AL-KUNOOZE SCIENTIFIC JOURNAL
ISSN: 2706-6231 E ,2706-6223 P
Vol.11 No3 (2025)

Enhancing IoT Smart Door Lock Security Using Chameleon Swarm Algorithm, SHA-256, and ECC

Arkan Kh Shakr Sabonchi
Department of Mathematics, Open Educational College, Kirkuk Branch, Kirkuk, 36001, Iraq

Corresponding Author Email Address: arkankhaleel@gmail.com

Abstract.
The rapid proliferation of Internet of Things (IoT) devices, spanning from smart home appliances to wearable technology, has significantly heightened concerns regarding security and privacy across various sectors. As cyber threats become increasingly sophisticated and frequent, the urgency for robust, adaptable security frameworks within IoT infrastructures is more critical than ever. This study introduces a cutting-edge security framework tailored for IoT-based smart door locks, which employs a novel integration of the Chameleon Swarm Algorithm (CSA), Secure Hash Algorithm SHA-256, and Elliptic Curve Cryptography (ECC). We conducted comprehensive performance evaluations in a Microsoft Visual Studio 2012 environment, where our proposed framework was benchmarked against conventional hybrid methods based on Genetic Algorithms (GA) and Firefly Algorithm such as - SHA-256-ECC-GA and SHA-256-ECC-FA. These evaluations demonstrated that our framework significantly enhances security performance, achieving up to 15.17% faster encoding times at 100 iterations and markedly quicker decoding times at 150 iterations compared to the benchmark techniques. The improvements confirm the framework's effectiveness in not only bolstering IoT device security but also in its potential for scalability and adaptability across diverse IoT applications.

Keywords: Information Security; Chameleon Swarm Algorithm (CSA); Internet of Things (IoT) security; Elliptic Curve Cryptography (ECC); SHA-256 algorithm

Introduction

The Internet of Things (IoT) has become a transformative force in modern technology, interconnecting an ever-growing number of devices from home appliances and vehicles to healthcare systems and industrial machinery [1-3]. This technological evolution is not just reshaping consumer behavior but also redefining the operational paradigms of numerous industries. As these devices proliferate, they form a network of interconnected digital entities that create and share data continuously. While this integration promises enhanced operational efficiency and access to real-time data, it also introduces significant security vulnerabilities that could be exploited to cause widespread harm. In the context of smart home technology, one of the most sensitive points of vulnerability lies in IoT-based smart door locks. These devices, pivotal in ensuring physical security, face unique challenges as they blend digital and mechanical functionalities. Given the potential consequences of security breaches, which can range from unauthorized home entry to data theft, the need for robust security measures is not merely beneficial but critical [4, 5]. Addressing these concerns, our paper introduces a novel security framework that integrates the CSA, SHA-256 hash function, and ECC. Each of these technologies has been specifically chosen for its strengths in securing digital communications and enhancing the integrity and confidentiality of data [6, 7]. CSA, for example, offers adaptive solutions that are highly effective in complex, dynamic environments like those found in IoT systems. SHA-256 provides a strong hashing mechanism ensuring data integrity and verifying authenticity. ECC is employed for its strength in creating secure cryptographic keys with relatively smaller key sizes, which is ideal for the resource-constrained environments typical of many IoT devices. The integration of these technologies aims to fortify the security of smart door locks significantly, thereby reducing the risk of unauthorized access while maintaining user convenience and efficiency. This paper not only discusses the technical implementation of these algorithms but also evaluates their effectiveness in real-world scenarios. Comparative analysis against existing security measures such as SHA-256-ECC-GA and SHA-256-ECC-Fais provided, illustrating our framework's improvements in encoding and decoding times, which are

critical for performance in real-time applications [8-10]. The findings and methodologies can be adapted for other IoT devices that require high levels of security, from wearable health monitors to automotive systems. By exploring the scalability and adaptability of the proposed security framework, this research contributes to the ongoing discourse on IoT security, proposing solutions that could be pivotal in securing a wide array of devices against an ever-evolving landscape of cyber threats [11-15]. This paper is structured as follows: Section 2 discusses related works. Section 3 explains the CSA, while Section 4 covers ECC. Section 5 proposes the CSA-ECC secure communication model. Section 6 evaluates its performance, and Section 7 concludes the study.

Related Works

From a detailed literature survey as shown in Table 1, it is clearly visible that there is a great advancement in securing IoT based on different cryptographic and optimization strategies. However, all of these studies focus on individual aspects of reducing resource utilization, secure data concealment, or power optimization rather than delivering an end-to-end security solution capable of addressing the ever-changing situation of IoT environments. The proposed solutions in all of these studies have an individual optimization algorithm dependency, which is not sufficiently resilient in addressing new security threats in a timely matter. Gaps in these fields are filled in this proposed study, which implements a hybrid security system based on CSA, SHA-256, and ECC. In a difference compared to other conventional methods, the adaptive nature of CSA facilitates timely action on threats and network alterations, and the integration of ECC results in maximum cryptographic power utilizing minimal resources.

Table 1. Comprehensive Comparison Table

Methodology	Techniques Used	Limitations	Advantages of the work
Binary image encryption [16]	Stream cipher, Ant Colony Optimization (ACO)	Limited to binary images; lacks scalability	Broader applicability in IoT environments.
Visual Secret Sharing (VSS) [17]	ECC with optimization techniques	Requires large numbers of shares	Reduces computational complexity and improves efficiency.

