Journal of Cyber Security

Tech Science Press

DOI: 10.32604/jcs.2023.041707
Article

Ensuring Information Security in Smart Door Lock Systems Using the Cuckoo Search Algorithm

Arkan Kh Shakr Sabonchi[1,*] and Zainab Hasim Obaid[2]

[1]Department of Mathematics, Open Educational College, Kirkuk Branch, Kirkuk, 36001, Iraq
[2]Mutafaweqat High School for Girls, Kirkuk Branch, Kirkuk, 36001, Iraq
*Corresponding Author: Arkan Kh Shakr Sabonchi. Email: arkankhaleel@gmail.com
Received: 03 May 2023; Accepted: 30 June 2023; Published: 10 August 2023

Abstract: The widespread use of Internet of Things (IoT) devices, such as smart appliances, phones, and watches, has brought about concerns regarding security and privacy. With the increasing prevalence of cyberattacks from both malicious and non-malicious sources, security has become a critical factor in the design of IoT systems. In particular, data security poses a significant challenge due to the growing amount of data stored in IoT systems. Inadequate security measures can allow hackers to take over IoT devices remotely, resulting in significant damage. To address these concerns and improve the security of IoT-based smart door lock systems, this research proposes utilizing the Cuckoo Search Algorithm (CS), the SHA-256 algorithm, and the Elliptic Curve Cryptography (ECC). The proposed design employs CS to generate the ECC private key, which can enhance data storage security in IoT systems. The study evaluates the proposed design in terms of encoding and decoding times and compares it with other encoding techniques such as (ECC-GA-SHA-256) and (ECC-FA-SHA-256). The results show that the proposed design achieves better encoding times with 125 iterations and better decoding times with 175 iterations. Furthermore, the proposed design has encoding and decoding times faster than other encoding techniques by more than 15.17%. These findings suggest that the proposed design can significantly enhance the security and performance of IoT-based smart door lock systems. The utilization of CS, SHA-256, and ECC can serve as promising methods to address the security challenges associated with IoT-based smart door lock systems.

Keywords: Information security; IoT; cryptography; Cuckoo Search Algorithms

1 Introduction

The IoT is a technological innovation that has brought about a significant transformation in communication and information industries. The technology operates through the interaction of intelligent servers, workstations, and sensor equipment, enabling the integration of various detecting objects [1]. The applications of IoT are diverse, spanning from home appliances, smartphones,

smartwatches, healthcare systems, to public services [2,3]. However, IoT has faced several security challenges, particularly in the areas of authentication, confidentiality, privacy, and integrity [4]. With the large volume of data generated by IoT, ensuring data security has become a major concern. Any unauthorized modification or replication of data in a network can pose a security risk and cause irreparable damage [5,6]. Therefore, securing IoT data requires robust security mechanisms, including network authentication protocols, key management mechanisms, and security measures between IoT gates and wireless sensors [7].

One of the critical applications of IoT is the smart door lock system. It is essential to ensure the security of the data in such systems, and security measures such as data integrity and anti-tampering must be put in place to safeguard communications [8,9]. While the compact size of IoT sensing devices limits hardware-related threats, cryptography remains the most effective means of protecting data [10]. Integrating cryptographic algorithms is crucial in IoT-based smart door lock systems as they have the potential to optimize and automate security measures [11].

To enhance IoT data security, a hybrid model has been proposed, which suggests a combined approach using SHA-256, ECC, and CS algorithms [12]. The private key for ECC is generated using a CS-based algorithm, which then hashes the data using SHA-256. The CS Algorithm is an optimization algorithm inspired by the behavior of cuckoo birds. It generates the optimal key by iteratively improving the quality of candidate solutions in the population, exploring new solutions using random walks and generating new solutions using the cuckoo egg replacement mechanism [12]. The algorithm iteratively improves the quality of solutions in the population and converges to the optimal solution over time.

ECC is a type of public key and asymmetric cryptography algorithm that ensures confidentiality, integrity, and authentication of data transmitted over the Internet [13]. SHA-256 is a widely used cryptographic hash function that produces a fixed-size output of 256 bits, regardless of the size of the input [14]. It ensures data integrity by detecting any modification of data during transmission. Even if a single bit of data is changed during transmission, the hash value produced by SHA-256 will be different from the original hash value, indicating that the data has been tampered with [15]. Using ECC for key exchange and digital signatures, and SHA-256 for data integrity can ensure secure communication over the Internet.

1.1 Motivation

Ensuring the security of IoT data from unauthorized access is imperative, and this can be achieved through the use of robust encoding algorithms during data storage and transmission. Breaching data security could pose significant challenges in maintaining confidentiality and accuracy of user data in IoT. Consequently, encoding of data before transmission becomes crucial. This study proposes an effective approach to data security by utilizing the CS algorithm to generate a private key for ECC and also integrating the SHA-256 algorithm.

1.2 Contributions

This research presents four primary outcomes. First, it proposes a novel approach to secure IoT data, which involves the utilization of ECC, CS algorithm, and SHA-256. Second, it devises an efficient private key for ECC with the use of the CS algorithm. Third, it enhances data security by encrypting the ciphertext based on ECC-CS with SHA-256. Finally, the effectiveness of the proposed method is demonstrated by comparing it with other existing models through empirical evaluation.

1.3 Organization

The paper is structured as follows: Related works are reviewed in Section 2. The CS algorithm and ECC are explained in Section 3. Section 4 presents the proposed model. Section 5 provides an evaluation and comparison of the proposed model with the other models. Finally, Section 6 concludes the study and discusses future work.

