Global Journal of Engineering and Technology Advances
eISSN: 2582-5003
Cross Ref DOI: 10.30574/gjeta
Journal homepage: https://gjeta.com/

(RESEARCH ARTICLE)

Check for updates

A system of symmetric key cryptography using deep learning

Lohith D K and Sanjay M *

Department of Computer Science and Engineering, Rajeev Institute of Technology, Hassan, India.

Global Journal of Engineering and Technology Advances, 2024, 20(02), 081–089

Publication history: Received on 16 June 2024; revised on 05 August 2024; accepted on 07 August 2024

Article DOI: https://doi.org/10.30574/gjeta.2024.20.2.0139

Abstract

Encryption is the process of making data unreadable, whereas decryption is the process of recovering the original data. Good cryptography encrypts data in a way that makes it impossible to conduct a brute force assault on the key or a cryptography method. Many ciphers have been proposed up to this point. But their weaknesses have also been made public. Novel cryptography techniques are therefore quite desirable. Within the suggested project, a symmetric A deep neural network-based cryptography algorithm is created. Our experiments demonstrate that the suggested approach outperforms conventional cryptographic methods and is very difficult to crack, especially when little text files are involved.

Keywords: Symmetric Encryption; Autoencoder Neural Networks; Cryptography; Cryptography Systems**.**

1. Introduction

Information security is becoming absolutely essential in today's digital environment, since billions upon terabytes of data from online transactions are continuously transferred over the Internet. Computer scientists are therefore strongly motivated to safeguard data against malevolent individuals, who develop rapidly along with technological advancements. The Cryptography's primary goal is to prevent unwanted access to data. By using intricate logic and math's to render information incomprehensible(decryption) and to get the original data back (encryption). Numerous cryptography algorithms have been created to date. like RSA, 3DES, and AES, each of which has advantages and disadvantages of its own. Encryption and decryption need a lot of processing power due to the extensive serial operations used by traditional cryptographic ciphers, which include complicated algorithms and large prime numbers and in some ways susceptible. However, recent advances in artificial intelligence and machine learning have led to notable advancements in cryptography. This Using an autoencoder neural network for data encryption and decryption, along with a "key generator" to set initial weights and determine the order in which information is presented to the network at each training phase epoch, this work proposes an alternative deep learning encryption system. According to experimental findings, the suggested system beats AES, RSA, 3DES, and DES when encrypting files that don't exceed 868 KB in size. Additionally, we demonstrate that there could be clear benefits to the suggested solution in terms of data security.

2. Related Works

The use of artificial neural networks (ANNs) for cryptography problems is at the forefront of deep learning-based cryptography. This introduces related work that investigates the concept of combining ANNs with encryption.

* Corresponding author: Sanjay M.

2.1. Neural Cryptography

Researchers, like Jogdand and Bisalapur [9] and Kinnzel and Kanter [11], have explored mutual learning neural networks for key management, generation, and exchange. They presented a neural key exchange system based on The study focuses on synchronizing two parity machines, a feed-forward neural network with one output neuron, K hidden neurons, and K*N input neurons.

Neural networks, for instance, are suggested to be able to function as a safe key exchange method [9]. The same input vector is received by two parity machines in this method, one for the sender and one for the recipient sides. Produce an output bit, which serves as the basis for training. Two neural networks that have been trained in silence appear to reach a synchronization stage in which their time-dependent synaptic weights are identical. Comparing the outputs of the matching tree parity machines of two partners, A and B, completes the training process. Lastly, using conventional methods like AES, the secret key that is created over a public channel is utilized for data encryption and decryption [28].

Similarly, [11] claimed that there is no proof that an algorithm doesn't exist.for a successful attack, however using raw force to break this strategy is quite difficult. An attacker cannot get the secret common key that A and B employ for encryption, even though he is aware of the input/output relationship and the technique. The only approach for generating keys over a public channel that isn't reliant on number theory is neural networks. Its primary advantages over conventional methods are that it is straightforward, requires little computing power during training, and generates a new key for every message exchange.