Image encoding [18]	Genetic Algorithm (GA), ECC	High computational costs	Efficient real-time key generation with adaptive performance.
Cryptanalysis of encryption keys [19]	Genetic and Memetic Algorithms	Key vulnerability detection limited to specific standards	Comprehensive cryptographic key generation.
Energy optimization for mobile devices [20]	ABC algorithm	Focus on energy optimization, lacks security	Integrated security and resource management.
Image encoding with chaotic map [21]	PSO-based chaotic map	Limited security coverage beyond image encoding	Broader security scope for IoT systems.
Enhanced cryptographic security [22]	PSO, Cuckoo Search, ECC	Complexity in algorithm integration	Simplified and scalable cryptographic framework.
Secure data hiding [23]	Fruit Fly Optimization, Seeker Algorithm	Risk of data loss in high-stress scenarios	Secure data embedding without quality degradation.
Image encoding [24]	Adaptive Elephant Herding Optimization	Scalability and real-time adaptation issues	Enhanced real-time response and key adaptability.
Cryptanalysis of stream ciphers [25]	PSO techniques	Identification of loopholes without solutions	Resilient stream cipher security mechanisms.
Big data access control [26]	ECC-based algorithm	Resource-intensive in large datasets	Lightweight ECC with CSA for big data.
Cluster head selection in HWSNs [27]	Genetic Algorithm (GA)	Sub-optimal in complex networks	Optimized head selection with extended network lifetime.
Smart irrigation system security [28]	ABC algorithm	Limited to confidentiality and authentication	Comprehensive IoT security framework.
Improved ECC encoding [29]	Cuckoo Search Algorithm	Focused only on performance improvement	Combines performance with real-time security.
IoT attack detection [30]	Firefly Optimization, Global Search	High computational demand	Real-time detection with lower computational cost.
Smart home automation analysis [31]	Architecture and technology review	Lack of strong security focus	Integrates comprehensive IoT security solutions.
Smart home automation challenges [32]	Systematic analysis of enabling technologies	Limited to system architecture insights	Practical implementation with robust security.

CS and GA have been shown to be effective for secure cryptographic key generation and system optimization. CS is distinguished by its capability for global search and convergence in some search environments, while GA is famous for its applicability and capability in solving complex optimization problems. However, these algorithms have severe

drawbacks in real-time IoT environments based on excessive computation and susceptibility to premature convergence [16, 19, 22]. While CS and GA have an exploration vs. exploitation trade-off, CSA attains a balance between these, resulting in faster convergence and minimal overhead computations.

Chameleon Swarm Algorithm

The CSA is one of the nature-inspired metaheuristic optimization algorithms developed recently [12], which draws inspiration from the adaptive and intelligent behavior of chameleons while hunting for food and/or survival. Chameleons possess a unique combination of rapid adaptability, effective camouflage, and precision in targeting prey. CSA exploits the unique ability of chameleons in balancing their exploration globally-contrasting against local exploitation-for refining what is possibly the best-found solution for solving different optimization problems in a highly effective manner. Another inspiration of CSA is found in the rapid adaptation of chameleons to their environment. Indeed, this algorithm dynamically tunes the trade-off between its exploration and exploitation phases, in that the moving pattern of the chameleons, or the candidate solutions, is updated differently in these two phases. The proposed mathematical models present a combination of equations defining both random movements related to the exploration phase and movements oriented towards promising areas related to the exploitation one. Let $X_i(t)$ represent the position of the i^{th} chameleon at iteration t, and $X_{best}(t)$ denote the best-known solution at that iteration. The movement of each chameleon is controlled by Eq. (1):

$$X_i(t+1) = X_i(t) + r_1 \cdot \text{Camouflage}(X_{best}(t) - X_i(t)) + r_2 \cdot \text{Hunting}(X_{\text{prey}}(t) - X_i(t)) + \alpha \cdot \text{RandomWalk} \quad (1)$$

where r_1 and r_2 are random numbers between 0 and 1, used to control the influence of camouflage and hunting behavior. Camouflage represents the chameleon's ability to blend with its environment, focusing on fine-tuning the solution by adjusting towards the best current position. Hunting describes the aggressive targeting of the best prey (solution), enhancing exploitation by directing solutions towards promising regions. RandomWalk introduces

stochasticity to ensure diversity in the search space, thus preventing premature convergence to local optima. α is the adaptive factor that controls the magnitude of random perturbations based on the progress of the search. The camouflage behavior of chameleons is modeled through a balance between exploration and exploitation. The adaptation of solutions is mathematically represented as Eq. (2):

$$\text{Camouflage}(X_{\text{best}}, X_i) = \gamma \cdot (X_{\text{best}} - X_i) + (1 - \gamma) \cdot \text{LevyFlight}(\lambda) \quad (2)$$

where γ is a weighting factor controlling the balance between direct exploitation of the best solution and stochastic exploration. Levy Flight represents a form of random walk characterized by Lévy flights, which allow for occasional large steps that help explore new areas in the search space. λ is a parameter governing the step size of the Lévy flight, facilitating long jumps to escape local optima. Similar to other algorithms using Lévy flights [30], CSA employs this mechanism to enhance exploration. Lévy flights enable occasional far-reaching jumps, modeled using a power-law distribution that favors small steps with rare but large jumps, allowing the algorithm to explore distant regions of the solution space. The step length L for the Lévy flight can defined as Eq. (3):

$$L = \frac{s}{|y|^{1/\beta}} \quad (3)$$

where s and y are random variables sampled from a normal distribution. β is the Lévy distribution parameter, typically set between 1 and 2, ensuring a balance between short exploratory movements and longer jumps. Chameleons are known for their precise and focused approach to hunting prey. In CSA, this behavior is mimicked by directing candidate solutions towards the best-known solution or prey. The hunting behavior introduces a controlled bias towards the current best solution in Eq. (4):

$$\text{Hunting}(X_{\text{prey}}, X_i) = \delta \cdot (X_{\text{prey}} - X_i) \quad (4)$$

where δ is a factor that controls how aggressively the algorithm exploits the best solution. X_{prey} represents the best-known solution or a near-optimal point in the solution space. To prevent the algorithm from getting stuck in local optima, CSA incorporates a

random walk component, which ensures diversity and allows the algorithm to escape suboptimal regions. The random walk is generated by Eq. (4).