2 Related Works

This section provides a comprehensive review of previous research relevant to information security in the context of the IoT. One study, documented in [16] described a new approach for encrypting binary images using a stream cipher and ant colony optimization (ACO) based key generation, which overcomes limitations of existing image encoding methods. The study also provides a detailed explanation of the proposed method. In [17], a novel secure visual secret sharing (VSS) scheme that employs e ECC and optimization techniques has been presented to overcome some limitations of existing VSS schemes, including the requirement of a large number of shares and the possibility of unauthorized parties reconstructing shares. In addition, reference [18] introduced an image encoding technique that utilized an optimized key generation process based ECC and Genetic Algorithm (GA) to enhance the security of image encoding. This method aimed to produce a highly randomized and complex key capable of withstanding attacks like brute-force and dictionary attacks. In reference [19], a cryptanalysis of Simplified Data Encoding Standard (SDES) using genetic and memetic algorithms has been used to identify weaknesses in the SDES algorithm by breaking the encoding key and revealing the plaintext message. In addition, a wireless network interface selection algorithm for industrial mobile devices has been proposed in [20], which is based on artificial bee colony optimization (ABC) and aims to minimize energy consumption by intelligently selecting the appropriate wireless interface based on the network conditions. In [21], the authors proposed a new image encoding method that utilized Particle Swarm Optimization (PSO) and a chaotic map to generate a highly randomized and complex key, which can resist various attacks such as brute-force and dictionary attacks. Additionally, reference [22] proposed a modified ECC algorithm for authentication and encoding in communication systems, which used a combination of particle swarm optimization (PSO) and CS algorithm to optimize the parameters of the ECC algorithm, with the aim of improving its efficiency and security. Moreover, reference [23] proposed a secure data hiding scheme using a hybridized algorithm that combines the Fruit Fly Optimization (FFO) algorithm and the Seeker Algorithm to hide sensitive data within an image in a secure and efficient manner while ensuring that the quality of the image remains unaffected. A medical image security scheme that uses a dual encoding approach combining Advanced Encoding Standard (AES) and Rivest-Shamir-Adleman (RSA) algorithms was also proposed [24]. The proposed scheme employed an Oppositional Based Optimization (OBO) algorithm to optimize the encoding parameters and enhance the security of the image. In [25], explored the use of the Harmony Search (HS) algorithm in Public Key Cryptography (PKC), a widely adopted cryptographic technique that enables secure communication over an insecure network through the use of public and private keys. The study proposed a novel method that was compared against the conventional RSA and ElGamal algorithms. Meanwhile, reference [26] proposed a new approach to image encoding that combined the signcryption method and adaptive elephant herding optimization (AEHO). Another research work has analyzed the Geffe generator cryptosystem using particle swarm optimization (PSO) techniques [27]. The Geffe generator is a widely used linear feedback shift register-based stream cipher for generating pseudo-random sequences. Additionally, researchers in [28] introduced a method to increase the energy efficiency of Cognitive Radio Networks (CRNs) that rely on IoT by utilizing a multi-objective ant colony optimization (MACO) algorithm

and double Q-learning algorithm. Both studies presented innovative and promising techniques for optimizing various aspects of network performance. In addition, an improved chaotic firefly algorithm (ICFA) based resource allocation solution for sensor networks in the IoT environment has been proposed in [29]. The proposed method aimed to optimize the resource allocation process by selecting the most suitable sensor nodes based on several criteria such as energy efficiency, network coverage, and data transmission rate. Furthermore, reference [30] proposed a new framework for optimizing resource allocation in Time Division Multiple Access (TDMA)-based Medium Access Control (MAC) protocols on the IoT that considered multiple objectives such as energy efficiency, network throughput, and fairness. The study used a multi-objective optimization algorithm to find the best resource allocation solution. Another study proposed a hybrid lightweight encoding and swarm optimization algorithm for medical data security in healthcare applications in [31]. The proposed algorithm aimed to provide secure transmission and storage of medical data while ensuring minimal computational overhead and energy consumption. Moreover, reference [32] proposed an energy-efficient clustering algorithm for the IoT using an Artificial Bee Colony (ABC) optimization approach to increase the lifetime of IoT networks by reducing the energy consumption of nodes while ensuring efficient data transmission and cluster formation. In [33], an optimal ECC algorithm for secure data storage and transmission in big data environments was proposed, which included an effective access control mechanism to restrict unauthorized access to the data. Another study, detailed in [34], employed an optimized GA-based approach for efficient cluster head election in Healthcare Wireless Sensor Networks (HWSNs) with movable sinks. The proposed method aimed to improve network lifetime and energy efficiency by selecting the most appropriate cluster heads based on factors such as residual energy, distance to the sink, and connectivity to other nodes. Moreover, an enhanced security scheme for RSA and ECC algorithms was proposed using an addition chain optimization approach based on Simplified Swarm Optimization (SSO) and Particle Swarm Optimization (PSO) [35]. The proposed scheme aimed to improve the security of RSA and ECC algorithms, which are commonly used for secure communication in mobile devices. Finally, reference [36] provided a practical solution for addressing security issues associated with the IoT systems, including confidentiality, integrity, and authentication. The proposed approach, which employs the ABC algorithm, has been evaluated on a smart irrigation system, and the results demonstrate its effectiveness in protecting the transmitted data between devices. In conclusion, these works present innovative techniques for addressing security concerns in different domains, which can enhance the security and reliability of the systems.

3 Preliminaries

This section presents an overview of the CS and ECC algorithms.

3.1 Cuckoo Search Algorithm

The Obligate Interspecific Brood Parasites are a group of bird species, including the cuckoo, which have evolved a parasitic behavior due to the cuckoo's rapid egg-laying process. This behavior involves laying their eggs in the nests of other birds and leaving them to hatch and grow without any parental care. As a result, the host's breeding success is negatively affected by decreasing their nesting success, which can occur in various ways, such as the female parasites potentially removing or damaging host eggs, or young parasites evicting or killing host young after hatching. In order to address optimization problems inspired by nature, Cuckoo Search [12] algorithm was developed, which works by creating solutions in nests, which are then improved by modifying certain parts of the solution and eliminating weaker nests. The CS algorithm uses Lévy flights and Markov chain random walk to create new solutions that are even more optimized and efficient.

The term "Lévy flights" was first introduced by the French mathematician Paul Pierre Lévy in the 1930s. Lévy flights refer to a type of random walk where movement lengths follow a probability distribution with a power law tail. This allows for rare but significant movements to occur, a pattern observed in a wide range of natural systems, including the foraging behavior of spider monkeys and the flight patterns of certain mammals. The power law distribution reflects the fact that occasional long-distance movements can be more beneficial than many short ones. Despite being a seemingly random process, Lévy flights have been found to be the best solution for optimizing certain search algorithms, leading to their application in various fields, including finance and biology [37].

Lévy [38] discovered that the equation $y = |x|^{-\alpha}$, where α is between 1 and 2, leads to a stable and balanced distribution. Mantegna [39] created an algorithm to simulate this distribution, and the size of each step is determined by a random variable described in Eq. (1).

$$\eta = \frac{x}{|y|^{1/\beta}} \tag{1}$$

The standard deviation of the variable σ_x can be determined by applying Eq. (2) when the variables x and y follow a normal distribution with a mean of 0 and the standard deviation of σ_y is 1.

$$\sigma_x = \left\{ \frac{\Gamma(1+\beta)\sin(\frac{\pi\beta}{2})}{[\Gamma(1+\beta)/2]\,\beta 2^{(\beta-1)/2}} \right\}^{1/\beta} \tag{2}$$

According to a recommendation made in reference [39], when the standard deviation of x (σx) is 0.6965 and the standard deviation of y (σy) is 1, it is advised to use $\beta = 1.5$ in Eq. (2). Under these conditions, when the absolute value of η is greater than or equal to 10, the probability distributions of η and the symmetrical Lévy stable process exhibit similar asymptotic behavior. In reference [40], the performance of three different methods for generating Lévy noise was compared, and Mantegna's algorithm was found to be the fastest.

