**DOI**: https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009 ***Int. J. Exp. Res. Rev.*, Special Vol. 31: 85-97 (2023)**

*Original Article* | *Peer Reviewed* | *Open Access*

International Journal of Experimental Research and Review (IJERR)

ISSN: 2455-4855 (Online)
www.iaph.in

ISSN 2455-4855

# Salp Swarm Algorithm to solve Cryptographic Key Generation problem for Cloud computing

**Waseem Kaleem[1], Mohammad Sajid[1*] and Ranjit Rajak[2]**

[1]Department of Computer Science, Aligarh Muslim University, Aligarh-202002, Uttar Pradesh, India; [2]Department of Computer Science and Applications, Dr. Harisingh Gour Central University, Sagar-470003, Madhya Pradesh, India

**E-mail/Orcid Id:**

*WK,* waseemkaleem14@gmail.com, https://orcid.org/0009-0006-1514-6119; *MS,* sajid.cst@gmail.com, https://orcid.org/0000-0001-8822-5332; *RR,* ranjit.jnu@gmail.com, https://orcid.org/0000-0003-2746-3278

**Article History:**

**Received:** 22nd May., 2023
**Accepted:** 26nd Jun., 2023
**Published:** 30thJul., 2023

**Keywords:**
Cryptography, randomness, key generation, Shannon entropy, transfer function, Quantization method

**Abstract:** Cryptographic keys are long strings of random bits generated using specialized algorithms and help secure data by making it unpredictable to any adversary. Cryptographic keys are used in various cryptographic algorithms in many domains, i.e., Cloud computing, Internet-of-Things (IoT), Fog computing, and others. The key generation algorithms are essential in cryptographic data encryption and decryption algorithms. This work proposed a cryptographic key generation algorithm based on Shannon entropy and the Salp Swarm algorithm (SSA) for generating randomized keys. The proposed Cryptographic Key Generation algorithm utilizes the dynamic movement of salps to create high-quality, robust, and randomized keys against attacks. The transfer function and quantization method convert a salp into a cryptographic key. The proposed Cryptographic Key Generation algorithm has been evaluated on four transfer functions against three state-of-the-art swarm intelligence metaheuristics, i.e., particle swarm optimization, BAT, and grey wolf optimization algorithms. The keys of eight different bit lengths, i.e., 512, 256, 192, 128, 96, 80, 64, were generated and evaluated due to their applications in the different encryption algorithms, i.e., AES, DES, PRESENT, SIMON, SPECK, and 3DES. The simulation study confirms that the proposed key generation algorithm effectively produces secure cryptographic keys.

## Introduction

Internet-of-Things (IoT), Fog Computing, and Cloud Computing have become indistinguishable components of every human's personal, social, and corporate sphere. IoT Analytics envisages that roughly 27 billion IoT devices will be available globally by 2025 and generate vast amounts of data (IoT Analytics, 2022). Gartner indicates that worldwide end-user expenditures on public cloud services will be increased by 20.7% to $591.8 billion in 2023 from $490.3 billion in 2022 (Gartner, 2021). Cloud computing allows IoT devices to manage, view, and store data remotely via the Internet instead of storing and managing locally on servers or hard drives. Due to rapid growth in cloud adoption and many issues like insecure APIs, data security, and hacker intervention, user data security and privacy have become significant concerns. Data security and confidentiality defend data from illicit access, disclosure, use, destruction, alteration, or disruption. It is important because data is a critical asset for businesses and individuals. It can be precious to attackers seeking to commit fraud, identity theft, corporate espionage, or other malicious activities (Tabrizchi et al., 2020; Alouffi et al., 2021). Thus, it is necessary to employ highly secure cryptographic algorithms for the encryption and decryption of data. Cryptographic algorithms provide a way to protect sensitive data by transforming it into an unreadable form without the proper decryption key. It guarantees that even if cyber criminals gain access to the data, they can decipher or utilize it without the appropriate authorization. Researchers and academicians have developed and implemented dozens of cryptographic algorithms yearly to provide data security. Each cryptographic algorithm uses a cryptographic key which

***Corresponding Author:** sajid.cst@gmail.com

must be randomized and un-predicable. The cryptographic key ensures data confidentiality, integrity, authenticity, and access control.

Without keys, cryptographic algorithms could not protect sensitive information effectively (Alouffi et al., 2021; Jawed and Sajid, 2022). Key generation is essential to cryptography, as the system's security depends on the keys' secrecy. Figure 1 depicts the expansion of scholarly study on the key generation, a research problem of interest for academics and scientists. The nature of the cryptographic key needs to be very random, devoid of patterns, and complicated in its makeup. The key's length is another factor that influences the encryption's efficacy. The challenge of decrypting data gets more intricate as key size increases. The generation of such cryptographic keys is classified as an NP-hard problem (Tahir et al., 2021; Jawed and Sajid, 2022).

Exact and metaheuristic algorithms are two approaches that can be utilized to deal with NP-Hard problems. By leveraging the power of metaheuristic algorithms, a category of computational intelligence, it is possible to solve NP-Hard problems within a reasonable timeframe. Metaheuristics algorithms are often population-based, i.e., they operate on a population of candidate solutions that are evolved and improved over time. The population is updated iteratively; with each iteration, the population changes and improves. Metaheuristic methods have been established to effectively resolve numerous optimization problems across various domains. In cloud security, innumerable metaheuristic algorithms have successfully addressed various security challenges, such as intrusion detection, access control, key management, data breaches, network security, and more. Additionally, these algorithms have been utilized to optimize cloud workload scheduling, resource allocation, substitution boxes, pseudo-number generation, and other related areas (Osaba et al., 2021; Dokeroglu et al., 2019; Ahsan et al., 2020; Cook et al., 2018). Salp Swarm Algorithm (SSA) is one of the population-based metaheuristic algorithms used extensively for various research problems (Mirjalili et al., 2017).

The significant contribution of this study is reported here. The Salp Swarm algorithm (SSA) and Shannon entropy are the foundations of the approach for generating randomized keys proposed in this study. It utilizes the dynamic movement of salps to create high-quality, robust, and randomized keys against attacks. The proposed algorithm employs a transfer function and quantization method to convert a salp into a cryptographic key. For the performance assessment, four transfer functions and three state-of-the-art swarm intelligence metaheuristics, i.e., grey wolf optimization (GWO) (Mirjalili et al., 2014), particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), and BAT (Yang, 2010) algorithms, have been considered. The keys of eight different bit lengths, i.e., 512, 256, 192, 128, 96, 80, 64, were generated and evaluated due to their applications in the different advanced encryption algorithms, i.e., DES, PRESENT, SIMON, SPECK, 3-DES, and AES (Osaba et al., 2021; Dokeroglu et al., 2019).

