International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Hybrid Encryption based Access Control Approach for Securing Cloud Computing

K. Shankar[1], M. Subhashini[2]
[1]Research Scholar in Department of Computer Science, Srimad Andavan Arts and Science College (Autonomous) (Affiliated to Bharathidasan University), Tiruchirappalli, Tamil Nadu, India.
[2]Assistant Professor in Department of Computer Science, Srimad Andavan Arts and Science College (Autonomous) (Affiliated to Bharathidasan University), Tiruchirappalli, Tamil Nadu, India.

Abstract

Cloud computing has become an integral part of modern technological infrastructure, facilitating the storage and processing of vast amounts of data. However, ensuring the security of sensitive information in the cloud remains a persistent challenge. This paper proposes a novel approach to enhance the security of cloud computing through hybrid encryption, leveraging the Whale Optimization Algorithm (WOA) and Cuckoo Search Optimization (CSO) algorithms. Hybrid encryption, combining symmetric and asymmetric cryptographic techniques, is employed to address the limitations of traditional encryption methods in cloud environments. The Whale Optimization Algorithm and Cuckoo Search Optimization are utilized to optimize key generation and management processes, enhancing the overall efficiency and security of the encryption scheme. The Whale Optimization Algorithm, inspired by the social behavior of humpback whales, is employed to optimize the parameters of the encryption algorithm. WOA's exploration and exploitation capabilities are leveraged to find an optimal balance in the encryption process, improving the overall robustness against potential attacks. Complementing WOA, the Cuckoo Search Optimization algorithm is applied to optimize the key distribution and update mechanisms. Modeled after the brood parasitism behavior of cuckoo birds, CSO excels in searching large solution spaces, making it suitable for refining the distribution of encryption keys and ensuring their constant adaptability to dynamic cloud environments.

Keywords: Cloud Computing, Optimization algorithms, Access Control, Encryption, Decryption, Key Management.

1. INTRODUCTION

In the era of digital transformation, where organizations increasingly rely on cloud computing for the storage, processing, and accessibility of data, the paramount concern is the security of sensitive information residing in the cloud. The dynamic and shared nature of cloud environments introduces unique challenges, making data security a top priority. Encryption stands as a fundamental pillar in fortifying the confidentiality and integrity of data, providing a robust defense against unauthorized access and cyber threats [1] [2].

Cloud computing offers unprecedented advantages such a s scalability, flexibility, and cost efficiency, revolutionizing the way businesses operate and individuals access services. However, this paradigm shift also brings forth security considerations, ranging from data breaches to insider threats [3] [4]. Encryption techniques serve as a crucial line of defense, ensuring that even if unauthorized entities gain access to cloud-stored data, the information remains indecipherable without the appropriate decryption keys.

The essence of encryption lies in the transformation of plaintext data into a ciphertext format using algorithms and cryptographic keys. This process renders the data unreadable to anyone lacking the corresponding decryption keys, thereby safeguarding it from unauthorized eyes. In the context of cloud computing, encryption plays a pivotal role in addressing data security concerns at rest, in transit, and during processing [5] [6].

Securing the cloud through encryption involves the application of various encryption techniques tailored to different facets of cloud data management. At rest, data encryption ensures that information stored in databases or object storage remains protected from unauthorized access [7] [8]. During transit, encryption protocols safeguard data as it travels between the user and the cloud server, mitigating the risks associated with interception and eavesdropping. Additionally, encryption techniques can be applied to specific data processing operations within the cloud environment, enhancing the overall security posture [9] [10] [16] [17] [18].

2. WHALE OPTIMIZATION ALGORITHM

The Whale Optimization Algorithm (WOA) is a nature-inspired optimization algorithm based on the social behavior of humpback whales. Introduced by Seyedali Mirjalili and Andrew Lewis in 2016, WOA aims to simulate the hunting behavior of humpback whales in finding prey [11]. This metaheuristic algorithm falls under the category of swarm intelligence, drawing inspiration from the natural world to solve optimization problems [15]. Key Components of the Whale Optimization Algorithm:

Initialization of the Population: A population of potential solutions, referred to as "whales," is randomly generated to represent the possible solutions to the optimization problem [19].