The proposed data security model incorporates multiple algorithms, such as the CS, ECC, and SHA-256. The CS algorithm generates private keys for ECC, while SHA-256 ensures privacy and security. This comprehensive solution ensures the confidentiality, integrity, and authenticity of data, which makes it suitable for a variety of applications requiring high-level security, such as online banking, e-commerce, and government systems.

3.2 Elliptic-Curve Cryptography

ECC is a cryptographic technique that employs mathematical operations on small fields to provide security. It is faster than other encoding techniques because of its short key length. The security of ECC is maintained due to the complex exponential discrete logarithm problem [13]. To apply ECC, two numbers (a and b) are necessary from a particular set called Fp. Moreover, an elliptical curve has a set of points denoted as E(Fp), with Q being one of these points as shown in Eq. (3).

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

The process of encrypting data using ECC involves selecting a random number x from the Fp set, between 1 and n−1, as a private key. The plaintext is first converted into bits and then mapped onto an elliptic curve as (x, y) points. These points are subsequently encrypted and converted back into bits. The process of encrypting an elliptic curve involves the following steps:

1. Initialization: The encoding algorithm is initialized by selecting appropriate parameters and initializing necessary data structures, such as E, Q, and p. In some cases, this step may require generating random numbers or other initial values.

2. Public Key Generation: A cryptographic key pair consisting of a public key and a private key is created. The public key is used for data encoding, while the private key is used for decoding. The keys are generated in a way that makes it computationally infeasible to derive the private key from the public key. The private key, x, is used to determine the public key (H), which is determined by the equation H = x.Q.
3. Encoding: Once the public key is generated, the plaintext data is encrypted using the public key. Encoding involves transforming the original data into an unreadable form without the corresponding private key, as depicted in Eq. (4), where r represents a random number.

$$Ciphertext = Enc\,(data) = \begin{cases} Ciphertext_1 = r.Q \\ Ciphertext_2 = data + r.H \end{cases} \tag{4}$$

4. Decoding: The encrypted data is decoded using the private key, as shown in Eq. (5). Decoding is the process of reversing the encoding process and transforming the encrypted data back into its original form. Only the owner of the private key can decrypt the data, ensuring confidentiality and security.

$$\begin{aligned} Dec\,(Ciphertext) &= Ciphertext_2 - x.Ciphertext_1 = data + r.H - x.r.Q \\ &= data + x.r.Q - x.r.Q = data \end{aligned} \tag{5}$$

4 The Proposed Model

In order to ensure the secure encoding and decoding of data using ECC, it is necessary to generate a random private key. To address this requirement, a proposed approach utilizes CS optimization to generate such a key. The quality of the encoding for sensitive data in ECC is significantly impacted by the randomness of the key used in the encoding process. CS is a mathematical technique that searches for solutions in the search space, which is analogous to the discrete nests or eggs of host birds. In this context, the objective of the CS algorithm is to identify the optimal private key (Pr) for ECC, which can allow for accurate and secure decoding of the ciphertext to the original plaintext. The fitness of a solution, i.e., a private key, is determined by comparing the decrypted plaintext with the original plaintext, and evaluating a fitness value based on the accuracy and security of the decoding. The objective function for the CS algorithm in this context can be formulated as follows:

$$f(Pr) = -h(Dec(CP, Pr), B) \tag{6}$$

where, CP is the ciphertext obtained after encoding, B is the original plaintext, Dec(CP, Pr) is the decoding of the ciphertext CP using the private key Pr, h(x, y) is a function that measures the similarity or distance between the decrypted plaintext x and the original plaintext y.

In this case, the negative sign is used to convert the fitness function into a minimization problem, where the goal is to minimize the distance between the decrypted plaintext and the original plaintext.

The following steps are involved in the proposed model:

1. The Cuckoo Search method starts by setting certain values: the number of nests (n), the size of each nest (m), the chance of finding foreign eggs (pa), the step size ($\alpha = 0.01$), and the maximum number of iterations.
2. The host bird's eggs are set up by picking a random sequence of numbers with the same length as the private key. For ECC, the private key is 256 bits long.
3. New eggs are created using Eq. (1), except for the best egg, which stays the same. If the new eggs are better than the current ones, they are selected randomly using Eq. (6).

4. Bad nests are removed by finding alien eggs through a process of generating a random number and comparing it to the chance of discovering alien eggs (pa). If the random number is less than pa, a new solution is created. If it is better than the current nests, it replaces them by taking small steps from their current positions.
5. The creation of new nests and the finding of alien eggs are repeated until either the maximum number of attempts is reached, or the best nest reaches the desired quality level. The values of n, pa, and α are determined through experimentation.

The new method saves time by automatically assigning strings without user input. It uses the CS algorithm to find the private keys for users 1 and 2, while the public key is based on Q and the private key. The algorithm runs for a maximum of 175 iterations.

Algorithm 1 initializes variables and functions. It starts by defining the ECC curve equation and calculating point Q. During encoding, the function CS() finds the best discrete nests or eggs for the private key, and then SHA-256 is used to hash and encrypt texts. During decoding, the hash function is applied first, employing XOR, and, right-shift, and left-shift procedures, to modify the initial hash values. ECC decoding is then performed.

The proposed method offers many benefits: it automates the string assignment process, reduces time, and ensures secure communication. It also uses CS to optimize the solution and employs hash functions for extra security. Overall, this new model is an efficient and secure way to assign strings and transmit data between users.

To increase data confidentiality, the suggested approach follows ECC encoding with the XOR and shift-left operators. There are four ECC-based phases in these operators: Encoding (1), Encoding (2), Decoding (1), and Decoding (2). This strengthens the model's core and ensures secure message transmission. During decoding, the encrypted text undergoes shift-left operations and XOR before using the ECC algorithm.

Algorithm 1:

1. ECC():
- a) Perform the initialization phase.
- b) Determine a large prime number (p) using the parameters E and Fp.
- c) Select public parameters.
- d) Produce a base point (Q).
- e) Generate a secret key (Pr) using the CS() function.
- f) Calculate a public key (Pu) by multiplying the base point (Q) with the private key (Pr).

