an output layer responsible for making decisions or predictions about the input, and one or more hidden layers positioned between these two, which are viewed as the core computational elements of the network. MLP employs a supervised learning method known as “Backpropagation” for its training, which is considered a crucial component of neural networks and is widely utilized for training feedforward neural networks.

The main goal of backpropagation is to adjust the network weights to effectively map inputs to the desired outputs. MLP is affected by feature scaling and requires tuning various hyperparameters such as the number of hidden layers, neurons, and iterations, which can make it computationally intensive for solving complex security models. Nonetheless, MLP is advantageous in learning non-linear models, even in real-time or online learning scenarios, by using partial fit [24].

2.3. Simplified data encryption standard (S-DES)

S-DES is a simplified version of the DES algorithm. While it shares characteristics with DES, it uses a smaller block size and key, operating on 8-bit message blocks with a 10-bit key. The algorithm produces an 8-bit ciphertext block as output. The decryption process is similar to encryption, with the primary distinction being that the steps are executed in reverse order [14]. S-DES was designed as a test block cipher for learning about modern cryptanalytic techniques [25].

2.4. ADAM optimization

This algorithm optimizes stochastic objective functions using a first-order gradient-based approach that relies on adaptive estimates of lower-order moments. It is easy to implement, computationally efficient, requires minimal memory, and remains unaffected by diagonal rescaling of gradients. It is particularly effective for large-scale problems involving extensive data or numerous parameters and performs well with non-stationary objectives or when gradients are highly noisy or sparse. Additionally, the hyper-parameters are intuitively meaningful and generally need minimal adjustment [26].

2.5. Activation function

The activation function determines whether the neuron is activated. This means using simpler mathematical operations to determine whether the input of neurons to the network is important in the prediction process. The role of the activation function is to get the output of a set of input values supplied to the node (or layer [27]). Rectified linear unit (ReLU), may appear to be a linear function, yet it has a derivative that supports backpropagation, making it both computationally efficient and suitable for training neural networks. Mathematically it can be represented as $f(x) = \max(0, x)$. A key aspect of the ReLU function is that it does not activate all neurons simultaneously; neurons become inactive only if the output of the linear transformation is below zero. Leaky ReLU enhances the ReLU function by addressing the dying ReLU issue, as it introduces a small positive slope in the negative region. Mathematically it can be represented as $f(x) = \max(0.1x, x)$. Its output is not 0 for negative inputs, so it is the improvement of ReLU function for the problem of “Dying ReLU”.

3. RESEARCH METHOD

The research method outlines the research framework in this study. The framework presents step by step procedures to be carried out at each stage of the research. The concept of DL for cryptanalysis in this study consist of two experiments, recovering the secret key and recovering the plaintext. The expected result is a comparison of the level of accuracy, time required and the resulting loss. The utilized research method will be explained in Figure 1. Figure 1(a) is a cryptanalytic DL concept for recovering keys. The network requires input in the form of a pair of plaintext and ciphertext. The output of the targeted network is a key. Figure 1(b) shows the concept of DL cryptanalysis to recover plaintext. Input on the network in the form of ciphertext. The targeted output is plaintext. To see the performance of DL-based cryptanalysis done with compare the ReLU and LeakyRelu activation functions. To improve network accuracy, ADAM optimization is added. Measurement of loss value is calculated using MSE. The success of model learning is largely determined by the loss achieved ≤ 0.01 .