Objective Function: The optimization problem is defined by an objective function that needs to be minimized or maximized. This function evaluates the fitness of each whale in the population.

Encircling Prey Behavior: The primary inspiration for WOA comes from the encircling behavior of humpback whales when hunting for prey. Whales collaborate to create a spiral or circular motion to encircle and catch prey. In the algorithm, this behavior is simulated to converge the population toward an optimal solution. The position of a whale is updated based on its current position, the position of the best solution encountered so far (global best), and the position of a randomly chosen whale in the population.

Bubble-net Feeding Behavior: Another behavior inspired by humpback whales is the bubble-net feeding technique. Whales create a net of bubbles to trap and concentrate prey. In WOA, this behavior is emulated to explore the search space efficiently. Exploration is encouraged by updating the position of a whale towards a randomly selected whale's position.

Movement Towards Prey: To balance exploration and exploitation, a linear decreasing function is employed to gradually decrease the exploration and exploitation rates. The position of a whale is updated using the formula:

$$X_{new} = X_{rand} - A.D \qquad (1)$$

Where X_{rand} is a randomly selected whale's position, A is a linearly decreasing parameter, and D is the distance between the current whale and the randomly selected whale.

Updating the Parameters: The parameters A and C are updated iteratively during the optimization process, allowing the algorithm to dynamically adjust its exploration and exploitation strategies.

Step by Step Procedure for WOA

Step 1: Initialization

Step 1.1: Initialization of the parameters: Define population size N, maximum iterations *MaxIter,* and linearly decreasing parameters a and b.

Step 1.2: Initialize Whale Positions: Randomly initialize the positions of *N* whales in the search space.

Step 2: Fitness Evaluation: Evaluate the fitness of each whale in the population using the objective function $f(x)$.

Step 3: Repeat until reaching the maximum iterations or a convergence control.

Step 4: Encircling Prey Behavior

Step 4.1: Update Whales' Positions: Encircling behavior simulates the movement of whales in encircling prey.

Step 4.2: for each whale i:

Calculate A and C (linearly decreasing parameters):

$$A = 2.a.|r_1 - 0.5| \qquad (2)$$

$C = 2.r_2$, where r_1 and r_2 are random numbers in [0,1].

Update position based on the best whale's position (P_{best}) and a randomly selected whale (X_{rand}):

$$D = |P_{Best} - X_i| \qquad (3)$$

$$X_{new} = P_{Best} - A.D \qquad (4)$$

Step 5: Bubble-net Feeding Behavior

Step 5.1: Explore New Solutions: Explore the search space by updating whale positions.

For each whale i:

Update the position towards a randomly selected whale's position (X_{rand})

$$X_{new} = X_{rand} - C.D \qquad (5)$$

Step 6: Convergence Control: Gradually decrease the linearly decreasing parameters

$$a = a.\frac{a}{MaxIter}.iter \qquad (6)$$

$$b = 0.5.\left(1 - \frac{iter}{MaxIter}\right) (7)$$

Where $iter$ is the current iteration. In these equations, r_1 and r_2 are random numbers in the range [0, 1], P_{Best} is the best whale's position, X_i is the current whale's position, X_{rand} is a randomly selected whale's position, and iter is the current iteration number.

3. CUCKOO SEARCH OPTIMIZATION

Cuckoo Search Optimization (CSO) is a nature-inspired optimization algorithm introduced by Xin-She Yang and Suash Deb in 2009. The algorithm is inspired by the reproductive strategy of cuckoo birds, which lay their eggs in the nests of other bird species, leaving the host birds to incubate and care for their eggs [12]. In the context of optimization, cuckoos represent candidate solutions, and the host nests are analogous to the search space [20] [21]. Key Components of Cuckoo Search Optimization:

Initialization: Generate an initial population of cuckoos (solutions) randomly in the search space. Evaluate the fitness of each cuckoo using the objective function.

