Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 3 (2023)

# Distributed Cipher Hashing Cryptographic Technique For Secure Data Storage In Cloud Computing Using XOR-MECC

**[1] A R Athira, [2] Dr. P. Sasikala, [3] Dr. R. Reka**

[1]Research Scholar, Department of Computer Science, Vinayaka Mission's Kirupananda Variyar Engineering College, India.
[2] Professor and Head, Department of Mathematics, Vinayaka Mission's Kirupananda Variyar Engineering College
[3]Associate Professor, Department of CSE, Mahendra College of Engineering,

Email id: [1]parvathyretnakaran@gmail.com, [2]rgsasi@gmail.com, [3]rekasatheesh@gmail.com

**Abstract:** The user is enabled by means of Cloud Computing (CC) to store and access the data as of anywhere. To secure the data amassed on the Cloud Server (CS), CC security encompasses a collection of policies and technologies. Most existing methods bring about a lower-Security Level (SL), the higher computation time for encryption together with decryption, and also Memory Usage (MU) issues, et cetera. Here, to resolve such issues in the existent methods, an effectual modified cryptographic is proposed in favor of the secured Data Storage (DS) in CC. Firstly, secure authentication is done via the Alder32 hash algorithm. The user is authenticated via registration, login, together with verification. Subsequently, the authenticated user commences uploading the file. Prior to uploading, the inputted file is bifurcated into sub-files counting on the size, and the features are extracted as of the split file and distributed CS. Next, the Brownian Motion-based BAT algorithm (BM-BAT) selects the appropriate distributed CS. Then, the Modified Cryptographic (XOR-MECC) uploads the files securely into the CS. The experimentation's outcomes state that the proposed work attains better performance when weighed against the prevailing works.

**Keywords:** Adler32 hash algorithm, Brownian Motion-based BAT algorithm (BM-BAT), Modified Elliptic Curve Cryptographic (XOR-MECC) method.

## 1. Introduction

CC renders effectual, reliable, together with scalable resources for DS as well as computational activities at an extremely low cost. Thus, it functions a considerable role in people's daily life [1]. For reducing their local storage load, increasingly people depend on cloud storage services [2]. Cloud storage inspires users to transfer their huge data as of local storage to isolated CS as an integral part of CC [3]. Users can benefit from cost savings and productivity amelioration utilizing cloud-centered services to direct projects and establish collaborations via shifting the local data management system into cloud storage [4]. CC provides seemly unlimited storage as well as calculation resource to users as services athwart the Internet whilst hiding platform in addition to implementation details as of users [5]. Unpredictable security along with privacy concerns becomes the most famous issue that holds back the extensive adoption of DS in public cloud infrastructure regardless of the marvelous economic in addition to technical benefits [6]. Since the DO might share their data with requesters or friends, the data outsourced confidentiality is vital to the Data Owner (DO). Therefore, there is an absolute requirement for a supple secure access control scheme [7]. Nevertheless, in securing cloud infrastructures, unauthorized access together with malicious users is a significant issue. The more meaningful information that is being amassed on the cloud infrastructure might be deleted or modified by these unwanted users [8]. Usually, end-users favor encrypting their data prior to uploading to the CS, which might be malicious as the cloud storage server might be untrusted along with the outsourced data might encompass responsive information (for instance, company financial data, medical records, et cetera) [9]. As of the cloud, the DO requires to download the encrypted data as well as re-encrypt them for sharing (the DO encompasses no local copies of the data) [10]. DO might be unwilling to outsource private or classified data to cloud storage without sufficient protection [11]. Thus, it is significant to plan some good security mechanisms. It can fully assure the provision of preferred security aspects [12].

Data replication, regenerating codes, Redundant Residue Number Systems (RRNS), erasure codes, Secret Sharing Schemes (SSS), homomorphic encryption, et cetera are utilized for ameliorating security and lessening the hazards of data loss together with corruption [13]. Presently, security audit, encryption, together with access control are the '3' methods that the academic researches on data safety of cloud storage chiefly concentrated on. A common technique that can guard each file or data block is Encryption [14]. Secure auditing is practical in verifying DS correctness [15]. A mechanism wherein an auditing authority is capable of auditing the data devoid of affecting its privacy is called Data auditing [16]. A trusted server is utilized by the access control technique wherein the receptive data are amassed in encrypted form. The responsive data will still be private albeit the cloud storage server is compromised [17]. Fragmentation is reasonably commenced for DS utilizing a public cloud intended for the less classified data fragments [18]. It is basically a technique wherein a document is bifurcated into numerous uniform or fixed-size blocks called partitions wherein the document's confidentiality level was not considered [19]. Nevertheless, the prevailing methods still have security problems for the secured DS on the cloud. For ameliorating the SL, the proposed work employs distributed cloud storage utilizing the BM-BAT and XOR-ECC.

The paper is prearranged as: In section 2, methods associated with cloud storage are explained. Section 3 explains the proposed BM-BAT and XOR-ECC. The proposed method's performance is examined in section 4. Section 5 deduces this paper with future enhancement.

## 2. Literature Survey

**Irfan Mohiuddin *et al.* [1]** modeled distributed storage allotment architecture intended for fair usage of storage resources. In addition, it generated integrated end-to-end security aimed at data on cloud storage for eliminating insider hazards. The data that ought to be amassed in the VM disks on the physical storage on the cloud data centre was divided into chunks. The data packages were classified into classes by the classifier. It corresponds to the classes generated on the physical storage. For forming a virtual pond of storage, the physical cloud storage servers were joined. Next, it was split into virtual blocks. Nevertheless, cloud data centers were extremely intricate, merely depending on initial placement that can't balance the network's load.