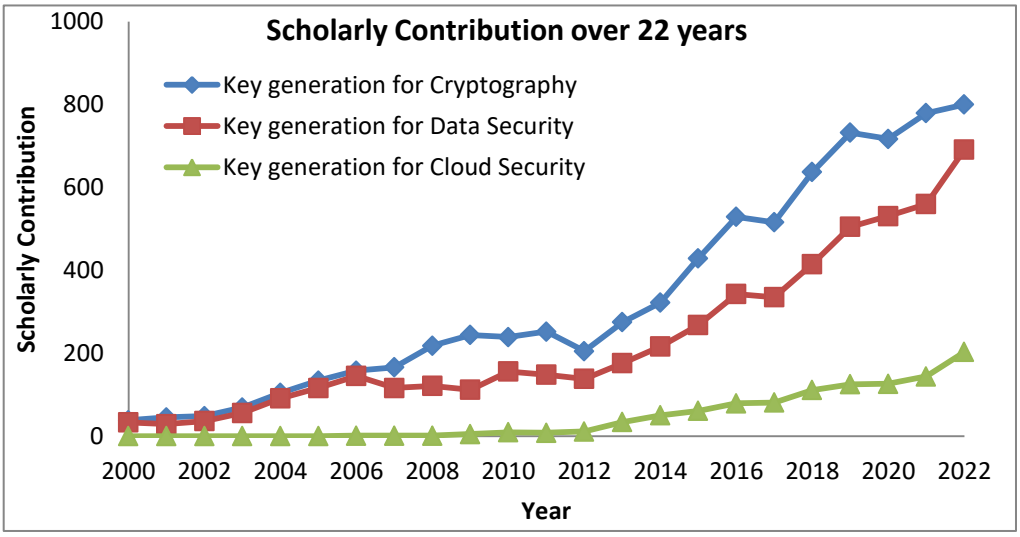


**Figure 1. Growth of scholarly work on the key generation problem from the year 2000 to 2022**

The paper's content is structured as follows: Section 2 outlines the research gap in the literature and summarizes some recent pertinent studies. Section 3 defines the mathematical formulation of the key generation problem, followed by a review of considered metaheuristic algorithms in Section 4. Section 5 has provided an illustrated description of the proposed algorithm. Section 6 discussed the findings of the simulation study based on numerous experiments. The

DOI: https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009

concluding remark and potential impending research recommendations are provided in Section 7.

## Related Work

It summarizes some of the current and pertinent studies and identifies the research gap in the literature. Many research surveys have been published on cloud computing security issues and challenges (Ali et al., 2015; Bhardwaj et al., 2016; Sicari et al., 2022; Khalid et al., 2021). Hashizume et al. (2013) discussed security issues related to cloud computing, which reports that essential services are outsourced to third parties leading to data security issues, service availability, and demonstrating compliance. Moreover, cloud computing's influence on various advanced technologies, i.e., virtualization, service-oriented architecture (SOA), advanced web, Blockchain technology, and others, also inherit that particular technology's security issue.

Tahir et al., 2021 suggested the usage of two levels of encryption in the CryptoGA cryptographic scheme. The first layer uses the shift cipher to encrypt the plaintext into ciphertext. Then, using a discrete genetic algorithm employing Shannon's Entropy as an optimization objective, the keys are extracted from the incoming ciphertext. Finally, an encryption method with randomized crossover and mutation operators is utilized to get the final ciphertext. Authors also claim that CryptoGA provides slower execution time and high throughput demanded by encryption and decryption than well-known traditional algorithms, i.e., DES and RSA cryptosystems.

Jawed and Sajid (2022) extended the CryptoGA block cipher and suggested that 128-bit random keys must be generated to increase Shannon's Entropy. Researchers analyzed XECryptoGA on CryptoGA and found that XECryptoGA generates keys six times faster than CryptoGA. In XECryptoGA and CryptoGA, the Avalanche effect is respectively close to 50% and less than 20%.CryptoGA can be cracked with a brute force attack with just 1456(i.e., 8 x 7 x 26) guesses, whereas XECryptoGA requires $2^{128}$.

Kalsi et al. (2017) presented a novel concept of DNA-based Deep Learning Cryptography involving the usage of DNA sequences and deep learning techniques to obscure data. This approach is designed to enhance data security by encoding DNA sequences and utilizing deep learning algorithms to protect the information from unauthorized access. The Genetic Algorithm with Needleman-Wunsch (NW) algorithm generates keys, which are then employed in the encryption and decryption processes along with numerous biological procedures, such as DNA sequencing, transcription, and translation.

Gupta et al. (2022) suggested protected key generation utilizing improved identity-based encryption for cloud systems to conceal the consumer's identity even if the attacker decodes the keys or data. According to the authors' assertions, the efficient selective-ID secure identity-based encryption solution, a competing strategy, takes longer to encrypt and decrypt data. According to the authors, the suggested method's most important feature is that the Lagrange coefficient hides the user's identity.

Singh and Chatterjee (2017) discussed the features, security issues, threats, and solutions of cloud computing and compared related works. Ali et al. (2015) surveyed the security issues arising from the cloud computing paradigm's shared, virtualized, and public nature. Bhardwaj et al. (2016) compared different symmetric and asymmetric algorithms used in cryptography. It also explains how symmetric algorithms are used in cloud-based applications and services. Thabit et al. (2021) proposed a new lightweight cryptographic method that draws inspiration from the Feistal and Substitution Permutation Architecture, which has been presented to ameliorate the data security of cloud systems. Authors claim that the new lightweight cryptographic algorithm (NLCA) provides high protection and low computational cost compared to frequently used algorithms, namely HIGHT, AES, DES, and Blowfish. Block size, key length, potential key, mathematical operations, cipher type, and security power were the six criteria used to evaluate NLCA.

Moreover, three-layer privacy-preserving (Wang et al., 2018), evolutionary game-based security mechanism (Sun et al., 2018), Session key-based lightweight encryption algorithm (Gupta et al., 2021), Elliptic curve Diffie Hellman cryptosystem (Subramanian and Tamilselvan, 2020), a novel secure fog-based cloud storage (Ahsan et al., 2022), and GA-based data security algorithm (Thabit et al., 2021) for cloud computing or fog computing were developed and tested.

Some algorithms that do not generate keys separately to produce ciphertext have various drawbacks, such as lack of confidentiality, limited security, limited use cases, lack of scalability, and lack of interoperability. As compared to keyless cryptographic algorithms, the key-utilized algorithms provide substantial protection and versatility, making them suitable for a wide range of applications that require data confidentiality and authentication. Moreover, the key generation algorithms have been embedded in the various existing cryptographic algorithms instead of separate lightweight

DOI: https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009

key generation algorithms. The Salp Swarm algorithm (SSA) and Shannon entropy are the foundations of the lightweight key generation approach for generating randomized cryptographic keys proposed in this study. It utilizes the dynamic movement of salps to create high-quality, robust, and randomized keys against attacks.