Levy Flight for Exploitation: Perform Levy flights to explore the search space efficiently. Levy flights are random walks that mimic the erratic foraging behavior of birds.

Cuckoo Levy Flights: Update cuckoo positions using Levy flights to explore new potential solutions. The new position x_i^{new} is calculated as:

$$x_i^{new} = x_i + \alpha \odot Levy\,(\lambda) \qquad (8)$$

Where x_i is the current position, α is the step size, and $Levy\,(\lambda)$ represents a Levy flight with a power-law distribution and $\odot$ denotes element-wise multiplication.

Cuckoo Fractional Change: Introduce a fractional change in the position of some cuckoos to diversify the search. A fraction of cuckoos (β) is randomly selected, and their positions are replaced by new ones.

$$x_i^{new} = x_i + \beta \odot rand() \quad (9)$$

Where $rand()$ is a random number in the range [0,1].

Evaluate and Selection: Evaluate the fitness of the newly generated cuckoos. Replace old cuckoos with new ones if they have better fitness values.

Stop Criterion: Check if the termination condition is met (e.g., a maximum number of iterations or a satisfactory solution is found).

Step by Step Procedure:

Step 1: Initialization

Step 1.1: Initialize Parameters: Set the population size N, maximum iteration *MaxIter*, Levy flight exponent λ, and step size α.

Step 1.2: Generate Initial Population: Randomly initialize the positions of *N* cuckoos in the search space. Evaluate the fitness of each cuckoo using the objective function.

Step 2: While (stopping criterion not met): Repeat until reaching the maximum iterations or a convergence criterion.

Step 3: Levy Flight for Exploitation

Step 3.1: Perform Levy Flights: Update cuckoo positions using Levy flights to explore the search space efficiently.

$$Levy(\lambda) = u.\left(\frac{1}{\sqrt{u}}\right)^{1/\lambda}$$

Where u is the random number from a standard normal distribution, and λ is the Levy flight exponent.

Step 3.2: The new position x_i^{new} is calculated using the Levy Flight equation using (8)

Step 4: Cuckoo Fractional Change for Exploration

Step 4.1: Introduce Fractional Change: Introduce a fractional change in the position of some cuckoos to diversify the search.

Step 4.2: A fraction of cuckoos (β) is randomly selected, and their positions are replaced by new ones using the fractional change equation (9).

Step 5: Evaluate and Selection

Step 5.1: Evaluate Fitness: Evaluate the fitness of the newly generated cuckoos.

Step 5.2: Replace Old Cuckoos: Replace old cuckoos with new ones if they have better fitness values.

Step 6: Stop Criterion: Check if the termination condition is met (e.g., a maximum number of iterations or a satisfactory solution is found).

4. RSA ENCRYPTION ALGORITHM

RSA (Rivest-Shamir-Adleman) is a widely used public-key cryptography algorithm that was introduced by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977. RSA is named after the initials of its inventors [13]. The algorithm is known for its security based on the difficulty of factoring the product of two large prime numbers, a problem that becomes computationally infeasible for sufficiently large primes.

Key Generation: RSA uses a pair of public and private keys.

Public Key (e, N): The public key consists of an exponent *e* and a modulus N. *e* is usually a small prime number, often 65537, and *N* is the product of two large prime numbers (*p* and *q*). **Private Key (d, N):** The private key consists of an exponent *d* and the same modulus *N*. The private exponent *d* is calculated such that $\big(mod\ \phi\ (N)\big)$, where (N) is Euler's totient function.

Steps in Key Generation

Step 1: Select two large prime numbers p and q.

Step 2: Compute *N=pq*.

Step 3: Computer $\phi\ (N) = (p-1)(q-1)$

Step 4: Choose e such that $1 < e < \phi\ (N)$ and e is coprime with $\phi\ (N)$.

