Mathematics and Statistics 12(5): 484-493, 2024
DOI: 10.13189/ms.2024.120510 http://www.hrpub.org

Elevating Data Entropy and Efficiency through Symmetric Cryptographic technique Based on Finite State Machine

A Yasmin, R Venkatesan*, K Gaverchand

Department of Mathematics, College of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur, 603203, Tamil Nadu, India

Received June 5, 2024; Revised August 16, 2024; Accepted August 24, 2024

Cite This Paper in the Following Citation Styles

***(a)**: [1] A Yasmin, R Venkatesan, K Gaverchand, "Elevating Data Entropy and Efficiency through Symmetric Cryptographic technique Based on Finite State Machine," Mathematics and Statistics, Vol.12, No.5, pp. 484-493, 2024. DOI: 10.13189/ms.2024.120510*

***(b)**: A Yasmin, R Venkatesan, K Gaverchand (2024). Elevating Data Entropy and Efficiency through Symmetric Cryptographic technique Based on Finite State Machine, Mathematics and Statistics, 12(5), 484-493. DOI: 10.13189/ms.2024.120510*

Abstract Globally, cryptosystems employ sophisticated techniques for encrypting and decrypting sensitive information, heavily relying significantly on secure secret keys. The generation of a randomized, secure 256-bit secret key is of paramount importance for maintaining data accuracy, confidentiality and fortification against a spectrum of security threats, making it challenging for potential hackers to anticipate key sequences. This proposed study endeavors to produce highly randomized ciphertext (CT), reduce time complexity as well as assure the efficiency and security of the cryptosystem. The key generation methodology involves considering the recipient's credentials and employing a codebook to strategically generate the secret key. The encryption process incorporates a finite state machine (FSM) to introduce randomness into the CT. Implementation is conducted in Python, with a meticulous evaluation of time complexity and a comparison with existing cryptographic methods to assess efficiency. The randomness test in the National Institute of Standards and Technology (NIST test) evaluates the proposed scheme, yielding a mean p-value of 0.6, surpassing the expected value. Nevertheless, Key space analysis is conducted to thwart unauthorized access through brute force attacks, revealing computational challenges inherent in the evaluation of 2^{256} possibilities. The security analysis examines the strength and vulnerabilities of various attacks on the model, thereby bolstering the model resilience against potential threats. Aggregate results robustly validate the elevated security and efficacy of the proposed model when contrasted with its antecedents.

Keywords Finite Automata, Encryption, Decryption, Key Generation, Randomness

1 Introduction

In the modern era, mathematics exerts a pervasive influence across various sectors, playing an indispensable role in decision-making, modeling, network analysis, transportation, construction, graphics, banking and the vast expanse of the internet. The internet, a fundamental aspect of daily life, functions as a vital channel for cashless transactions, communication and internet banking. The imperative need to secure, protect and maintain the confidentiality of online interactions has become paramount in today's society. While the internet provides various pros, it also introduces a range of cons, particularly in the form of security threats posed by intruders. Intruders can exploit vulnerabilities, resulting in unauthorized access and compromising sensitive information. Consequently, safeguarding against such intrusions becomes imperative to maintain the integrity and confidentiality of online data. Cryptography [1] is a pivotal technique employed to uphold the CIA triad (confidentiality, integrity and availability) of confidential data on the internet. Through encryption, it ensures that only authorized individuals can access and safeguard information. The overarching objective of cryptography is to address the unique requirements of legitimate users, fostering secure communication by implementing techniques that authenticate and authorize access [2]. This pivotal role positions cryptography as a cornerstone in ensuring the security and reliability of sensitive data, contributing significantly to the overall integrity and confidentiality of digital information in diverse online interactions.

Automata theory [3], a prominent branch of computer science, investigates abstract machines and computational process, focusing on understanding the capabilities and limitations of models like finite automata, pushdown automata for algorithm design. In the domain of cryptography, automata theory

assumes a pivotal role, serving as a foundational framework for the development and scrutiny of secure communication systems. Cryptographic algorithms heavily leverage concepts from automata theory, with finite automata playing a decisive role in crafting encryption and decryption process, enhancing the resilience of cryptographic systems against unauthorized access and cyber threats. Beyond algorithmic design, automata theory contributes to the creation of structured frameworks for cryptographic protocols, ensuring the secure transmission and storage of sensitive information during digital transactions. Its integration into cryptography underscores its essential role in fortifying the security and confidentiality of digital interactions in today's intricately connected world.

The objective of this paper is to develop a resilient data encryption technique by integrating cryptography with finite automata, generating a 256-bit secret key using receiver credentials. The encryption process is strengthened through several security layers, with a particular emphasis on the security enhancement brought about by finite automata in the overall proposed model. To assure security, efficiency and randomness, various assessments are performed on our proposed model.

The structure of the remaining sections in this research article is as follows. Section 2 offers a summary of existing studies related to the proposed model. Section 3 outlines the basic definitions, while Section 4 provides a brief description of the proposed model, accompanied by an illustration. Section 5 is dedicated to evaluating the results and analyzing the model. Finally, Section 6 concludes the proposed model and outlines the future directions for this article.

2 Literature Survey

Numerous researchers have innovated diverse cryptographic techniques through the application of automata theory. Some of the articles have been reviewed as follows.

