MJoSHT
Malaysian Journal of Science, Health & Technology
[mjosht.usim.edu.my]

Article

Enhancing Cryptographic Resilience through Symmetric Encryption Algorithm Utilizing Variable Length Chromosomes Genetic Algorithm

Ahmed Jobaer[1], Nur Hafiza binti Zakaria[1,2], Farida Ridzuan[1,2], and A H Azni[1,2]

[1]*Faculty of Science and Technology, Universiti Sains Islam Malaysia, Nilai 71800, Negeri Sembilan, Malaysia.*

[2]*CyberSecurity and Systems Research Unit, Faculty of Science and Technology, Universiti Sains Islam Malaysia, Nilai 71800, Negeri Sembilan, Malaysia.*

Correspondence should be addressed to:

Nur Hafiza binti Zakaria; mzhafiza@usim.edu.my

Article Info
Article history:
Received:15 August 2025
Accepted: 11 February 2026
Published:15 Mac 2026

Academic Editor:
Nurzi Juana Mohd Zaizi

Malaysian Journal of Science, Health & Technology

MJoSHT2025, Volume 11, Special Issue on the 5th International Conference on Recent Advancements in Science and Technology (ICoRAST 2025): Responsible Artificial Intelligence – Advancing Science and Technology for Humanity
eISSN: 2601-0003

https://doi.org/10.33102/mjosht.522

***Abstract*— Symmetric cryptography, particularly the Advanced Encryption Standard (AES), is widely used for secure data transmission due to its computational efficiency and strong encryption structure. However, limited internal randomness in AES makes it susceptible to advanced cryptanalytic attacks. To address this limitation, this research proposes an enhanced encryption approach that integrates Genetic Algorithm (GA) techniques into the AES framework to enhance randomness. The GA employs variable-length chromosomes and entropy-based fitness evaluation to evolve dynamic binary outputs through selection, crossover, and mutation operations. These evolved outputs are used to introduce controlled randomness into the encryption process, resulting in unpredictable and robust ciphertext. The expected outcome includes improved increased randomness, and stronger resistance to differential and linear attacks. In conclusion, the proposed GA-AES enhanced model offers a mechanism to strengthen symmetric encryption by introducing evolutionary randomness, making it more secure for modern data protection needs.**

***Keywords*— Symmetric Cryptography; Advanced Encryption Standard (AES); Genetic Algorithm; Enhanced Encryption; randomness.**

I. INTRODUCTION

In the modern era of digital communication, ensuring data confidentiality and integrity through robust cryptographic mechanisms is of critical importance. Among symmetric encryption algorithms, the Advanced Encryption Standard (AES) remains a cornerstone due to its efficiency, speed, and widespread adoption across applications [1-3]. However, the unchanging nature of AES components—such as predetermined substitution layers (S-Box) and transformation matrices (MixColumns)—limits its adaptability and introduces vulnerabilities to advanced cryptanalytic techniques, including differential and linear attacks [4, 5]. The predictable internal structures in standard AES can potentially be exploited if the key or internal transformations become compromised [6].

To overcome these limitations, this research introduces an enhanced cryptographic model that integrates Genetic Algorithm (GA) techniques into the AES framework [7, 8]. Rather than using GA for key generation, this study utilizes evolved binary offspring from GA to design new AES components, particularly a substitution box (S-Box) [2, 9]. Specifically, the proposed model explores three enhancement pathways: (1) designing an S-Box based on GA-generated offspring [10], (2) constructing a customized MixColumns matrix from the evolved sequences, and (3) modifying both components to maximize unpredictability and diffusion [1, 3]. The GA process applies variable-length chromosomes, entropy-based fitness functions, and genetic operators like crossover and bit-flip mutation to produce cryptographically resilient 128-bit offspring used in AES transformation logic [11, 12].

This integration of evolutionary computing with symmetric encryption aims to move beyond rigid designs by introducing genetic diversity into AES's internal operations [4]. The expected result is an encryption system with improved avalanche effect, higher entropy, and greater resistance to cryptanalytic and brute-force attacks [1, 2]. The work on the MixColumns transformation and other AES components will be addressed in future developments to further enhance AES's cryptographic strength [1, 6].

A. Research Objectives

1. To identify the critical components and optimal parameter configurations of Genetic Algorithms (GA) for implementing crossover and mutation operations using variable-length chromosomes.
2. To design an enhanced symmetric encryption model that integrates GA-evolved variable-length outputs into core AES components such as the S-Box and MixColumns transformations.
3. To evaluate the cryptographic strength of the proposed enhanced model based on randomness properties.

B. Research Questions

1. What are the essential genetic algorithm components and parameter settings required to effectively implement crossover and mutation using variable-length chromosomes?
2. How can an enhanced GA-AES encryption algorithm be designed by incorporating variable-length chromosome outputs into internal AES components to enhance randomness and security?
3. How effective is the proposed cryptographic framework in ensuring resilience, randomness, and efficiency against modern cyber threats?

II. LITERATURE REVIEW

This section provides a systematic review of 12 studies (2014–2024) on enhancing AES security through advanced S-Box design and the integration of Genetic Algorithms (GA), analyzing their methodologies, strengths, limitations, and applicability to the proposed GA-AES enhanced framework. The reviewed works illustrate how dynamic and key-dependent S-Boxes, chaotic maps, and GA-based key evolution significantly improve the nonlinearity, diffusion, and resistance of AES against differential and linear cryptanalysis, yet they also reveal persistent challenges in computational overhead, implementation complexity, and scalability. This research addresses these gaps by proposing a unified GA-driven model that generates customized S-Box and MixColumns transformations, introducing adaptive internal diversity to strengthen symmetric encryption. The accompanying tables categorize key findings into entropy enhancement, genetic optimization, S-Box nonlinearity, and cryptanalytic resistance, establishing the foundation for the novel contributions in robust and efficient AES encryption.