$$\text{RandomWalk} = \epsilon \cdot (X_{\text{rand}}(t) - X_i(t)) \quad (4)$$

where X_{rand} is a randomly selected solution from the population, and ϵ is a small perturbation factor that introduces randomness. For highly effective cryptographic key generation, particularly for enhancing the security of systems, the CSA has been used with ECC. By utilizing the CSA's adaptive hunting and camouflage mechanisms, highly unpredictable private keys can be generated, making it resistant to various types of attacks, including brute-force and dictionary attacks. In an encryption scheme using CSA for key generation, the private key K_{private} is generated by Eq. (5):

$$K_{\text{private}} = CSA(X_{\text{initial}}, N) \quad (5)$$

where X_{initial} is the initial population of candidate solutions (chameleons). N represents the number of iterations or chameleon movements. CSA is a powerful algorithm that solves complex optimization problems [33]. Adaptation is performed based on chameleon behavioral traits, thereby offering effective balancing between exploration and exploitation. By integrating Lévy flights, the hunting of prey, and camouflage behavior, the proposed algorithm will definitely work in different areas of applications where high unpredictability and robustness have to be considered, as in security systems. The application of CSA in key generation for cryptographic systems, such as ECC, provides better security with high performance and, hence, makes secure communications possible in IoT-based environments and other very high-security applications.

1. Elliptic-Curve Cryptography

ECC is a modern cryptographic technique that relies on the mathematics of elliptic curves, which are algebraic structures defined by equations over finite fields. By performing mathematical operations on these curves, ECC enables the secure encoding, encryption, and decryption of data. This method is highly efficient, offering strong security with smaller key sizes compared to other cryptographic algorithms, making it particularly useful in environments with limited computational resources, such as mobile devices and IoT

applications. The main advantage of ECC lies in its efficiency: strong security can be achieved with relatively small key sizes compared to other cr yptographic techniques. The security of ECC is based on the hardness of the elliptic curve discrete logarithm problem (ECDLP) [13]. To implement ECC, two constants, a and b, are selected from a finite field F_p, which defines an elliptic curve as represented in Eq (6):

$$y^2 = x^3 + ax + b \tag{6}$$

The points on the curve form a set denoted by $E(F_p)$, and Q is one such point on the curve. The ECC encryption process involves several key steps:

1. **Initialization**: The algorithm is initialized by setting parameters such as the elliptic curve E, the base point Q, and the field p. This step may include generating random numbers or other required initial values.
2. **Public Key Generation**: A key pair comprising a public key and a private key is generated. The private key, x, is a randomly selected number within the range 1 to $n-1$ (where n is the order of Q). The public key, H, is derived using the equation $H = x \cdot Q$. While the private key is used for decryption, the public key is used for encryption, ensuring the process remains secure and computationally resistant to deriving the private key from the public key.
3. **Encoding**: In this step, plaintext data is encrypted using the public key, then is first converted into bits and mapped to points (x, y) on the elliptic curve. These points are then encrypted using the public key. The ciphertext consists of two components, as shown in Eq (7):

$$\text{Ciphertext} = \text{Enc(data)} = \begin{cases} \text{Ciphertext}_1 = r \cdot Q \\ \text{Ciphertext}_2 = \text{data} + r \cdot H \end{cases} \tag{7}$$

Here, r is a random number selected during the encryption process.

4. **Decoding**: The decryption process involves recovering the original plaintext using the private key, x, as shown in Eq (8):

$$\text{Dec(Ciphertext)} = \text{Ciphertext}_2 - x \cdot \text{Ciphertext}_1 = \text{data} + r \cdot H - x \cdot (r \cdot Q) = \text{data} \tag{8}$$

Decryption is a procedure that turns encrypted data into its plaintext state in its native form. The decryption is obtained with a private key that makes it accessible to those with a valid key. The procedure protects data and keeps it confidential by not permitting unwarranted parties from having access to valuable information.

5. **The Proposed Model: Chameleon Swarm Algorithm for ECC-based Secure Communication**

In the proposed model, the CSA is employed to generate a random private key for ECC, ensuring secure data encoding and decoding in IoT systems. The randomness of the private key is critical to the security and robustness of ECC, and CSA is introduced as an optimization technique to improve the quality of the key generation process. By using CSA, the model aims to identify the optimal private key (denoted as Pr) that maximizes the accuracy and security of ECC-based encryption and decryption. The fitness of a generated private key is evaluated by comparing the original plain text with the decrypted ciphertext, ensuring that the private key produces a result as close as possible to the original message. The fitness function used in the CSA optimization is defined as Eq. (9):

$$f(Pr) = -h(Dec(CP, Pr), B) \quad (9)$$

In the proposed framework, CP denotes the ciphertext generated after the encoding process, while B represents the original plaintext. The function $Dec(CP, Pr)$ corresponds to the decrypted ciphertext, which is derived using the private key Pr. $h(x, y)$ is a metric function designed to evaluate the similarity or dissimilarity between the decrypted text x and the original text y. The primary objective of the CSA is to minimize the discrepancy between the decrypted plaintext and the original plaintext, thereby ensuring the accuracy and reliability of the decoding process. This objective is formalized as a minimization problem, where the inclusion of a negative sign in the objective function facilitates this transformation. Within this framework, a smaller value of the fitness function indicates a more optimal solution. The steps comprising the proposed model are as follows:

1. **Initialization**: The Chameleon Swarm Algorithm begins by setting important parameters, including the number of

solutions or nests n, the size of each nest m, the probability of discovering alien solutions p_a, the step size α, and the maximum number of iterations.