Rachna Navalakhe and Harsha Atre propose a cryptographic scheme in [4] that employs FSM and explicit recurrence relations through Bernoulli and Lucas numbers, ensuring heightened security with multiple key sets and diverse security levels. In a related context, authors in [5, 6] espouses a lightweight and highly secure cryptographic solution grounded in a sequential state machine, emphasizing computational efficiency and smaller key sizes. Ayush Mittal and Ravindra Kumar Gupta present a cryptographic technique in [7] that utilizes logical operators and finite fields, employing XOR operations with a 6×6 matrix and finite field GF(2^5) for enhanced security. This method, involving key sharing and bit-wise ASCII conversion, demonstrates resilience against known plaintext attacks and brute force techniques. Shengtao Geng et al. in [8] introduce an image encryption algorithm that employs block scrambling, discrete wavelet transform and a FSM, effectively enhancing image security through multi-layered encryption process.

Addressing critical security concerns in information technology services, Shandilya et al. [9] and Gopalan et al. [10] emphasize efficient digital data protection across platforms and explore the applications of automata in cryptography. Mohammad Mazyad et al. [11] propose an FSM and turbo code-based encryption algorithm with controlled puncturing and key generation from pre-existing data. This approach achieves high-security grayscale images through recurrence relations and LU decomposition. Agrawal and Vemuri [12] prsented a novel methodology aimed at harmonizing security and power efficiency in finite state controller encoding.

The generation of highly secured secret keys using cellular Automaton rules, with a focus on enhanced security and comparative analysis against established cryptosystems, is detailed in [13]. Additionally, the increasing importance of message encryption in electronic communication, employing FSM's and recurrence matrices for enhanced security, is underscored in [14, 15, 16]. Datta et al.[17] explored the intricacies of a cryptosystem based on chaotic dynamics. Dhanapal et al. [18] investigated quantum computing potential to surpass supercomputers and proposed a hybrid classical quantum AES-128 circuit on Qiskit, achieving reduced qubit count and faster encryption.

In the realm of DNA cryptography [19], Pramod et al. [20, 21] introduced a mealy and moore machine-based DNA cryptosystem, ensuring randomization and NIST test result analyses, though lacking in-depth security analysis. Kuldeep et al. [22] propose a cryptographic technique based on LU decomposition with automata, while Abhishek et al. [23] recommend a DNA-based encryption technique for secure cloud storage. The study includes strength and standard criteria analysis, with the caveat that the utilization of 2-D matrix operations increases temporal complexity. Suyel et al. [24] present a DNA cryptosystem using a Mealy machine, implementing security and comparative assessments. The authors in [25, 26, 27, 28, 29] explore various applications of cryptography including fingerprint based cryptosystems, IOT applications, blockchain and QR coding

The survey reveals that while various models offer enhanced security features, shortcomings persist, including smaller key sizes, high computational requirements, susceptibility to security attacks and a lack of consideration for the randomness of the model. This paper aims to systematically address these concerns, formulating a secure model utilizing FSM that collectively mitigates these identified issues.

The key contributions of the proposed model are outlined as follows:

1. Introducing an innovative cryptographic methodology employing the FSM.

2. Generation of a 256-bit symmetric secret key to ensure high-level security.

3. Assessment of the randomness of the proposed model through the NIST test.

4. Validation of the model's overall efficiency through various performance analysis.

3 The Basic aspects of FSM and cryptography

3.1 Finite Automata

FSM [30] also called finite automaton (FA), serves as a computational mathematical model utilized for delineating system behavior, transitioning from one state to another based on a designated transition function and specific inputs. FSM can be represented through graphical diagrams, tabular formats and mathematical functions. FSM is broadly categorized into two types. FSM with outputs and FSM without outputs. This computational framework offers a versatile means of illustrating dynamic state changes in a system, contributing to a thorough comprehension of its computational dynamics and responsiveness to inputs.

In this proposed model, we create a random FSM illustrated in Figure 2. The model processes a specific input string and the resulting states are regarded as the output string. Figure 2 comprises a finite set of states denoted as $Q = \{A, B, C, D, E\}$, a finite alphabet set $\sum = \{0, 1\}$, an initial state $I = A$, a finite set of outputs $O = \{A, B, C, D, E\}$ and a randomly constructed transition function δ, where $\delta(A, 0) = B$, $\delta(A, 1) = C$, $\delta(B, 0) = C$, $\delta(B, 1) = E$, $\delta(C, 0) = E$, $\delta(C, 1) = D$, $\delta(D, 0) = D$, $\delta(D, 1) = E$, $\delta(E, 0) = A$ and $\delta(E, 1) = B$.

Table 1. Input to next state / ouput state for generating FSM

State	**Input** 0 Next state/Output	**Input** 1 Next state/Output
A	B	C
B	C	E
C	E	D
D	D	E
E	A	B

3.2 Cryptography

Cryptography [31] is the process of securing information against unauthorized access, modification, interception and similar threats. It employs techniques for encoding/encrypting and decoding/decrypting confidential messages, ensuring that only legitimate recipients can revert the original information. The two primary classifications of cryptography are private (symmetric) key cryptography and public (asymmetric) key cryptography. The key distinction lies in the use of the same key for both encryption and decryption in private key cryptography, while different keys are employed for these process in public key cryptography. Cryptography serves various purposes, encompassing confidentiality, integrity, authentication and non-repudiation, collectively known as the CIA triad. The fundamental components of cryptography include plaintext, ciphertext, encryption algorithm, decryption algorithm and key.

The proposed model utilizes a 256-bit symmetric key cryptography for the precise and efficient encryption of textual data.

4 Proposed Model

The proposed scheme can be succinctly summarized as follows: Initially, the receiver securely transmits its credentials to the sender in an encrypted format. Subsequently, the sender conducts a validation process to verify the authenticity of the received credentials. Upon the receiver's request for data, the sender initiates the encryption process by generating a 256-bit secret key, utilizing the receiver's credentials and code book 2. This key plays a pivotal role in both the encryption and decryption process within the proposed model, ensuring a robust security foundation. The encryption and decryption procedures are further fortified by the incorporation of a randomly generated FSM. Finally, the sender discreetly transmits the CT, along with necessary parameters, to the recipient in an encrypted format, utilizing the recipient's public key and the sender's private key. The workflow of the proposed model is illustrated in Figure 1.