**Manoj Tyagi *et al.* [2]** incorporated server assortment approach, authentication, together with encryption schemes to attain integrity, confidentiality together with effectual computing. Utilizing the cuckoo algorithm, the Markov chains process, together with Levy's flight, the well-known server was selected via the Cloud services providers (CSP). Subsequent to server selection, the user's data was encrypted utilizing an Elliptic Curves (EC) integrated encryption at the user's side and transformed to CSP aimed at storage. Subsequent to applying 2$^{nd}$ encryption utilizing Advanced Encryptions Standard at the cloud's side, the CSP amassed the data. In that, the values were randomly selected that might bring about the cryptographic attacks.

**Abdulatif Alabdulatif *et al.* [3]** commenced privacy-preserving dispersed analytics aimed at big data on the cloud. '2' main objectives were: a) Preserving the users' data privacy while storing and processing on the cloud, in addition to b) lessening the considerable effect of processing encrypted data. Completely homomorphic encryption was employed as a powerful cryptosystem. It performed the examination tasks on encrypted data. Analysis's accuracy, execution time performance, and the resources' size was the '3' factors centered upon which the generated distributed approach regarded the scalability feature. The factors influenced each other, and users might desire to apply some restraints on those aspects. The Homomorphic encryption had cons of big key size and lower computation efficiency to secure CC.

**M. Sivaram *et al.* [4]** rendered a Fuzzy-centered Heuristic algorithm for solving the local in addition to global optimization issues in big networks on the cloud. The Sheepdog mechanism monitored the cluster information aimed at memory allocation on the cloud. A fuzzy-entered clustering approach formed the clusters. Centered upon CPU Core, Speed, along with RAM capacity of cluster nodes on the cloud, cluster heads were chosen utilizing fuzzy rules. Every cluster encompassed cluster-id with its members. It was employed to recognize which cluster made the storage request. The server-generated the distributed keys utilizing RSA to make sure security on the cloud storage and utilized KNN as a guide for effectual storage allotment. The KNN encompassed a time intricacy issue for effectual searching and wasn't appropriate for higher-dimensional data.

**Yongkai Fan *et al.* [5]** suggested a safe terminal architecture centered upon Trusted Executions Environment (TEE). An autonomous execution space TEE isolated as of Rich Executions Environment. It supported sufficient security on the data integrity verification process. The technique employed Sha TEE mir's (t, n) threshold scheme for encrypting the private key and transferred it to the cloud in lessening the storage utilize of the local side whilst outsourcing a larger-scale file to the cloud. The integrity corroboration for a single file was supported as well as concurrent verification for manifold files were attained.

**Wenting Shen *et al*. [6]** planned a realistic data integrity auditing devoid of private key storage aimed at secured cloud storage. In the registration and the signature generation phase, '2' fuzzy private keys were extracted as of the user. For generating '2' linear sketches, those '2' fuzzy private keys were employed. It encompassed coding and error correction procedures. Via eradicating the "noise" as of '2' sketches, the '2' fuzzy private keys were contrasted for confirming the user's identity. A customized BLS short signature was designed grounded on the notion of fuzzy signature buttressing blockless verifiability in addition to compatibility with the linear sketch. The signature scheme was merely appropriate for static data.

## 3. Proposed Distributed Cipher Hashing Cryptographic Method

Cloud security is the secured data storage as of any unauthorized access via a compilation of strategies. Cloud security is vital in CC for making sure that the data amassed in the cloud is secure. Information loss, random generation of private keys, together with insufficient memory storage, etc is some security issues that the disparate systems that were developed for cloud security had encountered. The proposed method's block diagram is exhibited in Figure 1,

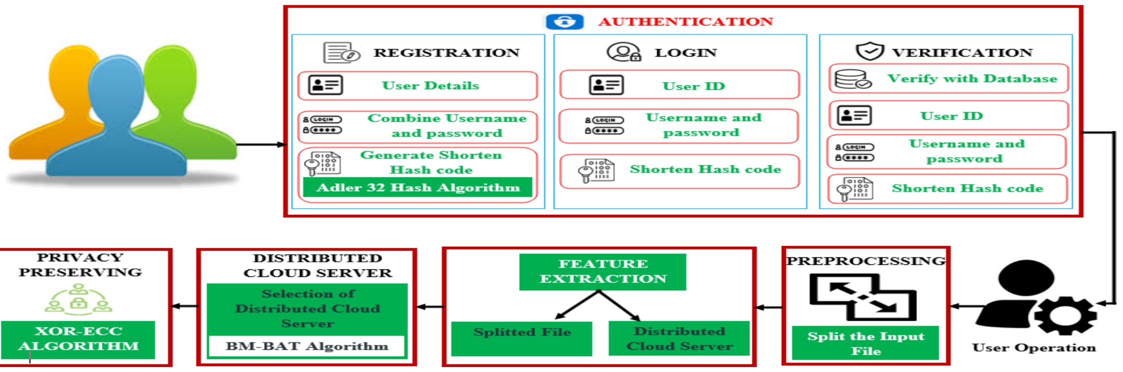


**Fig 1:** Block diagram of the proposed methodology

An effectual customized cryptographic is utilized for eliminating these issues and enhancing the SL in distributed cloud storage. Authentication, distributed cloud storage, together with privacy-preserving is the three phases that the proposed work encompasses. Via the registration, login, together with verification, the user can well be allowed to utilize the application in the authentication phase. The Adler32 hash algorithm generates a shorten Hash Code (HC) secure authentication during registration. Once the authentication is done, the user will choose the input file to upload on the CS. Prior to uploading, the inputted file is bifurcated into sub-files considering the size. In the succeeding phase, the split file as well as the distributed CS's features is extracted. Next, the split data will be amassed in the distributed CS, and the BMBAT selects it. In the '3$^{rd}$' phase, the file is uploaded into the distributed CS in a safe manner and the XOR-MECC ensures data privacy.

