Enhancing Data Security through Machine Learning-based Key Generation and Encryption

Abhishek Saini
USIC&T, Guru Gobind Singh Indraprastha University, India
abhishek.saini.00902@gmail.com

Ruchi Sehrawat
USIC&T, Guru Gobind Singh Indraprastha University, India
ruchi.sehrawat@gmail.com

Received: 2 March 2024 | Revised: 26 March 2024 | Accepted: 3 April 2024

ABSTRACT

In an era marked by growing concerns about data security and privacy, the need for robust encryption techniques has become a matter of paramount importance. The primary goal of this study is to protect sensitive information during transmission while ensuring efficient and reliable decryption at the receiver's side. To generate robust and unique cryptographic keys, the proposed approach trains an autoencoder neural network based on hashing and optionally generated prime numbers in the MNIST dataset. The key serves as the foundation for secure communication. An additional security layer to the cryptographic algorithm passing through the first ciphertext, was employed utilizing the XORed and Blum-Blum-Shub (BBS) generators to make the system resistant to various types of attacks. This approach offers a robust and innovative solution for secure data transmission, combining the strengths of autoencoder-based key generation and cryptographic encryption. Its effectiveness is demonstrated through testing and simulations.

Keywords-AES; autoencoder; DES; ElGamal; key generation; neural cryptography; RSA

I. INTRODUCTION

In an increasingly interconnected world, the transmission of sensitive information has become an integral part of indivinduals' daily life. Whether it involves personal messaging, confidential documents, or financial transactions, promising data security and privacy during its journey from the sender to the receiver has become an essential concern [1]. As digital communication continues to grow, so do malicious threats' methods, which aim to compromise the confidentiality and integrity of data [2-3]. To address these obstinate security challenges, this study presents an innovative approach that integrates neural networks and cryptographic algorithms [4]. By utilizing the input, hidden, and output layers of the network structure, this technique ensures secure data transfer over unreliable channels, enhancing the overall system resilience.

The foundation of the proposed method is the use of autoencoder neural network models, which are renowned for their ability to learn compact representations of complex data. Although traditional encryption methods often rely on manually generated cryptographic keys, this study introduces a novel method in which the cryptographic keys are autonomously generated by an autoencoder, an artificial neural network designed for data compression and reconstruction, thus introducing an additional security layer to the encryption process. The training of the autoencoder model is accomplished implementing the MNIST dataset, a well-known collection of handwritten digital images. By exploiting the autoencoder's ability, a unique cryptographic key for symmetric and asymmetric algorithms is generated. This key, shaped by the properties learned from the neural network, serves as the foundation for secure data transmission. The proposed encryption process consists of two crucial stages. The plaintext undergoes bitwise XOR operation and subsequent encryption deploying symmetric cryptographic algorithms, such as Advanced Encryption Standard (AES) [5] and Data Encryption Standard (DES) [6], and asymmetric cryptographic algorithms, such as Rivest-Shamir-Adleman (RSA) [7] and ElGamal [8]. This two-layer strategy ensures that the data are kept extremely secure.

This groundbreaking approach to secure data transmission incorporates autoencoder-based key generation with traditional symmetric or asymmetric encryption methods, striking a balance between enhanced data security and minimization of key interception risks. The merger of XOR-based preprocessing with a cryptographic algorithm promises robust protection against unauthorized access, thus offering a versatile solution for secure communication across various domains, from secure messaging platforms to confidential file-sharing networks. This study aims to contribute to the constantly changing field of data security at a time when protecting user privacy is of the utmost importance.

II. RELATED WORKS

In [9], an autoencoder neural network model was used in the proposed cryptographic approach. The sender creates a training set that consists of a binary representation of alphanumeric characters and special symbols, including padding and a shared secret key. This dataset is securely sent to the receiver, who also utilizes it to train their autoencoder neural network model. In the encryption process, the plaintext is transformed, including converted to ASCII and binary forms, and encrypted employing the trained autoencoder with the secret key. The resulting ciphertext is transmitted to the receiver and the whole process is reversed in the decryption phase, where the ciphertext is passed to the decoder of the autoencoder with the same secret key to generate the original plaintext. In [10], a method was proposed to secure the encryption of color images, addressing the issues of encryption speed and bandwidth usage. The encryption scheme was a combination of a convolutional autoencoder, DNA, and chaos theory. For key generation, the color image was converted into a grayscale image and passed to the SHA-256 hash function to obtain the key. This scheme proposed a dimensional conversion module that reduced the dimension of the input color image engaging the convolution autoencoder while maintaining image fidelity. Furthermore, this study deployed DNA and multiple chaotic sequences along with the substitution box of the encryption algorithm.