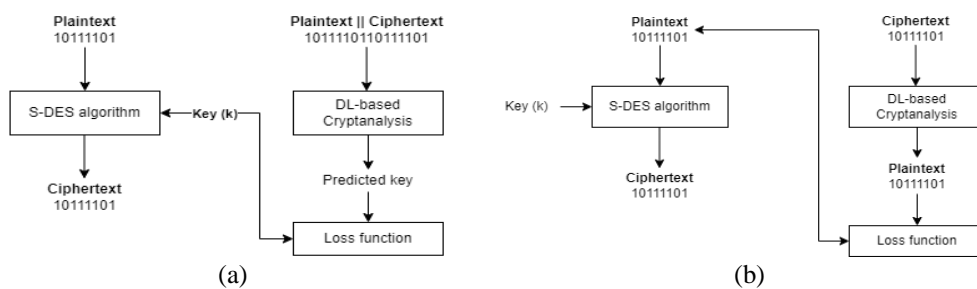


Figure 1. Research method of the DL-based cryptanalysis (a) to break the key and (b) to break the plaintext

3.1. Encryption using S-DES

S-DES is a simplified version of the DES algorithm designed for educational purposes. S-DES uses a 10-bit key and operates on 8-bit data blocks. The S-DES encryption process involves the following steps explained in the Figure 2.

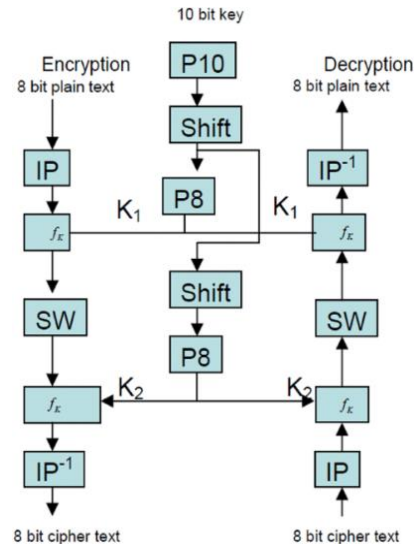


Figure 2. S-DES algorithm [14]

- Key generation: from the original 10-bit key, two 8-bit subkeys are generated using permutation and rotation operations. These are known as subkeys K_1 and K_2 .
- Initial permutation (IP): the 8-bit plaintext block undergoes an initial permutation called the IP to scramble the bits of the plaintext.
- Feistel rounds: the encryption process involves two Feistel rounds. In each round: the 8-bit block is divided into two 4-bit parts (L and R). The R part is processed using the Feistel function (F), which involves expansion/permutation, substitution using S-boxes (S0 and S1), and permutation. The result of the Feistel function is XOR with the L part to produce a new R part, and the old R becomes the new L part.
- Subkey application: subkey K_1 is used in the first round, and subkey K_2 is used in the second round.
- Switching (SW): after the first round, L and R are swapped before the second round begins.
- Inverse initial permutation (IP^{-1}): After the two rounds are completed, the resulting 8-bit block undergoes a final permutation called the IP^{-1} to produce the ciphertext.

Table 1 shows the 8 varying binary keys used in the S-DES encryption. The keys were generated randomly to produce varied binary plaintext. The plaintext used in this experiment consists of 94 ASCII character.

Table 1. S-DES binary key

| Binary key |
|------------|
| 0101010101 |
| 1111111111 |
| 0000000000 |
| 1010101010 |
| 1111100000 |
| 1100111010 |
| 1001111000 |

3.2. Deep learning cryptanalysis

In this study the architecture of the model used is MPL with 10 hidden layers and each consisting of 512 nodes. The architecture used is based on references from previous researchers [4]. The architecture is built using the tensorflow 2 library, the python programming language and the pycharm IDE. The training

process is carried out with the target of stopping two conditions, namely when the target error is 0.01 or reaches an epoch of 200,000. In detail the architecture of the DL model can be seen in Table 2.