2. **Random Initialization of Private Keys**: Random private key candidates are generated as binary sequences of length 256 bits (for ECC), representing the possible solutions (nests). These candidates are evaluated based on their ability to decode encrypted messages.
3. **Key Generation and Optimization**: New key candidates are generated based on the current best solution using a modified version of the Lévy flight equation, except for the best candidate, which remains unchanged. If a new candidate is found to perform better, it is selected, and the fitness function is evaluated using the key and ECC decryption.
4. **Elimination of Poor Solutions**: Bad solutions or nests are discarded using a probability p_a, simulating the discovery of alien solutions. If the fitness of a newly generated solution is superior to the current ones, the current solution is replaced.
5. **Iterative Optimization**: This process continues for a predetermined number of iterations or until the optimal private key achieves the desired quality level. The values of n, p_a, and α are fine-tuned through experimentation to achieve the best results in terms of computational efficiency and security. The proposed model incorporates ECC encoding and decoding with additional cryptographic functions for enhanced security. The procedure works as shown in Figure (1)

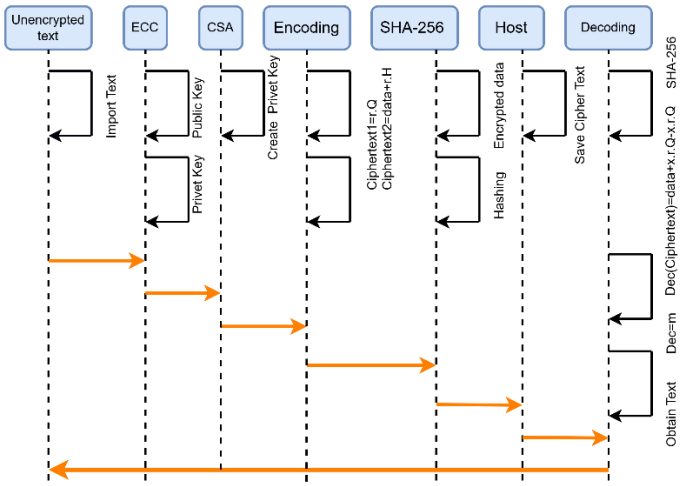

Figure1. Process diagram of the suggested framework

- **Public and Private Key Generation**: Using the CSA-optimized private key Pr, a public key H is computed as Eq. (10), where Q is a base point on the elliptic curve.

$$H = Pr \cdot Q \qquad (10)$$

- **Encryption**: The plaintext is mapped onto the elliptic curve and then encrypted using the public key. A random number r is selected to generate the ciphertext components Eq. (11)

$$\text{Ciphertext}_1 = r \cdot Q \text{Ciphertext}_2 = \text{data} + r \cdot H \qquad (11)$$

- **Decoding**: The recipient uses their private key Pr to decrypt the ciphertext, reversing the encryption process to retrieve the original data. This is done by calculating Eq. (12):

$$\text{Dec(Ciphertext)} = \text{Ciphertext}_2 - Pr \cdot \text{Ciphertext}_1 \qquad (12)$$

In a way to provide security for sent information, a new model does a left-shift and does an XOR on encrypted information. The processes make it hard for an attacker to reverse encrypting information. The secure communication method consists in four steps:

1. **Encoding (Phase 1)**: The plaintext is encrypted using ECC and the XOR operation.
2. **Encoding (Phase 2)**: A shift-left operation is applied to further obfuscate the encrypted data.
3. **Decoding (Phase 1)**: The shift-left operation is reversed before ECC decryption.
4. **Decoding (Phase 2)**: The XOR operation is reversed to fully recover the original plaintext.

By combining ECC with CSA and additional cryptographic techniques, the proposed model ensures secure and efficient communication in IoT systems. The optimized private key generation process guarantees high randomness, enhancing the overall security of the encrypted messages. This makes the model suitable for IoT applications where data confidentiality, integrity, and security are critical.

6. Evaluation and Results

The provided experimental was utilizing the C# programming paradigm in Microsoft Visual Studio 2012 outfitted in a Windows 10 professional operating system. The central process unit was an AMD®: E-350 processor 1.60 GHz, and 4 GB of RAM. The initial population was randomly generated, with its size dependent on the defined key length. The key length used in this work was defined through the random generation of integers in the range 0 to 127, while the size of the initial population is determined by that. In this way, the performance of encoding and decoding was systematically performed under the experimental conditions described in this study.

6.1. Performance Analysis (Decoding and Encoding Times)

This portion provides a detailed discussion of the proposed model's encoding performance. The parameters of its performance have been illustrated in Figure 2, in which time taken for encoding in milliseconds is illustrated along various counts of iterations, in which an initial population size of 25 is given. The figure clearly illustrates a definite trend in data, which is such that when iterations increase, time taken for encoding continues reducing, ultimately reaching a point of minimal time taken when there are 100 iterations. The trend is indicative of a situation in which successive optimization of the proposed model is being conducted, hence reducing overhead cost in computations and maximizing processing efficiency. This reduction of time taken for encoding can be of great utility in actual usage in which time and efficiency matter a great deal, so it can process large datasets or data encryptions in a timely and

hassle-free manner. Thus, results support the capability of the proposed work in providing an ideal solution for cryptographic purposes in which time is of great issue. Execution, for instance, has recorded that in the case of file size 50,000 KB, for iterations of 10, 100, and 200, 4200, 3800, and 4350 milliseconds are spent, respectively.

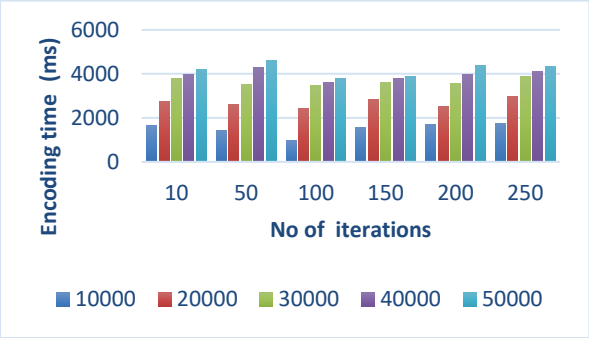

Figure 2. Time taken for encoding based on the number of iterations.