Step 5: Calculate d as the modular multiplicative inverse of e modulo $\phi\ (N)$, i.e., $d\ \equiv e^{-1}\big(mod\ \phi\ (N)\big)$

Step 6: Public key (e, N)

Step 7: Private key: (d, N)

Encryption: To encrypt a plaintext message *M*:

$$C \equiv M^e\ (mod\ N)$$

Where C is the cipher text, e is the public exponent, N is the modulus.

Decryption: To decrypt the ciphertext C and obtain the original plaintext *M*:

$$M \equiv C^d\ (mod\ N)$$

Where M is the decrypted plaintext, d is the private key and N is the modulus.

5. ADVANCED ENCRYPTION STANDARD (AES) ALGORITHM

AES (Advanced Encryption Standard) is a widely adopted symmetric encryption algorithm designed to provide secure and efficient encryption for electronic data [14]. It was established as the successor to the Data Encryption Standard (DES) by the National Institute of Standards and Technology

(NIST) in 2001. AES has become a standard encryption algorithm used in various applications, including securing communications, data storage, and cryptographic protocols.

5.1 Key Characteristics of AES

- **Symmetric Key Algorithm:** AES is a symmetric key algorithm, meaning the same key is used for both encryption and decryption.
- **Block Cipher:** AES operates on fixed-size blocks of data, with a block size of 128 bits.
- **Key Sizes:** AES supports key sizes of 128, 192, and 256 bits. The security strength increases with larger key sizes.
- **Rounds:** The number of rounds in AES depends on the key size: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.
- **Substitution-Permutation Network (SPN) Structure:** AES employs a Substitution-Permutation Network structure, which involves substitution (subBytes), permutation (shiftRows), mixing (mixColumns), and key addition (addRoundKey) operations.

5.2 Encryption Process

The AES encryption process involves several steps:

- **Key Expansion:** The original key is expanded into a set of round keys, one for each round of the encryption process.
- **Initial Round Key Addition:** The first round key is added to the initial block of plaintext.
- **Rounds:** Each round involves four main operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey.
    - **SubBytes:** Non-linear substitution step where each byte of the block is replaced with another according to a fixed lookup table.
    - **ShiftRows:** Bytes in each row are shifted by an offset.
    - **MixColumns:** Transformation mixing the data in each column.
    - **AddRoundKey:** XOR operation with the round key.
- **Final Round:** The final round skips the MixColumns step.
- **Ciphertext:** The final result after the specified number of rounds is the ciphertext.

5.3 Decryption Process

AES decryption is the reverse process of encryption, involving the inverse of the operations used in encryption. The decryption process includes:

- **Key Expansion:** The original key is expanded into the set of round keys.
- **Initial Round Key Addition:** The last round key is added to the ciphertext.
- **Rounds:** In each round, the following operations are applied in reverse order: InvShiftRows, InvSubBytes, InvMixColumns, and AddRoundKey.
- **Final Round:** The final round skips the InvMixColumns step.
- **Plaintext:** The result after the specified number of rounds is the decrypted plaintext.

6. PROPOSED ENHANCED ACCESS CONTROL WITH HYBRID ENCRYPTION APPROACH

This hybrid encryption approach involves using RSA for asymmetric encryption (for access control policies) and AES for symmetric encryption (for access control credentials). The keys for both RSA and AES are optimized using WOA and CSO, respectively. The following are the step-by-step procedure for the Enhanced Access Control with Hybrid Encryption approach.

Step 1: Access Control Policy Encryption (RSA): Encrypt access control policies using RSA.

$$C = P^e mod\ N$$

Where C is the ciphertext, P is the plaintext (access control policies), e is the public exponent, and N is the modulus.

Step 2: RSA Key generation and Optimization with WOA: Generate RSA keys (public and private). Apply WOA to optimize the RSA private key (e.g., adjust private exponent d).

$$X_i^{t+1} = X_i^t - A.D_i^t$$

Where X_i^t is the current position of the ith whale at iteration i. A is the coefficient. D_i^t is the distance vector.