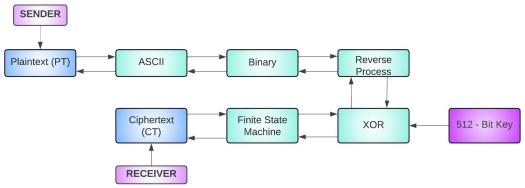

Figure 1. Workflow of the proposed scheme

4.1 Designing a FSM

The FSM was generated by the sender employing the subsequent steps.
Step:1 Define the FSM parameters

Begin by identifying the number of states, inputs and outputs using Table 1, denoted as A, B, C, D, E for states, 0 and 1 for inputs and A, B, C, D, E for outputs.
Step:2 State initialization

Introduce the states A, B, C, D, E as nodes and designate the initial state A using a right arrow.
Step:3 Transition diagram construction

Draw directed loops and edges connecting the states based on the Table 1.
Step 4: Edge labeling

Label the edges with the corresponding input values and consider the next state as the output values.
Step 5: Complete FSM diagram

Complete the FSM diagram construction using Table 1 until reaching the final state E.
The sender will transmit the randomly generated FSM given in Figure 2 to the receiver in an encrypted form, ensuring secure data transfer.

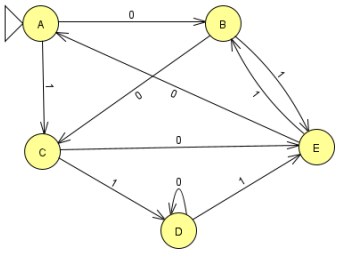

Figure 2. Finite state machine

Table 2. Random Conversion of Numbers into String of Alphabets

Numbers	String of alphabets	Numbers	String of alphabets
0	QE	5	UZ
1	PL	6	AW
2	FB	7	XP
3	UX	8	AY
4	YJ	9	GQ

4.2 Key Generation Process

Adhering to Kerckhoff's principle, emphasizing the supremacy of the secret key in bolstering overall model security, the strategy focuses on expanding the key space to diminish adversary success rates. In alignment with this principle, the key size is deliberately set at 256 bits, representing a computationally formidable challenge for potential intruders to decipher. Upon receiving the receiver's credentials, the sender randomly generates a string of length 10 and executes a swapping process between each pair of 2 bits within the 10-bit string. Subsequently, each character in the string is converted into its corresponding ASCII value and these ASCII values are concatenated into a unified string. Using a codebook 2, each numerical value in the string is then converted into its binary representation. The resulting binary values are concatenated and the middle 256-bit binary value is designated as the secret key, denoted as K. This K serves as the symmetric key for encryption and decryption.

4.3 Data Encryption

To secure the confidential information, the sender initiates the encryption process by converting the PT into ASCII values, followed by their conversion into equivalent 8-bit binary representations. The binary values are then amalgamated to form a binary string, which undergoes a reverse transformation. Subsequently, XOR operation is applied between the 256-bit secret key K and the reversed binary string. The resulting XOR output is utilized as input for a randomly generated FSM, producing its states as a new string. This ultimate string is recognized as the CT, composed of a series of characters.

Following this encryption process, the sender meticulously transmits the CT alongside essential parameters to the recipient. This transmission is executed with utmost security, employing the receiver's public key and the sender's private key.

4.4 Data Decryption

Upon receiving the data, the recipient employs their private key and the sender's public key to retrieve the CT along with the necessary parameters. The recipient then undertakes an inverse encryption process to restore the PT from the CT. Initially, utilizing a FSM, the decrypted data is transformed into a string of binary values. Following this, XOR operation is conducted between the binary string and the secret key K to yield a new binary string. The new string undergoes a reversal transformation, with each 8-bit segment converted into its corresponding ASCII value. Ultimately, the ASCII values are translated into their corresponding characters, thus restoring the original confidential data PT.

4.5 Algorithm

4.5.1 Algorithm: 1 Generation of 256-bit secret key K

1. Generate a string K_1, of length 10 utilizing receiver's credentials
2. Apply a swapping process on K_1, by exchanging each pair of characters to derive K_2
3. Convert K_2 into ASCII values and concatenate these values to form a single string K_3
4. Transform K_3 using Table 2 to produce K_4
5. Convert each character of K_4 into its corresponding 8-bit binary representation to create K_5
6. Extract the middle 256-bit segment from K_5 to serve as the secret key K

4.5.2 Algorithm: 2 Encrypting confidential data to yield CT

Input: Plaintext
Output: Ciphertext

1. Initiate with the confidential data PT
2. Convert PT into ASCII values
3. Transform these ASCII values into their binary representations and concatenate them into a single string
4. Rearrange the positions of the string using a reverse process
5. Perform an XOR operation between the key K and the reversed string
6. Pass the XOR result into a FSM to derive new string, which represents the corresponding states and results in the CT, composed of characters

7. End

4.5.3 Algorithm : 3 Retrivel of PT

Input: Ciphertext
Output: Plaintext

1. Begin with encrypted data CT
2. Use a FSM to transform CT into its binary representation
3. Perform an XOR amid binary value and key
4. Reverse the order of the result from the previous step
5. Convert each byte of the reversed order into its ASCII value
6. Translate these ASCII values into characters to recover the PT
7. End

4.6 Illustration of Proposed Model

This section exemplifies the proposed scheme through a sample implementation.