Table 2. DL-based cryptanalysis model setup

| Description | Amount |
|---------------|---------|
| Hidden layer | 10 |
| Hidden node | 512 |
| Error target | ≤0.01 |
| Epoch | 200,000 |
| Learning rate | 0.001 |

3.3. Loss function

The loss function addresses the core task of a neural network by measuring the gap between the predicted output and the target value. The target value comes with the training data but is not visible to the neural network during training, while the predicted value is generated by the network. Different types of loss functions exist, each serving specific purposes. The loss function to be used in this research is the mean squared error (MSE). MSE loss always produces a positive result, regardless of the sign of the actual and predicted values. To improve the model’s accuracy, the loss should be as small as possible or reach the perfect value of 0.0. The goal of the training phase in this DNN cryptanalytic research model is to get the same predicted value as the plaintext. The MSE loss function can be expressed as follows:

$$MSE = \frac{1}{N_r \cdot m} \sum_{j=1}^{N_r} \sum_{i=0}^{m-1} (k_i^{(j)} - \hat{k}_i^{(j)})^2$$

where N_r = number of training sampels; $k_i^{(j)}$ = i^{th} bit of the key corresponding to the j^{th} sample and $\hat{k}_i^{(j)}$ = i^{th} output of the DNN corresponding to the j^{th} sample.

4. RESULTS AND DISCUSSION

The first experiment was conducted to obtain the best parameters for the model by comparing the use of ReLU and LeakyReLU activation functions. To enhance the model’s performance, the author attempted to add ADAM optimization. The resulting loss has an average of 60.1784. Second experiment uses the LeakyReLU activation function and the ADAM optimizer. The resulting loss has an average of 40.2961. Third experiment uses the ReLU activation function and the ADAM optimizer. The resulting loss has an average of 0.04825. The results of this comparison can be seen in Figure 3.

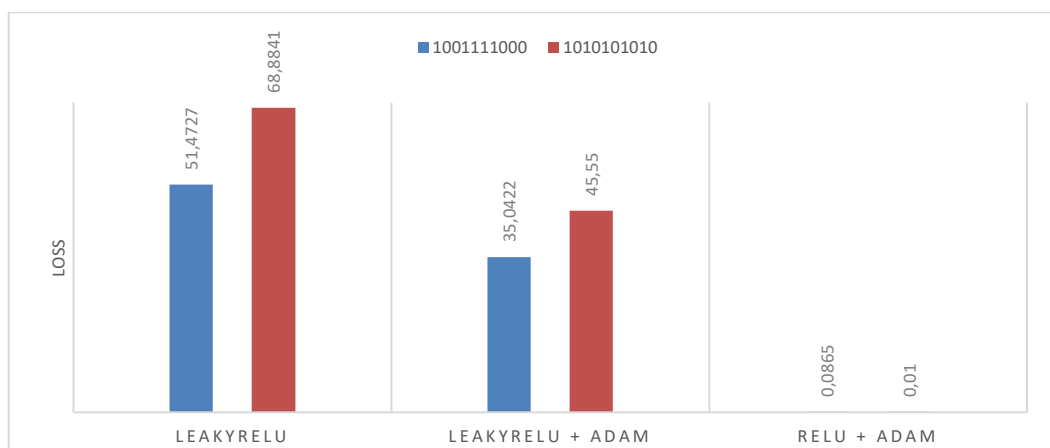


Figure 3. DL-cryptanalysis comparison chart of leakyrelu and relu using adam optimizer

From the three experiments, it can be concluded that the best loss resulting from the DL-based cryptanalysis model using the ReLU activation function and the ADAM optimizer is 0.04825. This is due to the target characteristics of the output data which must have a range between 0 and positive numbers,

ReLU is a suitable activation function in this case. ADAM is well known for achieving better performance than other methods in the non-convex case. ADAM combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients in noise problems. In addition to the adaptive learning rate, the weights in the neural network are also adjusted adaptively based on the first and second moments of the gradient. This helps achieve faster convergence, avoid large spikes in weight updates, and optimize overall model performance and yield. The second experiment was conducted to test the performance of DL-based cryptanalysis using the ReLU activation function and the ADAM optimizer. The experimental results are presented in Table 3.