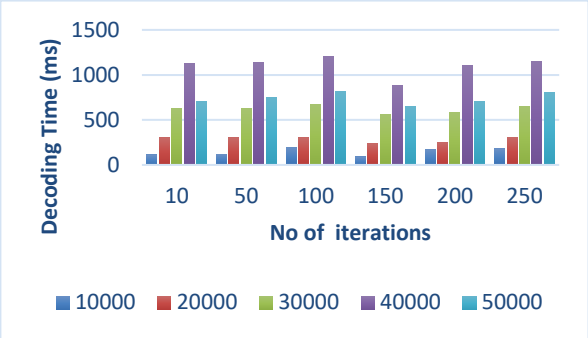

Figure 3. Decoding time as a function of the number of iterations.

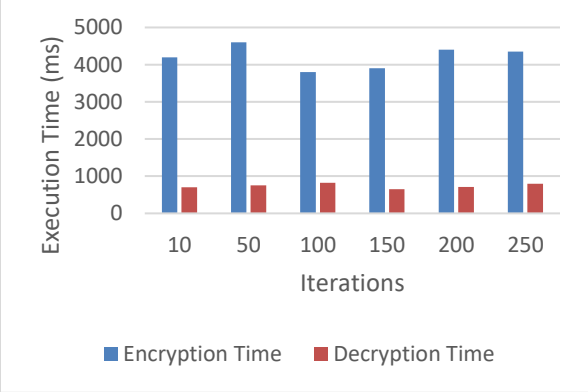

Figure 4. Times on average for encoding and decoding at various iterations.

Contrarily, Figure 3 is a detailed breakdown of the proposed model's decoding time, in milliseconds, for different numbers of iterations, when an initial population size of 25 is set. The figure illustrates a similar trend in the case of decoding time compared to encoding time based on differences in the number of iterations. Interestingly, it is shown here that the system is best when its decoding time is lower compared to its encoding time, which makes its total cryptographic process faster. With minimal decoding time, its entire process of encryption and decryption is faster, hence making its system perform well in actual applications, wherein there is a large amount of data, which must be handled in a timely and efficient manner. The result of this work is such that the shortest time of decoding is reached

when there are 150 iterations. For example, a file of size 50,000 KB needed 700.012 ms in 10 iterations, 650.001 ms in 150 iterations, and 800.022 ms in 250 iterations. Figure 4 illustrates the proposed model's mean time for encoding and for decoding based on various numbers of its iterations. The figure clearly indicates that time for encoding is best in its 100th position, which is its best point in the process. At this point, time for encoding is minimal, which indicates that the system is in a balance between functionality and computation efficiency. In all, hence, the proposed modeling shall make a realistic modeling and functionality of data encoding and data decryption process in a greater appropriateness in challenging cases of its requirements for its high levels of performances and improved cryptographic security. The results obtained through this assessment confirm that the proposed model is efficient in the processes of both encoding and decoding. The iteration count gives a minimum encoding time when set at 100, while for minimum decoding time, the iteration count works best when set at 150. Besides, given the high execution time of this proposed model, there is definitely an implication of significant practical advantages, especially within areas that involve large file sizes. In turn, this model will contribute much with regard to securing high performance in real-world data processing.

6.2. Comparison (Encoding and Decoding Based on AES-RSA and Proposed Model)

The following is a comparative study of this proposed system and the AES-RSA cryptographic system [34] in Table 2. The study is conducted based on a comparison of encoding time and decryption time for different input file sizes and numbers of iterations. The results point out differences in the performance of the two systems, noting in particular the speed and efficiency in the process of decryption and encryption. For the suggested model, time for encoding and decoding is quantified under various scenarios of iterations, depending on parameters that have been optimized in a quest for identifying best operation points. The suggested model, in a comparison of AES-RSA, demonstrates improvements in some instances, for example, in its time for decryption and encoding. The enhanced performance is clearly apparent when the best numbers of iterations for the model, in which its time for encoding is minimized, is reached, though this is attained without compromising the security of the system. Table 2 serves to further illustrate the relative strengths and weaknesses of each of these methods, producing insights on the applicability and utility of the proposed system

in cryptographic contexts. A total of 11 different-length files, based on a key size of 128 bits, were utilized in this study utilizing Visual Studio. The results show that the proposed model outperforms AES-RSA with a huge difference, especially for the case of 50 MB. For AES-RSA, encoding took 5.85081221 seconds, while the proposed model encoded it in 3.897667621 seconds, which is a huge difference in processing time. RSA is a widely known public-key cryptosystem that has been viewed as quite resistant to attacks, but its large key size renders the system computationally intensive and resource-consuming. In turn, the proposed model uses ECC, which allows for more efficiency since, for comparative security levels, ECC requires much smaller key sizes. This efficiency in computation means faster computation speed and lower usage of memory. Hence, ECC is perfectly suited to devices with low computational resources. This would, in turn, mean quicker computations for the proposed model, making it best suited for real-time applications.

Table 2. An evaluation of the suggested model in contrast to AES-RSA.

Input File Size (MB)	RSA-AES (second)		Proposed Model (second)	
	Encryption	**Decryption**	**Encryption**	**Decryption**
1	1.03309147	16.19099792	0.446219871	0.604796895
5	2.10843833	21.146642	1.088401662	1.198846913
10	2.86677779	23.72389669	1.597954877	1.609685631
15	3.43121086	25.37474864	2.000422897	1.912506901
20	3.89786828	26.61525524	2.346063845	2.16131668
25	4.30311578	27.61908005	2.654785801	2.376391037
30	4.6653103	28.46730536	2.936953916	2.567913253
35	4.99522496	29.20476144	3.198797134	2.741836407
40	5.29980983	29.85899916	3.444412382	2.901988874
45	5.58385215	30.44823232	3.676668189	3.051000595
50	5.85081221	30.98516549	3.897667621	3.190768115

Figure 5 shows the encoding time of the proposed model in comparison with AES-RSA. The results pointed out that the proposed model has a much lower encoding time and reduced the execution time by about 33.38% when encoding the file size to 50 MB. This high improvement indicates that the proposed model is efficient and further vindicates the capability

of the model to perform better than AES-RSA. The results hereby establish the proposed model as a better option than the current approach of AES-RSA in scenarios where high speed and efficiency in processing are needed.