Step 3: Symmetric Key Generation (AES): Generate a symmetric key for AES encryption.

Step 4: Optimization-Based Symmetric Key Optimization (CSO): Apply CSO to optimize the symmetric key for AES.

$$X_i^{t+1} = X_i^t + \alpha.L.Levy(\curlywedge)$$

X_i^t is the current position of the i-th cuckoo at iteration t, α is the step size, L is the scaling factor, and $Levy(\curlywedge)$ is the Levy flight.

Step 5: Access Control Credential Encryption (AES): Encrypt access control credentials using AES.

$$C_{AES} = AES_{Encrypt}(P_{credentials}, K_{AES})$$

Where C_{AES} is the AES ciphertext, $P_{credentials}$ is the plaintext (access control credentials), K_{AES} is the optimized symmetric key.

Step 6: Secure key storage: Store the optimized RSA private key and the optimized symmetric key securely on the cloud server and client devices.

Step 7: Transmission of Encrypted Policies and Credentials: Transmit the encrypted access control policies (RSA ciphertext) and credentials (AES ciphertext) to the cloud server.

Step 8: Cloud Server Processing: On the cloud server, decrypt the RSA private key using the optimized RSA private key (asymmetric decryption). Use the decrypted RSA private key to decrypt the access control policies (RSA decryption). Use the optimized symmetric key to decrypt the access control credentials (AES decryption).

7. RESULT AND DISCUSSION

The performance of the proposed Hybrid Encryption Approach is evaluated with other encryption techniques like RSA and AES with various performance metrics like Key Generation Time (in seconds), Key Updation time (in seconds), Encryption time (in seconds), Decryption time (in seconds), total encryption and decryption time (in seconds) and Throughput (in mbps) for various size of the dataset (in MB).

Table 1 depicts the Key Generation time (in seconds) by the proposed Hybrid Encryption approach, RSA and AES for various size of the dataset. From the table 1, The proposed hybrid encryption approach consistently demonstrates competitive or improved key generation times compared to both RSA and AES across all file sizes. As the file size increases, the key generation time for the proposed hybrid approach shows a moderate upward trend, indicating scalability.

RSA exhibits longer key generation times compared to the proposed hybrid approach, particularly as the file size increases. The key generation time for RSA shows a noticeable increase with larger datasets. AES key generation times are generally similar to RSA but slightly faster. Like RSA, the key generation time for AES also increases with larger file sizes, albeit at a slightly slower rate compared to RSA.

The proposed hybrid encryption approach appears to strike a balance between the key generation times of RSA and AES. As file sizes grow, the proposed hybrid approach maintains its efficiency relative to the individual RSA and AES methods. Depending on specific application requirements, the proposed hybrid approach may offer a favorable trade-off between key generation time and security.

Table 1: Key Generation Time (in seconds) by Proposed Hybrid Encryption approach, RSA and AES for various size of the dataset

File Size (in MB)	**Key Generation Time (in Seconds)**		
	Proposed Hybrid Approach	**RSA**	**AES**
100	378	532	641
200	415	689	692
300	564	745	787
400	632	814	891
500	786	1053	1052
600	814	1121	1128
700	863	1198	1196
800	901	1263	1262
900	924	1314	1322
1000	998	1382	1381

Table 2 depicts the Key Updation time (in seconds) by the proposed Hybrid Encryption approach, RSA and AES for various size of the dataset. From the table 2, The proposed hybrid encryption approach consistently demonstrates efficient key updation times compared to both RSA and AES across all file sizes. As the file size increases, the key updation time for the proposed hybrid approach shows a gradual but manageable increase, indicating scalability.

RSA exhibits longer key updation times compared to the proposed hybrid approach. The key updation time for RSA shows a noticeable increase with larger datasets, and the gap between RSA and the proposed hybrid approach becomes more pronounced.

AES key updation times are generally similar to RSA but slightly faster. Similar to RSA, the key updation time for AES also increases with larger file sizes, but the proposed hybrid approach maintains a competitive edge.