## Problem Formulation

Shannon developed Shannon's entropy E(X), which calculates the mean amount of "uncertainty" included in the given random sequence, to evaluate the unpredictable nature of any source "X". Shannon's entropy has wide applications in various fields, such as digital communication, data storage, machine learning and artificial intelligence (Tahir et al., 2021; Jawed and Sajid, 2022).

Suppose $K_{SSA}$ is the cryptographic key of length $n$-bit generated using the SSA-based key generation problem. The key generation problem in mathematical form can be given as:

$$Maximize: E(X) = -\sum p(k_i) \log_2 p(k_i) \dots\dots (1)$$

Where,

$$k_i \in \{0,1\} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2)$$

I=0,1,2,…….,n-1……...(3)

where $p(k_i)$ characterizes the probability of the bit $k_i$present in the cryptographic key$K_{SSA}$.

## Metaheuristic Algorithms

Metaheuristic algorithms are advanced optimization techniques that are employed to discover near-optimal solutions to intricate optimization problems. These algorithms operate at a top-level abstraction to explore a diverse range of potential solutions and evaluate them based on certain criteria, thereby enabling them to find approximate solutions to problems that are typically challenging to solve using conventional optimization methods. In contrast to exact optimization methods, metaheuristic algorithms do not guarantee finding the optimal global solution. Instead, they aim to find suitable solutions within a reasonable computation time. Metaheuristics algorithms are often population-based, meaning they operate on a population of candidate solutions that have evolved and improved over time. The population is updated iteratively and improves with each iteration. Metaheuristics have been used in several fields, such as operations research, computer science, engineering, etc., for solving optimization problems, including function optimization, constraint optimization, and combinatorial optimizations (scheduling, routing, network design, image processing, finance, and many more). Metaheuristics are suitable for high-dimensional, multi-modal, and non-linear problems, and they are widely used in many real-world applications (Crawford et al., 2017; Sajid et al., 2022; Sajid and Raza, 2017).

### Salp Swarm Algorithm (SSA)

Salp Swarm Algorithm is a bio-inspired optimization algorithm based on the swimming patterns of salp chains, which are planktonic tunicates in the ocean. The algorithm represents the optimization problem as a population of particles, each representing a candidate solution to the problem. Each particle's position is updated depending on its current location, personal best position, and the population's overall best position. In SSA, the particles move in a coordinated way, similar to how the individuals in a salp chain move. Each particle follows the leader in front of it while at the same time maintaining a distance from the leader behind it. It leads to the formation of a chain-like structure, where the particles move in a coordinated fashion to explore the search space. One of the characteristics of SSA is that the leader particle changes over time, allowing the whole population to examine different regions of the search space. SSA has been proficiently applied to unravel varied problems in engineering, computer science, and finance. It's suitable for optimization problem that has a complex landscape and high dimensionality problem (Mirjalili et al., 2017).

### Particle Swarm Optimization (PSO)

PSO is a metaheuristic optimization algorithm based on the behavior of a swarm of particles. It is used to determine a function's overall best solution. In PSO, each particle represents a candidate solution to the optimization problem. The position of each particle in the search space is updated based on its position, the position of the best solution found so far (called the "personal best" or "pbest"), and the position of the best solution found by the entire population (called the "global best" or "gbest"). The position of each particle is updated based on the positions of the personal best and global best particles, as well as a velocity component that allows the particle to move toward the best solutions. The velocity component is updated based on the current velocity, the cognitive component that guides the particle toward its pbest, and the social component that drives it toward the gbest. The PSO algorithm uses a random initialization of the particles and is sensitive to the initialization values, and different initialization values can lead to another solution. As the number of iterations increases, the particles converge toward the optimal solution, typically located near the gbest. PSO is considered a simple yet powerful optimization algorithm, and it's been used in

DOI: https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009

various areas such as function optimization, feature selection, neural networks, control systems, and many more. It's simple because of the minimal parameter setting required and easy to understand, yet it can find reasonable solutions for complex optimization problems (Kennedy and 1995).

### Grey Wolf Optimization (GWO)

GWO algorithm is a swarm optimization algorithm based on the grey wolves hunting behavior. It is used to find the global optimum of a given function. The algorithm uses a population of "wolves," each representing a candidate solution to the optimization problem. The position of each wolf in the search space is updated based on the current best position, the personal best position (the best position a particular wolf has found so far), and the global best position (the best position located by any wolf in the population). In GWO, leadership hierarchy among the wolves is used, similar to the real-world behavior of grey wolves in a pack. There are three types of wolves: Alpha, Beta, and Delta, each with different responsibilities and roles in the optimization process. The Alpha wolf is responsible for leading the pack and exploring the search space, the Beta wolves are responsible for exploiting the search space, and the Delta wolves are responsible for maintaining the diversity of the population. One of the characteristics of GWO is that it uses a simple random walk mechanism, similar to other metaheuristic optimization algorithms. It allows the algorithm to explore different regions of the search space and avoid getting stuck in local optima. GWO frequently outperforms other metaheuristic optimization algorithms when used for various optimization issues, including function optimization, feature selection, picture compression, and cluster analysis (Mirjalili et al., 2014).

### Bat Algorithm

Xin-She Yang (2010) introduced a swarm intelligence Bat algorithm inspired by the behavior of bats, specifically their echolocation abilities and hunting behavior. The algorithm is a metaheuristic optimization method that can unravel engineering design problems. It involves a population of virtual "bats" that search for the optimal solution to a given problem. There are several possible solutions to the population of bats. It updates its location within the search space based on its observations and those of the other bats in the colony. The algorithm works by adjusting the frequency and loudness of the bats' echolocation calls based on their current position and the position of the best bat in the population. It allows the bats to explore the search space more effectively and converge toward the optimal solution. The Bat Algorithm solves many optimization problems, including engineering design, data mining, and machine learning. It is also relatively simple to implement and can be adapted for various optimization problems (Yang, 2010).

### Salp-Swarm-Based Cryptographic Key Generation Algorithm

As shown in Algorithm 1, Salp Swarm Algorithm (SSA) based Cryptographic key generation algorithm starts with the salps' random population S of size $N$, the dimension of each salp which is the key length $n$. The values of salps are shown in Eq. (4):

$$S = \begin{bmatrix} S_1^1 & S_2^1 & \cdots & S_n^1 \\ S_1^2 & S_2^2 & \cdots & S_n^2 \\ \vdots & \vdots & \cdots & \vdots \\ S_1^N & S_2^N & \cdots & S_n^N \end{bmatrix} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

The cryptographic key is a binary problem, while the SSA is created for a real-valued problem. The transfer function followed by the discretization method is used on the real-valued salps. To compute the Shannon entropy, the transfer function $T(X)$ scales each salp between 0 and 1. The general equation is given as follows (Lanza-Gutierrez et al., 2017 and Beheshti, 2021),

$$T(S_i^j) = s_i^j \quad where, 0 \le s_i^j \le 1 \ldots\ldots\ldots\ldots\ldots\ldots (5)$$

Various transfer functions exist, such as S-type, V-type, and Q-type. It has been proven that V-type transfer functions perform well on many problems. Thus, four V-type transfer functions are considered for this study, and their graphical representation is also shown in Figure 2.