A public channel has also been employed to generate secure cryptographic secret keys through the use of simple neural networks. For example, the learning process where two perceptions receive a common input and adjust their weights based on their mutual output was described by Klein et al. [12]. Additionally, they proposed a novel method that blends chaos theory and neural networks. It appears that synchronization offers the best protection against intruders. At last, something functions. Describes the various attack methods and recommends that the system be made more secure by employing a large enough weight space.

Three techniques to breach this exchange protocol were suggested, despite the fact that this technology is secure in the sense that an attacker employing a similar neural network is unlikely to converge to the same produced key when both sides are synchronized. within [13]. That being said, the suggested methodology in this work is unaffected by this kind of issue because it encrypts and decrypts data in a manner akin to that suggested by Volna et al. [29] rather than dealing with key exchange protocols. Two neural networks are trained for encryption and decryption, respectively, in this study. According to this method, the key is regarded as a tuple made up of the network parameters.

2.2. Visual Cryptography

Not only has deep learning cryptography addressed message encryption and decryption, but it has also demonstrated the existence of cryptography systems. with an emphasis on visual content. Steganalysis is a method that makes it possible to conceal information in pictures in this way. The primary objective of steganography is to ensure An adversary is unable to decipher the hidden message or identify the pictures that contain it. confidential data [7]. A technique to determine whether or not an image contains concealed information was proposed by Shaohui, Hongxun, and Wen [26]. Comparable research, but with aShi et al. generated an alternative set of features for the images [27]. Neural networks were employed in this work of art as an image classifier, and a direct correlation was discovered.

More recently, Qian, Dong, Wang, and Tan [25] prepared a study using a customized Convolutional Neural Network (CNN). Among deep learning models, CNN is one of the most well-known. Its foundation in the convolution matrix operator of mathematics gives rise to its name. Convolutional, non-linearity, pooling, and fully connected layers make up the layers of a CNN, or hierarchical neural network [3]. This kind of neural network may adapt how trainable filters are applied and pooling operations are performed to learn the feature representations of complicated hierarchical images [25].Qian et al.'s major goal was to automate feature representation learning.Ye, Ni, and Yi created a CNN-based steganalysis technique for digital photos [30]. The developed CNN learns hierarchical data representation and optimizes essential processes for steganographic detection. In Ye et al. compare the suggested model to three modern steganographic algorithms: WOW, S-UNIWARD, and HILL, similar to Qian et al.'s approach.

Hu, Wang, Xu, Pu, and Peng found that a stacked autoencoder can effectively encrypt images [8]. Traditional cryptography techniques, such as AES or DES, are developed for text encryption and not suitable for picture encryption (15).

Deep learning has been used in digital watermarking, a popular technique for protecting photos from attacks. Image watermarking involves embedding information into an image. The trademark can then be identified and extracted to make a claim about the transmitted image. The four primary requirements for watermarking systems are domain trademark insertion, watermark detection and extraction, watermark resilience to attacks, and watermark visibility [7]. Mohananthini1 and Yumana [19] provided a review of watermark embedding and extraction strategies for digital images.

2.3. DNA cryptography and Deep Learning

DNA cryptography is a new approach to addressing information security challenges. This new approach is inspired by DNA's ability to store, process, and transmit information. DNA cryptography combines standard cryptographic techniques with the unique chemical properties of DNA molecules [20]. Essentially, in DNA cryptography, information is encoded on a nucleotide basis (A, C, G, T), utilizing DNA computational algorithms [24]. A single cubic decimeter of DNA solution may hold trillions of bits, allowing enabling faster processing of large amounts of data than existing methods [22]. Kalsi, Kaur, and Chnag proposed merging DNA cryptography and deep learning [10].

Basu, Karuppiah, Nasipuri, Halder, and Radhakrishnan also created a neural network key generator for DNA cryptography.[4]. Maddodi, Awad A., D., Awad M., and Lee [17] investigated the use of neural networks, chaotic systems, and DNA computing for picture encryption.