4.6.1 Key Generation

The sender executes the subsequent process to generate the key.
a) Utilizing the receiver's credentials encompassing name, date of birth, contact number, email ID and passport ID, the sender arbitarly constructs a string of length 10
$K_1 = xyz * 123ABC$
b) Execute pair swapping on K_1 to yield K_2.
$K_2 = yx * z21A3CB$
c) K_2 is transformed into its ASCII value and then concatenated
$K_3 = 12112042122504965516766$
d) Table 1 is employed to convert each numerical value into its corresponding string
$K_4 = PLFBPLPLFBQEYJFBPLFBFBUZQEYJ$
$GQAWUZUZPLAWXPAWAW$
e) Each character of K_4 undergoes conversion into its binary representation
$K_5 = 01010000010011000100011001000001001010000010$
$01100010100000100110001000110010000100101000101010$
$00101010110010100101001000110010000100101000001010$
$01100010001100100001001000110010000100101010101010$
$11010010100010100010101011001010010100100011101010$
$10001010000010101011101010101010110100101010101010$
$11010010100000100110001000001010101110101100001010$
$10000010000010101011101000001010101111$
f) The middle 256-bits of K_5 is designated as the secret key K
$K = 01001100010001100100001001010001010001010101$
$10010100101001000110010000100101000001001100010$
$01100100001001000110010000100101010101011010010$
$00010100010101011001010010100100011101010001010$
$00010101011101010101010110100101010101011010010$
00000100110001000001
The aforementioned 256-bit key is employed in both the encryption and decryption process.

4.6.2 Data Encryption

The sender encrypts the data using the following steps.
a) The sender is required to encrypt the PT
$PT = Mathematics$
b) The PT is transformed into ASCII values
$ASCII = 77, 97, 116, 104, 101, 109, 97, 116, 105, 99, 115$
c) The ASCII values are converted to their corresponding 8-bit binary representations
$Binary = 01001101011000010111010001101000011001$
$01101101011000010111010001101001011000110111001 1$
d) The binary values are transformed into a reversed string
$Reversed\ Binary = 11001110110001101001011000101111$
$01000011010110110101001100001011000101110100001 10$
10110010
e) Extract K' from K, where K represents the initial n bits of a 256-bit key, and n is the length of reversed binary string
$K' = 01001100010001100100001001010001010001010101$
$11001010010100100011001000010010100000100110 0$
f) The XOR operation is executed between K' and the reversed binary
$XOR = 10000010100000001101010001111111110000111$
$11011111110110001010000011011001101011011111110$
g) Utilize a FSM for converting XOR values into characters. Employ the XOR value as FSM input, where its states manifested as characters, serve as the output, eventually concatenated into a string
$FSM = ACEABCEBCDDDDDDDDEBCDDEAB$
$CDEBEBEBEBCEABEBEBEACDEBEBEACD$
$DDDEACEABCEBEACDDDEBCDDEBCDEBE$
$BEBC$
Therefore, FSM is regarded as CT

4.6.3 Data Decryption

The recipient is required to execute the inverse process of encryption in order to retrieve the PT
a) The sender possesses the CT
$CT = ACEABCEBCDDDDDDDDEBCDDEABC$
$DEBEBEBEBCEABEBEBEACDEBEBEACDD$
$DDEACEABCEBEACDDDEBCDDEBCDEBEB$
EBC
b) The receiver utilizes CT as input to an FSM for binary string extraction and the executes the inverse process by employing Table 1 for FSM execution. The exemplary execution process is as follows.
1. Convert each character of CT into a binary string. For the first 2 characters i.e., AC, where the start state is A and the next state is C, using Table 1, the input value from state A to C is 1 by Table 1. Consequently, A is translated into 1. $A \rightarrow 1$,

The initial bit of binary string is 1
2. Considering the subsequent state E, with the current state as C and using Table 1, the input value for the transition from C to E is 0. This leads to the conversion of C into 0, resulting in the binary string = 10
3. Sequentially applying the aforementioned procedure to each subsequent character in CT yields the binary string that evolves based on the prescribed state transitions delineated in Table 1
$Binary\ string = 10000010100000001101010001111111110000111101111111011000101000001101100110101101111110$
c) The XOR is executed between binary string and K'
$XOR = 1100111011000110100101100010111010000011010110010110100110000101100010111010000110101100100$
d) Create the reversed XOR string
$Reversed\ string = 0100110101100001011101000110100011001010110110101100001011101000110100101100011011110011$
e) Translate every 8-bit segment of the reversed string into its respective ASCII values
$ASCII = 77, 97, 116, 104, 101, 109, 97, 116, 105, 99, 115$
f) Transform the ASCII values into their corresponding characters
Decrypted text = $PT = Mathematics.$

5 Results and Analysis

5.1 Experimental Setup

To execute the proposed methodologies, the Python software has been deployed on an HP laptop, featuring a cutting-edge 12th-generation Intel Core i5 processor and a spacious 512GB SSD. Python valued for readability and versatility, is chosen for its robust capabilities across diverse domains, from web development to data science, due to its elegant syntax and extensive libraries. The execution of the proposed model takes place within the Python environment, leveraging various sets of textual data acquired from the 20 newsgroups dataset [32].

In Figure 3 , the application window features an intuitively designed interface allowing users to input the original message into a designated text box. Initiating the encryption process is achieved by pressing the "Encrypt" button, displaying the output in the encrypted text box. Conversely, the "Decrypt" button is used to transform encrypted text, revealing the original message in the decrypted text box.