The proposed hybrid encryption approach consistently outperforms both RSA and AES in terms of key updation time across various file sizes. The efficiency of the proposed hybrid approach makes it a strong candidate for applications where frequent key updation is required, striking a balance between RSA and AES.

Table 2: Key Updation Time (in seconds) by Proposed Hybrid Encryption approach, RSA and AES for various size of the dataset

File Size (in MB)	**Key Updation Time (in Seconds)**		
	Proposed Hybrid Approach	**RSA**	**AES**
100	239	392	421
200	248	463	498
300	261	518	532
400	275	592	603
500	283	684	685
600	297	735	745
700	310	821	843
800	321	906	918
900	342	998	1015
1000	356	1024	1035

Table 3 depicts the Encryption time (in seconds) by the proposed Hybrid Encryption approach, RSA and AES for various size of the dataset. From the table 3,

Table 3: Encryption Time (in seconds) by Proposed Hybrid Encryption approach, RSA and AES for various size of the dataset

File Size (in MB)	Encryption Time (in Seconds)		
	Proposed Hybrid Approach	RSA	AES
100	28	68	69
200	34	83	88
300	49	96	98
400	63	118	128
500	82	138	145
600	98	167	175
700	109	181	192
800	122	203	214
900	141	234	245
1000	156	263	275

From the table 3, The proposed hybrid encryption approach consistently demonstrates significantly faster encryption times compared to both RSA and AES across all file sizes. As the file size increases, the encryption time for the proposed hybrid approach shows a moderate increase, but it remains notably lower than both RSA and AES.

RSA exhibits longer encryption times compared to both the proposed hybrid approach and AES. The encryption time for RSA shows a substantial increase with larger datasets, making it less efficient for larger file sizes. AES encryption times are generally lower than RSA but higher than the proposed hybrid approach. Similar to RSA, the encryption time for AES also increases with larger file sizes, although the proposed hybrid approach maintains a significant speed advantage.

The proposed hybrid encryption approach outperforms both RSA and AES in terms of encryption time across various file sizes, making it a highly efficient choice for encryption tasks. The efficiency of the proposed hybrid approach makes it well-suited for applications where fast encryption is crucial.

Table 4 depicts the Decryption time (in seconds) by the proposed Hybrid Encryption approach, RSA and AES for various size of the dataset. From the table 3,

Table 4: Decryption Time (in seconds) by Proposed Hybrid Encryption approach, RSA and AES for various size of the dataset

File Size (in MB)	Decryption Time (in Seconds)		
	Proposed Hybrid Approach	RSA	AES
100	21	44	53
200	29	53	60
300	42	72	75
400	54	83	90
500	69	95	102
600	82	123	128
700	96	145	150
800	112	165	173
900	128	187	192
1000	132	198	204

From the table 4, The proposed hybrid encryption approach consistently demonstrates significantly faster decryption times compared to both RSA and AES across all file sizes. As the file size increases, the decryption time for the proposed hybrid approach shows a moderate increase, but it remains notably lower than both RSA and AES.

RSA exhibits longer decryption times compared to both the proposed hybrid approach and AES. The decryption time for RSA shows a substantial increase with larger datasets, making it less efficient for larger file sizes. AES decryption times are generally lower than RSA but higher than the proposed hybrid approach. Similar to RSA, the decryption time for AES also increases with larger file sizes, although the proposed hybrid approach maintains a significant speed advantage.

The proposed hybrid encryption approach outperforms both RSA and AES in terms of decryption time across various file sizes, making it a highly efficient choice for decryption tasks. The efficiency of the proposed hybrid approach makes it well-suited for applications where fast decryption is crucial.

Table 5 depicts the Total time taken for encryption and decryption (in seconds) by the proposed Hybrid Encryption approach, RSA and AES for various size of the dataset.