In [11], an approach based on Ant Colony Optimization (ACO) with Encryption Curve Cryptography (ECC)-based steganography was presented with a specific focus on improving the security of medical image management, particularly concentrating on a diabetic retinopathy dataset. The proposed method combined two main techniques, encryption with ECC by maximizing the Peak Signal-to-Noise Ratio (PSNR) to maintain image quality during transmission, and key generation with ACO, where the encryption key is created by finding the best coefficients to hide information in the integer wavelet transform of the source image. This strategy aims to secure complex regions of the image while applying LSB steganography efficiently. ACO-based key generation effectively creates a secure key stream for binary image encryption by coordinating ant agents to produce pheromones. In [12], a 3D cube algorithm was introduced, which employed deep neural network learning to facilitate the creation of symmetric keys without sharing the pre-shared key between systems. The 3D cube algorithm hashed and XORed patterns related to shuffle operations in an initially arranged 3D cube to create a 256-bit secret key, providing an impressive level of security. This approach used AES-256-based symmetric key encryption and the DeepCube AI technique for shuffled pattern resolution. The former achieved the highest recognition rate with four shuffles and three hidden layers in the DNN model. In [13], a bioinspired cryptography system that used hybrid key generation methods was presented. This system utilized DNA encoding, translation, and transcription processes for encryption and decryption based on the Central Dogma of Molecular Biology (CDMB) and the Central Molecular Biology (CMB) principle. The Whale Optimization Algorithm (WOA) was applied to optimize the weight vectors to achieve an effective encryption and decryption process. Additionally, a Bidirectional Associative Memory Neural Network (BAMNN) was put into service for key generation and storage during the encryption and decryption processes. In [14], a private key was generated using the features of color images. The frequencies of RGB coolers in the image were calculated, and the maximum frequencies of each color were multiplied by the number of frequencies for each of their colors. The result represented the key, which was converted into binary form. The generated key remains unchanged even after hiding a text in the image.

In [15], the BAMNN model, which saves memory space by remembering and recreating sets of keys repeatedly in a recurrent manner, was implemented for key generation. A bioinspired cryptosystem based on CDMB was proposed for encryption and decryption. The encryption process involves converting binary input into DNA bases, simulating the genetic coding process, followed by transcription and translation to generate protein-based ciphertext. The decryption procedure involved the reverse of the encryption process. In [16], two distinct disciplines, machine learning and DNA-based encryption, were combined. A modified Hebbian neural network was used for message encryption, and a ciphering neural network was employed to encode plaintext utilizing LNN-generated keys and DNA techniques to enhance data confusion and compression. The cipher neural network transformed the plaintext into ASCII code and engaged ciphered LNN keys as weights.

ALGORITHM 1: PROPOSED METHOD

```
#Key Generation
Generate-Key ():
  Random-Integer = Autoencoder (MNIST-Dataset)
  Hash-Integer = SHA512 (Random-Integer, size)
  If (Required Prime Number):
    prime-integer = Convert-prime (Hash-Integer)
    return (prime-integer)
  else:
    key = Hash-Integer
    return key

#Encryption
Encryption (plaintext):
  Key = Generate-Key ()
  Tweak-value = BBS-Generator ( )
  #Obtain first cipher text
  Hash-key = SHA_512 (Tweak-value, size)
  Cipher_text_1= XOR (plaintext, Hash-key)
  # Pass through Symmetric/ Asymmetric
  Cryptographic Algorithm (e.g. AES, DES, RSA,
    ElGamal)
  Cipher_text_2 =
    Cryptographic-Algorithm (Cipher_text_1, Key)
  Return (Cipher_text_2, Tweak-value, Key)

#Decryption
Decryption (Cipher_text_2, Tweak-value, Key):
  # Pass through Symmetric/ Asymmetric
      Cryptographic Algorithm
  Decrypted_Cipher_text_1 = Cryptographic-Algorithm
    (Cipher_text_2, Key)
  #Obtain Final Plain text
  Hash-key = SHA_512 (Tweak-value, size)
  Original Plain Text =
    XOR(Decrypted_Cipher_text_1, Hash-key)
  return (Original Plain Text)
```