2. Encoding():
- a) Obtain the plaintext from a text file (B).
- b) Generate a private key (PrB) using the CS() function.
- c) Convert the plaintext to an integer value (m).
- d) Generate a random number (r).
- e) Calculate the first part of the ciphertext1 by multiplying the base point (Q) with the random number (r).
- f) Calculate the second part of the ciphertext2 by adding the plaintext (m) with the product of the secret key (PrA) and the public key (Pu).
- g) Store the ciphertext as a pair of ciphertext1 and ciphertext2 (CP).
- h) Calculate the SHA-256 hash of the plaintext to obtain the encrypted text.

(Continued)

Algorithm 1 (continued)

3. Decoding():

a) Calculate the SHA-256 hash of the encrypted text to obtain the original plaintext.
b) Obtain the ciphertext (CP).
c) Extract the left part of the ciphertext1.
d) Extract the right part of the ciphertext2.
e) Obtain the original plaintext (m) by subtracting the product of the secret key (PrA) and the base point (ciphertext1) from the second part of the ciphertext (ciphertext2).
f) Convert the integer value of the plaintext (m) to its corresponding plain text.

4. CS():

a) Initialize the current solutions.
b) Initiate a loop.
c) Evaluate the fitness for each solution in the population of size n.
d) Identify the best solution.
e) For each solution in the population, take a step towards the best solution using a normally distributed random number to calculate the step size.
f) Obtain the new solution by rounding off the result of the step.
g) Keep the solution within the bounds of the problem.
h) Evaluate the fitness of the new solution.
i) If the fitness of the new solution is better than the fitness of the previous solution, accept the new solution.
j) Choose two random solutions from the population.
k) Generate a random scaling factor.
l) Generate a new candidate solution by adding the scaled difference between the two solutions to each solution in the population.
m) Evaluate the fitness of the candidate solution.
n) If the fitness of the candidate solution is better than the fitness of the previous solution, accept the candidate solution.
o) Randomly permute the population.
p) Obtain another random permutation of the population.
q) Generate a random number.
r) Evaluate the fitness of the candidate solution.
s) Generate another random number.
t) If the random number is less than a certain probability (pa), accept the candidate solution.
u) Keep the solution within the bounds of the problem.
v) Evaluate the fitness of the best solution.
w) Continue the loop until either the maximum number of iterations is reached or the fitness threshold is achieved.
x) Evaluate the fitness of the best solution.
y) End the loop.

5. SHA-256():

a) Initialize the hash values.
b) Initialize the round constants.
c) Initialize the XOR and shift right operators.
d) Initialize the working variables.

(Continued)

Algorithm 1 (continued)

e) Run the main loop of the compression function.
f) Add the compressed chunk to the current hash value.
g) Produce the final hash value.
h) End the function.

5 Evaluation and Results

This section aims to evaluate the encoding and decoding throughput and speed of the proposed model. The experiments were executed on a Windows 10 Pro machine, utilizing C# programming language via Microsoft Visual Studio 2012. The computer was equipped with an AMD E-350 Processor, running at a frequency of 1.60 GHz, and 4 GB of RAM.

The initial population size, randomly chosen by the algorithm, was influenced by the key length. In this specific model, the key length was determined by selecting random integers between 0 and 127. The size of the initial population was therefore dependent on this key length. The performance evaluation of the encoding and decoding process was based on these experimental settings.

5.1 Performance Analysis

5.1.1 Decoding and Encoding Times

This section provides an assessment of the encoding and decoding speeds of the proposed model. The outcomes are depicted in Figs. 1 and 2, correspondingly. Fig. 1 illustrates the encoding time (measured in milliseconds) of the proposed model at various iterations with an initial population of 20. The results indicate that the encoding time diminishes as the number of iterations increases, and it is minimized at 75 iterations.

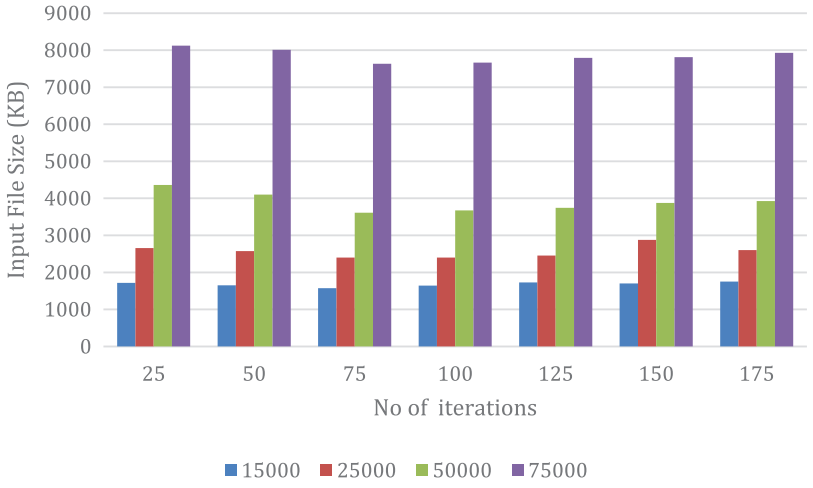

Figure 1: Encoding time based on iterations

For example, when encoding a file with a size of 75,000 KB, the encoding time is 8,125 ms for 25 iterations, 7,635 ms for 75 iterations, and 7,930 ms for 175 iterations. The findings are supported by the data presented in the figures, which were obtained through a comprehensive experimental study conducted on the proposed model.

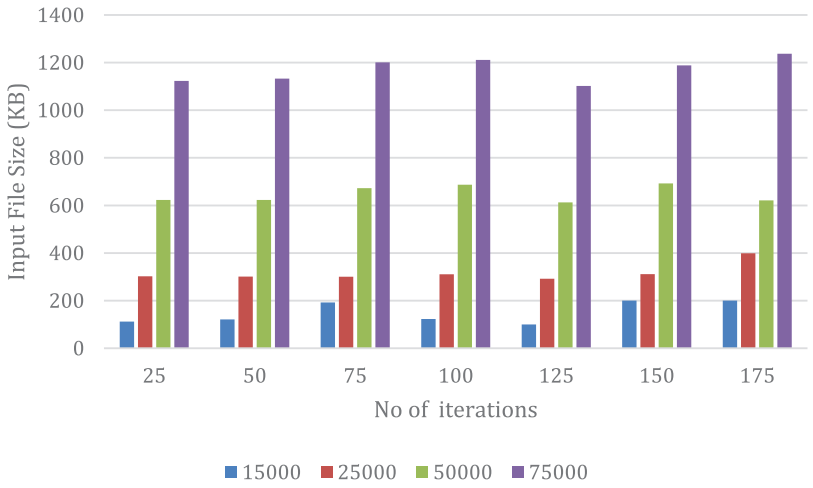

Figure 2: Decoding time based on iterations

On the contrary, Fig. 2 illustrates the decoding time in milliseconds of the suggested model at different iterations while employing an initial population of 20. The optimal condition is when the decoding time is less than the encoding time, which results in a prompt and efficient system. The findings indicate that the decoding time reaches its minimum at 125 iterations. For instance, in decrypting a file of 75,000 KB, the decoding time measures at 1123.201 ms for 25 iterations, 1102.012 ms for 125 iterations, and 1237.072 ms for 175 iterations.