2.4. Symmetric Key Cryptography Model

The generated key determines the starting weights and order of information provided to the network during each epoch. Figure 1 shows a graphical depiction of the suggested method for a symmetric key encryption model.

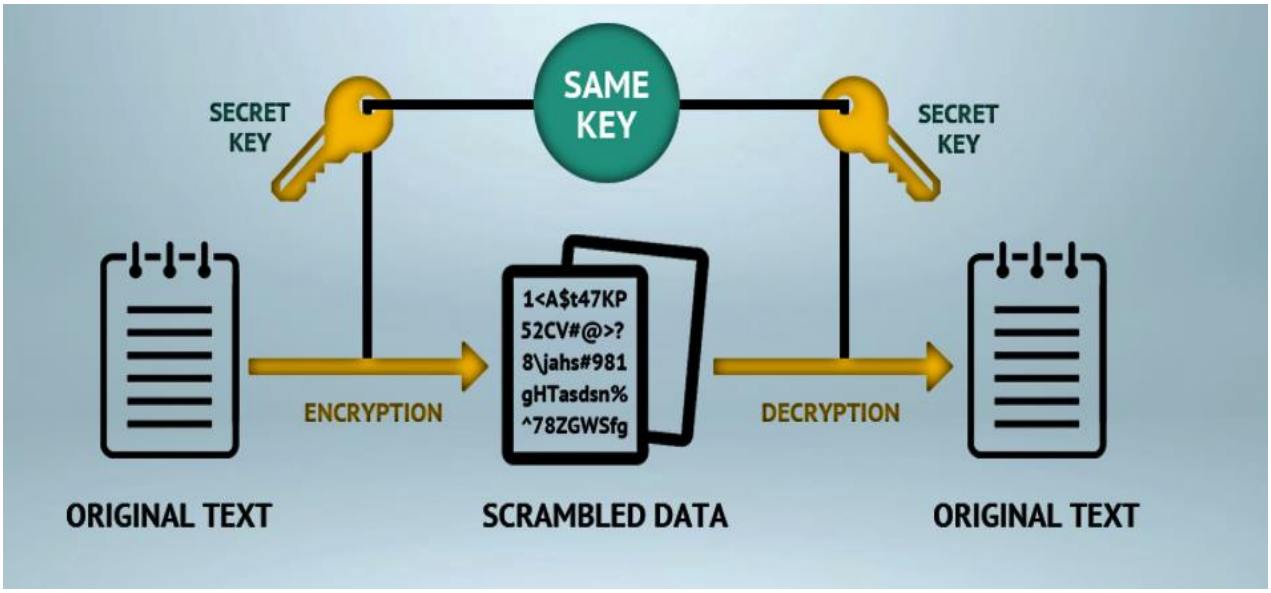


Figure 1 Symmetric Key Analogy Implemented in the Proposed Model.

2.5. Autoencoder architecture

In this work, a sparse audio encoder with 8 neurons in the input layer, 9 neurons in the hidden layer, and 8 neurons in the output layer was selected. The reason for choosing a single layer is that a single hidden layer keeps the system simple and powerful enough when more neurons than the input layer are added. An autoencoder is a feed-forward artificial neural network that uses backpropagation to be trained in an unsupervised fashion [21]. The autoencoder is trained to generate its own input at the output [6].

The activation function for this implementation is a sigmoid function with a gain value: β according to the following formula:

$$f(x) = \frac{1}{1 + e^{-\beta x}} \quad (1)$$

It is possible to have layers with various sigmoid function versions when the gain value is variable. A method like this makes it possible to mitigate the impact of the gradient descent issue. Specifically, the output layer utilizes a gain value of 5.5, which was chosen after multiple testing, but the concealed layer uses a gain value of 10.5. The initial synaptic weight interval was fixed at [-0.5, 0.5], and the chosen learning rate was 0.25. The spare autoencoder's hyperparameters are compiled in Table 1.