Table I shows 17 recent studies (2014–2024) on enhancing the security of AES through Genetic Algorithm (GA)-based approaches, emphasizing key findings and methodologies relevant to the development of a GA-AES enhanced framework. Recent studies have demonstrated the effectiveness of integrating GA techniques into cryptography, particularly in generating dynamic S-Boxes and optimizing other internal transformations like MixColumns, which significantly improve AES's resistance to both cryptanalysis and brute-force attacks [4][7][13]. These findings reflect the growing interest in enhancing traditional cryptographic algorithms with evolutionary computing techniques to overcome AES's limitations, specifically its reliance on static, predictable components.

TABLE I. LITERATURE REVIEW TABLE (2014–2025)

Ref	Domain	Key Size	Algorithm Used	Fitness Score Formula	Best Fittest Selection	Crossover Method	Mutation Technique	Strengths	Weaknesses
[7]	Sym. Crypt	48 bits, 128 bits	Genetic Algorithm (GA)	Gap Test & K-S Test	Max fitness + Hamming distance	One-point	Bit flip	High Randomness	High Computation Time
[9]	Stream Cipher	Variable (LFSR-based)	GA + LFSR	Randomness tests	Non-repeating sequence generation	Custom (nibble shift)	Two-stage mutation	Long Key Sequence	Complexity
[3]	Sym. Crypt.	Variable	GA	Entropy-based Fitness	Fitness threshold satisfaction	Ring-type (one-point)	Modular arithmetic	Strong Entropy	Processing Overhead
[8]	Asym. Crypt. (ECC)	159–521 bits	ECC + GA	Distance to target ECC point	Tournament selection	Uniform crossover	Random mutation	Small Key Size, Fast ECC	Complexity
[5]	DNA Crypt.	Variable	DNA-based OTP & RSA	Not GA (biological encoding)	N/A	N/A	N/A	High Density Encoding	Complex Lab Procedures
[6]	Sym. Crypt.	Variable	GA	Entropy-based Fitness	Fitness proportionate selection	One/two/unif orm crossover	Bit inversion	High Flexibility	Increased Computational Load
[7]	Sym. Crypt.	Variable	GA	Entropy-based Fitness	Random pairing	Multi-point crossover	Bit flip	High Randomness	Time Complexity
[8]	Sym. Crypt. (Cloud)	128 bits	GA + Caesar cipher	None specified (simple XOR-based)	N/A	Random point crossover	Bit flip	Simplicity	Weak Security (XOR only)
[12]	Crypt. Review	N/A	N/A	N/A	N/A	N/A	N/A	Comprehensive Comparison	No Implementation
[4]	AES Security Enhanceme nt	128/192/2 56 bits	Advanced S-Box design + GA	N/A	N/A	N/A	N/A	High Nonlinearity, Dynamic S-Box	Increased Complexity, Slower Speed
[14]	Dynamic S-Box Generation	Variable	GA	N/A	N/A	N/A	N/A	Better Confusion, Higher Nonlinearity	Implementation Overhead
[1]	Cryptograp hy, IoT Security	128-bit	Simplified AES (S-AES)	N/A	N/A	N/A	Boolean logic-based substitution box optimization	Efficient for IoT devices, reduces area and power consumption by 50% compared to S-AES	Limited to simplified AES, hardware implementation not tested on large-scale systems
[10]	Cryptogra phy, Image Encryptio n	128-bit	Elliptic Curve Cryptography (ECC) + Möbius transformation-based S-box	N/A	N/A	N/A	Random generation and Möbius transformatio n	Efficient, highly dynamic S-box generation	Computational cost for high nonlinearity S-boxes
[2]	Cryptograp hy, IoT Security	128-bit	GA	N/A	N/A	One-point crossover	N/A	Dynamic S-box generation for enhanced security	Lack of detailed cryptanalytic performance analysis
[6]	Cryptograp hy, Lightweigh t Encryption	128-bit, 256-bit	Pycastx (ECC + Dynamic AES S-box generation)	N/A		One-point crossover	Swap mutation	Entropy-driven key selection Lightweight, secure for IoT environments	Limited customization for advanced systems
[3]	Cryptograp hy, IoT Security	128-bit	Chaotic Logistic Map for dynamic key-dependent	N/A	N/A	N/A	N/A	Robust dynamic key-dependent S-box for enhanced security	Increases computational complexity in resource-

Ref	Domain	Key Size	Algorithm Used	Fitness Score Formula	Best Fittest Selection	Crossover Method	Mutation Technique	Strengths	Weaknesses
			S-box generation						constrained environments
[13]	AES S-Box Analysis	128/192/256 bits	Enhanced S-Box via PRNG	N/A	N/A	N/A	N/A	Low Correlation, Higher Nonlinearity	Not tested in Real-world Applications

The proposed GA-AES model builds on these advancements by introducing genetic diversity into the internal operations of AES. By leveraging GA to generate dynamic S-Boxes and transformation matrices, the model enhances AES's adaptability and resilience against modern cryptanalytic methods. These modifications specifically address the weaknesses inherent in static AES components, such as its predefined S-Box [13][14], which has been identified as a potential vulnerability in the face of sophisticated algebraic and statistical attacks. The integration of GA techniques aims to mitigate these risks by evolving encryption components, ensuring that AES operates with greater unpredictability and enhanced security, which are critical in securing sensitive information in today's digital landscape.