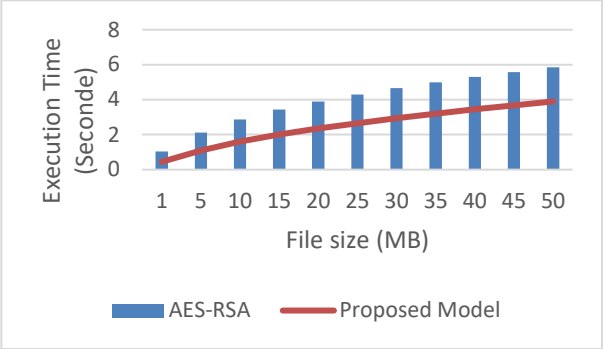

Figure 5. Comparison of Decoding Time by Proposed Model and AES-RSA

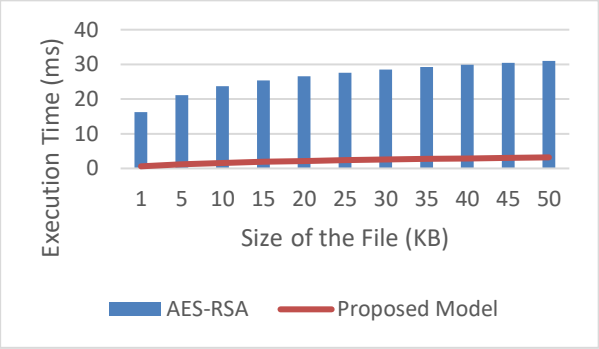

Figure 6. Comparison of decoding times between the proposed model and AES-RSA.

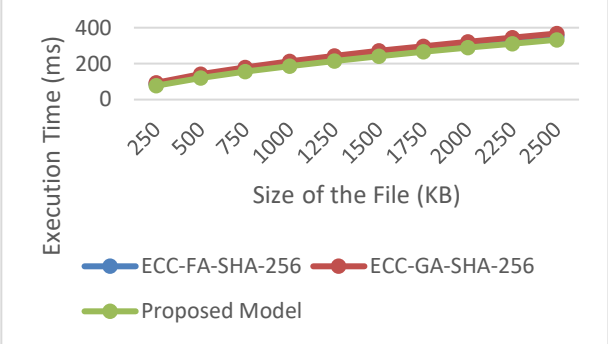

Figure 7. Comparison of encoding times for the proposed model, GA, and FA.

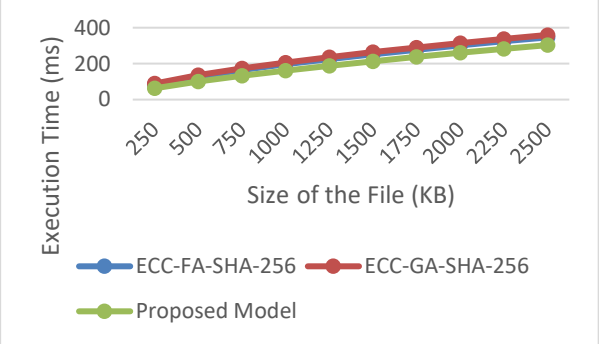

Figure 8. Comparison of decoding times for the proposed model, GA, and FA.

Figure 6 illustrates a direct comparison of the decoding times between the proposed model and the AES-RSA encryption scheme. The results clearly indicate that the proposed model outperforms AES-RSA in terms of speed. This performance advantage can be attributed to the inherent computational overhead associated with the RSA algorithm. Specifically, RSA suffers from slow processing times due to the extensive computational requirements for encoding and decoding. These operations involve modular exponentiation and calculation of the private key exponent, which are computationally intensive operations. Thus, RSA involves a large number of computations, which makes decryption processing time quite time-consuming. For this reason, unlike it, the system proposed here makes decryptions in milliseconds, owing to its optimization of vital operations. This makes it a more effective

solution for time-critical scenarios. The above difference highlights the potential of this proposed system in making cryptographic systems more effective, particularly for environments in which data decryption must be conducted quickly. The proposed model provides speeds during processing because computations set in its encoding and decoding processes are very few compared to RSA. Thus, the proposed model provides better performance for these applications where fast and efficient cryptographic processing is required. However, the proposed model, at a file size of 50 MB, recorded approximately 89.75% encoding execution time reduction against AES-RSA. Thus, it became very much proven that this preferred model outpaces AES-RSA at the highest grade in terms of efficiency by its operational processes concerning both encoding and decoding of data. Integration with ECC reduces the computational overhead; therefore, it increases the speed in the proposed model. Therefore, this is very suitable for real-time applications and also for those applications under constrained resources.

6.3. Comparison of Encoding and Decoding Times Based on FA, GA, and the Proposed Model

This section provides a detailed comparative analysis of two well-known metaheuristic algorithms, the FA [35] and the GA [36], alongside the proposed model for generating ECC keys. The analysis evaluates the performance, efficiency, and effectiveness of these algorithms in the context of ECC key generation, highlighting their strengths and weaknesses in comparison to the novel approach being proposed. Both FA and GA employ selection, crossover, and mutation operators to optimize ECC solutions. The results, summarized in Table 3, demonstrate that the proposed CSA surpasses FA and GA in terms of encoding and decoding efficiency. Additionally, CSA achieves superior solutions due to its enhanced search efficiency and stronger convergence capabilities. The concrete outcome of the experiment indicates that the proposed CSA achieves considerably quicker encoding and decoding times, followed by FA, compared with GA. For example, the encoding of a 2500 KB file took 352.9763444 ms for both the SHA-256-ECC-FA and SHA-256-ECC-GAmodels, with decoding taking 345.9605204 ms and 359.5499456 ms correspondingly. In contrast, the proposed model shows a much-reduced encoding time of 6.27047476 ms and decoding time of 7.076964611 ms.