Table 1 Hyper parameters List of the Proposed Neural Network

	Parameter	Value
Autoencoder	Training Algorithm	Backpropagation
	Activation Function	Sigmoid
	Number of Neurons	8-9-8
	Gain Values	10.5-5.5
	Initial Weight Range	[-0-5-0-5]
	Learning Rate	0.25

2.6. Key generator

The programming language utilized for this implementation was C++. As previously stated, a neural network's synaptic weights should be randomly initialized. In C++, every random initialization requires a seed, which defines the values of the pseudo-random sequence of initial synaptic weights. Because the same seed values generate the same pseudo-random sequence, a seed generator based on the secret password was created to boost the entropy level during training. In addition, the sequence in which inputs are delivered to the network influences how the network produces internal representation. This knowledge is used to invigorate the system by creating a key generator that shuffles the data set each time the entire set is submitted to the network during training. The seed sequence, which has the same length as the user's login password, is incremented after each usage to prevent repeating seeds in subsequent searches. Figure 2 displays the key generation algorithm developed in this study.

2.7. Encryption and Decryption Approach

This section provides a detailed description of the encryption technique used for message exchange. Figure 3 shows a flowchart of the encryption process. The simple text is first translated into 8-bit ASCII representations for each character.

For example the text “Hello” is encoded into binary ASCII sequences as:

P =01001000011001010110110001101100011011111

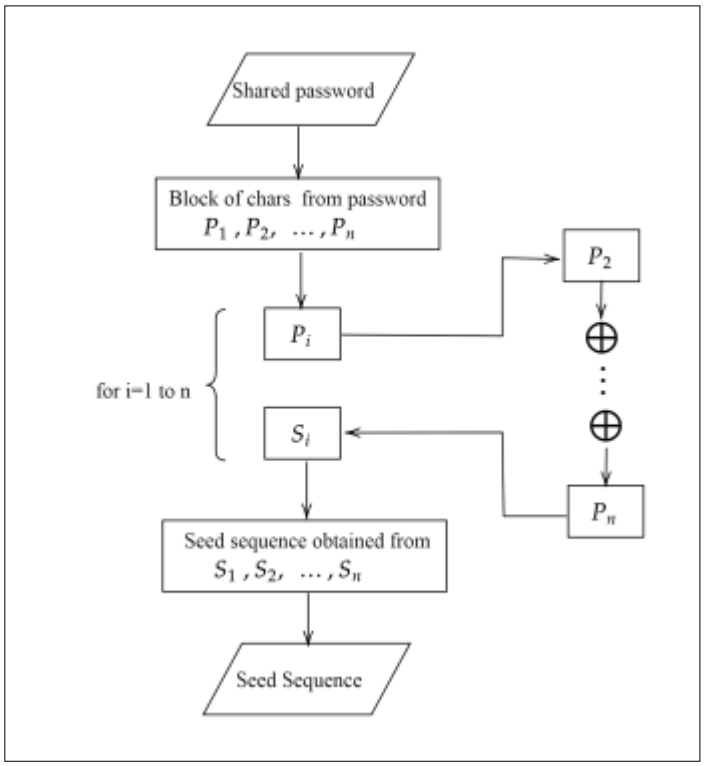


Figure 2 Seed Sequence Generation Flowchart

After converting sequence in to binary form the sequence is divided in to blocks of 8-bit each:

P` =01001000 01100101 01101100 01101100 01101111

The weight matrix for the hidden layer, denoted as WE, is multiplied by each 8-dimensional binary vector to generate encrypted versions. The resulting 9-dimensional vector of floating-point numbers between 0 and 1 corresponds to the range of the sigmoid function used as the network's transfer function. Finally, the encryption text is completed. Concatenating each 9-dimensional vector results in an array of floating points. Numbers have a length of 9 times the number of characters, while plain text has:

C = 0.999846 0.882371 0.220441 0.000862 0.870736 0.369639 0.002191 0.491044 0.542451 0.992447 0.070593 0.000766 0.992716 0.156359 0.549780 0.549430 0.500320 0.995227 0.999920 0.907750 0.000229 0.049527 0.617286 0.029654 0.452724 0.598132 0.507328 0.999920 0.907750 0.000229 0.049527 0.617286 0.029654 0.452724 0.598132 0.507328