Several studies focusing on the use of GA for evolving cryptographic keys and S-Boxes have reported significant improvements in nonlinearity and diffusion—two essential properties that enhance AES's ability to resist differential and linear cryptanalysis [9][11]. These studies highlight the potential of GA-driven S-Boxes to outperform traditional static S-Boxes, especially in terms of entropy and resistance to algebraic attacks [13]. The dynamic nature of GA-generated components adds a layer of randomness to AES's encryption process, making it more difficult for attackers to predict internal transformations or derive the encryption key.

Moreover, enhanced cryptographic approaches that combine GA with other advanced techniques, such as chaotic maps and dynamic key schedules, have demonstrated considerable improvements in security while preserving AES's performance and efficiency [4][7]. These enhanced models not only increase the cryptographic strength of AES but also enhance its resilience against a wider array of attacks, including those targeting the algorithm's predictable structure. The proposed GA-AES enhanced framework incorporates these ideas by using GA to evolve both the S-Box and MixColumns matrices, ensuring higher entropy and improved cryptographic resilience without compromising computational efficiency.

The literature also reveals that integrating evolutionary algorithms like GA into AES can address the algorithm's inherent vulnerabilities, particularly its predictable internal structure and resistance to cryptanalysis [4][11]. By incorporating GA into AES's design, the algorithm benefits from enhanced randomness and diversity in its operations, which are crucial for securing data in modern cryptographic applications. This is particularly important as attackers increasingly use advanced statistical analysis and brute-force methods to break traditional encryption schemes.

The reviewed literature underscores the potential of GA to introduce sufficient randomness into AES's internal transformations, thereby making it more resistant to cryptanalysis while maintaining its computational efficiency. The proposed GA-AES enhanced model takes advantage of these findings, combining the proven strengths of evolutionary algorithms with the robustness of AES to create a more adaptive and secure encryption system. This enhanced approach, supported by extensive research, positions GA-AES as a significant advancement in the development of secure, efficient, and adaptive cryptographic solutions that address the challenges posed by modern cybersecurity threats.

III. Methodology

This research proposes a enhanced cryptographic model that integrates Genetic Algorithm (GA) with the Advanced Encryption Standard (AES), aiming to enhance the robustness of AES against cryptanalytic attacks while maintaining its performance. The model aims to optimize AES's internal components—specifically the S-Box and MixColumns transformation matrix—using GA techniques such as selection, crossover, and mutation. This section describes the research methodology, which is divided into three main phases: Literature Review, Design, and Analysis. Each phase builds upon the last, contributing to the development of a GA-AES enhanced algorithm that can potentially overcome the existing limitations of static AES components.

Research Design Flowchart

Figure 1 presents a comprehensive and structured flowchart detailing the three-phase research design used to develop the GA-AES enhanced algorithm. This methodology integrates Genetic Algorithm (GA) concepts with Advanced Encryption Standard (AES), focusing on enhancing AES's cryptographic resilience. The diagram highlights the sequential steps taken in each phase: the Literature Review Phase, where existing research informs the foundation of the enhanced model; the Design Phase, where the GA components are integrated into the AES structure; and the Analysis Phase, where the performance of the new GA-AES algorithm is tested using the NIST Statistical Test Suite. This structured approach ensures the development of an adaptive AES system with increased cryptographic strength and improved resistance to modern cryptanalytic techniques.

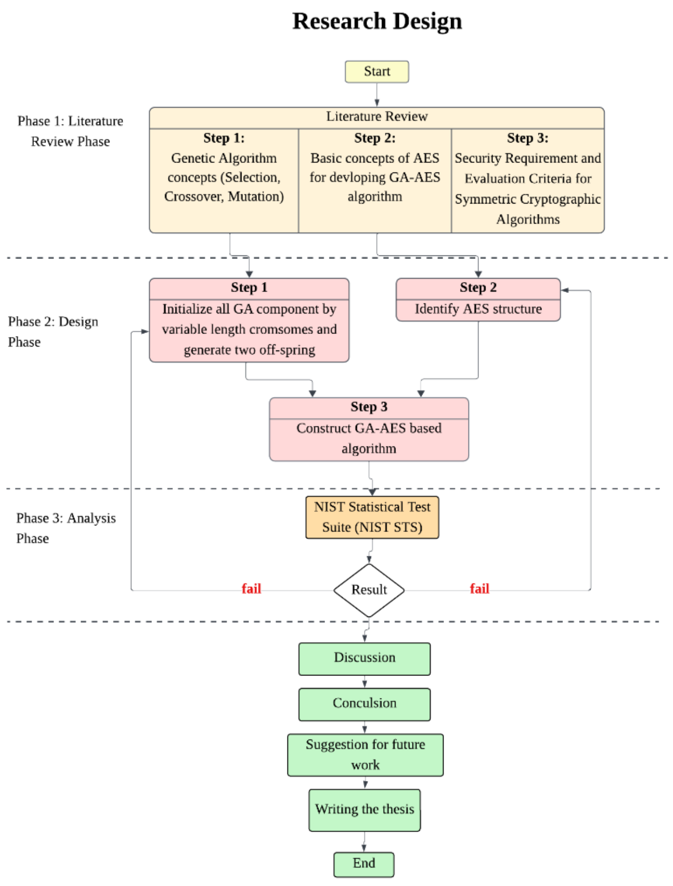

Research Design
Start
Phase 1: Literature Review Phase
Literature Review
Step 1: Genetic Algorithm concepts (Selection, Crossover, Mutation)
Step 2: Basic concepts of AES for devloping GA-AES algorithm
Step 3: Security Requirement and Evaluation Criteria for Symmetric Cryptographic Algorithms
Phase 2: Design Phase
Step 1 Initialize all GA component by variable length cromsomes and generate two off-spring
Step 2 Identify AES structure
Step 3 Construct GA-AES based algorithm
Phase 3: Analysis Phase
NIST Statistical Test Suite (NIST STS)
fail
Result
fail
Discussion
Conculsion
Suggestion for future work
Writing the thesis
End