To convert the scaled salp $s_i$ into the discrete binary population $y_i$, the standard binarization technique (Quantization method) is as follows-

$$y_i^j = \begin{cases} 1 & ; if\ rand(0,1) \le s_i^j \\ 0 & ; else \end{cases} \ldots\ldots\ldots\ldots\ldots (6)$$

Where $rand(0,1)$ denotes a chance number between 0 and 1.

The converted population Y is given as follows-

$$Y = \begin{bmatrix} y_1^1 & y_2^1 & \cdots & y_n^1 \\ y_1^2 & y_2^2 & & y_n^2 \\ \vdots & \vdots & \cdots & \vdots \\ y_1^N & y_2^N & \cdots & y_n^N \end{bmatrix} where\ y_i^j \in \{0,1\}\ \forall i,j \ldots\ldots (7)$$

After converting the real-valued salp into a discrete binary salp, Shannon entropy E(X) determines each salp

DOI: https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009

fitness value using equation (1). Figure 3 shows the conversion process from a salp to a cryptographic key.

**Algorithm 1: SSA-Based Key Generation Algorithm**

**Input:** Population Swith size N, dimension n, upper bound ub, lower bound lb

**Output:** Key ($K_{SSA}$) with maximum Shannon entropy

**Begin:**

1. Initialize the salp $x_i (i = 1,2,3, \dots, N)$ with population S, dimension n, upper bound *ub*, and lower bound *lb*;
2. While the end condition is not satisfied, do
3. Calculate the fitness of each search agent (salp) using the Transfer function (Eq. 5), Quantization method ( Eq. 6), and Shannon's Entropy (Eq. (1));
4. F = the best search agent with maximum Shannon entropy;
5. Do for every salp $x_i$:

If salp $x_i$ is leader salp:

Update salp $x_i$'s position using Eq. (8);

else

Update salp $x_i$'s position using Eq. (10);

6. end
7. end do loop
8. Adjust all salps dimension values based on the lower and upper bounds;
9. End while
10. return F as Key ($K_{SSA}$) and Shannon entropy

**End:**

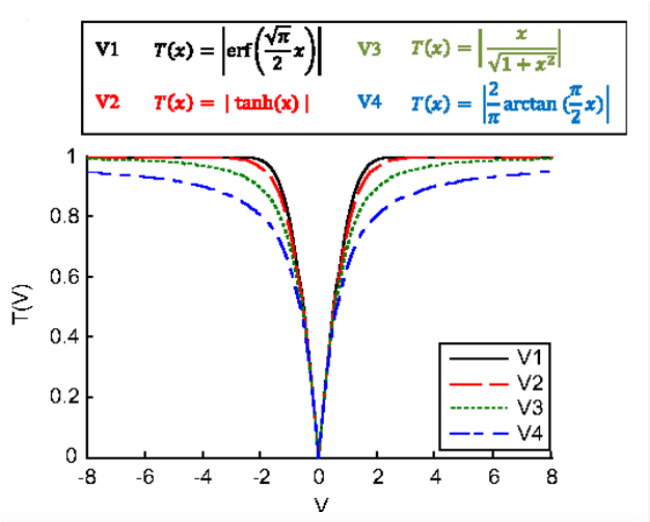


**Figure 2. V-Type Transfer Functions**

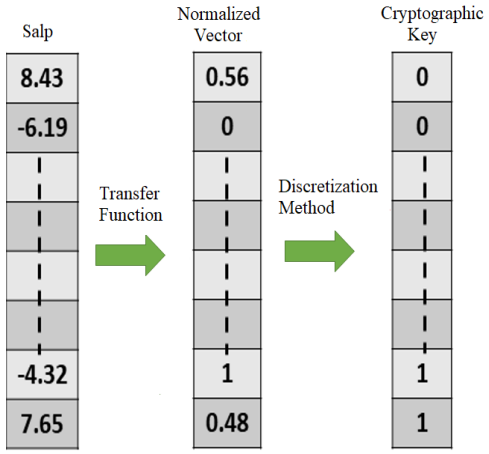


**Figure 3. Salp to cryptographic key conversion**

According to step 6 of Algorithm 1, the position of salps is updated to generate the new population. The position of the leader salp is calculated by equation (8):

$$p_j^i = \begin{cases} S_j + c_1\left((ub_j - lb_j)c_2 + lb_j\right) c_3 \geq 0.5 \\ S_j - c_1\left((ub_j - lb_j)c_2 + lb_j\right) c_3 < 0.5 \end{cases} \dots\dots\dots (8)$$

The new position of the leader is shown by $p_j^i$ and food source's position vector in the $j^{th}$ dimension is shown by $S_j$, the upper and lower bounds of the $j^{th}$ dimension is shown by $ub_j$and $lb_j$, respectively, $c_2$ and $c_3$ are random values inside [0, 1], in Eq(9), $c_1$ is the algorithm's primary parameter.

$$c_1 = 2e^{-\left(\frac{4t}{T_{max}}\right)^2} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (9)$$

$T_{max}$ displays the maximum number of iterations, whereas t represents the current iteration. This value reduces as the iteration count rises. As a result, it can draw attention to the tendency towards diversification in the early stages of optimization and the tendency towards intensification in the latter stages.

The followers' location is modified by Eq…… (10):

$$p_j^i = \frac{p_j^i + p_j^{i-1}}{2} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (10)$$

Where $i \geq 2$ and $p_j^i$ is the location of the $i^{th}$ follower salp at the $j^{th}$dimension.

DOI: https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009

The above processes keep on repeating until the maximum iteration is reached. The fittest salp of the final population is considered as the cryptographic key. The cryptographic key and its Shannon entropy are returned.

## Simulation Results

This section depicts and discusses the system parameters, obtained results in the form of entropy values and execution times of algorithms, and the study's observations.

### System Parameters

The system used for obtaining the simulation results is Intel Core i5-8250U, 12 GB RAM, Windows 10 OS, and Python v3.11.0. The four algorithms, SSA, BAT, GWO, and PSO, have been modified for the cryptographic key generation problem for various sizes of the key length. Table 1 depicts the parameters and their values corresponding to four different meta-heuristics. The keys of eight different bit lengths, i.e., 512, 256, 192, 128, 96, 80, 64, were generated and evaluated due to their applications in the different cryptographic algorithms, i.e., AES, DES, PRESENT, SIMON, SPECK, and 3DES. The simulation tests were run 20 times for each cryptographic key length, and mean and standard deviation values for entropy and execution durations were recorded and presented (Krishna et al., 2018).