5.2 Complexity Analysis

Complexity analysis typically assesses both runtime and space requirements of an algorithm, facilitating comparisons and evaluations. It aids in predicting the most suitable algorithm by examining the relationship between processing time and input size. In this context, we analyze the time complexity of proposed algorithms (key generation, encryption and decryption) with the existing algorithms. The overall time complexity of the proposed model is $O(n)$, indicating a linear rise in execution time with input size, while the encryption and decryption algorithms of the Pramanik model [33], along with the key generation process of the Paul model [34], exhibit a time complexity of $O(n^2)$, signifying a quadratic correlation between input size and processing time. Consequently, the proposed model exhibits significantly reduced time complexity compared to previous models.

5.3 Key Space Analysis

Key space analysis in cryptography entails an evaluation of the total possible keys within an algorithm, offering insights into its resilience against brute-force attacks. However, in standard cryptographic models like AES, DES and Triple DES, key sizes are conventionally set to 256, 56 and 168 bits, respectively. In contrast, our model employs a more robust key size of 256 bits, resulting in a key space KS.

$$KS = L^N = 2^{256}$$

where L represents the count of potential symbols in each key position, such as 2 for binary or 16 for hexadecimal and N signifies the total number of key positions.

This engenders an expansive key space, exponentially elevating the complexity of exhaustive key search attempts and fortifying security. A comprehensive key space analysis is indispensable for appraising the cryptographic system's strength and ensuring steadfast protection against unauthorized access or decryption endeavors. The utilization of 256-bit key size in our model guarantees a robust key space, making exhaustive key exploration computationally infeasible for adversaries.

5.4 Comparative Analysis

The comparative analysis in cryptography methodically evaluates the computational efficiency and security attributes of diverse cryptographic algorithms. This rigorous examination involves scrutinizing key operations, including encryption and decryption, with considerations for algorithmic intricacy and implementation. This paper specifically focuses on comparing the time complexity of the proposed model with existing models. This evaluation incorporates data from dataset [32], iterated up to 50 times and relies on the average of these iterations, are presented in Table 3 and Table 4. The graphical representation is illustrated in Figure 4 and Figure 5.

Table 3. Time comparative analysis for encrypting the data of different length (in ms)

PT Length	Paul Model[34]	Pramanik Model[33]	Proposed Model
100	49.87	35.76	22.64
200	86.32	54.98	41.28
300	132.16	79.42	74.32
400	176.88	118.65	99.07
500	217.37	142.09	129.67
Average	132.52	86.18	73.39

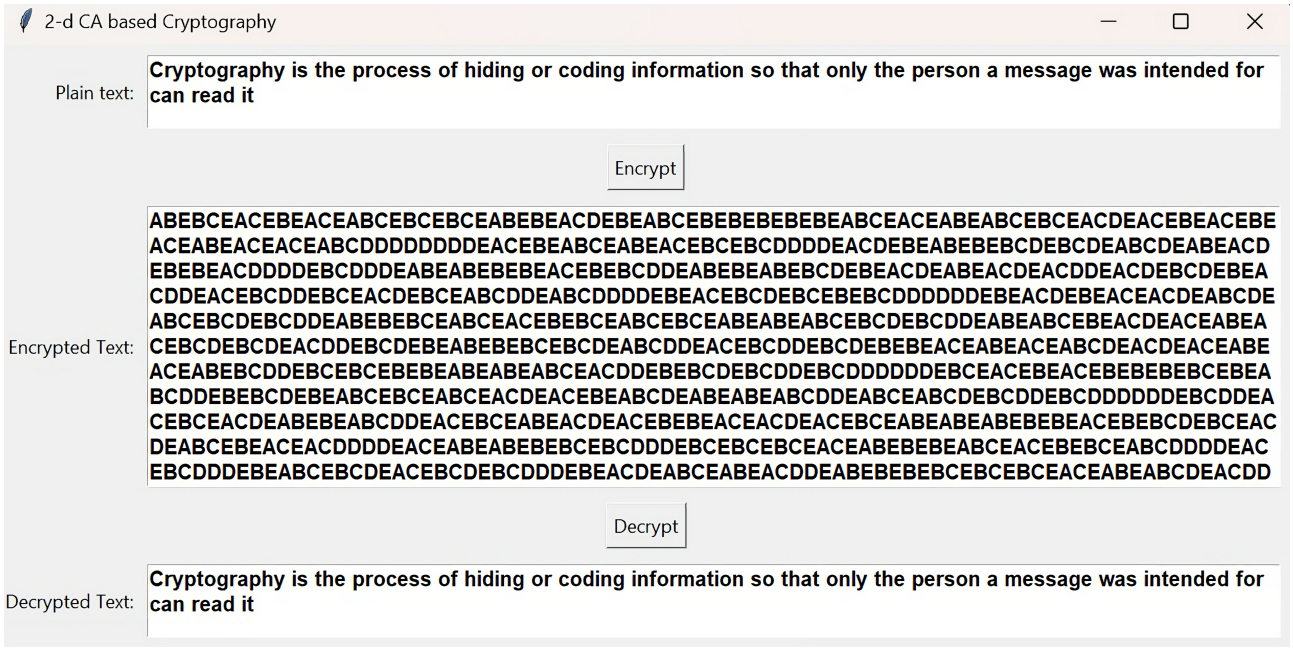

Figure 3. Illustration using python software

Table 4. Time comparative analysis for decrypting the data of different length (in ms)

PT Length	Paul Model[34]	Pramanik Model[33]	Proposed Model
100	54.87	41.67	27.49
200	91.72	79.23	46.09
300	137.87	97.87	61.49
400	171.32	123.09	78.41
500	208.90	172.84	103.74
Average	132.93	102.94	63.44