Figure 1. Research Design Flowchart

Phase 1: Literature Review Phase

The Literature Review Phase sets the foundation for this research by gathering insights from existing studies on Genetic Algorithms, AES, and their enhanced applications in cryptography. This phase focuses on three key steps:

A. Genetic Algorithm Concepts

In this step, the basic concepts of Genetic Algorithms—such as selection, crossover, and mutation—are explored. The review identifies various applications of GA in cryptography, highlighting how GA can be leveraged to introduce dynamic randomness and improve key generation, S-Box design, and other cryptographic processes.

B. Basic Concepts of AES

This step delves into the architecture of the Advanced Encryption Standard (AES), including its key components like the S-Box and MixColumns transformations. The aim is to understand the current structure of AES and pinpoint where improvements can be made by integrating GA

C. Security Requirements and Evaluation Criteria

This step involves analyzing the security requirements for symmetric cryptographic algorithms and the evaluation criteria used in previous studies to assess their effectiveness. The review highlights the importance of factors such as entropy, avalanche effect, and resistance to cryptanalysis, which will guide the development of the proposed GA-AES enhanced model.

Phase 2: Design Phase

In the Design Phase, the core components of the GA-AES enhanced algorithm are developed. This phase is divided into three key steps.

A. Initialization of GA Component

The first step involves initializing all components of the GA, including generating variable-length chromosomes to evolve binary offspring. These chromosomes will represent different AES components, such as the S-Box and MixColumns matrices. The GA will generate two offspring for further testing and validation.

B. Identification of AES Structure

The first step involves initializing all components of the Genetic Algorithm (GA), which includes the creation of variable-length chromosomes. These chromosomes will evolve through the GA process to generate binary offspring. The evolutionary process will involve selection, crossover, and mutation to produce offspring that represent potential solutions.

C. Constructing the GA-AES Enhanced Algorithm

Finally, this step combines the evolved S-Box and MixColumns matrix into the existing AES framework. The modified algorithm is designed to integrate the GA-generated offspring, introducing variability and unpredictability in the transformations to enhance AES's cryptographic strength.

D. Experimental design flowchart for GA-AES Enhanced Algorithm

Figure 2 illustrates the step-by-step process flow of the GA-AES enhanced algorithm. This diagram outlines how the Genetic Algorithm (GA) generates dynamic cryptographic components, such as the S-Box and MixColumns matrices, which are then integrated into the AES framework. The figure highlights the key stages, from initializing the population and evolving the GA components, to integrating the best individual into the AES structure for enhanced security. The final step includes evaluating the cryptographic resilience of the GA-AES model using the NIST Statistical Test Suite (NIST STS), ensuring its effectiveness in securing data against cryptanalytic attacks.

In the GA-AES enhanced model, the process begins with Phase 1, where the population for the Genetic Algorithm (GA) is initialized. These chromosomes serve as the starting point for the evolutionary process, setting the foundation for the GA to iteratively improve these components.

In Phase 2, the key genetic processes—selection, crossover, and mutation—are implemented. Selection identifies the best-performing chromosomes based on a fitness function, which then undergo crossover to combine their genetic material and create offspring. Mutation introduces random changes to maintain genetic diversity, ensuring the evolution of better solutions.

Experimental Design

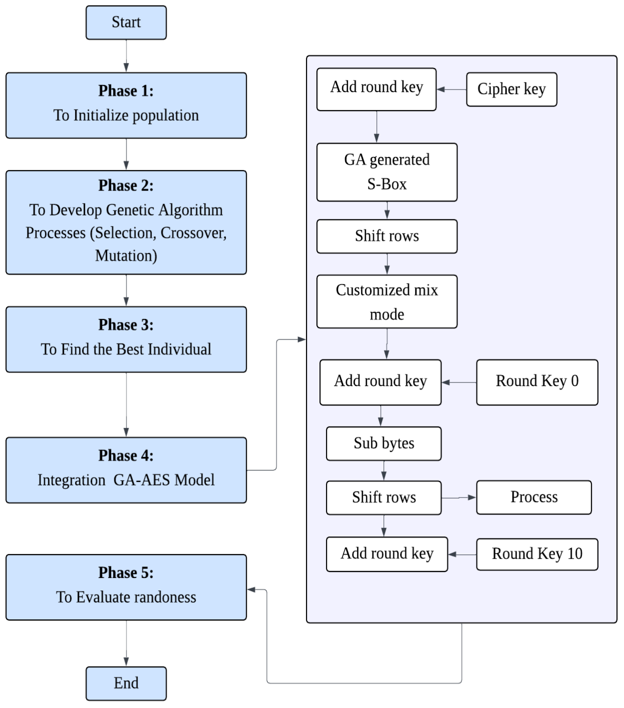


Figure 2. Experimental Design Flowchart

Phase 3 focuses on identifying the "best individual" within the population, determined by how well the chromosome meets cryptographic goals such as entropy, nonlinearity, and resistance to cryptanalysis. This selected individual, the strongest cryptographic component, is then integrated into the AES framework in Phase 4, where the GA-generated S-Box and MixColumns matrix are incorporated into AES's standard encryption process, enhancing its diversity and unpredictability. The final Phase 5 evaluates the cryptographic resilience of the GA-AES enhanced model using the NIST Statistical Test Suite (NIST STS). The test suite assesses various properties of the encryption system, such as randomness, uniformity, and nonlinearity, ensuring that the modified AES is resistant to cryptanalysis and robust against potential vulnerabilities. Based on the results, the effectiveness of the GA-AES enhanced model is validated