III. THE PROPOSED METHOD

This study introduces a novel approach to secure data transmission by integrating cryptographic algorithms with the strength of machine learning through an autoencoder model. The MNIST dataset is implemented for secure key generation, and the entire approach encompasses several distinct phases, as shown in Figure 1 and the above pseudocode.

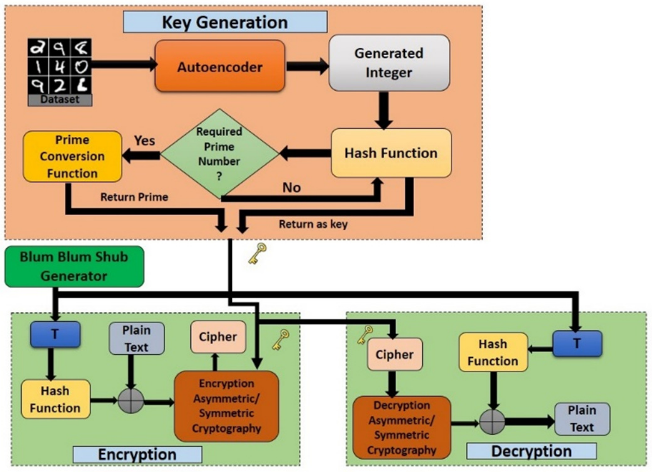

Fig. 1. Flowchart of the proposed method.

A. Key Generation

The main objective of this approach lies in the generation of a highly secure cryptography key. This approach generates a distinctive and reliable cryptographic key using the wide range of patterns and digits in the MNIST dataset. This phase deals with the key distribution, as well as with ensuring the confidentiality of the key. The following processes are deployed to create a key:

- An appropriate autoencoder architecture [17-18] is designed and implemented. The model is trained on the MNIST dataset to obtain a high dimensional integer representation based on the learned representations, ensuring their uniqueness and security, as illustrated in Figure 2.

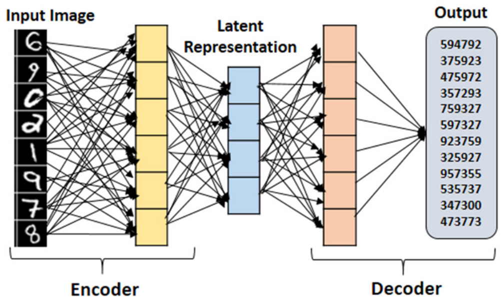

Fig. 2. Proposed Autoencoder Model.

- SHA-512, a secure and robust hashing algorithm [19], is applied to the integer representation to produce a hash value between 8 and 512 bits, per user requirements, that can be computed as:

Initialize:

$$bytes\_value = UTF\_8(key\_str) \quad (1)$$

Hashing:

$$hash\_res = H(bytes\_value) \quad (2)$$

$$Byte\_key = hash\_res[:key\_size\,/\,8] \quad (3)$$

Conversion:

$$Key = bytes\_to\_integer(Byte\_key) \quad (4)$$

where *key_str* is an integer received from the autoencoder, *bytes_value* contains the converted byte value of *key_str*, *H*(*x*) represents the SHA-512 hash function, and *key_size* is defined by the user. Up to a 512-bit long key can be generated, which is stored in *Byte_key* in bytes form, and the *Key* stores the converted integer value of *Byte_key*. Users have the option to convert the hash value into a prime number. The following pseudocode gives the next prime of the given hash value.