Table 3. Comparison of Encoding and Decoding Times: Proposed Model, FA, and GA

File Size (KB)	SHA-256- ECC- FA		SHA-256- ECC- GA		Proposed Model	
	Encryption	**Decryption**	**Encryption**	**Decryption**	**Encryption**	**Decryption**
250	86.46676147	83.35066309	93.47184056	89.25566265	77.35256967	62.79172011
500	132.0527154	127.9325222	141.1438787	135.7675834	120.0227795	100.8356139
750	169.1690279	164.3713225	179.6219345	173.5213979	155.1915433	133.0293592
1000	201.6719414	196.3599285	213.1293699	206.5172802	186.2312739	161.9293278
1250	231.1252841	225.4003772	243.3673958	236.3735728	214.5225408	188.6035425
1500	258.3562646	252.2887884	271.2318102	263.9449437	240.8002792	213.6284382
1750	283.8680733	277.5101883	297.2656287	289.7506501	265.5137394	237.3601233
2000	307.9949687	301.3871755	321.8285394	314.1352742	288.9625414	260.0381572
2250	330.9729603	324.1477696	345.1738956	337.3419079	311.3589534	281.8335025
2500	352.9763444	345.9605204	367.4884113	359.5499456	332.8601974	302.8735949
13750	2354.654342	2298.709254	2473.722705	2406.158218	2192.816418	1942.92338
Throughput	5.839498289	5.981617716	5.558424141	5.714503684	6.27047476	7.076964611

The results in Figures 7 and 8 once again confirm that the proposed model outperforms the existing methods regarding execution time and encoding efficiency. Hence, the proposed model provides better benefits compared with conventional approaches toward key generation within the CSA algorithm framework.

6.4. Throughput

In this paper, we have measured the proposed model's performance in its velocities of encoding and decoding, which have been

estimated by Eqs. (13) and (14) [37]. The higher these velocities, the better its performance. The proposed model is compared on a file of size 13,750 KB and compared with SHA-256-ECC-FA and SHA-256-ECC-GA models. The proposed model encodes and decodes in 6.27047476 ms and 7.076964611 ms, respectively, which is a result in a favorable direction compared to SHA-256-ECC-FA(encoding time = 5.839498289 ms, time of decoding = 5.981617716 ms) and SHA-256-ECC-GA(encoding time = 5.558424141 ms, time of decoding = 5.714503684 ms). The results indicate the enhanced efficiency of the proposed model in its encoding and decoding.

$$En - coding\ Throughput = \sum(Input\ File)/\sum(En - coding\ Time) \quad (12)$$

$$De - coding\ Throughput = \sum(Input\ File)/\sum(De - coding\ Time) \quad (13)$$

It attained an encoding throughput of virtually 7.38% higher and a decoding throughput of virtually 18.31% higher compared to the model identified as ECC-FA-SHA-256. The levels of its encoding and decoding throughput were also virtually 12.81% and 23.84% higher, respectively, compared to the model identified as ECC-GA-SHA-256. The results clearly demonstrate that the proposed model is indeed more efficient and effective in doing faster and secure encoding and decoding compared to other models.

Conclusion

Nowadays, IoT finds a deep root in everyday life. However, ensuring security for IoT devices themselves is still a big challenge due to their limited computation and storage resources. IoT devices send messages, images, and audio; thus, there is a dire need for confidentiality and integrity. This problem led to the finding that some security algorithms are more efficient as compared to others. The goal of this paper is to propose a new scheme for securing the data transmitted by resource-constrained IoT-based door locks with the aid of ECC and SHA-256 integrated with CSA. Accordingly, this mechanism is developed in such a manner that the operations of encoding and decoding can be performed with the least time consumption. The performed simulations showed optimum performance at 100, and the proposed model achieved optimal performance at 150 iterations for both encoding and decoding. When compared to the other methods, the proposed model demonstrated significant improvements in throughput. Specifically, the encoding throughput was approximately 7.38% higher, and the decoding throughput was about 18.31% better than the

SHA-256-ECC-FA model. Additionally, the proposed model outperformed the SHA-256-ECC-GA model with encoding and decoding throughputs that were approximately 12.81% and 23.84% higher, respectively. These results underscore the proposed model's superior efficiency in both encoding and decoding processes, highlighting its potential for faster data handling in cryptographic applications. These affirm that the proposed approach enhances both speed and security. The focus in the future will be on investigation into alternative private key generation methods that also enhance the model's flexibility as well as robustness.

References

[1] Belli L, Cilfone A, Davoli L, Ferrari G, Adorni P, Di Nocera F, et al. IoT-enabled smart sustainable cities: Challenges and approaches. Smart Cities. 2020;3(3):1039–71.

[2] Lin J, Yu W, Zhang N, Yang X, Zhang H, Zhao W. A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. IEEE Internet Things J. 2017;4(5):1125–42. doi:10.1109/JIOT.2017.2683200.

[3] Mylonas G, Kalogeras A, Kalogeras G, Anagnostopoulos C, Alexakos C, Muñoz L. Digital twins from smart manufacturing to smart cities: A survey. IEEE Access. 2021;9:143222–49.

[4] Chanal PM, Kakkasageri MS. Security and privacy in IoT: A survey. Wireless Pers Commun. 2020;115:1667–93.

[5] Sarkar KR. Assessing insider threats to information security using technical, behavioural and organisational measures. Inf Secur Tech Rep. 2010;15(3):112–33.

[6] Makhdoom M, Abolhasan H, Abbas H, Ni W. Blockchain's adoption in IoT: The challenges, and a way forward. J Netw Comput Appl. 2019;125:251–79.

[7] Ghani K, Mansoor S, Mehmood S, Chaudhry SA, Rahman AU, et al. Security and key management in IoT-based wireless sensor networks: An authentication protocol using symmetric key. Int J Commun Syst. 2019;32(16):e4139.