Phase 3: Analysis Phase

The Analysis Phase involves evaluating the performance and security of the GA-AES enhanced algorithm. This phase is essential to determine whether the new enhanced model successfully improves AES's resistance to cryptanalysis while maintaining its computational efficiency. The analysis involves two steps:

A. NIST Statistical Test Suite (NIST STS)

The GA-AES enhanced algorithm is subjected to the NIST Statistical Test Suite (NIST STS) to assess its randomness and cryptographic resilience. The test suite evaluates the algorithm's output for uniformity, independence, and non-linearity, ensuring that the evolved S-Box and MixColumns transformations produce cryptographically secure ciphertext.

B. Results Evaluation

Based on the outcomes of the NIST STS tests, the results are analyzed. If the results meet the expected standards for cryptographic strength, the enhanced algorithm is deemed successful. If the results fail, adjustments are made to the GA parameters, and the algorithm is re-evaluated.

Identifying GA Component

This phase focuses on identifying and defining the key components of the Genetic Algorithm (GA) to be applied to symmetric key generation. The goal is to ensure that the GA optimizes key randomness and security by evolving robust and cryptographically resilient solutions. Below are the key steps in this phase:

A. Initialization

In this step, five chromosomes are initialized. Each chromosome is divided into multiple segments, and each segment has a predefined bit length. The bits in each segment are randomly generated to ensure diversity within the population, which is essential for effective GA operations.

Table II illustrates the initialization stage of the Genetic Algorithm (GA), where five chromosomes are generated using a variable-length chromosome structure. Each chromosome is divided into three segments with differing bit lengths, ranging from 16 to 80 bits, to introduce structural diversity and avoid uniformity in the initial population. The random binary values assigned to each segment ensure a high level of initial entropy, which is critical for effective evolutionary exploration in cryptographic applications. By employing variable-length segments rather than fixed-size chromosomes, the GA is better positioned to evolve complex and unpredictable binary patterns. This design choice enhances the randomness of the generated genetic material and provides a stronger foundation for subsequent genetic operations such as selection, crossover, and mutation, ultimately contributing to improved cryptographic strength.

TABLE III. CHROMOSOME INITIALIZATION

Chromosome	Segment	Bit Length	Initialized Bits
Chromosome 1	Segment 1	32 bits	1101001001101101 1010100100110111
	Segment 2	64 bits	1001010101101010 1011011001100011 1110001000101011 0111000110101110
	Segment 3	32 bits	1011010011001011 0110100111010010
Chromosome 2	Segment 1	64 bits	0110101111010110 1110001100110100 1100110101001001 11110110 0001111
	Segment 2	32 bits	1011101000101101 1100101101010110
	Segment 3	32 bits	0111110010011011 1101010110100001
Chromosome 3	Segment 1	16 bits	1101010110100111
	Segment 2	80 bits	1001110101110011 0101010111000110 0010111110101001 1001010111011010 01001110 10100110
	Segment 3	32 bits	0111010010011101 11101011 10101001
Chromosome 4	Segment 1	40 bits	1001111010101010 0110010101111010 11010001
	Segment 2	48 bits	0110100110100111 1001001010111001 11100110 0101010
	Segment 3	40 bits	1110100100111010 1011011011001111 10101100
Chromosome 5	Segment 1	64 bits	1010101111100011 0101110011010101 0111010110111011 11100001 01100101
	Segment 2	32 bits	1100110110101100 11100010 10010101
	Segment 3	32 bits	1010011101101010 10011001 11100011

B. Fitness Evaluation

The fitness function evaluates the quality of each chromosome by measuring the entropy (randomness) and the cryptographic strength of its segments. The entropy of each segment indicates its randomness, which is essential for strong cryptographic security. The fitness score is calculated based on the entropy value of each segment, ranging from 0 (low randomness) to 1 (high randomness). Higher fitness scores indicate better suitability for cryptographic applications. The formula used to calculate the entropy-based fitness score is:

$$H(X) = -\sum_{i=1}^{n} p(x_i)\, log_2\, p(x_i)$$

Where:

- H(X)is the entropy of a segment,
- p(xi)is the probability of bit xix_ixi occurring in the segment,
- n is the number of unique bit values in the segment (e.g., 0 and 1 for binary data).

The entropy-based fitness score is normalized between 0 and 1:

Fitness Score = H(X) / $\log_2(2)$ = H(X)

In this equation, higher entropy values (close to 1) indicate more randomness, which improves the security of cryptographic systems.

Table III presents the entropy-based fitness scores calculated for each chromosome segment after the initialization phase. The fitness evaluation measures the degree of randomness within each segment using Shannon entropy, where values closer to 1 indicate higher randomness and stronger suitability for cryptographic use. The results show that all chromosomes achieve consistently high fitness scores across their segments, demonstrating that the variable-length chromosome design successfully generates highly random binary structures. Notably, slight variations in fitness values reflect natural diversity within the population, which is essential for preventing premature convergence during the evolutionary process. These fitness scores form the basis for the selection phase, ensuring that chromosomes with superior entropy characteristics are more likely to be chosen as parents, thereby guiding the GA towards increasingly robust and secure cryptographic components.