### 3.1 Authentication

For getting access to information in CC, authentication is an indispensable step. Via verifying the user's identity, authentication allows just the authenticated users to access the amassed data. Registration, login, along with verification are the '3' steps involved in User authentication.

***Registration***

Here, every user will generate their identity via rendering the username and password to the CS. The combined user ID $U_{id}$ as well as password $U_{pw}$ is rendered to the CS for registration as,

$$C_U \xrightarrow{\{U_{id}, U_{pw}\}} C_s \tag{1}$$

Wherein, $C_U$ signifies the user and $C_s$ implies the cloud server. The Adler32 hashing algorithm generate a shorten HC, through which the server reacts to the success andfailure of the registration. In the Adler32, '2' 16-bit checksums $C_1$ and $C_2$ are computed for the combined user id and password called as $S_i$ and are combined into a 32-bit integer. The function is derived as,

$$C_1 = 1 + S_1 + S_2 + .........S_n (\bmod 65521) \tag{2}$$

Wherein, $C_1$ signifies the sum of every byte. $C_2$ implies the sum of every individual values of $C_1$ computed as,

$$\begin{aligned} C_2 &= (1+S_1) + (1+S_1+S_2) + ..........(1+S_1+S_2+......S_n)(\bmod 65521) \\ &= n \times S_1 + (n-1) \times S_2 + (n-2) \times S_3 + .........+ S_n + n(\bmod 65521) \end{aligned} \tag{3}$$

Wherein, $S$ signifies the bytes string and $n$ implies the length of $S$ . Lastly, the HC is generated as,

$$\vec{A}(S) = C_2 \times 65536 + C_1 \tag{4}$$

Wherein, $\vec{A}(S)$ signifies the generated HC. Next, the server transmits the generated HC to the user and stores $U_{id}, U_{pw}$ and $\vec{A}(S)$ of the registered user in its database.

***Login***

The user is subjected to the login phase once the registration is finished. Login methods aid the user to get access to the CS. During login, the user must enter the username, password, in addition to HC that is rendered in the registration phase.

$$C_U \xrightarrow{\{U_{id}, U_{pw}, \vec{A}(S)\}} C_s \tag{5}$$

Subsequent to log in, the authentication is performed via verification.

***Verification***

This is carried out to refuse the unregistered user who endeavors to log into the system. Here, entered user id, username, password, along with HC of user id is matched with the saved values.

$$\{U_{id}, U_{pw}, \vec{A}(S)\}_{matched} \xrightarrow{response} U_{authoried} \tag{6}$$

$$\{U_{id}, U_{pw}, \vec{A}(S)\}_{notmatched} \xrightarrow{response} U_{unauthorized} \tag{7}$$

The user is authorized to log in the system if the match is found. Otherwise, the login is refuted to the user. The authentication will end subsequent to the verification.

### 3.2 User Operation

The CS allows the authenticated user for uploading the files easily to the cloud. The authenticated user chooses the input file that is essential to be uploaded to the server as of their device storage. It can well be of any sort. After choosing the file, the file can well be uploaded to the server utilizing the subsequent procedures. The inputted file chosen via the user is signified as $F$

### 3.3 Preprocessing

Then, the chosen file is preprocessed wherein the input file $F$ is split into manifold sub-files as,

$$F_i = \{F_1, F_2, F_3, ........, F_n\} \tag{8}$$

Wherein, $F_n$ signifies the $n^{th}$ subfile. Here, centered upon the size, the file is bifurcated. Next, the split file is rendered to the distributed CS wherein the significant features of the split file are extracted for additional processing.

### 3.4 Distributed Cloud Server

Here, the pre-processed file is utilized to choose distributed CS. The server can't check the likelihood of data storing into the distributed CS since the server didn't have the knowledge about the split file's size. The BM-BAT takes care of this issue. Initially, as of the split file and the distributed CS, the features are extracted. The BM-BAT decides the distribution of split data to the server relying on these features.

#### *3.4.1 Feature Extraction*

Here, the split file together with the distributed CS's features is extracted. From the split file, Speed, Task Cost, Weight, size data, etc are extracted, which are signified as,

$$P_f = \{g_1, g_2, g_3, .........g_n\} \tag{9}$$

Wherein, $P_f$ signifies the extracted feature set of the split file, $g_n$ implies the $n^{th}$ feature in the feature set $P_f$. CPU resources, storage resource, memory, memory resources, total requests, bandwidth, processing speed as well as cycle, disk space, load on VM, CPU usage is extracted as of the distributed CS. The feature set extracted as of the distributed cloud server $P_{csf}$ is,

$$P_{csf} = \{h_1, h_2, h_3, ..........h_n\} \tag{10}$$

Wherein, $h_n$ signifies the $n^{th}$ feature on the feature set $P_{csf}$. Subsequent to the extraction of the features as of the split file and distributed CS, the total features extracted is expressed as,

$$W_i = \{P_f, P_{csf}\} \tag{11}$$

Wherein, $W_i$ implies the last feature set of the split file and distributed CS.

#### 3.4.2 Selection of distributed cloud server

Here, the BM-BAT selects the equivalent server aimed at the distribution of split data. Centered upon the echolocation behaviour of the microbat, the BAT is derived. The new solution will be randomly formed in the prevailing BAT. Thus, the outcome might be varying counting on the arbitrary values. The BM is employed for resolving the arbitrary selection of new solutions.