According to the results depicted in Fig. 3, the proposed model exhibits varying mean encoding and decoding times at different iterations. The most efficient encoding time is observed at the 75th iteration, whereas the optimal decoding time is achieved at the 125th iteration. These findings indicate that the proposed model satisfies the desired security level while also demonstrating enhanced execution time. Hence, it can be inferred that the proposed model is a viable solution for achieving secure and efficient data encoding and decoding.

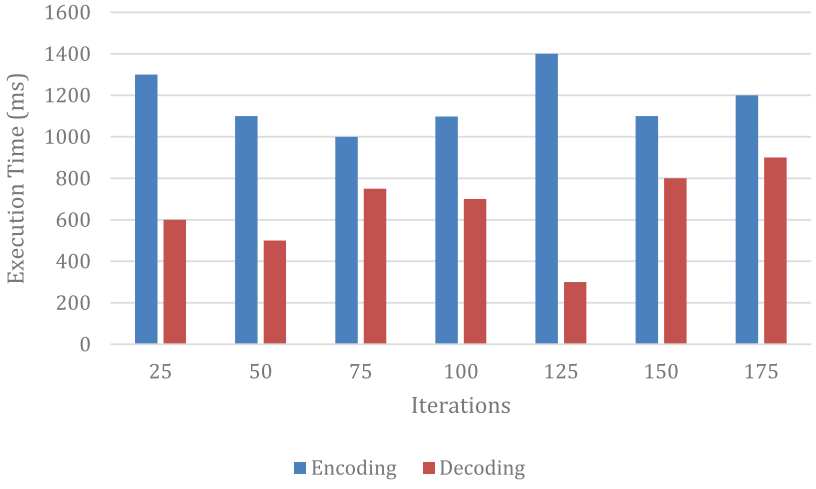

Figure 3: Times on average for encoding and decoding at various iterations

The findings of this assessment indicate that the suggested model is effective in terms of both encoding and decoding speed. The optimal number of iterations for minimizing encoding time is 75, while the optimal number for minimizing decoding time is 125. These results can guide the development of more efficient and effective encoding algorithms in the future, and contribute to the advancement of the field of cryptography. Furthermore, the proposed model's improved execution time can provide practical benefits for various applications, especially those involving large file sizes.

5.2 Comparison and Evaluation

5.2.1 Encoding and Decoding Based on AES-RSA and Proposed Model

In this section, a comparative analysis of the suggested model and AES-RSA [41] is presented in Table 1, based on the evaluation of 14 files with varying sizes and a 128-bit key length using Visual Studio. The main objective of this study was to compare the efficiency of the proposed model with AES-RSA in terms of encoding and decoding time. The results indicate that the proposed model outperforms AES-RSA, specifically when dealing with a data size of 50 MB. The encoding time of AES-RSA in this scenario is 6.53 s, while the proposed model only takes 3.29 s, demonstrating a significant improvement in processing time.

Table 1: An evaluation of the suggested model in contrast to AES-RSA

File size (MB)	AES-RSA		Proposed model	
	Encoding	Decoding	Encoding	Decoding
1	1.73	18.92	0.87	0.75
3	2.01	19.72	1.11	1.01
7	2.05	18.49	1.35	1.22
10	3.01	21.08	1.92	1.89
17	3.61	21.91	2.04	1.99
21	3.72	29.01	2.21	2.11
25	3.73	32.28	2.41	2.23
28	4.57	32.01	2.79	2.43
32	4.42	31.08	2.31	2.37
35	5.29	28.1	3.12	2.79
39	4.97	27.93	3.24	2.87
42	5.52	27.88	3.53	3.12
46	5.62	29.73	3.91	3.51
50	6.53	31.59	4.62	3.29

While RSA is a well-known heavyweight cryptography system due to its large key size, the ECC offers a more efficient alternative with a smaller key size. This results in faster processing speeds and less memory usage [42]. Therefore, the ECC can be implemented on devices with limited resources, leading to quicker computations in real-time.

According to the results presented in Fig. 4, the encoding time of the proposed model was compared to that of AES-RSA. The findings indicate that the proposed model's encoding time is considerably shorter, resulting in a reduction of about 29.24% in the execution time of encoding on a

file size of 50 MB. Furthermore, the proposed model has been demonstrated to outperform AES-RSA, establishing its superiority over the current AES-RSA approach in terms of efficiency.

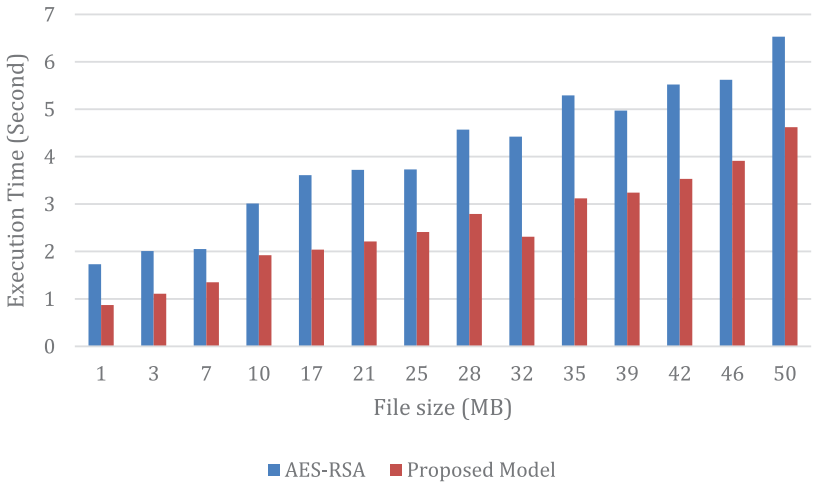

Figure 4: A comparison of the suggested model's and AES-RSA's encoding times

The comparison of decoding time between the proposed model and AES-RSA is depicted in Fig. 5, indicating that the proposed model outperforms AES-RSA in terms of speed. The RSA algorithm's slower performance can be attributed to its computationally intensive encoding and decoding processes, which involve modular exponentiation operations and private key exponent calculations. These operations require a significant number of computations, resulting in a slower processing speed. In contrast, the proposed model requires fewer computations during its encoding and decoding processes, resulting in faster processing speeds when compared to RSA. The proposed model's superior performance is therefore attributed to its reduced computational requirements.