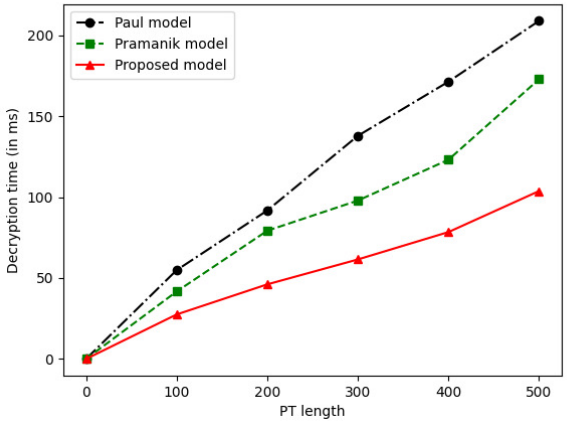

Figure 5. Comparative analysis for decryption

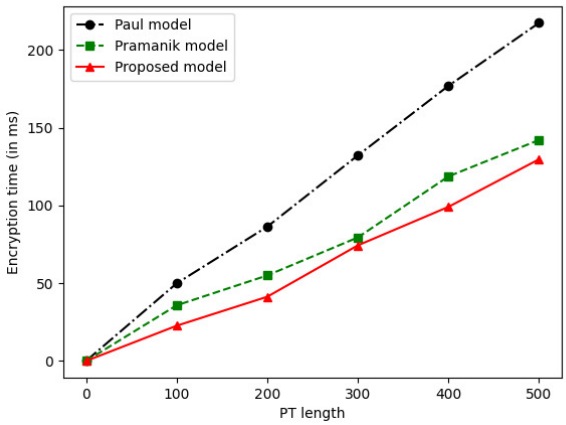

Figure 4. Comparitive analysis for encryption

5.5 Entropy Test

The entropy test in cryptography for CT assesses the randomness and unpredictability of encrypted data, playing a vital role in ensuring the robustness of cryptographic systems by detecting patterns or predictability in the CT. High entropy in the CT signifies increased security, while low entropy could indicate potential vulnerabilities. This test is crucial for validating the effectiveness of encryption algorithms and maintaining the integrity of secure communication. Regular entropy testing is essential to uphold the resilience of cryptographic protocols against evolving threats.

This paper utilizes NIST test suite's randomness assessments, encompassing frequency and block frequency to appraise the randomness of the proposed cryptographic scheme. These tests, executed on binary representations of CT, generate P-values, where success in randomness necessitates a P-

value surpassing 0.01. Table 5 delineates notations and descriptions for these assessments. In both tests, an identical set of PT data undergoes evaluation through 30 experiments, generating respective p-values. The outcomes reveal that the proposed model demonstrates higher p-values, affirming its superior randomness. Consequently, the proposed scheme exhibits robust randomness and resilience against entropy tests.

Table 5. Notations and descriptions of NIST Test analysis

Notation	Description
i	Length of binary string
S_i	Absolute value
N	Number of blocks in the input
erfc	Complementary error function
M	Number of a bits in a block
S_{obs}	Test static in frequency test
π	Proportion of ones in binary string
ϵ	Sequence of bits
π_n	Proportion of ones in the i^{th} block
igamc	Incomplete gamma function
P_{value}	Probability of test static
χ^2_{obs}	Test statistic in block frequency test

5.5.1 Frequency Test

The frequency test, synonymous with the monobit test, examines the balance between occurrences of 1's and 0's in a binary sequence, serving as a foundational metric for randomness assessment. P-values for this test are calculated through equation 1. The P-values derived from the CT of the proposed model are illustrated in Figure 6 and their averages are detailed in Table 6.

$$\begin{aligned} S_i &= \sum_{k=1}^{i} X_k \quad \forall\, X_k = 2\epsilon - 1 \\ S_{obs} &= \frac{|S_i|}{\sqrt{i}} \\ P_{value} &= erfc\left(\frac{S_{obs}}{\sqrt{2}}\right) \end{aligned} \tag{1}$$

5.5.2 Frequency Test within A Block

The block frequency test evaluates the distribution of ones across M-blocks, aiming for a frequency near M/2. The P-value computed through equation 2, gauges the performance of the CT randomness. Figure 6 visually portrays consistently heightened P-values for the proposed model, with the corresponding average delineated in Table 6.

$$\begin{aligned} \pi_n &= \frac{\sum_{j=1}^{M} \epsilon_{(n-1)} M + j}{M} \quad \forall\, 1 \le n \le N \\ \chi^2_{obs} &= 4M \sum_{n=1}^{N} (\pi_n - 1/2)^2 \\ P_{value} &= igamc\left(\frac{N}{2}, \frac{\chi^2_{obs}}{2}\right) \end{aligned} \tag{2}$$

Table 6. Average P-value of NIST test

Test	Average P-value
Monobit	0.672
Block frequency	0.731

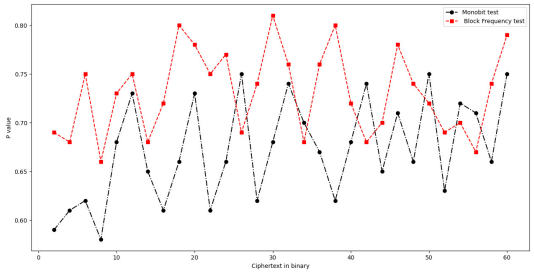

Figure 6. Entropy test

5.6 Security Analysis

Security analysis in cryptography ensures robustness against attacks and vulnerabilities, bolstering reliance and fostering innovation in digital security. The proposed cryptographic model undergoes rigorous evaluation against familiar cryptographic attacks, with the aggregate results demonstrating its resilience and affirming the strength and efficiency of the model.