Table 3. DL-based cryptanalysis comparison on plaintext and key recovery on 26 characters

| Key | Plaintext recovery | | | Key recovery | | |
|------------|--------------------|---------|----------------|--------------|--------|----------------|
| | Loss | Epoch | Time (minutes) | Loss | Epoch | Time (minutes) |
| 0101010101 | 0.01 | 100,661 | 6.71 | 0.010 | 12,241 | 0.8 |
| 1111111111 | 0.01 | 35,721 | 2.3 | 0.008 | 24,239 | 1.6 |
| 0000000000 | 0.01 | 32,316 | 2.15 | 0.002 | 10 | 0.0006 |
| 1010101010 | 0.01 | 101,546 | 6.7 | 0.009 | 5,923 | 0.39 |
| 1111100000 | 0.01 | 83,708 | 5.5 | 0.010 | 28,679 | 1.9 |
| 1100111010 | 0.600 | 119,062 | 7.9 | 0.009 | 22,110 | 1.4 |
| 1001111000 | 0.086 | 81,756 | 5.4 | 0.010 | 19,571 | 1.3 |

Table 3 showing the comparison results between key recovery and plaintext recovery in terms of loss, epoch and time required. In this experiment, DL-based cryptanalysis attempted to compare the performance in recovering plaintext and recovering keys. The data set tested was 26 characters consisting of the letters a to z. Given seven different kinds of keys from patterned to random. The DL-based cryptanalysis experiment in recovering plaintext resulted in an average loss of 0.105. While the experiment in recovering the key resulted in a smaller loss, namely an average of 0.008. In terms of the epochs in the DL-based cryptanalysis experiment to recover plaintext, the average epoch was 28,738. While the experiment in recovering the key resulted in smaller epochs, namely an average of 17 epochs. The time required in the DL-based cryptanalysis experiment to recover the plaintext resulted in an average of 5.23 minutes. While the experiment in recovering the key resulted in a shorter time, namely 1.05 minutes on average.

The experimental results in Table 3 are visualized graphically in Figure 4. Figure 4(a) is a graph that visualizes the comparison of errors in key and plaintext recovery. The key recovery error shows good and consistent results. However, plaintext recovery displays a random pattern, where the 6th and 7th keys used for encryption affect the resulting loss. Figure 4(b) is a graph that visualizes the comparison of epochs between key and plaintext recovery. Figure 4(c) is a graph that visualizes the comparison of time between key and plaintext recovery. Plaintext recovery shows that more epochs and time are needed compared to key recovery. The last experiment uses the ReLU activation function and the ADAM optimizer, the character is expand to 94 characters consisting of a combination of letters, numbers and symbols. The experimental results are presented in Table 4.

Table 4 showing the comparison results between key recovery and plaintext recovery in terms of loss, epoch and time required with expanded data. The DL-based cryptanalysis experiment in recovering plaintext with 94 characters experienced a very large increase in loss, namely an average of 301,285. While the experiment in recovering the key resulted in a smaller loss, namely an average of 0.007. In terms of epochs required in DL-based cryptanalysis experiments to recover plaintext yielded average epochs above 200,000. While the experiment in recovering the key resulted in smaller epochs, namely an average of 49 epochs. The time required for DL-based cryptanalysis experiments to recover plaintext results in an average of over 30 minutes. While the experiment in recovering the key resulted in a shorter time, which was an average of 0.178 minutes. The experimental results in Table 4 are visualized graphically in Figure 5. Figure 5(a) is a graph that visualizes the comparison of errors in key and plaintext recovery. Figure 5(b) is a graph that visualizes the comparison of epochs for key and plaintext recovery. Figure 5(c) is a graph that visualizes the comparison of time for key and plaintext recovery. We found that increasing the amount of data affects the error, epochs, and time required for plaintext recovery. The model was unable to recover the plaintext and failed to learn the provided data patterns. The highly random nature of large data sets requires further research to construct effective data and models with enhanced hyperparameters. Future studies may explore using more varied data and an improved model.