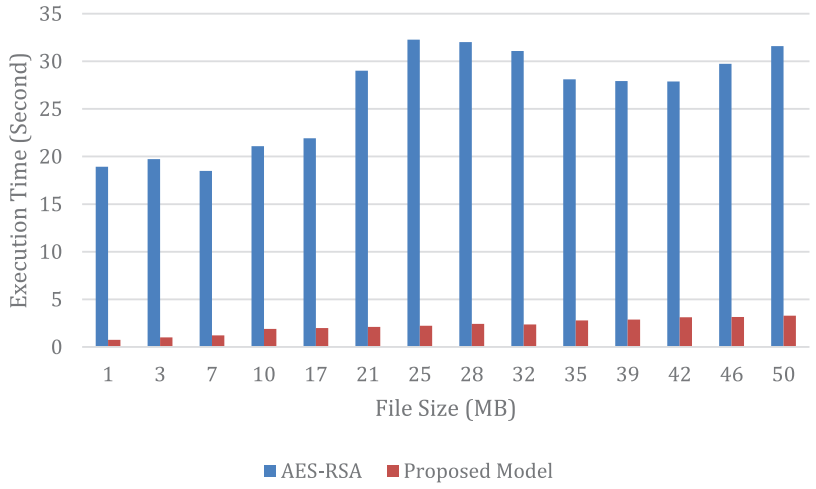

Figure 5: Evaluating the differences between the suggested model's and AES-RSA's decoding times

In brief, the model suggested in this study surpasses AES-RSA in both encoding and decoding speed, thereby proving to be a more efficient and feasible alternative for cryptographic applications, particularly in situations with limited resources. The utilization of ECC in the suggested model curtails

the computational burden, rendering it more rapid and well-suited for real-time scenarios. These advantages make the proposed model a promising solution for secure data communication.

5.2.2 Encoding and Decoding Based on FA, GA, and Proposed Model

This section presents a comparative analysis of two metaheuristic algorithms, namely the Firefly Algorithm (FA) [43] and Genetic Algorithm (GA) [44], in conjunction with the proposed model for generating ECC keys. These algorithms utilize selection, crossover, and mutation operators to search for an optimal ECC solution. The findings, presented in Table 2, indicate that the proposed CS algorithm outperforms both FA and GA in terms of encoding and decoding times, thereby yielding better solutions, search, and power convergence. More specifically, the results demonstrate that the proposed CS algorithm shows faster encoding and decoding times than both FA and GA, although FA performs better than GA. For instance, when encoding a 10,000 KB file, ECC-FA-SHA-256 and ECC-GA-SHA-256 required 4224 and 4370 ms, respectively, while their decoding times were 4151 and 4292 ms, respectively. In contrast, the proposed model exhibited encoding and decoding times of 4011 and 3726 ms, respectively.

Table 2: Presents a comparison of the encoding and decoding times (in milliseconds) of the proposed model with those of the (FA) and (GA)

File size (KB)	ECC-FA-SHA-256		ECC-GA-SHA-256		Proposed model	
	Encoding	Decoding	Encoding	Decoding	Encoding	Decoding
1000	105	101	103	99	75	63
2000	172	164	198	183	141	121
3000	237	228	251	247	221	193
4000	313	310	317	307	290	253
5000	391	387	402	398	374	350
6000	447	438	501	489	467	429
7000	580	563	583	571	527	513
8000	607	599	623	617	593	529
9000	671	667	679	673	645	597
10000	701	694	713	708	678	678
55000	4224	4151	4370	4292	4011	3726
Throughput	13.02083333	13.24981932	12.58581236	12.81453868	13.7122912	14.76113795

The results presented in Figs. 6 and 7 offer supplementary proof that the suggested model surpasses alternative approaches in both execution time and encoding efficiency. As a result, the suggested model may be considered a superior choice when compared to existing techniques for generating ECC keys within the context of the CS algorithm. These findings strongly suggest that the implementation of the suggested CS algorithm leads to an increase in the security and efficiency of cryptographic systems.

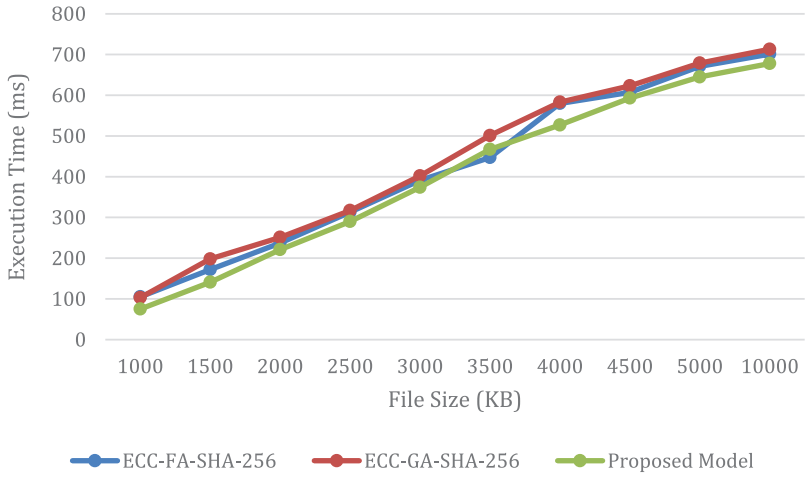

Figure 6: Illustrates a comparison of the encoding time between the proposed model and GA and FA

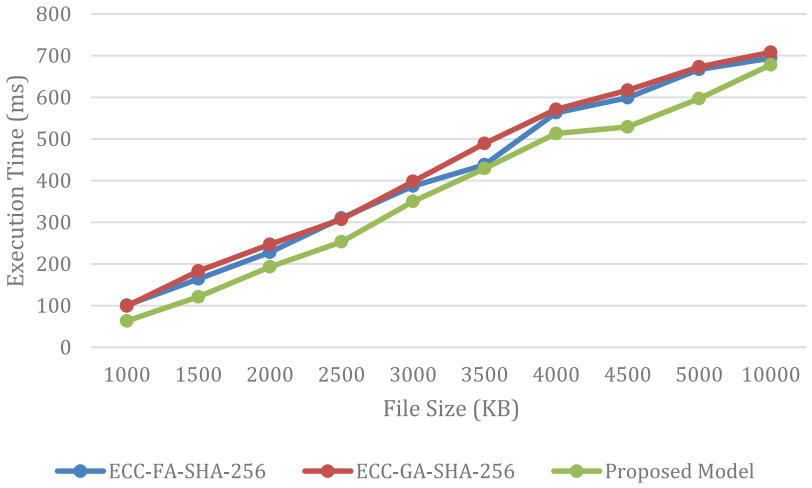

Figure 7: Illustrates a comparison of the decoding time of the proposed model with that of GA and FA