[8] Aldawira R, Putra HW, Hanafiah N, Surjarwo S, Wibisurya A. Door security system for home monitoring based on ESP32. Procedia Comput Sci. 2019;157:673–82.

[9] Ha J. Security and usability improvement on a digital door lock system based on internet of things. Int J Secur Appl. 2015;9(8):45–54.

[10] Jacobsson M, Boldt M, Carlsson B. A risk analysis of a smart home automation system. Future Gener Comput Syst. 2016;56:719–33.

[11] Ahtsham M, Yan HY, Ali U. IoT-based door lock surveillance system using cryptographic algorithms. In: 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC); 2019. p. 448–53.

[12] Braik MS. Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems. Expert Syst Appl. 2021;174:114685.

[13] Miller VS. Use of elliptic curves in cryptography. In: 12th International Workshop SAC; 2005. p. 417–26.

[14] Yoshida H, Biryukov A. Analysis of a SHA-256 variant. In: 12th International Workshop SAC; 2006. p. 245–60.

[15] Prasanna SR, Premananda BS. Performance Analysis of MD5 and SHA-256 Algorithms to Maintain Data Integrity. In: 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT); 2021. p. 246–50.

[16] Sreelaja NK, Pai GV. Stream cipher for binary image encryption using ant colony optimization based key generation. Appl Soft Comput. 2012;12(9):2879–95.

[17] Shankar K, Eswaran P. A secure visual secret share (VSS) creation scheme in visual cryptography using elliptic curve cryptography with optimization technique. Aust J Basic Appl Sci. 2015;9(36):150–63.

[18] Shankar K, Eswaran P. An efficient image encryption technique based on optimized key generation in ECC using genetic algorithm. In: Artificial Intelligence and Evolutionary Computations in Engineering Systems: Proceedings of ICAIECES 2015. Springer India; 2016. p. 705–14.

[19] Dworak K, Nalepa J, Boryczka U, Kawulok M. Cryptanalysis of SDES using genetic and memetic algorithms. In: Recent Developments in Intelligent Information and Database Systems; 2016. p. 3–14.

[20] Kalaiselvi K, Kumar A. An empirical study on effect of variations in the population size and generations of genetic algorithms in cryptography. In: 2017 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC). IEEE; 2017. p. 1–5.

[21] Ahmad M, Alam MZ, Umayya Z, Khan S, Ahmad F. An image encryption approach using particle swarm optimization and chaotic map. Int J Inf Technol. 2018;10:247–55.

[22] Kota S, Padmanabhuni VNR, Budda K. Authentication and encryption using modified elliptic curve cryptography with particle swarm optimization and cuckoo search algorithm. J Inst Eng India Ser B. 2018;99(4):343–51.

[23] Kiruba R, Sharmila TS. Secure data hiding by fruit fly optimization improved hybridized seeker algorithm. Multidimens Syst Signal Process. 2020;33(3):1423–43.

[24] Shankar K, Elhoseny M, Perumal E, Ilayaraja M, Kumar KS. An efficient image encryption scheme based on signcryption technique with adaptive elephant herding optimization. In: Hassanien A, Elhoseny M, editors. Cybersecurity and secure information systems. Cham: Springer; 2019. p. 47–72. doi:10.1007/978-3-030-16837-7_3.

[25] Din M, Pal SK, Muttoo SK. Applying PSO-based technique for analysis of Geffe generator cryptosystem. In: Harmony Search and Nature Inspired Optimization Algorithms: Theory and Applications, ICHSA 2018. Springer Singapore; 2019. p. 741–49.

[26] Yousefi S, Derakhshan F, Aghdasi HS, Karimipour H. An energy-efficient artificial bee colony-based clustering in the internet of things. Comput Electr Eng. 2020;86:106733.

[27] Verma OP, Jain N, Pal SK. Design and analysis of an optimal ECC algorithm with effective access control mechanism for big data. Multimed Tools Appl. 2020;79:9757–83.

[28] Mullai M, Mani K. Enhancing the security in RSA and elliptic curve cryptography based on addition chain using simplified Swarm Optimization and Particle Swarm Optimization for mobile devices. Int J Inf Technol. 2021;13:551–64.

[29] Sabonchi KS, Obaid ZH. Ensuring Information Security in Smart Door Lock Systems Using the Cuckoo Search Algorithm. J Cybersecur. 2022;4(4).

[30] Braik MS, Awadallah MA, Al-Betar MA, Hammouri AI, Zitar RA. A non-convex economic load dispatch problem using chameleon swarm algorithm with roulette wheel and levy flight methods. Appl Intell. 2023;53(14):17508–47.

[31] Sabonchi KS. Optimizing internet of things device security with a globalized

firefly optimization algorithm for attack detection. J Artif Intell. 2024;6(1):261–82. doi:10.32604/jai.2024.056552.

[32] Ezugwu AE, Taiwo O, Egwuche OS, Abualigah L, Van Der Merwe A, Pal J, et al. Smart Homes of the Future. Trans Emerg Telecommun Technol. 2025;36(1):e70041.

[33] Amara M, Siad A. Elliptic Curve Cryptography and its applications. In: Proc Int Workshop on Systems, Signal Processing and their Applications (WOSSPA); 2011. p. 247–50. doi:10.1109/WOSSPA.2011.5931464.

[34] Zou L, Ni M, Huang Y, Shi W, Li X. Hybrid encryption algorithm based on AES and RSA in file encryption. In: Frontier Computing: Theory, Technologies and Applications (FC 2019). Springer Singapore; 2020. p. 541–51.

[35] Yang XS. Firefly algorithms for multimodal optimization. In: Int Symp Stochastic Algorithms. Springer Berlin Heidelberg; 2009. p. 169–78.

[36] 36. Holland JH. Genetic algorithms. Sci Am. 1992;267(1):66–73.

[37] Lemma M, Tolentino M, Mehari G. Performance analysis on the implementation of data encryption algorithms used in network security. Int J Comput Inf Technol. 2015;4(4):711–17.