Initially, every bat in the population $W_i$ (explicitly, the extracted features) moves with the frequency, $f_{fi}$, velocity $V_{vi}$, and loudness $L_{li}$ in a d-dimensional search space. Centered upon the extracted features, every bat's fitness is estimated. The bats' movement at every time step $n$ is formed via updating their velocity along with position as,

$$P_i^n = P_i + V_{vi}^n \tag{12}$$

$$V_{vi}^n = f_{fi}\left(\frac{V_{vi}^{n-1}}{f_{fi}} + (P_i - \overline{P})\right) \tag{13}$$

$$f_{fi} = \gamma\left(\frac{f_{f\min}}{\gamma} + (f_{f\max} - f_{f\min})\right) \tag{14}$$

Wherein, $P_i$ implies the location of the $ith$ bat, $\gamma$ signifies the arbitrary parameter as well as $\gamma \in (0,1)$, $\overline{P}$ implies the best solution attained subsequent to comparing every solutions of $m$ bats and $f_f \in (f_{f\max}, f_{f\min})$. Next, the BM technique $D(a)$ selects the best solution amongst the current best solutions as,

$$D(a)\,\alpha\,\sqrt{a} \tag{15}$$

$$D(a) = j\sqrt{a} \tag{16}$$

Wherein, $j$ signifies constant, $a$ signifies scaling factor. Next, a new solution for every bat is formed by means of local search as,

$$P^{new} = P^{old} + D(a)L_{li}^{avg} \tag{17}$$

Wherein, $L_{li}^{avg}$ signifies the average loudness of every bats, $j$ signifies constant, $P^{new}, P^{old}$ are the bats' new and old positions. Then, the old and new positions' fitness is estimated and contrasted with one another as,

$$P^{cur} = \begin{cases} P^{new} & if\ (\eta_{old} < \eta_{new}) \\ P^{old} & otherwise \end{cases} \tag{18}$$

Wherein, $\eta_{old}, \eta_{new}$ implies the old and new positions' fitness, $P^{cur}$ signifies the current position. Additionally, the bat's loudness and pulse emission rate are updated when it finds its prey. The bat's loudness is lessened as well as the pulse emission rate augments as,

$$L_{li}^{n+1} = \rho L_{li}^{t} \tag{19}$$

$$z_{ri}^{n+1} = z_{ri}^{0}(1 - e^{-\delta n}) \tag{20}$$

Wherein, $\rho, \delta$ implies constants, $z_{ri}$ implies the pulse emission rate. Lastly, the bats are ranked as well as the current best solution is found. Aimed at the current best solution, the fitness is estimated. The BM-BAT selects the distributed CS to upload the file via following the way of finding the best solution. The proposed BM-BAT pseudo-code is delineated in Figure 2,

```
Input: Extracted features $W_i$
Output: Selected distributed cloud server

Begin
    Initialize population $W_i$, frequency $f_{fi}$, velocity $V_{vi}$, loudness $L_{li}$, maximum number
    of iteration $i_{max}$
    Calculate fitness for each bat
    Set $i = 0$
    While ($i \le i_{max}$)
        Update $V_{vi}$ , $f_{fi}$ , $P_i$
        Update new solution by local search
        Evaluate fitness of new solution $\eta_{new}$
            If ($\eta_{old} < \eta_{new}$) {
                If ($L_{li} > n_{ran}$) {
                    Replace $P^{cur} = P^{new}$
                } End if
            Else
            {
                $P^{cur} = P^{old}$
            } End if
        Rank the bats
        Update current location
        Calculate fitness of the current best location
    Set $i = i + 1$
    End while
    Return selected distributed cloud server
End
```

**Fig 2:** Pseudocode for the BM-BAT algorithm

Figure 2 signifies the way of selection of distributed CS.

### 3.3 Privacy preserving

The file is uploaded securely to the chosen CS. The XOR-MECC ensures security on the transformation of information. The prevailing ECC is more responsive to vulnerabilities as it encompasses a poor design of the system together with procedures. The appropriate design is fixed via the XOR-MECC to resolve this issue. An alternate technique to implement public-key cryptography is the ECC. The mathematical expression of the EC is,

$$s^2 = t^3 + ut + v \tag{21}$$

Wherein, $u, v$ signifies the integers, the standard variables $s, t$ are utilized to state the function. The ECC generates the public along with the private key since the key generation is fundamental to encrypt and also decrypt the data. The XOR key is generated along with the public along with private keys to ameliorate security. The public key $K_p$ can well be generated as,

$$K_p = R_{ran} * q \tag{22}$$

Wherein, $R_{ran}$ signifies the arbitrary number that is termed a private key and $R_{ran} \in (1, n-1)$, $q$ implies the points on the curves. '2' alphanumeric numbers are randomly generated to produce an XOR key $K_{xor}$ and are computed as,

$$K_{xor} = A \oplus B \tag{23}$$

Wherein, $\oplus$ implies the XOR, $A$ and $B$ implies the alphanumeric numbers. Next, the encryption together with decryption for the files is done. Amid encryption, the ciphertexts will be formed in addition to the $K_{xor}$ is added with the encryption formula as,

$$CT_1 = c * q \tag{24}$$

$$CT_2 = F + c * K_p + K_{xor} \tag{25}$$

Wherein, $CT_1, CT_2$ implies the ciphertexts, $c$ signifies the arbitrary number on the gamut $(1, n-1)$, $F$ signifies the actual file. The split files as of the distributed CS are gathered once the user endeavors for downloading the file. The gathered files are decrypted and joined for downloading. In the decryption, the actual file can be recovered by subtracting $K_{xor}$ as of the equation as,

$$F = CT_2 - R_{ran} * CT_1 - K_{xor} \tag{26}$$

Wherein, $F$ signifies the final outcome of the decryption.

## 4. Result And Discussion

The proposed model's performance is examined. The proposed secure DS utilizing the XOR-MECC is developed in Java.