**Table 1. System Parameters**

| Algorithm | Parameter | Value |
|---|---|---|
| Standard Parameters for GWO, BAT, PSO, and SSA | Population Size | 50 |
| | Maximum Iteration | 100 |
| | Dimension | Key Length |
| | Lower Bound (lb) | -100 |
| | Upper Bound (lb) | +100 |
| | Objective Function | Shannon Entropy |
| GWO | Alpha (Initial Score) | -inf |
| | Beta (Initial Score) | -inf |
| | Delta (Initial Score) | -inf |
| | Convergence constant | Linearly decreases from 2 to 0 |
| BAT | Loudness | 0.5 |
| | Pulse Rate | 0.5 |
| | Minimum Frequency | 0 |
| | Maximum Frequency | 2 |
| PSO | Inertia | [0.2, 0.9] |
| | Cognition of particle ($c_1$) | 2 |
| | Social influence of swarm ($c_2$) | 2 |
| SSA | Initial Speed | 0 |
| | Uniformly Generated Random Numbers ($c_2$& $c_3$) | (0, 1) |

### Results Obtained

Table 2 consists of the mean and standard deviation of entropy produced by GWO, PSO, BAT, and SSA algorithms for key sizes (in bits) 512, 256, 192, 128, 96, 80, and 64 corresponding to four transfer functions, i.e., $V_1$, $V_2$, $V_3$, and $V_4$. Table 2 illustrates that the SSA-based cryptographic key generation algorithm offers the best mean values of entropy corresponding to the transfer function $V_1$ followed by PSO-, GWO- and BAT-based cryptographic key generation algorithms. However, the SSA-based cryptographic key generation algorithm offers the second-best standard deviation values of entropy corresponding to the transfer function $V_1$ followed by GWO- and BAT-based cryptographic key generation algorithms. The PSO-based cryptographic key generation algorithm offers the best standard deviation values of entropy corresponding to the transfer function $V_1$. For transfer functions $V_2$, $V_3$, and $V_4$, the best mean entropy values were offered by the SSA-based cryptographic key generation algorithm corresponding to the transfer function $V_1$ followed by PSO-, GWO-, and BAT-based cryptographic key generation algorithms. Regarding standard deviation entropy values, the PSO-based cryptographic key generation algorithm offers the best values, followed by SSA-, GWO-, and BAT-based algorithms.

Figure 4 depicts the different convergence curves offered by SSA-, PSO-, GWO- and BAT-based cryptographic key generation algorithms corresponding to 4-transfer functions, i.e., $V_1$, $V_2$, $V_3$, and $V_4$. As can be seen from Figure 4, the SSA-based cryptographic key generation algorithm offers the best entropy values compared to PSO, GWO, and BAT algorithms. Generally, the performance order from best to worst is SSA, PSO, GWO, and BAT algorithms. However, GWO and BAT algorithms are competitive for different instances, as seen in Figure 4.

Table 3 displays the mean and standard deviation of execution time consumed over 100 iterations by GWO, PSO, BAT, and SSA algorithms for key sizes (bits)512, 256, 192, 128, 96, 80, and 64 corresponding to four transfer functions, i.e., $V_1$, $V_2$, $V_3$, and $V_4$. The SSA and BAT-based cryptographic key generation algorithms consume minimum and almost equal time corresponding to the transfer function $V_1$, $V_2$, $V_3$, and $V_4$, followed by PSO-and GWO-based cryptographic key generation algorithms.

DOI: https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009

**Table 2. Entropy Values for 4-Algorithms Corresponding to Four Transfer Functions**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Entropy Values for Transfer Function $V_1$** | | | | | | | |
| Key Size (Bits) / Algorithms | | 64 | 80 | 96 | 128 | 192 | 256 | 512 |
| GWO | Mean | 0.393 | 0.396 | 0.335 | 0.295 | 0.276 | 0.255 | 0.225 |
| | Stdev | 0.057 | 0.046 | 0.042 | 0.033 | 0.033 | 0.043 | 0.059 |
| PSO | Mean | 0.541 | 0.509 | 0.411 | 0.419 | 0.369 | 0.378 | 0.306 |
| | Stdev | 0.047 | 0.088 | 0.055 | 0.064 | 0.073 | 0.044 | 0.023 |
| BAT | Mean | 0.286 | 0.270 | 0.313 | 0.265 | 0.220 | 0.218 | 0.193 |
| | Stdev | 0.091 | 0.077 | 0.097 | 0.072 | 0.025 | 0.030 | 0.025 |
| SSA | Mean | 0.555 | 0.563 | 0.519 | 0.481 | 0.410 | 0.379 | 0.328 |
| | Stdev | 0.084 | 0.074 | 0.065 | 0.054 | 0.047 | 0.033 | 0.027 |
| | **Entropy Values for Transfer Function $V_2$** | | | | | | | |
| Key Size (Bits) / Algorithms | | 64 | 80 | 96 | 128 | 192 | 256 | 512 |
| GWO | Mean | 0.458 | 0.414 | 0.416 | 0.326 | 0.296 | 0.302 | 0.278 |
| | Stdev | 0.087 | 0.077 | 0.078 | 0.039 | 0.033 | 0.067 | 0.101 |
| PSO | Mean | 0.579 | 0.542 | 0.501 | 0.470 | 0.440 | 0.3947 | 0.329 |
| | Stdev | 0.054 | 0.073 | 0.087 | 0.033 | 0.051 | 0.033 | 0.041 |
| BAT | Mean | 0.353 | 0.293 | 0.307 | 0.296 | 0.258 | 0.237 | 0.200 |
| | Stdev | 0.105 | 0.069 | 0.0769 | 0.058 | 0.033 | 0.034 | 0.033 |
| SSA | Mean | 0.593 | 0.586 | 0.561 | 0.496 | 0.478 | 0.417 | 0.380 |
| | Stdev | 0.071 | 0.0427 | 0.055 | 0.045 | 0.047 | 0.037 | 0.030 |
| | **Entropy Values for Transfer Function $V_3$** | | | | | | | |
| Key Size (Bits) / Algorithms | | 64 | 80 | 96 | 128 | 192 | 256 | 512 |
| GWO | Mean | 0.468 | 0.442 | 0.475 | 0.389 | 0.340 | 0.333 | 0.356 |
| | Stdev | 0.035 | 0.050 | 0.064 | 0.061 | 0.041 | 0.054 | 0.083 |
| PSO | Mean | 0.605 | 0.593 | 0.519 | 0.508 | 0.456 | 0.457 | 0.384 |
| | Stdev | 0.074 | 0.068 | 0.086 | 0.089 | 0.055 | 0.022 | 0.051 |
| BAT | Mean | 0.435 | 0.391 | 0.358 | 0.390 | 0.321 | 0.310 | 0.271 |
| | Stdev | 0.063 | 0.092 | 0.061 | 0.050 | 0.037 | 0.028 | 0.032 |
| SSA | Mean | 0.649 | 0.647 | 0.630 | 0.582 | 0.487 | 0.509 | 0.415 |
| | Stdev | 0.072 | 0.044 | 0.049 | 0.044 | 0.035 | 0.038 | 0.021 |
| | **Entropy Values for Transfer Function $V_4$** | | | | | | | |
| Key Size (Bits) / Algorithms | | 64 | 80 | 96 | 128 | 192 | 256 | 512 |
| GWO | Mean | 0.588 | 0.560 | 0.547 | 0.509 | 0.482 | 0.458 | 0.474 |
| | Stdev | 0.039 | 0.029 | 0.053 | 0.053 | 0.032 | 0.060 | 0.086 |
| PSO | Mean | 0.718 | 0.691 | 0.650 | 0.629 | 0.582 | 0.556 | 0.512 |
| | Stdev | 0.056 | 0.050 | 0.046 | 0.047 | 0.026 | 0.042 | 0.025 |
| BAT | Mean | 0.544 | 0.559 | 0.530 | 0.493 | 0.466 | 0.433 | 0.406 |
| | Stdev | 0.067 | 0.078 | 0.072 | 0.056 | 0.058 | 0.060 | 0.037 |
| SSA | Mean | 0.774 | 0.730 | 0.717 | 0.672 | 0.644 | 0.627 | 0.566 |
| | Stdev | 0.047 | 0.057 | 0.042 | 0.041 | 0.034 | 0.027 | 0.023 |