5.6.1 Known Plaintext Attack

Attackers leverage information of PT-CT pairs to deduce the secret key or PT. However, the proposed model dependency on FSM and XOR operation with a secret key significantly impedes the extraction of pertinent information without access to the secret key, thereby enhancing data security.

5.6.2 Chosen Plaintext Attack

Adversaries have the capability to choose PT and observe corresponding CT to scrutinize the encryption process or deduce the key. However, the utilization of XOR operation with a secret key substantially hampers the adversary's capacity to extract confidential information, thereby bolstering the proposed model against such attacks.

5.6.3 Man-in-the-Middle Attack

Through intercepting communication between parties, adversaries endeavor to eavesdrop or manipulate data. However, the proposed scheme reliance on a secret key mitigates the risk of interception, as intercepted CT remains unintelligible without access to the key and FSM, thus reinforcing data confidentiality.

5.6.4 Differential Cryptanalysis

The proposed model utilizing a 256-bit secret key and randomization, combats differential cryptanalysis effectively. Adhering to Kerckhoff's principle, its complexity deters adversary

success. Incorporating XOR operation with the secret key enhances security against PT-CT pattern exploitation, while the inclusion of a FSM fortifies cryptographic resilience.

6 Conclusion and Future Scope

In the presented inquiry, an innovative cryptography paradigm was introduced, employing a randomly generated finite state machine (FSM). This methodology facilitated the formation of a 256-bit random key, subsequently applied in the cryptographic process. The randomness of the resulting ciphertext (CT) was methodically evaluated using the NIST test suite. The key space evaluations emphasized the resilience of the proposed model against brute-force attacks, showcasing its robustness. Comparative analysis demonstrated a notable improvement in time complexity compared to existing cryptographic algorithms. The security analysis ensures the model's resilience against various attacks. This study revealed a significant correlation amid cryptography and FSMs, showcasing a distinctive feature in its proficiency to generate a stochastic secret key, a pivotal component for fortifying cryptographic integrity.

These findings significantly impact secure data communication, enhancing encryption, integrity assurance, and confidentiality measures. The model excels with fortified security, heightened resilience, accelerated processing, reduced storage demands, and a sophisticated randomized secret key generation, which is commonly referred to as key management in cryptography.

In future endeavors, exploring diverse automata methods, including pushdown automata and Turing machines, aims to enhance the stability and efficiency of cryptographic models. Implementing these models on hardware accelerators such as GPUs and FPGAs is expected to expedite encryption and decryption processes and facilitate the application of the proposed method to larger datasets. Anticipated benefits include robust, innovative designs that advance cryptographic technologies and ensure sustained efficacy against evolving security challenges and emerging threats.

REFERENCES

[1] Kahate Atul, "Cryptography and Network Security," Tata McGraw Hill, 2003.

[2] Alia Mohammad, Tamimi Abdelfatah, Alallaf Almustawi Omaima, "Cryptography Based Authentication Methods," Proceedings of the World Congress on Engineering and Computer Science, vol. 1, 2014.

[3] Linz Peter, "An Introduction to Formal Languages and Automata," 4th ed, Narosa Publishing, 2009.

[4] Rachna Navalakhe, Harsha Atre, "Cryptographic Algorithms using FSM, Bernoulli and Lucas Numbers," South East Asian Journal of Mathematics and Mathematical Sciences, vol. 17, no. 3, pp. 17–28, 2021.

[5] Basappa B Kodada, Demian Antony D'Mello, "Symmetric Key Cryptosystem based on Sequential State Machine," IOP Conference Series, Materials Science and Engineering, vol. 1187, 012026, 2021. https://doi.org/10.1088/1757-899X/1187/1/012026.

[6] Domosi Pal, "A novel cryptosystem based on finite automata without outputs," In Automata, Formal Languages and Algebraic Systems, pp. 23-32, 2010. https://doi.org/10.1142/9789814317610-0002.

[7] Ayush Mittal, Ravindra Kumar Gupta, "Cryptographic Technique involving finite fields and Logical Operators," Indian Journal of Science and Technology, vol. 13, no. 3, pp. 316–32, 2020. https://doi.org/10.17485/ijst/2020/v13i03/148904.

[8] Geng Shengtao, Wu Tao , Wang Shida, Zhang Xuncai, Wang Yanfengs, "Image Encryption Algorithm based on Block Scrambling and Finite State Machine," IEEE, vol. 8, pp. 831–844, 2020. https://doi.org/10.1109/ACCESS.2020.3045101.

[9] Shandilya S. K., Datta A., Nagar A. K., "Learning-Based Cryptography. In: A Nature-Inspired Approach to Cryptology. Studies in Computational Intelligence," Springer, vol. 1122, 2023. https://doi.org/10.1007/978-981-99-7081-0_3.

[10] Gopalan N. P., Kumerasan G., "An Analytical study of Cellular Automata and its applications in Cryptography," International Journal of Computer Network and Information Security, vol. 12, pp. 45–54, 2017. https://doi.org/10.5815/ijcnis.2017.12.06.

[11] Hazzazi M. M., Budaraju R. R., Bassfar Z., Albakri A., Mishra S., "A Finite State Machine-Based Improved Cryptographic Technique," Mathematics, vol. 11, no. 10, 2225, 2023. https://doi.org/10.3390/math11102225.

[12] Agrawal R., Vemuri R., "Encoding of Finite-State Controllers for Graded Security and Power. Behavioral Synthesis for Hardware Security," Springer, 2022. https://doi.org/10.1007/978-3-030-78841-4_8.