Table 5: Total Encryption/Decryption Time (in seconds) by Proposed Hybrid Encryption approach, RSA and AES for various size of the dataset

File Size (in MB)	Total Encryption/Decryption Time (in seconds)		
	Proposed Hybrid Approach	RSA	AES
100	49	112	122
200	63	136	148
300	91	168	173
400	117	201	218
500	157	233	247
600	180	290	303
700	205	326	342
800	234	368	387
900	264	421	437
1000	294	461	479

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

From the table 5, The proposed hybrid encryption approach consistently demonstrates significantly faster total encryption/decryption times compared to both RSA and AES across all file sizes. As the file size increases, the total time for the proposed hybrid approach shows a moderate increase, but it remains notably lower than both RSA and AES.

RSA exhibits longer total encryption/decryption times compared to both the proposed hybrid approach and AES. The total time for RSA shows a substantial increase with larger datasets, making it less efficient for larger file sizes. AES total encryption/decryption times are generally lower than RSA but higher than the proposed hybrid approach. Similar to RSA, the total time for AES also increases with larger file sizes, although the proposed hybrid approach maintains a significant speed advantage.

The proposed hybrid encryption approach outperforms both RSA and AES in terms of total encryption/decryption time across various file sizes, making it a highly efficient choice for overall cryptographic operations. The efficiency of the proposed hybrid approach makes it well-suited for applications where both fast encryption and decryption are crucial.

Table 6 depicts the Throughput (in mbps) by Proposed Hybrid Encryption approach, RSA and AES for various size of the dataset.

Table 6: Throughput (in mbps) by Proposed Hybrid Encryption approach, RSA and AES for various size of the dataset

File Size (in MB)	Throughput (in mbps)		
	Proposed Hybrid Approach	**RSA**	**AES**
100	1436	1128	1113
200	1487	1186	1172
300	1521	1245	1230
400	1545	1288	1273
500	1598	1326	1314
600	1628	1378	1368
700	1675	1432	1426
800	1708	1483	1467
900	1734	1535	1518
1000	1789	1590	1575

From the table 6, The proposed hybrid encryption approach consistently demonstrates higher throughput (in mbps) compared to both RSA and AES across all file sizes. As the file size increases, the throughput for the proposed hybrid approach remains consistently higher, showcasing its efficiency in data processing.

RSA exhibits lower throughput compared to both the proposed hybrid approach and AES. The throughput for RSA shows a moderate increase with larger datasets, but it remains lower than the proposed hybrid approach.

AES throughput is generally lower than the proposed hybrid approach but higher than RSA. Similar to RSA, the throughput for AES also increases with larger file sizes, but the proposed hybrid approach maintains a throughput advantage. The proposed hybrid encryption approach consistently outperforms both RSA and AES in terms of throughput across various file sizes, indicating its superior data processing efficiency. The high throughput of the proposed hybrid approach makes it well-suited for applications where rapid data transfer is critical.

8. CONCLUSION

In this research work, the hybrid encryption approach integrating Whale Optimization Algorithm (WOA) for RSA and Cuckoo Search Optimization (CSO) for AES presents a robust and innovative solution for securing access control in cloud computing environments. This method leverages the strengths of both optimization algorithms and combines the security features of asymmetric (RSA) and symmetric (AES) encryption. The step-by-step procedure outlined for RSA encryption, key optimization using WOA, AES encryption, and key optimization using CSO provides a comprehensive framework for implementing a secure access control system. The use of RSA ensures secure and efficient key exchange for access control policies, while AES efficiently encrypts access control credentials. The integration of optimization algorithms such as WOA and CSO enhances the security of the cryptographic keys by optimizing their parameters. WOA, inspired by the hunting behavior of whales, and CSO, mimicking the brood parasitic behavior of cuckoos, contribute to the robustness of the encryption keys. From the results obtained, it is clear that the proposed Hybrid Encryption based Access Control approach consumes less key generation time, updation time, encryption time, decryption and total time taken for encryption and decryption than the existing encryption techniques like RSA, and AES.