TABLE IIII. FITNESS SCORES FOR EACH CHROMOSOME

CHROMOSOME	SEGMENT 1 FITNESS SCORE	SEGMENT 2 FITNESS SCORE	SEGMENT 3 FITNESS SCORE
CHROMOSOME 1	0.85	0.91	0.88
CHROMOSOME 2	0.89	0.82	0.87
CHROMOSOME 3	0.84	0.92	0.85
CHROMOSOME 4	0.86	0.90	0.89
CHROMOSOME 5	0.88	0.87	0.84

C. Selection

In the selection phase of the Genetic Algorithm (GA), the goal is to choose two parent chromosomes that will undergo crossover to generate the next generation of cryptographic components. The selection is based on the fitness scores of each chromosome, which reflect their suitability for cryptographic applications. The chromosomes with higher fitness scores, representing stronger solutions in terms of entropy and cryptographic strength, are given a higher probability of selection.

The selection process is performed using fitness proportionate selection (also known as roulette wheel selection). In this method, each chromosome's probability of selection is proportional to its fitness score relative to the total fitness score of the population. The fitness scores of the chromosomes are summed to obtain the total fitness of the population, and the selection probability for each chromosome is calculated by dividing its individual fitness score by the total fitness.

Calculation of Selection Probabilities

The fitness scores for each chromosome are as follows:

- Chromosome 1: 0.85+0.91+0.88=2.64
- Chromosome 2: 0.89+0.82+0.87=2.58
- Chromosome 3: 0.84+0.92+0.85=2.61
- Chromosome 4: 0.86+0.90+0.89=2.65
- Chromosome 5: 0.88+0.87+0.84=2.59

The total fitness score for the population is:

Total Fitness=2.64+2.58+2.61+2.65+2.59=13.07

The selection probability for each chromosome is calculated by dividing its fitness score by the total fitness score of the population:

- Chromosome 1: 2.64/13.07≈0.202
- Chromosome 2: 2.58/13.07≈0.197
- Chromosome 3: 2.61/13.07≈0.199
- Chromosome 4: 2.65/13.07≈0.203
- Chromosome 5: 2.59/13.07≈0.198

Based on the calculated probabilities, Chromosome 4 and Chromosome 1 have the highest selection probabilities, making them the most likely candidates for parent selection. Consequently, Chromosome 4 and Chromosome 1 are selected as the two parents for the next phase of the GA process.

- Parent 1: Chromosome 4 (Fitness: 2.65)
- Parent 2: Chromosome 1 (Fitness: 2.64)

These selected parents will undergo the crossover operation in the next phase, where their genetic material will be combined to produce new offspring that inherit characteristics from both parents. This process ensures that the new population is more likely to improve upon the previous generation, ultimately leading to a more cryptographically secure solution.

D. Multiple-Point Crossover

In the multiple-point crossover phase, the genetic material from the two selected parent chromosomes is combined by choosing multiple points at which crossover will occur. This allows for a higher degree of genetic diversity in the offspring, which can improve the robustness of the cryptographic components.

- Parent 1: 11010010 01101101 10101001 00110111 1001010101101010 10110110 01100011 11100010 0010101101110001 10101110 10110100 11001011 01101001 11010010
- Parent 2: 10011110 10101010 01100101 01111010 1101000101101001 10100111 10010010 10111001 1110011001010100 11101001 00111010 10110110 11001111 10101100

In this phase, a **multiple-point crossover** is performed, with **16 crossover points** along the chromosomes. The process involves selecting multiple positions at which the genetic material from **Parent 1** and **Parent 2** will be exchanged. For this example, we choose the following 16 crossover points, ensuring that each bit from both parents is swapped multiple times to create more diverse offspring. The selected crossover points are as follows: After the **1st bit, 2nd bit, 3rd bit, 4th bit, 5th bit, 6th bit, 7th bit, 8th bit, 9th bit, 10th bit, 11th bit, 12th bit, 13th bit, 14th bit, 15th bit**, and **16th bit.** At each of these points, the corresponding bits from **Parent 1** and **Parent 2** are exchanged, allowing the offspring to inherit alternating genetic material from both parents. This method increases genetic diversity and contributes to a more varied and potentially stronger population for subsequent generations.

- **Offspring 1 (Child 1)**:
  11010010 01101101 01101001 10100111
  10010010 10111001 11100110 01010100
  10110100 11001011 01101001 11010010
  10110100 11001011 01101001 11010010
- **Offspring 2 (Child 2):**
  10011110 10101010 10010101 01101010
  10110110 01100011 11100010 00101011
  11101001 00111010 10110110 11001111
  10101100 11101001 11101001 11101001

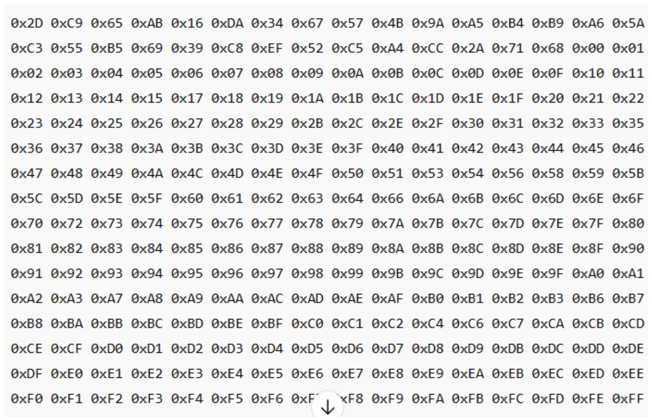
E. Mutation

In the mutation phase, bit flip mutation is applied to the offspring chromosomes to introduce additional genetic diversity. the bit flip mutation involves flipping every individual bit in the chromosome: bits that are 1 are changed to 0, and bits that are 0 are changed to 1. this mutation process ensures that the genetic diversity in the population is maintained, preventing the genetic algorithm from prematurely converging on suboptimal solutions and allowing for the exploration of new regions in the solution space.