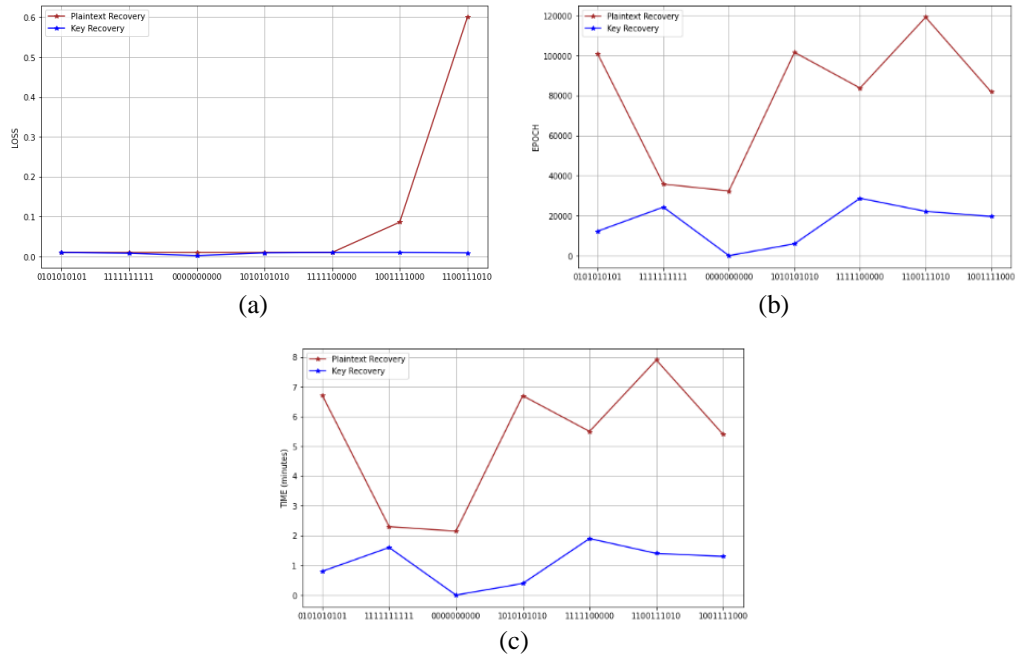


Figure 4. Comparison on plaintext and key recovery: (a) loss, (b) epoch, and (c) time required on 26 characters

Table 4. DL-based cryptanalysis comparison on plaintext and key recovery

| Key | Plaintext recovery | | | Key recovery | | |
|------------|--------------------|---------|----------------|--------------|-------|----------------|
| | Loss | Epoch | Time (minutes) | Loss | Epoch | Time (minutes) |
| 0101010101 | 752 | 200.000 | > 30 | 0.0082 | 318 | 0.02 |
| 1111111111 | 331 | 200.000 | > 30 | 0.0088 | 5,429 | 0.36 |
| 0000000000 | 81 | 200.000 | > 30 | 0.0090 | 7 | 0.0004 |
| 1010101010 | 215 | 200.000 | > 30 | 0.0025 | 2,051 | 0.13 |
| 1111100000 | 145 | 200.000 | > 30 | 0.0085 | 7,629 | 0.5 |
| 1100111010 | 273 | 200.000 | > 30 | 0.0083 | 1,316 | 0.087 |
| 1001111000 | 312 | 200.000 | > 30 | 0.0088 | 2,356 | 0.15 |