DOI: https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009

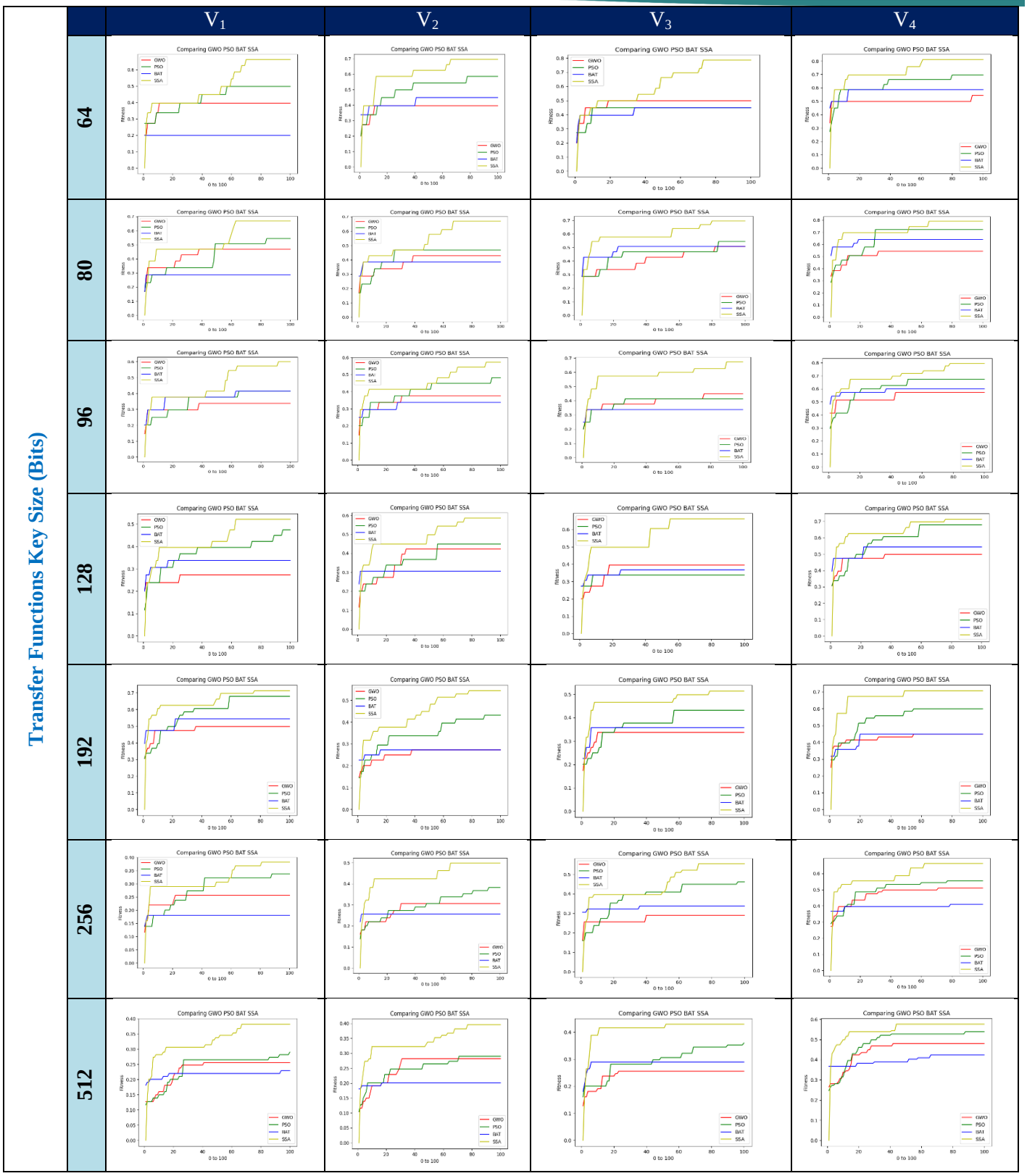


**Figure 4. Convergence Curves for GWO, PSO, BAT, and SSA corresponding to 4-Transfer Functions**

DOI: https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009

**Table 3. Execution Times for 4-Algorithms Corresponding to 4-Transfer Functions**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Execution Times(seconds) for Transfer Function $V_1$** | | | | | | | |
| Key Size (Bits) / Algorithms | | 64 | 80 | 96 | 128 | 192 | 256 | 512 |
| GWO | Mean | 4.781 | 5.946 | 7.908 | 9.906 | 14.248 | 18.896 | 44.681 |
| | Stdev | 0.052 | 0.036 | 1.94 | 0.553 | 0.089 | 0.118 | 16.736 |
| PSO | Mean | 4.561 | 5.634 | 7.790 | 9.180 | 13.704 | 18.012 | 49.348 |
| | Stdev | 0.089 | 0.026 | 3.087 | 0.104 | 0.502 | 0.102 | 25.684 |
| BAT | Mean | 3.859 | 4.721 | 6.414 | 7.749 | 11.292 | 14.989 | 39.401 |
| | Stdev | 0.073 | 0.029 | 2.313 | 0.212 | 0.088 | 0.115 | 18.432 |
| SSA | Mean | 3.827 | 4.730 | 6.441 | 7.598 | 11.204 | 15.158 | 41.061 |
| | Stdev | 0.142 | 0.238 | 2.456 | 1.922 | 0.069 | 0.216 | 18.249 |
| | **Execution Times (seconds) for Transfer Function $V_2$** | | | | | | | |
| Key Size (Bits) / Algorithms | | 64 | 80 | 96 | 128 | 192 | 256 | 512 |
| GWO | Mean | 5.911 | 5.987 | 7.379 | 9.460 | 14.212 | 18.771 | 37.442 |
| | Stdev | 2.154 | 0.077 | 0.161 | 0.049 | 0.100 | 0.102 | 0.376 |
| PSO | Mean | 6.163 | 5.667 | 7.575 | 8.956 | 13.567 | 17.951 | 35.719 |
| | Stdev | 2.930 | 0.060 | 1.964 | 0.065 | 0.0792 | 0.129 | 0.217 |
| BAT | Mean | 5.157 | 4.806 | 5.813 | 7.432 | 11.309 | 14.943 | 29.704 |
| | Stdev | 2.336 | 0.090 | 0.044 | 0.052 | 0.075 | 0.100 | 0.190 |
| SSA | Mean | 5.236 | 4.930 | 5.856 | 7.520 | 11.388 | 15.558 | 29.987 |
| | Stdev | 2.446 | 0.431 | 0.0322 | 0.050 | 0.069 | 1.330 | 0.219 |
| | **Execution Times (seconds) for Transfer Function $V_3$** | | | | | | | |
| Key Size (Bits) / Algorithms | | 64 | 80 | 96 | 128 | 192 | 256 | 512 |
| GWO | Mean | 4.820 | 5.962 | 8.141 | 9.706 | 13.935 | 19.481 | 43.732 |
| | Stdev | 0.033 | 0.040 | 2.046 | 0.969 | 0.093 | 0.293 | 9.566 |
| PSO | Mean | 4.567 | 5.644 | 9.136 | 8.923 | 13.335 | 18.431 | 42.595 |
| | Stdev | 0.067 | 0.037 | 3.791 | 0.0461 | 0.105 | 0.232 | 10.538 |
| BAT | Mean | 3.855 | 4.776 | 8.258 | 7.483 | 11.032 | 15.758 | 32.978 |
| | Stdev | 0.026 | 0.033 | 3.969 | 0.048 | 0.071 | 0.347 | 4.151 |
| SSA | Mean | 3.843 | 4.787 | 8.297 | 7.503 | 11.264 | 15.733 | 33.229 |
| | Stdev | 0.027 | 0.059 | 3.999 | 0.038 | 0.379 | 0.241 | 4.082 |
| | **Execution Times (seconds) for Transfer Function $V_4$** | | | | | | | |
| Key Size (Bits) / Algorithms | | 64 | 80 | 96 | 128 | 192 | 256 | 512 |
| GWO | Mean | 4.817 | 5.960 | 7.180 | 9.409 | 14.403 | 19.955 | 37.307 |
| | Stdev | 0.078 | 0.050 | 0.049 | 0.083 | 0.290 | 0.161 | 0.336 |
| PSO | Mean | 4.549 | 5.632 | 7.168 | 9.088 | 13.639 | 19.153 | 35.541 |
| | Stdev | 0.021 | 0.041 | 1.118 | 0.598 | 0.086 | 0.197 | 0.222 |
| BAT | Mean | 3.840 | 4.747 | 5.719 | 7.405 | 11.367 | 15.991 | 29.669 |
| | Stdev | 0.038 | 0.041 | 0.041 | 0.051 | 0.080 | 0.133 | 0.280 |
| SSA | Mean | 3.871 | 4.745 | 5.736 | 7.491 | 11.450 | 16.140 | 29.978 |
| | Stdev | 0.113 | 0.048 | 0.047 | 0.077 | 0.081 | 0.138 | 0.294 |

DOI: https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009