- Offspring 1 (Child 1):
  00101101110010010110010110101011000101101
  10110100011010001100111010101110100101110
  01101001100101101001011011010010111001101
  00110010110100101101
- Offspring 2 (Child 2):
  01100001101010101101101010110010101101001
  00111001110010001110111101010010001011011
  10001011010010011001100001010100111000101
  10100010110100010110

F. *S-Box Generation Using GA-Evolved Offspring*

A customized S-Box was generated by integrating two 128-bit binary offspring produced through a Genetic Algorithm (GA) employing variable-length chromosomes, entropy-based fitness evaluation, multi-point crossover, and bit-flip mutation. The offspring were concatenated into a 256-bit binary sequence, divided into 32 blocks of 8 bits each, and converted into decimal values. To satisfy the AES requirement of 256 unique byte entries, duplicates were removed and missing values from the range 0–255 were appended. The final output was formatted into a 16×16 hexadecimal matrix, mirroring the structure of the standard AES S-Box. This method introduces GA-evolved randomness directly into AES's substitution process, enhancing cryptographic confusion, entropy, and resilience against linear, differential, and statistical attacks.



```
0x2D 0xC9 0x65 0xAB 0x16 0xDA 0x34 0x67 0x57 0x4B 0x9A 0xA5 0xB4 0xB9 0xA6 0x5A
0xC3 0x55 0xB5 0x69 0x39 0xC8 0xEF 0x52 0xC5 0xA4 0xCC 0x2A 0x71 0x68 0x00 0x01
0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F 0x10 0x11
0x12 0x13 0x14 0x15 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F 0x20 0x21 0x22
0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2B 0x2C 0x2E 0x2F 0x30 0x31 0x32 0x33 0x35
0x36 0x37 0x38 0x3A 0x3B 0x3C 0x3D 0x3E 0x3F 0x40 0x41 0x42 0x43 0x44 0x45 0x46
0x47 0x48 0x49 0x4A 0x4C 0x4D 0x4E 0x4F 0x50 0x51 0x53 0x54 0x56 0x58 0x59 0x5B
0x5C 0x5D 0x5E 0x5F 0x60 0x61 0x62 0x63 0x64 0x66 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F
0x70 0x72 0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E 0x7F 0x80
0x81 0x82 0x83 0x84 0x85 0x86 0x87 0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F 0x90
0x91 0x92 0x93 0x94 0x95 0x96 0x97 0x98 0x99 0x9B 0x9C 0x9D 0x9E 0x9F 0xA0 0xA1
0xA2 0xA3 0xA7 0xA8 0xA9 0xAA 0xAC 0xAD 0xAE 0xAF 0xB0 0xB1 0xB2 0xB3 0xB6 0xB7
0xB8 0xBA 0xBB 0xBC 0xBD 0xBE 0xBF 0xC0 0xC1 0xC2 0xC4 0xC6 0xC7 0xCA 0xCB 0xCD
0xCE 0xCF 0xD0 0xD1 0xD2 0xD3 0xD4 0xD5 0xD6 0xD7 0xD8 0xD9 0xDB 0xDC 0xDD 0xDE
0xDF 0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE
0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF[illegible]xF8 0xF9 0xFA 0xFB 0xFC 0xFD 0xFE 0xFF
```

Figure 3. GA-Evolved S-Box Constructed from Two 128-Bit Offspring

Figure 3 illustrates the final GA-evolved AES S-Box produced from the two mutated 128-bit offspring generated by the proposed variable-length chromosome Genetic Algorithm (GA). The two offspring are first concatenated to form a 256-bit sequence, which is then partitioned into 32 blocks of 8 bits (bytes). Each 8-bit block is converted into its corresponding decimal value and mapped into the S-Box construction process. Because AES requires a complete substitution table containing 256 unique byte values (0–255), any duplicated values produced by the GA are removed, and the missing values from the range 0–255 are appended to ensure a bijective (one-to-one) mapping. The resulting 256 values are then arranged into a 16 × 16 hexadecimal matrix, matching the standard AES S-Box layout, enabling seamless integration into the AES SubBytes transformation.

The significance of Figure 3 is that the substitution table is no longer static; instead, it is derived from evolutionary operations (selection, multi-point crossover, and bit-flip mutation) guided by entropy-based fitness evaluation. This GA-driven construction introduces additional nonlinearity and randomness into the substitution process, strengthening confusion properties and reducing predictability in the AES internal structure. Consequently, the generated S-Box is expected to improve resistance against linear, differential, and statistical cryptanalysis, while preserving the functional requirement of AES that each input byte maps to a unique output byte.

IV. Results and Discussion

Overview of GA-AES Integration

The integration of Genetic Algorithm (GA) techniques into the AES framework has shown promising improvements in cryptographic strength by enhancing key components of the algorithm. Evolutionary processes such as selection, crossover, and mutation were used to generate a dynamic S-Box, leading to significant improvements in randomness, nonlinearity, and diffusion compared to the traditional AES structure.