5.3 Throughput

In this study, the proposed model was assessed in terms of its encoding and decoding capabilities, as measured by speed, which was determined using Eqs. (7) and (8) [45]. A higher measurement was indicative of a superior model. To evaluate the model's performance, it was tested on a file with a size of 55000 KB, and the results were compared with those of the (ECC-FA-SHA-256) and (ECC-GA-SHA-256) models. The proposed model demonstrated encoding and decoding throughputs of 13.71 and 14.76 ms, respectively. In contrast, the (ECC-FA-SHA-256) model had encoding and decoding throughputs of 13.02 and 13.24 ms, while the (ECC-GA-SHA-256) model had encoding and decoding throughputs of 12.58 and 12.81 ms, respectively. Therefore, the proposed model outperformed both of the comparison models in terms of encoding and decoding speed.

$$\text{Encoding Throughput} = \Sigma(\text{Input File})/\Sigma(\text{Encoding Time}) \tag{7}$$

$$\text{Decoding Throughput} = \Sigma(\text{Input File})/\Sigma(\text{Decoding Time}) \quad (8)$$

The proposed model's encoding and decoding throughputs were found to be approximately 5.31% and 11.41% better than those of the (ECC-FA-SHA-256) model, respectively. Similarly, the proposed model's encoding and decoding throughputs were approximately 8.95% and 15.17% better than those of the (ECC-GA-SHA-256) model, respectively. These findings indicate that the proposed model is significantly more efficient and effective in encoding and decoding data quickly and securely than the other models.

6 Conclusion and Future Works

In contemporary times, the IoT has become ubiquitous in various aspects of daily life. Nevertheless, ensuring security on IoT devices remains a significant challenge due to their constrained resources. IoT devices transmit diverse types of data, including messages, images, and sounds, and it is crucial to maintain the confidentiality and integrity of such data. In order to accomplish this goal, certain security algorithms exhibit superior efficiency compared to others. This study presents a novel approach to secure data transmitted by small IoT-based door locks utilizing ECC and SHA-256 with the CS algorithm. The proposed approach is efficient and ensures that the encoding and decoding of data require less time. The investigation indicates that utilizing 75 iterations for encoding and 125 iterations for decoding is optimal. When compared to other techniques, the proposed model is 15.17% more efficient and provides adequate protection against attacks, making it suitable for devices with limited resources. To verify the efficacy of the proposed method, the study conducted simulations, and future research will explore alternative ways to generate private keys.

Acknowledgement: Grateful thanks to the Department of Mathematics, Open Educational College, Kirkuk Branch, and Mutafaweqat High School for Girls, Kirkuk Branch, for their invaluable support.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm their contributions to the paper as follows: study conception and design: Arkan Kh Shakr Sabonchi, Zainab Hasim Obaid; data collection: Zainab Hasim Obaid; analysis and interpretation of results: Arkan Kh Shakr Sabonchi, Zainab Hasim Obaid; draft manuscript preparation: Zainab Hasim Obaid. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and materials utilized in this study are available upon request from the corresponding author, in compliance with the relevant data protection and privacy regulations. Due to confidentiality and ethical considerations, certain data may not be made publicly accessible. However, efforts have been made to ensure transparency and reproducibility, and interested researchers can contact the corresponding author to discuss access to the data within the bounds of ethical and legal restrictions.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] L. Kong and B. Ma, "Intelligent manufacturing model of construction industry based on Internet of Things technology," *The International Journal of Advanced Manufacturing Technology*, vol. 107, pp. 1025–1037, 2020.

[2] A. Jha, A. Athanerey and A. Kumar, "Role and challenges of internet of things and informatics in healthcare research," *Health and Technology*, vol. 12, no. 4, pp. 701–712, 2022.

[3] A. Kamilaris and A. Pitsillides, "Mobile phone computing and the internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 885–898, 2016.

[4] P. M. Chanal and M. S. Kakkasageri, "Security and privacy in IoT: A survey," *Wireless Personal Communications*, vol. 115, pp. 1667–1693, 2020.

[5] K. R. Sarkar, "Assessing insider threats to information security using technical, behavioural and organisational measures," *Information Security Technical Report*, vol. 15, no. 3, pp. 112–133, 2010.

[6] I. Makhdoom, M. Abolhasan, H. Abbas and W. Ni, "Blockchain's adoption in IoT: The challenges, and a way forward," *Journal of Network and Computer Applications*, vol. 125, pp. 251–279, 2019.

[7] A. Ghani, K. Mansoor, S. Mehmood, S. A. Chaudhry, A. U. Rahman *et al.,* "Security and key management in IoT-based wireless sensor networks: An authentication protocol using symmetric key," *International Journal of Communication Systems*, vol. 32, no. 16, pp. e4139, 2019.

[8] C. R. Aldawira, H. W. Putra, N. Hanafiah, S. Surjarwo and A. Wibisurya, "Door security system for home monitoring based on ESP32," *Procedia Computer Science*, vol. 157, pp. 673–682, 2019.

[9] I. Ha, "Security and usability improvement on a digital door lock system based on internet of things," *International Journal of Security and its Applications*, vol. 9, no. 8, pp. 45–54, 2015.

[10] A. Jacobsson, M. Boldt and B. Carlsson, "A risk analysis of a smart home automation system," *Future Generation Computer Systems*, vol. 56, pp. 719–733, 2016.

[11] M. Ahtsham, H. Y. Yan and U. Ali, "IoT based door lock surveillance system using cryptographic algorithms," in *2019 IEEE 16th Int. Conf. Networking, Sensing and Control (ICNSC)*, Banff, AB, Canada, pp. 448–453, 2019.

[12] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, Coimbatore, India, pp. 210–214, 2009.

[13] V. S. Miller, "Use of elliptic curves in cryptography," in *12th Int. Workshop*, Canada, SAC, pp. 417–426, 2005.

[14] H. Yoshida and A. Biryukov, "Analysis of a SHA-256 variant," in *12th Int. Workshop*, Canada, SAC, pp. 245–260, 2006.

[15] S. R. Prasanna and B. S. Premananda, "Performance analysis of MD5 and SHA-256 algorithms to maintain data integrity," in *2021 Int. Conf. on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, Bangalore, India, pp. 246–250, 2021.

[16] N. K. Sreelaja and G. V. Pai, "Stream cipher for binary image encryption using ant colony optimization based key generation," *Applied Soft Computing*, vol. 12, no. 9, pp. 2879–2895, 2012.