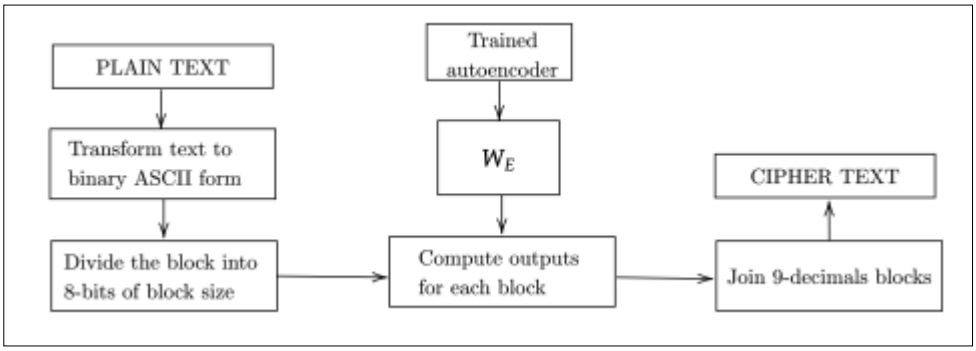


Figure 3 Encryption Process

The decryption procedure is similar to the encryption process, as illustrated in Figure 4. First, the receiver divides the received array into blocks of nine floating point integers and multiplies them by the output layer's weight matrix,

abbreviated as WD. After processing each block, the output layer's 8-dimensional vector is turned into a character using the ASCII table. Finally, the collected characters are concatenated to yield plain text.

This technique allows all parties involved in the transmission to encrypt and decode data. Passwords can be changed with each discussion or set to expire after a certain time.

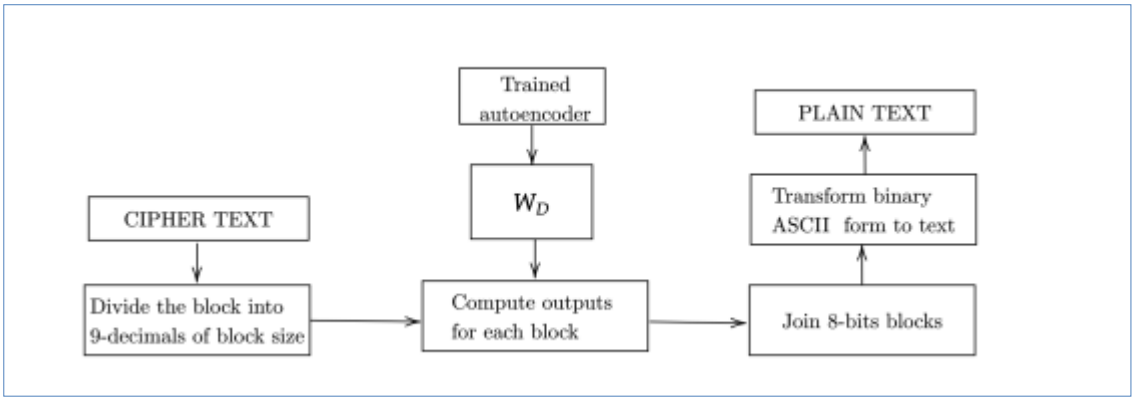


Figure 4 Decryption Process

3. Experimental Study

Evaluation parameters were compared to standard cryptographic methods, including AES, DES, 3DES, and RSA. Because every symmetric encryption algorithm is open, an eavesdropper in the proposed system already knows the network's design, hyperparameters, and seed generation algorithm. In this sense, an attacker attempting to break the proposed system desires to obtain the network's weights. This can be accomplished by trial and error. An experiment was conducted to determine the feasibility of using brute force to brake the system.

3.1. Evaluation Parameters

This work examines the time required for encryption and decryption. Two experiments with varying input sizes were carried out, and the findings were compared to the state of the results under various implementations and hardware to obtain a general picture of the proposed system's behavior when compared to classical cryptography techniques.