ALGORITHM 2: NEXT PRIME CALCULATION

```
next prime (n):
  if n < 2:
    return 2
  if n == 2:
    return 3
  if n % 2 == 0
    n = n + 1
  else
    n + 2
  while True:
    if is prime (n):
      return n
    n += 2
```

If the user needs a key for symmetric cryptography then the generated hash value will go as the key. In addition, if the user wants prime numbers for asymmetric cryptography, then the next prime number of the hash value will be passed.

B. Encryption

This study adopts the properties of the tweakable block cipher model [20] to strengthen the security and versatility of the proposed cryptographic framework. A BBS generator [21-22] is employed to produce a 32- or 64-bit number using the following procedure.

ALGORITHM 3: BBS GENERATOR

```
$n = p*q$
where $p$ & $q$ is a large prime number such that
  p ≡ q ≡ 3 mod 4
Select random seed value $S_0$ such that gcd($S_0$, $n$) = 1
$X_0 = (S_0)^2 \bmod n$
for $i$ =1 to ∞
        $X_i = (X_{i-1})^2 \bmod n$
        Append $Xi$ to the list $L$
Where $X_i$ is the pseudo random number
```

This produces a list of multiple random numbers, while a random number is taken from the list as tweak *T*, which is passed to the hash function SHA-512 *H*(*T*). The plaintext data

P undergoes a bitwise XOR operation with $H(T)$ to obtain the first cipher text C_1, represented as:

$$C_1 = H(T) \oplus P \quad (5)$$

The first ciphertext C_1 is passed to the cryptographic algorithm to create the final ciphertext C_2. In the case of symmetric cryptography, this algorithm uses the generated key from the key generation model. For asymmetric cryptography, the algorithm deploys the generated prime numbers from the key generation model to produce the final key. The cipher text C_2 will be:

$$C_2 = E(K, C_1) \quad (6)$$

where E represents the encryption. The sender transmits the cipher text C_2 along with the key K and the tweak value T.

C. Decryption

To retrieve the original plaintext data from the encrypted cipher text, the recipient will perform inverse operations, including symmetric or asymmetric decryption, as well as XOR decryption. At first, the recipient will determine which type of encryption algorithm was applied and then perform the decryption operation to calculate the recovered text RT:

$$RT = D(K, C_2) \quad (7)$$

To get the original text, tweak T is passed to the SHA-512 hash function, i.e. $H(T)$. The recovered text RT is bitwise XORed with $H(T)$ to obtain the original plain text P:

$$P = H(T) \oplus RT \quad (8)$$

IV. RESULTS & DISCUSSION

The proposed method was implemented in Python on a machine equipped with an Intel Core i3-2328M processor running at 2.20 GHz, 4 GB of RAM, and 256 GB SSD. An autoencoder neural network model was trained on the MNIST dataset to generate keys, with an input dimension of 784 and an encoding dimension of 32, consisting of two dense layers, one for encoding and one for decoding. The MNIST dataset was loaded and preprocessed, transforming the pixel values to the [0, 1] range. The training process utilizes the Adam optimizer and binary cross-entropy loss over 10 epochs, indicating its effective data reconstruction capabilities. The model achieved a final training loss of 0.0974 and a validation loss of 0.0953. As displayed in Figure 3, these values emphasize the capability of the model, exhibiting a minimal reconstruction error. In the end, the model produced approximately a 7554-digit long integer key, which passed through the compression function.

A. Keyspace Analysis

Keyspace analysis is a crucial aspect in addressing the security and effectiveness of a cryptographic system. Keyspace refers to the total number of possible keys that can be utilized in a cryptographic algorithm and determines the size and complexity of the key, providing information about the security and usability of the encryption scheme. For example, a 512-bit key is capable of producing a total of 2^{512} distinct keys, which is a huge key space and so it is difficult to search the full key space thoroughly with a brute-force attack. Suppose that an attacker can test one trillion (10^{12}) keys per second. Then, it would take an impractical amount of time, such as 4.23254×10^{141} years to search the entire 512-bit key space. The proposed neural network model produces up to 512-bit dynamic keys and provides an exceptional level of security, making it significantly more difficult for adversaries to predict the key, which is highly suitable for safeguarding sensitive data in various applications.