### 4.1 Performance Analysis

The XOR-MECC's performance is examined with the Diffie-Hellman, ElGamal, Rivest Shamir Adleman (RSA), together with EC Cryptography (ECC).

**Table 1:** analysis of the performance of XOR-MECC with existing algorithms

**(a)** Encryption Time

| Data size in MB | Diffie-Hellman | ElGamal | RSA | ECC | Proposed XOR-MECC |
|---|---|---|---|---|---|
| **10** | 1978 | 1784 | 1541 | 1345 | 1087 |
| **20** | 2784 | 2589 | 2432 | 2143 | 1867 |
| **30** | 3897 | 3645 | 3245 | 2974 | 2612 |
| **40** | 4923 | 4678 | 4432 | 4124 | 3875 |
| **50** | 5867 | 5521 | 5234 | 4932 | 4521 |

**(b)** Decryption Time

| Data size in MB | Diffie-Hellman | ElGamal | RSA | ECC | Proposed XOR-MECC |
|---|---|---|---|---|---|
| **10** | 1945 | 1768 | 1574 | 1325 | 1098 |
| **20** | 2765 | 2651 | 2378 | 2178 | 1847 |
| **30** | 3851 | 3689 | 3247 | 2917 | 2667 |
| **40** | 4968 | 4612 | 4467 | 4127 | 3856 |
| **50** | 5846 | 5539 | 5274 | 4915 | 4589 |

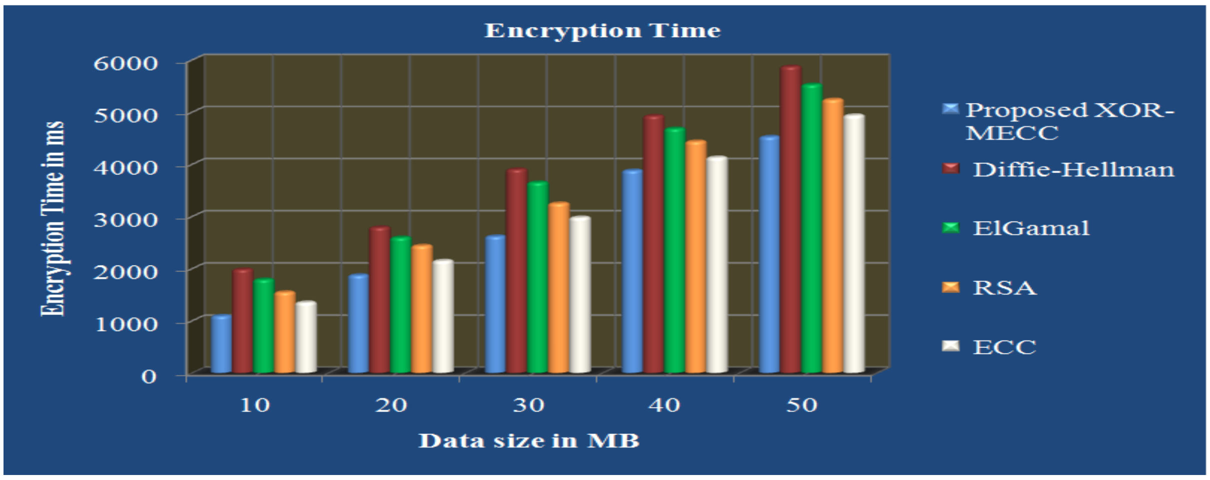
**(c)** Security analysis

| Techniques | Security (%) |
|---|---|
| Diffie-Hellman | 90.124578 |
| ElGamal | 91.324578 |
| RSA | 93.567845 |
| ECC | 95.234578 |
| Proposed XOR-MECC | 96.325678 |

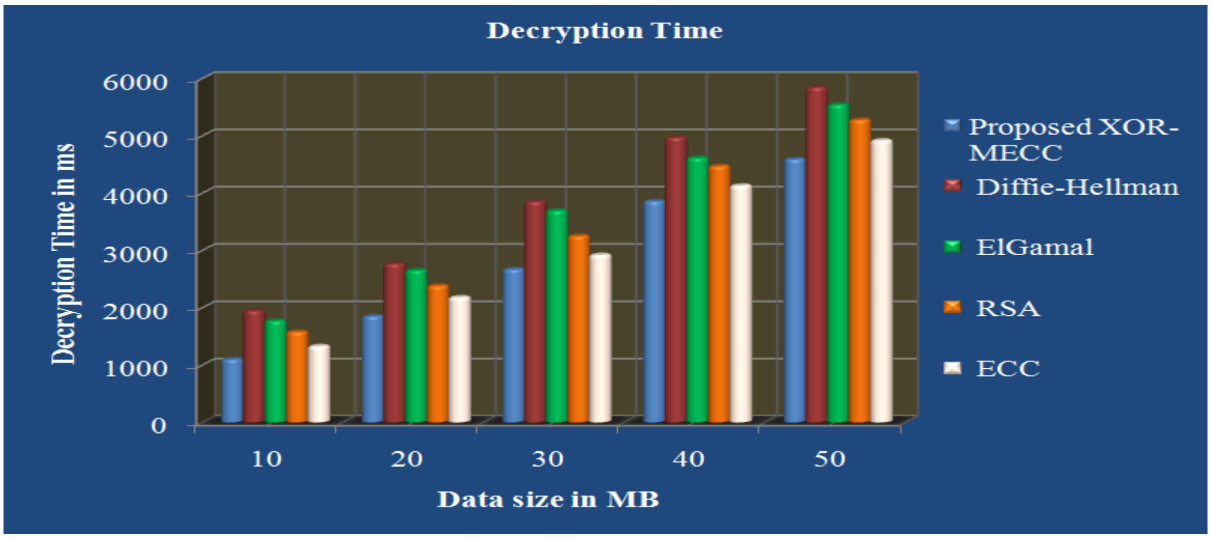
**Discussion:** Table 1 (a) and table 1(b) shows the examination of the XOR-MECC's performance centered upon Encryption Time (ET) and Decryption Time (DT). ET indicates the encryption's speed and the DT denotes the decryption's speed. Concerning data size varies as of 10Mb to 50Mb, the ET together with DT are computed. For the 10Mb data, the proposed one takes 1087ms aimed at encryption and 1908ms for decryption when comparing the ET and DT with existent algorithms. Aimed at the same data size, the prevailing Diffie-Hellman technique takes a larger time contrasted with the other ElGamal, RSA, together with ECC that takes a medium time aimed at encryption as well as decryption. The proposed work's performance centered on SL is showed in Table1 (c). The XOR-MECC together with ECC has an enhanced SL contrasted with the Diffie-Hellman, ElGamal, together with RSA. As of the above-stated discussion, it is apparent that the XOR-MECC attains better performance contrasted with the other prevailing methods.