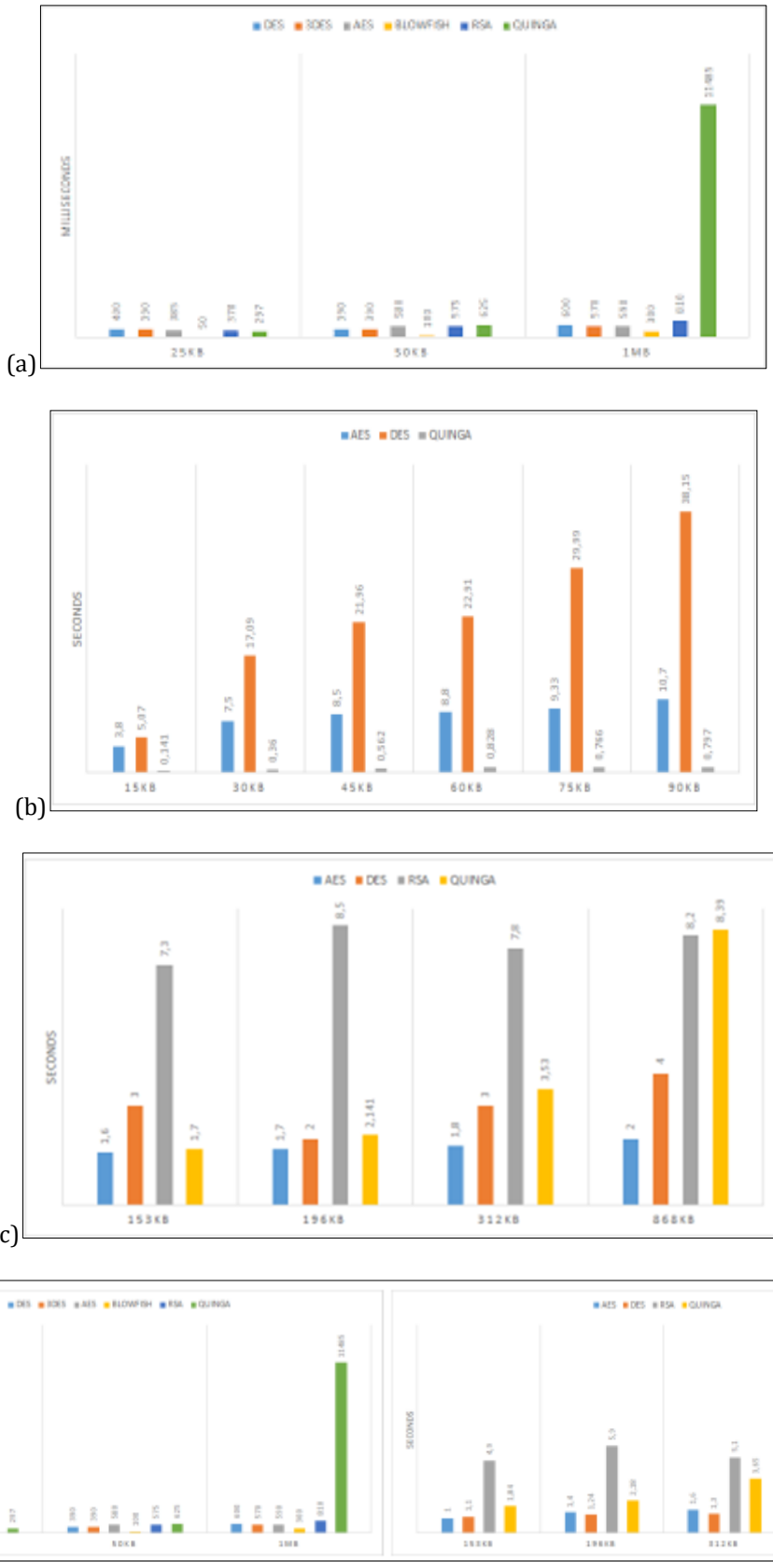
Experiment 1: Encryption time: The execution time for the suggested encryption mechanism. The plain text was assessed across a set of data files. File sizes range from 15 KB to 1 MB and contain alphanumeric characters. The elapsed times were measured using built-in C++ methods and compared to encryption times given by Patil et al. [23], Rihan et al. [1], and Mahajan and Sachdeva [18]. Figure 5 summarizes the findings of the experiment.