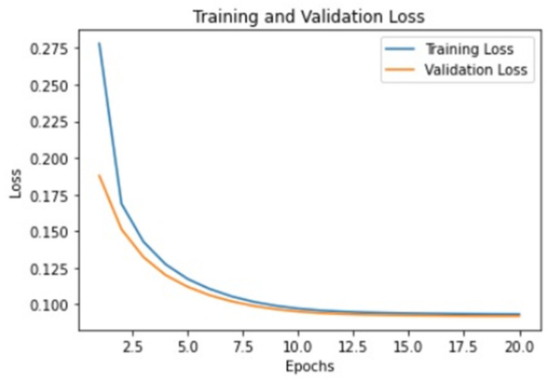

Fig. 3. Model loss.

B. Binary Entropy Analysis

In information theory, a higher entropy value shows better randomness. Binary entropy is a measure of the amount of uncertainty or randomness in a binary system. It is also known as binary information entropy. For a binary key with probability P being state 1 and probability (1 - P) being state 0, binary entropy $H(P)$ can be calculated using:

$$H(P) = -[P * log_2(P) + (1 - P) * log_2(1 - P)] \quad (9)$$

A binary entropy of approximately 1.00 bits indicates that the binary key is essentially a random sequence, which means a high level of information content in the key [22-23]. This study evaluated the entropy of different key lengths, ranging from 16- to 512-bit dynamic binary keys, and the resulting entropy of different key lengths is nearly 1.00, as observed in Figure 4, representing maximum unpredictability and a truly random sequence. These results highlight the significance of key length in cryptographic applications and the inherent connection between the key length and the degree of randomness, with a 512-bit key providing the maximum level of unpredictability.

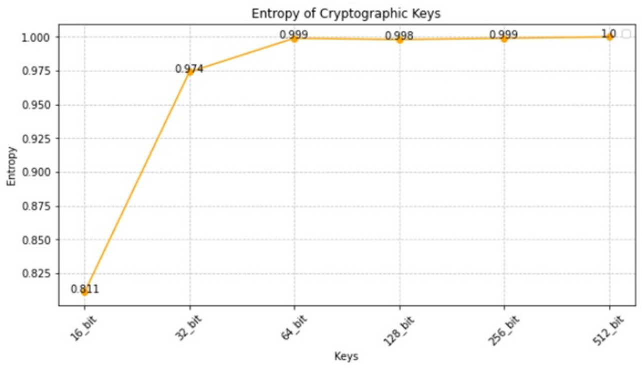

Fig. 4. Entropy analysis of cryptographic keys.

C. NIST Test

The NIST randomness tests are designed to evaluate the quality of random sequences, including 15 different statistical tests, each designed to detect specific patterns or biases in a sequence of binary data. The latter are important in various fields, such as cryptography applications, secure communications, and random number generation. Eleven (11) NIST tests were performed on 128, 256, and 512-bit long binary keys. As portrayed in Table I, the p-value of the tests was greater than 0.01. This indicates that the generated keys are random. Figure 5, provides information on the performance of various key lengths and demonstrates the distribution of p-values for each key size. Certain NIST tests, like the binary rank test, the overlapping template, the universal statistical test, and the linear complexity test, were not performed, as they require longer sizes of binary digits up to 10^6 to satisfy the randomness requirements [24].

TABLE I. NIST TEST RESULTS

Tests	***p*-value**			**Conclusion**
	128-bit sequence	**256-bit sequence**	**512-bit sequence**	
Frequency test	0.59588	0.61707	0.72367	Random
Frequency test within a block	0.59588	0.53526	0.36536	Random
Runs test	0.83985	0.79032	0.99559	Random
Longest run	0.65552	0.93063	0.94052	Random
Discrete Fourier transform (spectral) test	0.33039	0.81854	0.93535	Random
Non-overlapping template matching test	0.9999	0.99998	0.50922	Random
Serial test (p-value-1)	0.49896	0.84134	0.69077	Random
Serial Test (p-value-2)	0.49853	0.75967	0.85568	Random
Approximate entropy test	1.0	1.0	1.0	Random
Cumulative sums test (forward)	0.89202	0.74584	0.81876	Random
Cumulative sums test (backward)	0.65476	0.97420	0.92314	Random
Random excursions test (average)	0.885645	0.703873	0.447325	Random
Random excursions variant test (average)	0.524083	0.173436	0.370241	Random