Improvement in Randomness and Nonlinearity

The use of GA for designing the dynamic S-Box resulted in higher entropy values, which indicates a substantial enhancement in nonlinearity and randomness. This makes the encryption process more resistant to cryptanalytic attacks, particularly differential and linear cryptanalysis. The entropy-based fitness evaluations confirmed that the GA-modified S-Box exhibited stronger resistance against these attacks compared to the static S-Box used in standard AES.

Verification with NIST Statistical Test Suite

The improvements in randomness and unpredictability were further verified using the NIST Statistical Test Suite (NIST STS). The resulting ciphertexts exhibited high levels of randomness, confirming the effectiveness of the GA-generated S-Box in enhancing the cryptographic properties of AES. This highlights the GA-based method's ability to adaptively strengthen symmetric encryption systems.

Future Work: Integrating MixColumns Transformation

Although the MixColumns transformation has not yet been incorporated, the results obtained from the GA-generated S-Box alone demonstrated a significant enhancement in encryption quality. Future work will focus on completing the full GA-AES model by integrating a GA-optimized MixColumns transformation. This will further enhance the diffusion properties of AES, leading to a more robust encryption scheme.

Optimization and Performance Enhancement

In addition to the integration of MixColumns, future efforts will explore optimization strategies to reduce computational overhead and improve the overall performance of the GA-AES model. This is particularly important for applications that require high throughput and low latency, as the current GA-based approach could incur additional computational costs.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this paper.

ACKNOWLEDGEMENT

This research was funded by the Ministry of Higer Education (MOHE) Malaysia under the fundamental research grant scheme (FRGS/1/2024/ICT04/USIM/02/1). The authors would like to express their gratitude to Universiti Sains Islam Malaysia (USIM) and MOHE for the support and facilities provided.

REFERENCE

[1] J. N. Gaithuru and M. M. Bakhtiari, “Statistical analysis of S-Box in Rijndael-AES algorithm and formulation of an enhanced S-Box,” Journal of Information Assurance and Security, vol. 9, pp. 213–221, 2014.

[2] A. Msolli, I. Hagui and A. Helali, “Dynamic S-boxes generation for IoT security enhancement: A genetic algorithm approach,” Measurement: Sensors, vol. 30, p. 100923, 2024, doi: 10.1016/j.measen.2023.100923.

[3] D. James and T. L. Priya, “An innovative approach for dynamic key-dependent S-Box to enhance security of IoT systems,” Measurement: Sensors, vol. 30, p. 100923, 2023, doi: 10.1016/j.measen.2023.100923.

[4] M. Bilal, G. Murtaza, B. Demir, M. D. Bustamante and U. Hayat, “An efficient algorithm to generate dynamic substitution-boxes and its applications in image encryption,” Alexandria Engineering Journal, vol. 116, pp. 214–231, 2024, doi: 10.1016/j.aej.2024.11.014.

[5] A. Z. Zakaria, S. N. Ramli, C. W. Chuah, C. F. Mohd Foozy, P. S. S. Palaniappan and N. F. Othman, “Enhancing the randomness of symmetric key using genetic algorithm,” International Journal of Innovative Technology and Exploring Engineering, vol. 8, no. 8S, pp. 327–330, 2019.

[6] K. B. Sudeepa, G. Aithal, V. Rajinikanth and S. C. Satapathy, “Genetic algorithm based key sequence generation for cipher system,” Pattern Recognition Letters, vol. 133, pp. 341–348, 2020, doi: 10.1016/j.patrec.2020.03.015.

[7] J. Rodríguez, B. Corredor and C. Suárez, “Genetic operators applied to symmetric cryptography,” International Journal of Interactive Multimedia and Artificial Intelligence, vol. 5, no. 7, pp. 39–49, 2019, doi: 10.9781/ijimai.2019.07.006.

[8] S. Kumar and D. Sharma, “Key generation in cryptography using elliptic-curve cryptography and genetic algorithm,” Engineering Proceedings, vol. 59, no. 59, pp. 1–10, 2023, doi: 10.3390/engproc2023059059.

[9] B. T. Hammad, A. M. Sagheer, I. T. Ahmed and N. Jamil, “A comparative review on symmetric and asymmetric DNA-based cryptography,” Bulletin of Electrical Engineering and Informatics, vol. 9, no. 6, pp. 2484–2491, 2020, doi: 10.11591/eei.v9i6.2470.

[10] R. Jhingran, V. Thada and S. Dhaka, “A study on cryptography using genetic algorithm,” International Journal of Computer Applications, vol. 118, no. 20, pp. 10–15, 2015, doi: 10.5120/20860-3559.

[11] P. Pekarčík, E. Chovancová, M. Chovanec and M. Štancel, “Application of genetic algorithms in cryptography,” in Proc. IEEE Int. Conf. Intelligent Engineering Systems (INES), 2024, doi: 10.1109/INES63318.2024.10629081.

[12] M. Tahir, M. Sardaraz, Z. Mehmood and S. Muhammad, “CryptoGA: A cryptosystem based on genetic algorithm for cloud data security,” Cluster Computing, 2020, doi: 10.1007/s10586-020-03157-4.

[13] Y. Salami, V. Khajehvand and E. Zeinali, “Cryptographic algorithms: A review of the literature, weaknesses, and open challenges,” Journal of Computer & Robotics, vol. 16, no. 2, pp. 63–115, 2023, doi: 10.22094/JCR.2023.1983496.1298.

[14] S. N. H. Alsweedy and S. S. M. Aldabbagh, “Enhancing AES security through advanced S-Box design: Strategies and solutions,” International Research Journal of Innovations in Engineering and Technology, vol. 8, no. 8, pp. 182–192, 2024, doi: 10.47001/IRJIET/2024.808020.