[17] K. Shankar and P. Eswaran, "A secure visual secret share (VSS) creation scheme in visual cryptography using elliptic curve cryptography with optimization technique," *Australian Journal of Basic & Applied Science*, vol. 9, no. 36, pp. 150–163, 2015.

[18] K. Shankar and P. Eswaran, "An efficient image encryption technique based on optimized key generation in ECC using genetic algorithm," in *Artificial Intelligence and Evolutionary Computations in Engineering Systems: Proc. of ICAIECES 2015*, India, Springer, pp. 705–714, 2016.

[19] K. Dworak, J. Nalepa, U. Boryczka and M. Kawulok, "Cryptanalysis of SDES using genetic and memetic algorithms," in *Recent Developments in Intelligent Information and Database Systems*, Springer, Cham, pp. 3–14, 2016.

[20] K. Kalaiselvi and A. Kumar, "An empirical study on effect of variations in the population size and generations of genetic algorithms in cryptography," in *2017 IEEE Int. Conf. on Current Trends in Advanced Computing (ICCTAC)*, Bangalore, India, pp. 1–5, 2017.

[21] M. Ahmad, M. Z. Alam, Z. Umayya, S. Khan and F. Ahmad, "An image encryption approach using particle swarm optimization and chaotic map," *International Journal of Information Technology*, vol. 10, pp. 247–255, 2018.
[22] S. Kota, V. N. R. Padmanabhuni and K. Budda, "Authentication and encryption using modified elliptic curve cryptography with particle swarm optimization and cuckoo search algorithm," *Journal of the Institution of Engineers (India): Series B*, vol. 99, no. 4, pp. 343–351, 2018.
[23] R. Kiruba and T. Sree Sharmila, "Secure data hiding by fruit fly optimization improved hybridized seeker algorithm," *Multidimensional Systems and Signal Processing*, vol. 33, no. 3, pp. 1423–1443, 2020.
[24] T. Avudaiappan, R. Balasubramanian, S. S. Pandiyan, M. Saravanan, S. K. Lakshmanaprabu *et al.,* "Medical image security using dual encryption with oppositional based optimization algorithm," *Journal of Medical Systems*, vol. 42, pp. 1–11, 2018.
[25] S. Mitra, G. Mahapatra, V. E. Balas and R. Chattaraj, "Public key cryptography using harmony search algorithm," in *Innovations in Infrastructure*, Singapore, Springer, pp. 1–11, 2019.
[26] K. Shankar, M. Elhoseny, E. Perumal, M. Ilayaraja and K. Sathesh Kumar, "An efficient image encryption scheme based on signcryption technique with adaptive elephant herding optimization," in *Cybersecurity and Secure Information Systems: Challenges and Solutions in Smart Environments*. Springer, Cham, pp. 31–42, 2019.
[27] M. Din, S. K. Pal and S. K. Muttoo, "Applying PSO based technique for analysis of geffe generator cryptosystem," in *Harmony Search and Nature Inspired Optimization Algorithms: Theory and Applications, ICHSA 2018*. Singapore: Springer, pp. 741–749, 2019.
[28] S. Vimal, M. Khari, R. G. Crespo, L. Kalaivani, N. Dey *et al.,* "Energy enhancement using multiobjective ant colony optimization with double Q learning algorithm for IoT based cognitive radio networks," *Computer Communications*, vol. 154, pp. 481–490, 2020.
[29] Z. Wang, D. Liu and A. Jolfaei, "Resource allocation solution for sensor networks using improved chaotic firefly algorithm in IoT environment," *Computer Communications*, vol. 156, pp. 91–100, 2020.
[30] P. Siddavaatam and R. Sedaghat, "A novel multi-objective optimizer framework for TDMA-based medium access control in IoT," *CSI Transactions on ICT*, vol. 8, pp. 319–330, 2020.
[31] K. Tamilarasi and A. Jawahar, "Medical data security for healthcare applications using hybrid lightweight encryption and swarm optimization algorithm," *Wireless Personal Communications*, vol. 114, pp. 1865–1886, 2020.
[32] S. Yousefi, F. Derakhshan, H. S. Aghdasi and H. Karimipour, "An energy-efficient artificial bee colony-based clustering in the internet of things," *Computers & Electrical Engineering*, vol. 86, pp. 106733, 2020.
[33] O. P. Verma, N. Jain and S. K. Pal, "Design and analysis of an optimal ECC algorithm with effective access control mechanism for big data," *Multimedia Tools and Applications*, vol. 79, pp. 9757–9783, 2020.
[34] A. S. Nandan, S. Singh and L. K. Awasthi, "An efficient cluster head election based on optimized genetic algorithm for movable sinks in IoT enabled HWSNs," *Applied Soft Computing*, vol. 107, pp. 107318, 2021.
[35] A. Mullai and K. Mani, "Enhancing the security in RSA and elliptic curve cryptography based on addition chain using simplified swarm optimization and particle swarm optimization for mobile devices," *International Journal of Information Technology*, vol. 13, pp. 551–564, 2021.
[36] S. K. Mousavi and A. Ghaffari, "Data cryptography in the internet of things using the artificial bee colony algorithm in a smart irrigation system," *Journal of Information Security and Applications*, vol. 61, pp. 102945, 2021.
[37] D. W. Sims, D. Righton and J. W. Pitchford, "Minimizing errors in identifying Lévy flight behaviour of organisms," *Journal of Animal Ecology*, vol. 76, no. 2, pp. 222–229, 2007.
[38] P. Lévy, "Théorie de l'addition des variables aléatoires," *Bulletin de la Société Mathématique de France*, vol. 67, pp. 1–41, 1939.
[39] X. S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.
[40] M. Conti, N. Dragoni and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[41] L. Zou, M. Ni, Y. Huang, W. Shi and X. Li, "Hybrid encryption algorithm based on AES and RSA in file encryption," in *Frontier Computing: Theory, Technologies and Applications (FC 2019)*, vol. 8. Singapore: Springer, pp. 541–551, 2020.
[42] M. Bafandehkar, S. M. Yasin, R. Mahmod and Z. M. Hanapi, "Comparison of ECC and RSA algorithm in resource constrained devices," in *2013 Int. Conf. on IT Convergence and Security (ICITCS)*, Macao, China, pp. 1–3, 2013.
[43] X. S. Yang, "Firefly algorithms for multimodal optimization," in *Int. Symp. on Stochastic Algorithms*, Sapporo, Japan, pp. 169–178, 2009.
[44] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.
[45] A. Lemma, M. Tolentino and G. Mehari, "Performance analysis on the implementation of data encryption algorithms used in network security," *International Journal of Computer and Information Technology*, vol. 4, no. 4, pp. 711–717, 2015.