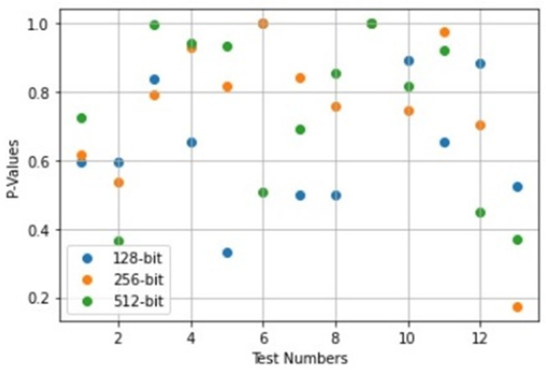

Fig. 5. Relationship between distinct keys and p-value.

D. Avalanche Effect

An algorithm's performance and security are often evaluated by its ability to exhibit a robust avalanche effect, certifying that small changes in the plaintext or encryption key result in significant variations in the ciphertext. Change in ciphertext should be at least 50% [25]. Table II reveals that the proposed cryptosystem with the existing algorithms modified the ciphertext by 50% on average after changing one bit in the plaintext and key, indicating that the proposed cryptosystem has stronger unpredictability than that of the existing algorithms.

TABLE II. AVALANCHE EFFECT

Algorithm		**Original plaintext: WELCOME! (64-BIT)**			
		Plaintext sensitivity (Change in plaintext)		**Key Sensitivity (Change in key)**	
Proposed cryptosystem with existing algorithm	**Existing algorithm**	**Proposed**	**Existing**	**Proposed**	**Existing**
DES		57.81%	45.16%	53.13%	0
AES		50.0%	44.44%	53.25%	50.79%
RSA		50.53%	7.63%	52.24%	55.93%
ElGamal		50.88%	48.11%	50.96%	50.04%

E. Execution Speed Analysis

The execution time of each algorithm revealed the efficiency, indicating how well the algorithm works. Table III compares the average encryption and decryption times of the proposed cryptosystem with the existing ones. Figures 6 and 7 disclose that the proposed cryptosystem is comparably excellent.

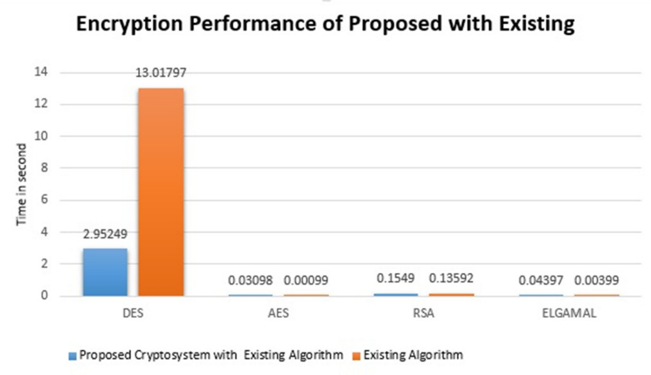

Fig. 6. Encryption performance comparison of the proposed and existing cryptosystems.

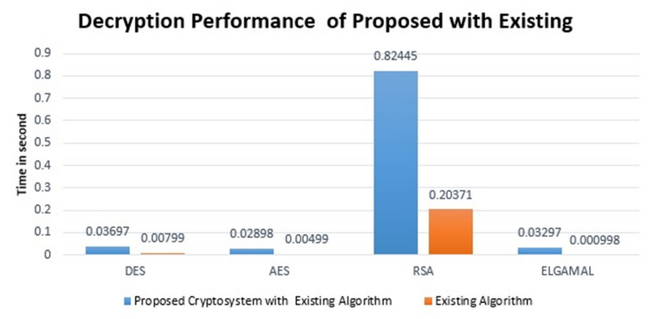

Fig. 7. Decryption performance comparison of the proposed and existing cryptosystems.