## 4.2 Comparative Analysis

The comparative assessment of the XOR-MECC with the Diffie-Hellman, ElGamal, RSA, together with ECC centered on ET, DT, SL, MU on encryption in addition to decryption. Next, the performance examination of the BM-BAT and the prevailing Shuffled Frog Leaping Algorithms (SFLA), Cockroach swarm optimizations (CSO), Glowworm Swarm Optimization Algorithms (GSO), and BAT grounded on the assortment of distributed CS and Fitness vs Iteration is exhibited.



**(a)**



**(b)**

**Fig 3**: illustrate the (a) ET (b) DT for the proposed and existing methods.

**Discussion:** The ET in addition to DT of the XOR-MECC with the prevailing ones for the data on the 10Mb to 50Mb size is compared, which are delineated in Figure 3. The proposed work takes 1087ms for encryption together with 1098ms for decryption for the data size of 10 Mb. Aimed at the same size, the other works take 1945ms, 1784ms, 1541ms, 1345ms for encryption, and 1945ms, 1768ms, 1574ms, and 1325ms for decryption. The proposed one takes 1867ms, 2667ms, 3856ms, and 4589ms for encryption and 1847ms, 2667ms, 3856ms, and 4589ms for decryption while the prevailing RSA and ECC take a medium time for encryption together with decryption that is lesser when weighed against the Diffie-Hellman together with ElGamal methods. As of figure 3, it is apparent that the proposed work can endure malevolent activities with a sensible ET and DT.