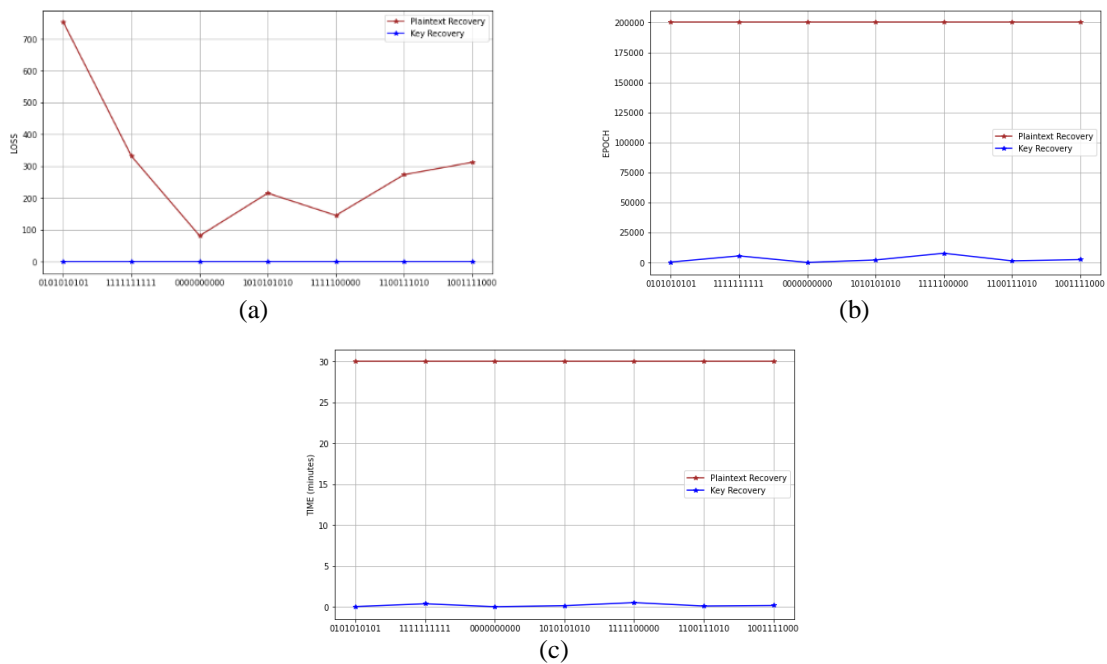


Figure 5. Comparison on plaintext and key recovery: (a) loss, (b) epoch, and (c) time required on 94 characters

5. CONCLUSION

The first objective of this study is to determine the influence of optimization and activation functions on the MLP model in performing cryptanalysis. The second objective is to compare cryptanalysis in terms of recovering keys and plaintext in terms of time, epochs, and accuracy. Based on the experimental results, it is shown that the selection of the appropriate activation function and optimization plays a crucial role in improving the accuracy of the network model. In this study, the best results were obtained using the ReLU activation function and ADAM optimization, as evidenced by a significant reduction in loss. The research results also indicate that cryptanalysis is more effective at recovering keys than plaintext. DL-based cryptanalysis successfully recovered keys with an average loss of 0.007, an average of 49 epochs, and an average time of 0.178 minutes.

Future research can be developed by attempting hyperparameter selection using optimization algorithms. The appropriate selection of hyperparameters will maximize the model's performance in conducting cryptanalysis. Cryptanalysis experiments on other lightweight cryptographic algorithms can also be conducted considering the emergence of new algorithms.