Although the proposed method outperforms DES, 3DES, AES, and RSA when encrypting 25 KB of data in just 297 milliseconds, it falls behind when dealing with large files, as illustrated in Figure.5(a). On the other hand, Figure. 3(b) shows that the suggested technique significantly outperforms AES and DES, independent of the file size. For 90KB encryption, AES and DES took 10.7 and 38.15 seconds, respectively, whereas the suggested technique took only 0.797 seconds. Figure. 5(c) demonstrates that the proposed system maintains encryption timings comparable to AES and DES for packet sizes of 153KB, 196KB, and 312KB. However, speed decreases for text files larger than 868 KB.

Experiment 2: Decryption time: The suggested system's execution time for decrypting the cipher texts generated in Experiment 1 was measured and compared to encryption times provided by Patil et al. [23] and Mahajan & Sachdeva [18]. This experiment indicates that standard algorithms require less time to decode than encryption, whereas the proposed system remains almost same (see Figure. 5).

Figure 5(d) shows that the proposed approach performs optimally for files larger than 25 KB.

Figure 5(e) shows that the proposed system requires less time than RSA for 153KB, 196KB, and 312KB. The proposed approach requires somewhat longer decryption time than DES and AES. However, using an 868 KB text file causes the recommended solution to take significantly longer. Compare the times taken by AES, DES, and RSA.



(a)

(b)

(c)

(d)

(e)

Figure 5 Comparison of encryption time with (a) Patil et al. (b) Rihan et al.; (c) Mahajan & Sachdeva. Comparison of decryption time with (d) Patil et al. and (e) Mahajan & Sachdeva.

3.2. Cryptanalysis

The suggested approach's strength depends on the synaptic weights and key generation process, as the autoencoder architecture and hyperparameters remain open. We examined the entropy of the key generation algorithm, as well as the key and synaptic weight spaces.

Experiment 3: Key generator: The system's overall security relies on its ability to generate unique synaptic weights after each initialization. The key generator should be capable of producing distinct synaptic weights even if the user's login password differs by a single word. To investigate synaptic weights, a user password was fixed and one word was arbitrarily changed. During this trial, the user. The average Euclidean distance between synaptic weights was calculated using the password "hello" and the passwords "hella," "yello," and "henlo.".

Table 2 summarizes the findings from this investigation. Using different log-in passwords leads to significant fluctuation. The suggested key generator can generate unique seed sequences and synaptic weights with minimum alterations. Enter the login password.

Table 2 Average Distance Between Synaptic with Respect to the Password “hello”.

	Average Distance	
Password	W_E	W_D
“hella”	0.27	0.99
“yello”	0.15	0.66
“henlo”	0.25	0.95
Average	0.22	0.87

Experiment 4: Brute force attack: The proposed system's cipher codes are: key={N, S}, where N is the number of possible seeds in C++, and S is the number of used seeds, which are 144 for weight initialization and 14 for input data shuffling during 14 epochs (the average number of epochs required to learn the model). Hence, for this experimental work, the key space approximates 10^{761}.

The proposed system's weight codes include synaptic weight values WE and WD. That is, weights = {E, D}.After testing with various user passwords, the approximate ranges for WE and WD are [-0.50; 0.45] and [-1.74; 1.69], respectively. Then, E = 95 and D = 393. This produces a weighted space approximation of $8.56*10^{324}$.

The suggested key and weights space analysis yields amounts of 10^{761} and 10^{324}, making the system exceedingly difficult to crack using brute-force methods.