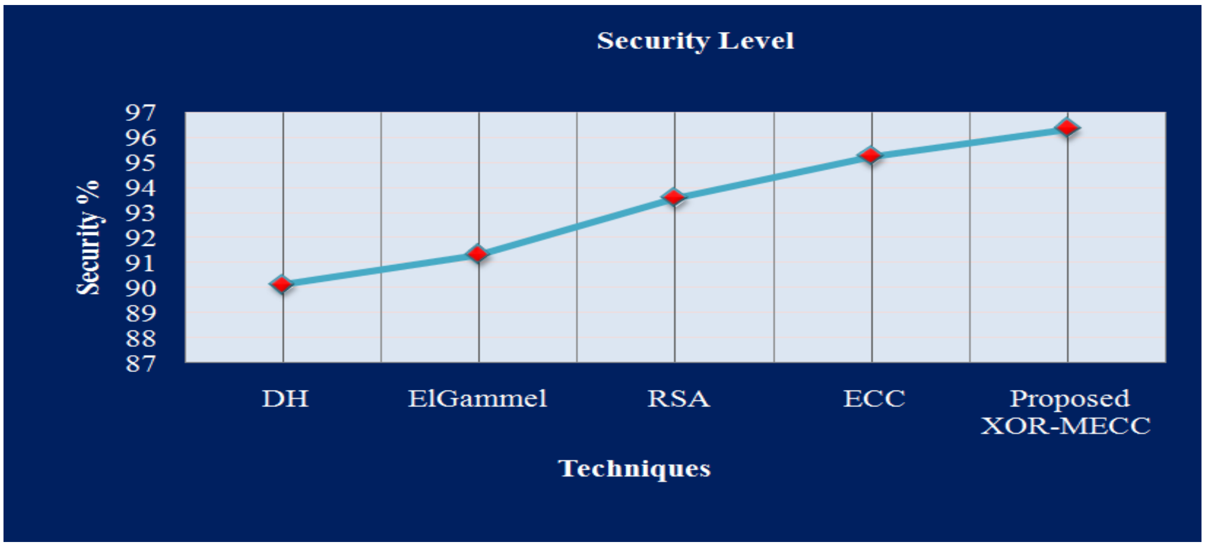
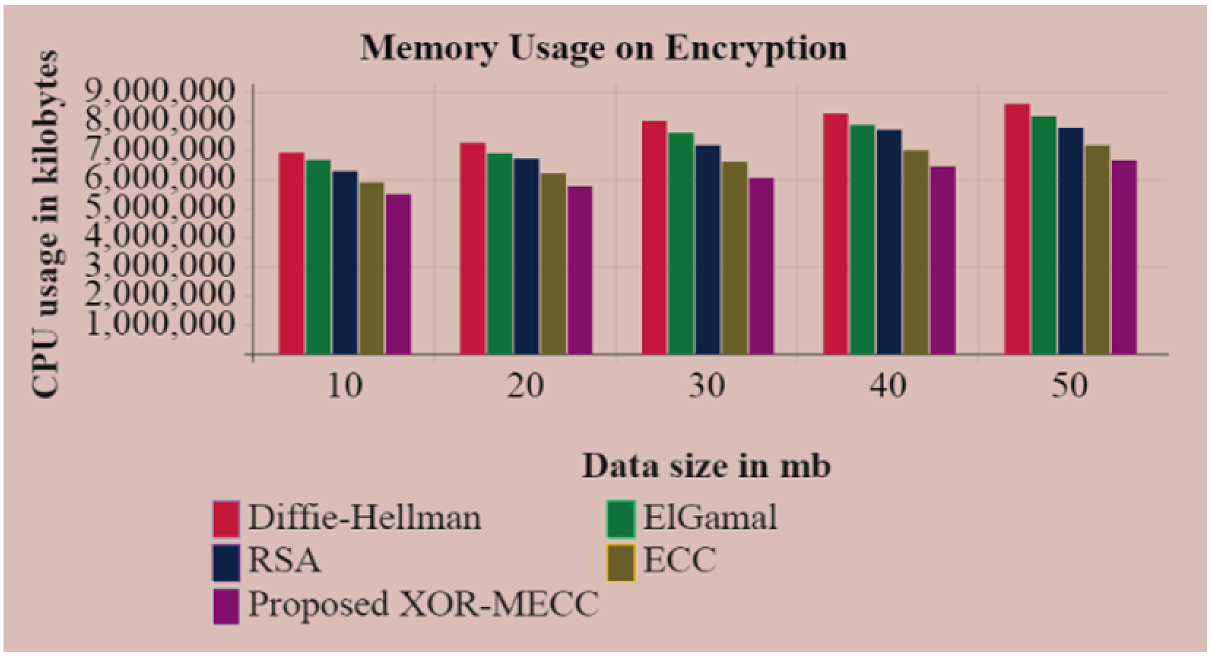
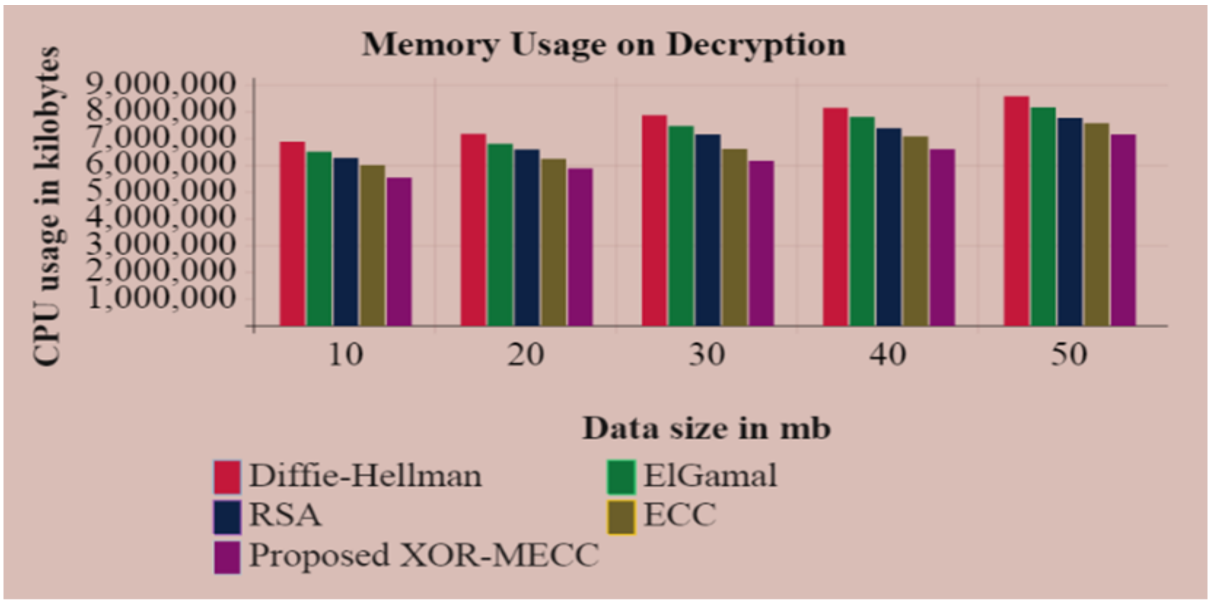


**Fig 4:** compares the SL of the proposed technique with existing techniques

**Discussion:** Figure 4 exhibit the graphical delineation of the SL for the XOR-MECC and prevailing Diffie-Hellman, ElGamal, RSA, together with ECC. The proposed XOR-MECC has an enhanced SL of 96.325678% and the existent methods have a SL of 91.324578% for ElGamal, 90.124578% for Diffie-Hellman, 93.567845% for RSA, and 95.234578% for ECC. Furthermore, the proposed work has a better SL when weighed against the existent methods.



**(a)**



**(b)**

**Fig 5:** demonstrate the (a) MU on encryption and (b) MU on decryption for the proposed and existing methods.

**Discussion**: The graphical depiction of proposed and prevailing methods for MU on encryption in addition to decryption is delineated in Figure 5. Aimed at the data sizes differs as of 10Mb to 50Mb, the proposed work has the CPU utilization of 5477445kb, 5748596kb, 6025887kb, 6425784kb, and 6635887kb on encryption and 5521142kb, 5864787kb, 6145788kb, 6578455kb, and 7122547kb on decryption. Contrasted to the XOR-MECC, the prevailing Diffie-Hellman, ElGamal, RSA, and ECC have a larger MU on encryption as well as decryption when comparing for the differing data sizes.