## Observations

As observed from the results, the SSA-based cryptographic key generation algorithm offers the best mean values of entropy corresponding to the transfer functions compared to PSO, GWO, and BAT algorithms. The performance order for the standard deviation entropy values is PSO, SSA, GWO, and BAT-based cryptographic key generation algorithms. SSA leads can also be observed compared to GWO, PSO, and BAT for the key size (bits) of 512, 256, 192, 128, 96, 80, and 64.GWO and BAT algorithms are competitive for different instances and PSO remains the second best to offer the best entropy values. For 100 iterations, the performance order from best to worst for execution time consumed by the four algorithms is SSA, BAT, PSO, and GWO -based cryptographic key generation algorithms. From entropy values and execution times, it can be established that the SSA-based cryptographic key generation algorithm performs better compared to GWO, PSO, and BAT algorithms. Due to low computational requirements, the SSA-based cryptographic key generation algorithm can be utilized for different encryption algorithms, i.e., AES, DES, PRESENT, SIMON, SPECK, and 3DES. Moreover, the transfer function $V_1$ and $V_4$ with SSA-based key generation algorithm performs well for small and large size keys, respectively.

## Conclusion

Cloud computing, IoT, and Fog computing are relatively young research areas offering quick and effective online services on a pay-as-you-go-payment model. Data security, in transit or storage, has always been a fundamental obstacle to adopting cloud computing, and it demands strong cryptographic encryption and key generation algorithms. The Salp Swarm algorithm (SSA) and Shannon entropy are the foundations of the approach for generating randomized cryptographic keys proposed in this study. The proposed Cryptographic Key Generation algorithm has been evaluated on four transfer functions against three state-of-the-art swarm intelligence metaheuristics, i.e., grey wolf optimization (GWO), particle swarm optimization (PSO) and BAT algorithms. The simulation study observed that SSA performs better in terms of entropy value and execution times than the GWO, PSO, and BAST algorithms for the keys of eight different bit lengths, i.e., 512, 256, 192, 128, 96, 80, and 64. Due to low execution time requirement, the SSA-based Cryptographic Key Generation algorithm can be used for various encryption algorithms, i.e., AES, DES, PRESENT, SIMON, SPECK, and 3DES in cloud computing and IoT domain. For future research directions, S-type, Q-type, and other transfer functions can also generate cryptographic keys. The Algebraic and similarity-based meta-heuristics algorithms can also be utilized for the cryptographic key generation problem.

## Conflict of interest

None

## References

Ahsan, M. M., Gupta, K. D., Nag, A. K., Poudyal, S., Kouzani, A. Z., & Mahmud, M. A. P. (2020). Applications and evaluations of bio-inspired approaches in cloud security: A review. *IEEE Access*, *8*, 180799-180814. https://doi.org/10.1109/ACCESS.2020.3027841

Alouffi, B., Hasnain, M., Alharbi, A., Alosaimi, W., Alyami, H., & Ayaz, M. (2021). A systematic literature review on cloud computing security: threats and mitigation strategies. *IEEE Access*, *9*, 57792-57807. https://doi.org/10.1109/ACCESS.2021.3073203.

Ali, M., Khan, S. U., & Vasilakos, A. V. (2015). Security in cloud computing: Opportunities and challenges. *Information Sciences*, *305*, 357–383. https://doi.org/10.1016/j.ins.2015.01.025.

Beheshti, Z. (2021). UTF: Upgrade transfer function for binary meta-heuristic algorithms. *Applied Soft Computing*, *106*, 107346. https://doi.org/10.1016/j.asoc.2021.107346

Bhardwaj, A., Subrahmanyam, G., Avasthi, V., & Sastry, H. (2016). Security Algorithms for Cloud Computing. *Procedia Computer Science*, *85*, 535–542. https://doi.org/10.1016/j.procs.2016.05.215.