REFERENCES

- [1] Y. Zhao and S. Fan, "Analysis of cryptosystem recognition scheme based on Euclidean distance feature extraction in three machine learning classifiers," *Journal of Physics: Conference Series*, vol. 1314, no. 1, 2019, doi: 10.1088/1742-6596/1314/1/012184.
- [2] W. Stallings, *The william stallings books on computer data and computer communications*, Eighth Edition, 5th ed. New York: Pearson, 2011.
- [3] A. Benamira, D. Gerault, T. Peyrin, and Q. Q. Tan, "A deeper look at machine learning-based cryptanalysis," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2021, vol. 12696 LNCS, pp. 805–835, doi: 10.1007/978-3-030-77870-5_28.
- [4] J. So, "Deep learning-based cryptanalysis of lightweight block ciphers," *Security and Communication Networks*, vol. 2020, pp. 1–11, Jul. 2020, doi: 10.1155/2020/3701067.
- [5] Y. Fatma, R. Wardoyo, and H. Mukhtar, "An approach to cryptography based on neural network," *AIP Conference Proceedings*, vol. 2601, no. 1, pp. 1–9, 2023, doi: 10.1063/5.0130464.
- [6] R. L. Rivest, "Cryptography and machine learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 739 LNCS, 1993, pp. 427–439, doi: 10.1007/3-540-57332-1_36.
- [7] S. Baek and K. Kim, "Recent advances of neural attacks against block ciphers," *Proceedings of the 2020 Symposium on Cryptography and Information Security*, 2020.
- [8] E. M. Meno, "Neural cryptanalysis for cyber-physical system ciphers," Virginia Polytechnic Institute and State University, 2021.
- [9] B. Y. Chong and I. Salam, "Investigating deep learning approaches on the security analysis of cryptographic algorithms," *Cryptography*, vol. 5, no. 4, p. 30, Oct. 2021, doi: 10.3390/cryptography5040030.
- [10] M. Danziger and M. A. Amaral Henriques, "Improved cryptanalysis combining differential and artificial neural network schemes," in *2014 International Telecommunications Symposium (ITS)*, Aug. 2014, pp. 1–5, doi: 10.1109/ITS.2014.6948008.
- [11] L. Lerman, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked AES: Reaching the limit of side-channel attacks with a learning model," *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 123–139, 2015, doi: 10.1007/s13389-014-0089-3.
- [12] S. Amic, K. M. S. Soyjaudah, and G. Ramsawock, "Binary cat swarm optimization for cryptanalysis," in *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Dec. 2017, pp. 1–6, doi: 10.1109/ANTS.2017.8384120.
- [13] R. Focardi and F. L. Luccio, "Neural cryptanalysis of classical ciphers?," *CEUR Workshop Proceedings*, vol. 2243, pp. 104–115, 2018.
- [14] R. Kamal, M. Bag, and M. Kule, "On the cryptanalysis of S-DES using nature inspired optimization algorithms," *Evolutionary Intelligence*, vol. 14, no. 1, pp. 163–173, 2021, doi: 10.1007/s12065-020-00417-5.
- [15] W. Tian and B. Hu, "Deep learning assisted differential cryptanalysis for the lightweight cipher SIMON," *KSII Transactions on Internet and Information Systems*, vol. 15, no. 2, pp. 600–616, 2021, doi: 10.3837/tiis.2021.02.012.
- [16] H. Kim, S. Lim, and Y. Kang, "Deep learning based cryptanalysis of lightweight block ciphers, revisited," *Cryptology ePrint Archive*, no. 886, pp. 1–15, 2022, doi: doi.org/10.3390/e25070986.
- [17] S. Andonov, J. Dobрева, L. Lumburovska, S. Pavlov, and A. Popovska-mitrovikj, "Application of machine learning in DES cryptanalysis," in *ICT-Innovations 2020*, 2020, pp. 124–134.
- [18] A. Al Bataineh, D. Kaur, and S. M. J. Jalali, "Multi-layer perceptron training optimization using nature inspired computing," *IEEE Access*, vol. 10, pp. 36963–36977, 2022, doi: 10.1109/ACCESS.2022.3164669.
- [19] R. Y. Sun, "Optimization for deep learning: an overview," *Journal of the Operations Research Society of China*, vol. 8, no. 2, pp. 249–294, 2020, doi: 10.1007/s40305-020-00309-6.
- [20] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, pp. 1–74, Mar. 2021, doi: 10.1186/s40537-021-00444-8.
- [21] S. R.-Salzedo, *Cryptography*. Gewerbestrasse 11, 6330 Cham, Switzerland: Springer International Publishing AG, 2018, doi: https://doi.org/10.1007/978-3-319-94818-8.
- [22] S. Fan and Y. Zhao, "Analysis of des plaintext recovery based on BP neural network," *Security and Communication Networks*, vol. 2019, pp. 1–5, 2019, doi: 10.1155/2019/9580862.
- [23] C. Zhu, G. Wang, and K. Sun, "Cryptanalysis and improvement on an image encryption algorithm design using a novel chaos based s-box," *Symmetry*, vol. 10, no. 9, p. 399, 2018, doi: 10.3390/sym10090399.
- [24] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed., vol. 1–3. Pearson Education, Inc., publishing as Prentice Hall., 2009.