REFERENCES

[1] Daniel, Aliu, et al. "A computer security system for cloud computing based on encryption technique." Computer Engineering and Applications 10.1 (2021): 41-53.
[2] Kumar, Sarvesh, et al. "Chaos based image encryption security in cloud computing." Journal of Discrete Mathematical Sciences and Cryptography 25.4 (2022): 1041-1051.
[3] Mehrtak, Mohammad, et al. "Security challenges and solutions using healthcare cloud computing." Journal of medicine and life 14.4 (2021): 448.
[4] Chinnasamy, P., et al. "Efficient data security using hybrid cryptography on cloud computing." Inventive Communication and Computational Technologies: Proceedings of ICICCT 2020. Springer Singapore, 2021.
[5] Abroshan, Hossein. "A hybrid encryption solution to improve cloud computing security using symmetric and asymmetric

cryptography algorithms." International Journal of Advanced Computer Science and Applications 12.6 (2021): 31-37.

[6] Liu, Pengtao. "Public-key encryption secure against related randomness attacks for improved end-to-end security of cloud/edge computing." IEEE Access 8 (2020): 16750-16759.

[7] Shabbir, Maryam, et al. "Enhancing security of health information using modular encryption standard in mobile cloud computing." IEEE Access 9 (2021): 8820-8834.

[8] Velmurugadass, P., et al. "Enhancing Blockchain security in cloud computing with IoT environment using ECIES and cryptography hash algorithm." Materials Today: Proceedings 37 (2021): 2653-2659.

[9] Thabit, Fursan, et al. "A new lightweight cryptographic algorithm for enhancing data security in cloud computing." Global Transitions Proceedings 2.1 (2021): 91-99.

[10] Subhash, Ligade Sunil, and R. Udayakumar. "Sunflower whale optimization algorithm for resource allocation strategy in cloud computing platform." Wireless Personal Communications 116 (2021): 3061-3080.

[11] Krishnadoss, Pradeep, et al. "CCSA: Hybrid cuckoo crow search algorithm for task scheduling in cloud computing." International Journal of Intelligent Engineering and Systems 14.4 (2021): 241-250.

[12] Kumar, Y. Kiran, and R. Mahammad Shafi. "An efficient and secure data storage in cloud computing using modified RSA public key cryptosystem." International Journal of Electrical and Computer Engineering 10.1 (2020): 530.

[13] Awan, Ijaz Ahmad, et al. "Secure framework enhancing AES algorithm in cloud computing." Security and communication networks 2020 (2020): 1-16.

[14] Poornappriya, T.S., Selvi, V., Evolutionary Optimization of Artificial Neural Network for Diagnosing Autism Spectrum Disorder, International Journal of Electrical Engineering and Technology (IJEET), 11(7), 47-61 (2020).

[15] Subhashini, M., & Gopinath, R., Mapreduce Methodology for Elliptical Curve Discrete Logarithmic Problems – Securing Telecom Networks, International Journal of Electrical Engineering and Technology, 11(9), 261-273 (2020).

[16] Upendran, V., & Gopinath, R., Feature Selection based on Multicriteria Decision Making for Intrusion Detection System, International Journal of Electrical Engineering and Technology, 11(5), 217-226 (2020).

[17] Upendran, V., & Gopinath, R., Optimization based Classification Technique for Intrusion Detection System, International Journal of Advanced Research in Engineering and Technology, 11(9), 1255-1262 (2020).

[18] Subhashini, M., & Gopinath, R., Employee Attrition Prediction in Industry using Machine Learning Techniques, International Journal of Advanced Research in Engineering and Technology, 11(12), 3329-3341 (2020).

[19] Priyadharshini, D., Poornappriya, T.S., & Gopinath, R., A fuzzy MCDM approach for measuring the business impact of employee selection, International Journal of Management (IJM), 11(7), 1769-1775 (2020).

[20] Poornappriya, T.S., Gopinath, R., Application of Machine Learning Techniques for Improving Learning Disabilities, International Journal of Electrical Engineering and Technology (IJEET), 11(10), 392-402 (2020).