TABLE III. ENCRYPTION AND DECRYPTION TIMES OF PROPOSED AND EXISTING CRYPTOSYSTEMS

Algorithm		**Original Plaintext: WELCOME! (64-BIT)**			
		Encryption Time		**Decryption Time**	
Proposed cryptosystem with existing algorithm	**Existing algorithm**	**Proposed**	**Existing**	**Proposed**	**Existing**
DES		2.95249	13.01797	0.03697	0.00799
AES		0.03098	0.00099	0.02898	0.00499
RSA		0.15490	0.13592	0.82445	0.20371
ElGamal		0.04397	0.00399	0.03297	0.000998

V. CONCLUSION

In an era defined by the ubiquitous exchange of digital information, the need to protect sensitive data during transmission has never been more pressing. This study presents an innovative and scalable solution for information security that uses modern technology to increase data protection. The proposed method combines the strength of modern cryptographic practices with the feature extraction capabilities of the autoencoder model implementing the MNIST dataset to transform it into a key generation source. Utilizing SHA-512 for hashing and applying both symmetric (AES and DES) and asymmetric (RSA and ElGamal) encryption with tweakable block cipher characteristics, this dual-layered encryption system guarantees robust data security and exhibits superior performance compared to existing systems, namely AES, DES, RSA, and ElGamal. The addition of a BBS-based random number generator makes it a promising solution for data security in various applications. The system demonstrated resistance to interception and tampering and encompassed key aspects of cryptosystem performance. The binary entropy analysis manifested a high level of unpredictability, with an average value of 0.96. Keyspace analysis revealed a well-distributed key space, such as the 512-bit key size boasting a total key space of 2^{512}, ensuring robust defense against brute-force attacks. The avalanche effect test for both plaintext and key sensitivity averaged 52% compared to the existing algorithms. For plaintext sensitivity, the proposed cryptosystem with RSA showed 50.53%, while the existing RSA algorithm showed 7%, indicating a significant improvement. Similarly, for DES, the key sensitivity analysis is 0%, while the proposed cryptosystem with DES produced 53.13%, indicating increased security. The encryption and decryption times of the proposed cryptosystem are lower than those of the existing cryptosystems, except for the encryption time of the DES algorithm. However, it is important to note that the decryption time performance of the existing cryptosystems is approximately 80% higher compared to that of the proposed cryptosystem, which demonstrates excellent performance. Additionally, the proposed cryptosystem passed the NIST tests with key sizes of 128, 256, and 512 bits, highlighting its applicability for various security uses and its efficacy in safeguarding confidential information during transmission. Finally, the performance bar graph emphasized the system's efficiency in terms of encryption and decryption speed.

ACKNOWLEDGMENT

The authors express their gratitude for the IPRF award granted by Guru Gobind Singh Indraprastha University, New Delhi, India.

REFERENCES

[1] R. J. Rasras, Z. A. AlQadi, and M. R. A. Sara, "A Methodology Based on Steganography and Cryptography to Protect Highly Secure Messages," *Engineering, Technology & Applied Science Research*, vol. 9, no. 1, pp. 3681–3684, Feb. 2019, https://doi.org/10.48084/etasr.2380.

[2] A. H. Al-Omari, "Lightweight Dynamic Crypto Algorithm for Next Internet Generation," *Engineering, Technology & Applied Science Research*, vol. 9, no. 3, pp. 4203–4208, Jun. 2019, https://doi.org/10.48084/etasr.2743.

[3] A. S. Alshammari, "Comparison of a Chaotic Cryptosystem with Other Cryptography Systems," *Engineering, Technology & Applied Science Research*, vol. 10, no. 5, pp. 6187–6190, Oct. 2020, https://doi.org/10.48084/etasr.3745.

[4] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. New York, NY, USA: Pearson, 2008.

[5] A. M. Abdullah, "Advanced encryption standard (AES) algorithm to encrypt and decrypt data," *Cryptography and Network Security*, vol. 16, no. 1, 2017.

[6] D. Coppersmith, "The Data Encryption Standard (DES) and its strength against attacks," *IBM Journal of Research and Development*, vol. 38, no. 3, pp. 243–250, May 1994, https://doi.org/10.1147/rd.383.0243.

[7] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Oct. 1978, https://doi.org/10.1145/359340.359342.