4. Conclusion

We investigate the use of an autoencoder, a deep neural network, for addressing cryptography challenges. Our goal is to create a new encryption technology that combines artificial neural networks with cryptography to its full potential. We tested the suggested approach on real-world text files of varying sizes. Experiments were conducted to assess the system's robustness against brute force attacks.

Experiments show that the suggested system outperforms DES, 3DES, AES, and RSA for encrypting text files ranging from 15 KB to 90 KB. The suggested method performs similarly to standard systems for files ranging from 153KB to 312KB in size. The proposed system's performance diminishes when data files exceed 868 KB in size.

Furthermore, traditional ciphers are easily broken. For instance, DES can be broken with 243 or 211 plain strings [2] [14]. Similarly, the faster attack for 3DES takes 290 single encryptions (16). Similarly, a key attack for AES requires 299.5 time and data complexity [5]. The suggested system is exceedingly difficult to crack using brute-force methods, with key and weight space analysis yielding amounts of 10^{761} and 10^{324}, respectively. This is especially significant when contrasted to the number of attacks required for traditional ciphers.

Furthermore, classical ciphers are easily compromised. For example, DES can be broken with 243 or 211 simple strings [2] [14]. Similarly, the faster attack for 3DES uses 290 single encryptions (16). Similarly, an AES key attack takes 299.5

millisecond's and is data-complexity-dependent [5]. The proposed system is extremely difficult to crack using brute-force approaches, with key and weight space analyses producing values of 10761 and 10324, respectively. This is particularly noteworthy when compared to the number of attacks necessary for standard ciphers.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Ahmed, K., Rihan, S.D.: A Performance Comparison of Encryption A Performance Comparison of Encryption Algorithms AES and DES. International Journal of Engineering Research & Technology (IJERT) 4(December), 151–154 (2015)

[2] Alani, M.M.: Neuro-cryptanalysis of des and triple-DES. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7667 LNCS(PART 5), 637–646 (2012). https://doi.org/10.1007/978-3-642-34500-5 75

[3] Albawi, S., Mohammed, T.A., Al-Zawi, S.: Understanding of a convolutional neural network. Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017 2018-January(August), 1–6 (2018). https://doi.org/10.1109/ICEngTechnol.2017.8308186

[4] Basu, S., Karuppiah, M., Nasipuri, M., Halder, A.K., Radhakrishnan, N.: BioInspired Cryptosystem with DNA Cryptography and Neural Networks. Journal of Systems Architecture (2019). https://doi.org/10.1016/j.sysarc.2019.02.005, https://doi.org/10.1016/j.sysarc.2019.02.005

[5] Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5912 LNCS, 1–18 (2009). https://doi.org/10.1007/978-3-642-10366-7 1

[6] Caliskan, A., Yuksel, M.E.: Classification of coronary artery disease data sets by using a deep neural network. The EuroBiotech Journal 1(4), 271–277 (2017). https://doi.org/10.24190/issn2564-615x/2017/04.03

[7] Hadke, P.P., Kale, S.G.: Use of Neural Networks in cryptography: A review. IEEE WCTFTR 2016 - Proceedings of 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare pp. 1–4 (2016). https://doi.org/10.1109/STARTUP.2016.7583925

[8] Hu, F., Wang, J., Xu, X., Pu, C., Tao, P.: Batch Image Encryption Using Generated Deep Features Based on Stacked Autoencoder Network. Mathematical Problems in Engineering 2017, 256–261 (2017). https://doi.org/10.1155/2017/3675459

[9] Jogdand, R.M., Bisalapur, S.S.: Design of An Efficient Neural Key Generation. International Journal of Artificial Intelligence & Applications 2(1), 60–69 (2011). https://doi.org/10.5121/ijaia.2011.2105

[10] Kinzel, W., Kanter, I.: Neural cryptography. ICONIP 2002 - Proceedings of the 9th International Conference on Neural Information Processing: Computational Intelligence for the E-Age 3(November), 1351–1354 (2002). https://doi.org/10.1109/ICONIP.2002.1202841