- [25] K. Raj, B. Sharma, N. Kumar, and D. D. Kaur, "Differential cryptanalysis on S-DES," *International Journal of Management & Information Technology*, vol. 1, no. 2, pp. 42–45, 2012, doi: 10.24297/ijmit.v1i2.1445.
- [26] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015, pp. 1–15, doi: doi.org/10.48550/arXiv.1412.6980.
- [27] J. Feng and S. Lu, "Performance analysis of various activation functions in artificial neural networks," *Journal of Physics: Conference Series*, vol. 1237, no. 2, p. 022030, Jun. 2019, doi: 10.1088/1742-6596/1237/2/022030.

BIOGRAPHIES OF AUTHORS






Yulia Fatma    completed education Bachelor's degree in the Department of Informatics Engineering, University of Amikom Yogyakarta. Master's degree in Master of Computer Science at Gadjah Mada University. Now working as a lecturer in the Department of Informatics, University of Muhammadiyah Riau. With research interests in the field of Cryptography and AI. She can be contacted at email: yuliafatma@umri.ac.id.






Muhammad Akmal Remli    joins Institute for Artificial Intelligence and Big Data (AIBIG), Universiti Malaysia Kelantan (UMK) as a fellow researcher in early 2020 and now he is AIBIG's director. He is also a senior lecturer at Faculty of Data Science and Computing, U K. He received a Master and a Ph.D. degree in Computer Science from Universiti Teknologi Malaysia in 2014 and 2018 before joining Universiti Malaysia Pahang from 2018 until 2020. In 2016, he worked at The Bioinformatics, Intelligent Systems and Educational Technology (BISITE) Research Group at University of Salamanca, Spain as research attachment and was working in cancer bioinformatics. His main research interests are artificial intelligence, data science, business intelligence and computational systems biology. He has published numerous scientific research papers indexed by Scopus and Clarivate Web of Science including in Expert Systems with Applications (ESWA) and Engineering Applications of Artificial Intelligence (EAAI) journals. He can be contacted at email: akmal@umk.edu.my.



Mohd Saberi Mohamad    is a professor of artificial intelligence and health data science. He is currently the Director of the Health Data Science Laboratory, Department of Genetics and Genomics. Before joining CMHS-UAEU, he was with several universities in Malaysia as the Director of the Institute for AI and big data, the head of the AI and Bioinformatics Research Group, a Founder of the Department of Data Science, the Manager of IT, and the Deputy Director (Academic) for Centre of Computing and Informatics. In appreciation of his leadership, the university has appointed him as a member of University Senate. Additionally, he is a member of the Advisory Board for the AI Research Institute and IoT Digital Innovation Hub in Europe. In addition to being the principal investigator on 21 research grants and the co-PI on 20 research grants, he has published 304 articles in international refereed journals and international conferences, book chapters, and 17 books. His research interests include artificial intelligence, data science, and bioinformatics. He can be contacted at email: saberi@uaeu.ac.ae.



Januar Al Amien    completed education Bachelor's degree in the Department of Informatics Engineering, STMIK-AMIK Riau. Master's degree in Master of Information Technology at Putra Indonesia University Padang. Now working as a lecturer in the Department of Computer Science, University Muhammadiyah of Riau. With research interests in the field of machine learning algorithms and AI. He can be contacted at email: januaralamien@umri.ac.id.