Cook, A., Robinson, M., Ferrag, M., Leandros, H., & Ying, J. (2018). Internet of Cloud: Security and Privacy Issues. *IEEE Internet Computing*, *15*(1), 73-76.

Crawford, B., Soto, R., Astorga, G., García, J., Castro, C., & Paredes, F. (2017). Putting continuous metaheuristics to work in binary search spaces. *Complexity*, 8404231.

Del Ser, J., Osaba, E., Dondo, J., López-Guede, J. M., & Molina, D. (2018). Bio-Inspired Computation: Where We Stand and What's Next. *Complexity*, *2018*, 1-16.

https://doi.org/10.1155/2018/9343095.

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., & Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, *137*, 106040.

Gartner. (2021). Forecast: IT Services for IoT, Worldwide, 2019-2025. [Online]. Available: https://www.gartner.com/en/documents/4004741

Gupta, M., Gupta, K. K., & Shukla, P. K. (2021). Session key-based fast, secure, and lightweight image encryption algorithm. *Multimedia Tools and Applications*, *80*(7), 10391–10416. https://doi.org/10.1007/s11042-020-10116-z.

Gupta, R. K., Almuzaini, K. K., Pateriya, R. K., Shah, K., Shukla, P. K., & Akwafo, R. (2022). An improved secure key generation using enhanced identity-based encryption for cloud computing in large-scale 5G. *Wireless Communications and Mobile Computing*, *2022*, 7291250. https://doi.org/10.1155/2022/7291250.

Hashizume, K., Rosado, D. G., Fernández-Medina, E., & Fernández, E. B. (2013). An analysis of security issues for cloud computing. *Journal of Internet Services and Applications*, *4*, 5.

Jawed, M. S., & Sajid, M. (2022). A comprehensive survey on cloud computing: architecture, tools, technologies, and open issues. *International Journal of Cloud Applications and Computing*, *12*(1), 1–33. https://doi.org/10.4018/IJCAC.308277.

Jawed, M.S., & Sajid, M. (2022). XECryptoGA: A Metaheuristic algorithm-based Block Cipher to Enhance the Security Goals. Evolutionary Systems. https://doi.org/10.1007/s12530-022-09462-0.

Kalsi, S., Kaur, H., & Chang, V. (2017). DNA cryptography and deep learning using genetic algorithm with NW algorithm for key generation. *Journal of Medical Systems*, *42*(1). https://doi.org/10.1007/s10916-017-0851-z.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*, *IV*, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, *5*, 4104–4108.

Khalid, T., Abbasi, M.A.K., Zuraiz, M., Khan, A.N., Ali, M., Ahmad, R.W., Rodrigues, J.J.P.C., Aslam, M. (2021). A survey on privacy and access control schemes in fog computing. *International Journal of Communication Systems*, *34*(2), e4181. https://doi.org/10.1002/dac.4181.

Krishna, G. J., Ravi, V., & Bhattu, S. N. (2018). Key generation for plain text in stream cipher via bi-objective evolutionary computing. *Applied Soft Computing*, *70*, 301–317. https://doi.org/10.1016/j.asoc.2018.05.025.

Lanza-Gutierrez, J. M., Crawford, B., Soto, R., Berrios, N., Gomez-Pulido, J. A., & Paredes, F. (2017). Analyzing the effects of binarization techniques when solving the set covering problem through swarm optimization. *Expert Systems with Applications*, *70*, 67–82. https://doi.org/10.1016/j.eswa.2016.10.054

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, *114*, 163-191. https://doi.org/10.1016/j.advengsoft.2017.07.002

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, *69*, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Osaba, E., Villar-Rodriguez, E., Del Ser, J., Nebro, A. J., Molina, D., LaTorre, A., Suganthan, P. N., Coello-Coello, C. A., & Herrera, F. (2021). A tutorial on the design, experimentation, and application of metaheuristic algorithms to real-world optimization problems. *Swarm and Evolutionary Computation*, *64*, 100888. https://doi.org/10.1016/j.swevo.2021.100888

Sicari, A. S., Rizzardi, A., & Coen-Porisini, A. (2022). Insights into security and privacy towards fog computing evolution. *Computers*

*& Security*, *120*, 102822. https://doi.org/10.1016/j.cose.2022.102822.

Singh, A., & Chatterjee, K. (2017). Cloud security issues and challenges: A survey. *Journal of Network and Computer Applications*, *79*, 88–115. https://doi.org/10.1016/j.jnca.2016.11.027.

Subramanian, E. K., & Tamilselvan, L. (2020). Elliptic curve Diffie-Hellman cryptosystem in big data cloud security. *Cluster Computing*, pp.1–11. https://doi.org/10.1007/s10586-020-03069-3

Sun, Y., Lin, F., & Zhang, N. (2018). A security mechanism based on evolutionary game in fog computing. Saudi *Journal of Biological Sciences*, *25*(2), 237–241. https://doi.org/10.1016/j.sjbs.2017.09.010.

Tabrizchi, H., & Kuchaki Rafsanjani, M. (2020). A survey on security challenges in cloud computing: issues, threats, and solutions. *Journal of Supercomputing*, *76*, 9493–9532. https://doi.org/10.1007/s11227-020-03213-1

Tahir, M., Sardaraz, M., Mehmood, Z., & Muhammad, S. (2021). CryptoGA: a cryptosystem based on genetic algorithm for cloud data security. *Cluster Computing*, *24*(2), 739–752. https://doi.org/10.1007/s10586-020-03157-4.

Thabit, F., Alhomdy, S., & Jagtap, S. (2021). A new data security algorithm for cloud computing based on genetics techniques and logical-mathematical functions. *International Journal of Intelligent Networks*, *2*, 18–33. https://doi.org/10.1016/j.ijin.2021.03.001

Thabit, F., Alhomdy, S., Al-ahdal A.H.A., Jagtap, S. (2021), A new lightweight cryptographic algorithm for enhancing data security in cloud computing. *Global Transitions Proceedings*, 2(1), 91–99. https://doi.org/10.1016/j.gltp.2021.01.013.

Wang, T., Zhou, J., Chen, X., Wang, G., Liu, A., & Liu, Y. (2018). A three-layer privacy-preserving cloud storage scheme based on computational intelligence in fog computing. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*(1), 3–12. https://doi.org/10.1109/TETCI.2017.2764109.

Yang, X. S. (2010). A New Metaheuristic Bat-Inspired Algorithm. In Nature Inspired Cooperative Strategies for Optimization (NISCO 2010). *Studies in Computational Intelligence*, *284*, 65–74.

**How to cite this Article:**

Waseem Kaleem, Mohammad Sajid and Ranjit Rajak (2023). Salp Swarm Algorithm to solve Cryptographic Key Generation problem for Cloud computing. *International Journal of Experimental Research and Review*, 31, 85-97.

**DOI** : https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.009