[8] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985, https://doi.org/10.1109/TIT.1985.1057074.

[9] V. Sagar and K. Kumar, "Autoencoder Artificial Neural Network Public Key Cryptography in Unsecure Public channel Communication," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 11, pp. 4023–4032, Sep. 2019, https://doi.org/10.35940/ijitee.K1456.0981119.

[10] F. Ahmed *et al.*, "A DNA Based Colour Image Encryption Scheme Using A Convolutional Autoencoder," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 19, no. 3s, Oct. 2023, https://doi.org/10.1145/3570165.

[11] S. Karthikeyini, R. Sagayaraj, N. Rajkumar, and P. K. Pillai, "Security in Medical Image Management Using Ant Colony Optimization," *Information Technology and Control*, vol. 52, no. 2, pp. 276–287, Jul. 2023, https://doi.org/10.5755/j01.itc.52.2.32532.

[12] J. Jin and K. Kim, "3D CUBE Algorithm for the Key Generation Method: Applying Deep Neural Network Learning-Based," *IEEE Access*, vol. 8, pp. 33689–33702, 2020, https://doi.org/10.1109/ACCESS.2020.2973695.

[13] M. Indrasena Reddy, A. P. Siva Kumar, and K. Subba Reddy, "A secured cryptographic system based on DNA and a hybrid key generation approach," *Biosystems*, vol. 197, Nov. 2020, Art. no. 104207, https://doi.org/10.1016/j.biosystems.2020.104207.

[14] W. A. Shukur, "A Proposed Method for Generating a Private Key Using Digital Color Image Features," *International Journal of Applied Engineering Research*, vol. 12, no. 16, pp. 6235–6240, 2017.

[15] S. Basu, M. Karuppiah, M. Nasipuri, A. K. Halder, and N. Radhakrishnan, "Bio-inspired cryptosystem with DNA cryptography and neural networks," *Journal of Systems Architecture*, vol. 94, pp. 24–31, Mar. 2019, https://doi.org/10.1016/j.sysarc.2019.02.005.

[16] S. A. Kadum, A. Y. Al-Sultan, and N. A. Hadie, "Data protection based neural cryptography and deoxyribonucleic acid," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 3, pp. 2756-2764, Jun. 2022, https://doi.org/10.11591/ijece.v12i3.pp2756-2764.

[17] U. Michelucci, "An Introduction to Autoencoders." arXiv, Jan. 11, 2022, https://doi.org/10.48550/arXiv.2201.03898.

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

[19] C. Dobraunig, M. Eichlseder, and F. Mendel, "Analysis of SHA-512/224 and SHA-512/256," in *Advances in Cryptology – ASIACRYPT 2015*, Berlin, Heidelberg, 2015, pp. 612–630, https://doi.org/10.1007/978-3-662-48800-3_25.

[20] M. Liskov, R. L. Rivest, and D. Wagner, "Tweakable Block Ciphers," in *Advances in Cryptology — CRYPTO 2002*, Santa Barbara, CA, USA, 2002, pp. 31–46, https://doi.org/10.1007/3-540-45708-9_3.

[21] L. Blum, M. Blum, and M. Shub, "A Simple Unpredictable Pseudo-Random Number Generator," *SIAM Journal on Computing*, vol. 15, no. 2, pp. 364–383, May 1986, https://doi.org/10.1137/0215025.

[22] W. Stallings, Cryptography and Network Security: Principles and Practice, Global Ed, 8th edition. Harlow, UK: Pearson, 2022.

[23] M. Imdad, S. N. Ramli, and H. Mahdin, "Increasing Randomization of Ciphertext in DNA Cryptography," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 10, pp. 423–429, 2021, https://doi.org/10.14569/IJACSA.2021.0121047.

[24] L. E. Bassham *et al.*, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," National Institute of Standards and Technology, Sep. 2010. [Online]. Available: https://www.nist.gov/publications/statistical-test-suite-random-and-pseudorandom-number-generators-cryptographic.

[25] B. Banerjee, "Avalanche effect: A judgement parameter of strength in symmetric key block ciphers," *International Journal of Engineering Development and Research*, vol. 7, no. 2, pp. 116–121, 2019.