[13] Gaverchand K., Venkatesan R., "A novel approach of 1-D cellular automata in cryptosystem," Mathematical Modelling of Engineering Problems, vol. 10, no. 6, pp. 2121–2126, 2023. https://doi.org/10.18280/mmep.100623.

[14] Krishna Gandhi B., Chandra Sekhar A., Srilakshmi S., "Cryptographic Scheme for Digital Signals using Finite State Machines," International Journal of Computer Applications, vol. 29, no. 6, pp. 61–63, 2011. https://doi.org/10.5120/3565-4904.

[15] Jyotirmie P. A., Chandra Sekhar A., Uma Devi S., "Application Of Mealy Machine And Recurrence Relations In Cryptography," International journal of engineering research and technology, vol. 2, no. 5, pp. 1286–1290, 2013.

[16] Karudaiyar G., Karthikeyan S., Sainath B., "Encryption and Decryption Scheme by Using Finite State Machine," Biosciences Biotechnology Research Asia, vol. 11, no. 3, pp. 1861–1865, 2014. https://doi.org/10.13005/bbra/1595.

[17] Datta A., Shandilya S. K., Nagar A. K., "Chaos Cryptography. In: A Nature-Inspired Approach to Cryptology. Studies in Computational Intelligence," Springer, vol. 1122, 2023. https://doi.org/10.1007/978-981-99-7081-0_7.

[18] Dhanapal A. D., Anantha Ramanujan S., Jeyalakshmi V., "Advanced Encryption Standard on a Quantum System," International Conference on Integrated Intelligence and Communication Systems, pp. 1–7, 2023. https://doi.org/10.1109/ICIICS59993.2023.10421474.

[19] Mukherjee P., Garg H., Pradhan C., Ghosh S., Chowdhury S., Srivastava G., "Best Fit DNA-Based Cryptographic Keys: The Genetoc Algorithm Approach," Sensors, vol. 22, no. 19, 7332, 2022. https://doi.org/10/3390/s22197332.

[20] Pramod Pavithran, Mathew Sheena, Namasudra Suyel, Singh Ashish, "Enhancing randomness of the ciphertext generated by DNA based cryptosystem and finite state machine," Cluster Computing, vol. 26, pp. 1035–1051, 2022. https://doi.org/10.1007/s10586-022-03653-9.

[21] Namasudra S, "Perspective of DNA Computing in Computer Science. Advances in Computers," Elsevier, vol. 129, pp. 1–387, 2023. https://doi.org/10.1016/bs.adcom.2022.08.001.

[22] Kuldeep Vayadande, Agarwal Kirti, Kabra Aadesh, Gangwal Ketan, Kinage Atharv, "Cryptography using Automata Theory," ITM Web of conferences ICAECT, vol. 50, pp. 1–9, 2022. https://doi.org/10.1051/itmconf/20225001007.

[23] Majumdar A., Biswas A., Majumder A., "A novel DNA-inspired encryption strategy for concealing cloud Storage," Frontiers of Computer Science, vol. 15, 153807, 2021. https://doi.org/10.1007/s11704-019-9015-2.

[24] Suyel Namasudra, Pramod Pavithran, Sheena Mathew, Pascal Lorenz, "A Novel Cryptosystem based on DNA cryptography and randomly generated mealy machine," Computer and security, vol. 104, 102160, 2021. https://doi.org/10.1016/j.cose.2020.102160.

[25] Hashem M. I., Alibraheemi K., "Literature Survey: Biometric Cryptosystems Based on Fingerprint Processing Techniques," IEEE International Conference on Data Science and Intelligent Computing, 2022, pp. 198–201. https://doi.org/10.1109/ICDSIC56987.2022.10076184.

[26] Bhardwaj C., Garg H., Shekhar S., "An Approach for Securing QR code using Cryptography and Visual Cryptography," IEEE International Conference on Computational Intelligence and Sustainable Engineering Solutions, pp. 284–288, 2022. https://doi.org/10.1109/CISES54857.2022.9844332.

[27] Fares N., Wang B., Bakiras S., "Design and implementation of certificateless cryptography for IOT applications," IEEE International Midwest Symposium on circuits and systems, pp. 933–937, 2023. https://doi.org/10.1109/MWSCAS57524.2023.10405921.

[28] Venugopal L. K., Rajaganapathi R., Birjepatil A., Raja S. E., Subramaniam G., "A Novel Information Security Framework for Securing Big Data in Healthcare Environment Using Blockchain," Engineering Proceedings, vol. 59, no. 1, 107, 2023. https://doi.org/10.3390/engproc2023059107.

[29] Lu X., Sami H. U., Guler B., "SCALR: Communication-Efficient Secure Multi-Party Logistic Regression," IEEE Transactions on Communications, vol. 72, no. 1, pp. 162–178, 2024. https://doi.org/10.1109/TCOMM.2023.3308954.

[30] Hopcroft J. E., Ullman J. D., "Introduction to Automata, Languages and Computation," Narosa Publishing House, 1987.

[31] William Stallings, "Cryptography and Network Security," 3rd ed, Pearson Education, 2003.

[32] Lang K., 20 Newsgroups, Accessed on May 2024, https://qwone.com/jason/20Newsgroups.

[33] Pramanik Sabari, Setua Sanjit, "DNA cryptography," IEEE 7^{th} International Conference on Electrical and Computer Engineering, pp. 551–554, 2012. https://doi.org/10.1109/ICECE.2012.6471609.

[34] Paul Sanchita, Anwar Tausif, Kumar Abhishek, "An innovative DNA cryptography technique for secure data transmission," International Journal of Bioinformatics Research and Applications, vol. 12, no. 3, pp. 238, 2016. https://doi.org/10.1504/IJBRA.2016.078235.