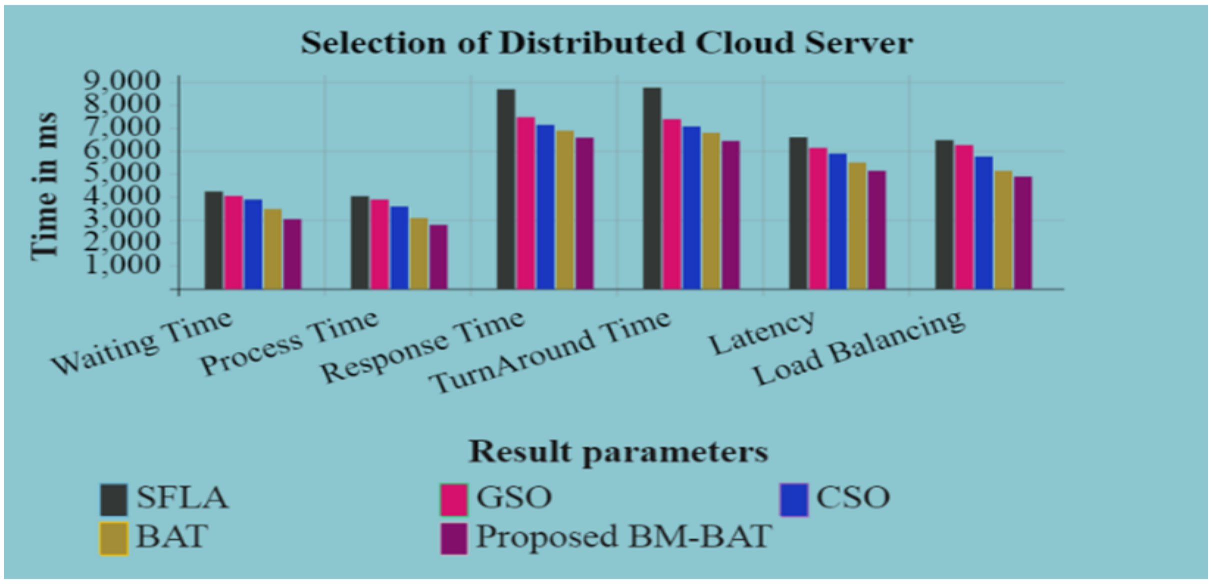


**Fig 6**: performance analysis based on the selection of distributed CS

**Discussion**: The assortment of distributed CS concerning the outcome parameters, say waiting time, response time, Latency, process time, turnaround time, together with Load balancing is exhibited in Figure 6. The techniques take a time with minor differences whilst the proposed one takes lesser time than the existent ones for the waiting time together with process time. The SFLA takes a larger time along with the proposed work has the 6574ms and 6417ms time for the assortment of distributed CS for the response time together with turnaround time. Next, the assortment time of the proposed one is 5122ms and 4875ms aimed at Latency and also loads balancing together with the prevailing techniques take a longer time contrasted with the proposed work. In figure 6, the proposed BM-BAT has a lessened time than the prevailing methods for all outcome parameters analyzed.

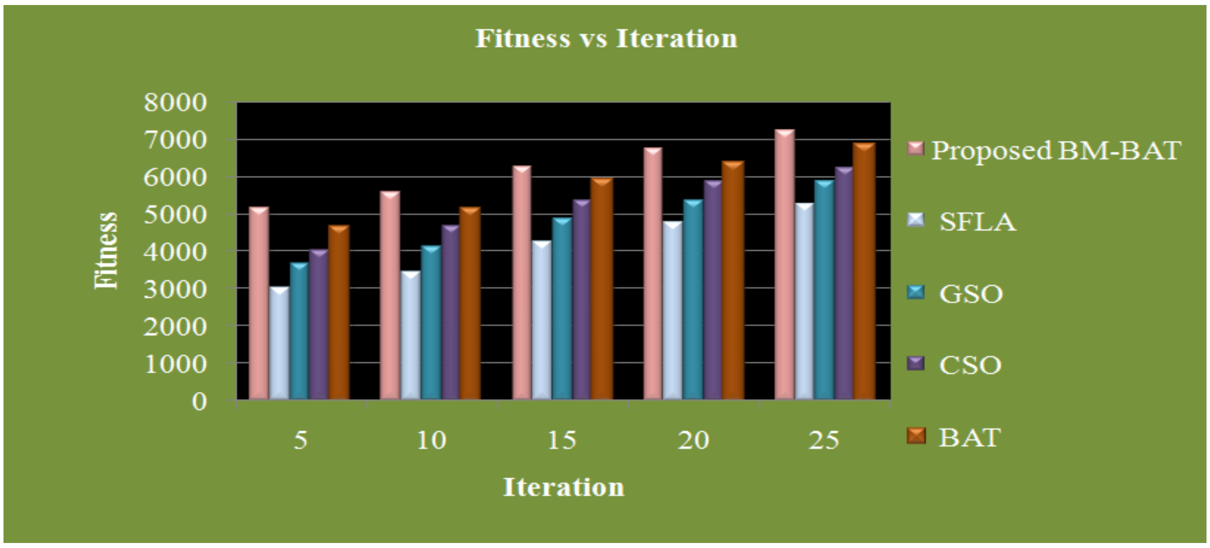


**Fig 7**: fitness vs iteration analysis

**Discussion**: The total iterations taken for analysis are differed as 5, 10, 15.20, and 25 is delineated in figure 7. In the 5 iterations, the proposed one has the fitness value of 5147 and the existent techniques have the fitness value of 3031 aimed at SFLA, 3678 aimed at GSO, 4021 aimed at CSO, together with 4623 aimed at BAT. Additional comparisons for the remaining iterations clearly exhibits that the proposed work has the fitness values of 5567 aimed at 10 iterations, 6254 aimed at 15 iterations, 6745 aimed at 20 iterations, and 7240 aimed at 25 iterations. As of the above-said discussion, it is recognized that the existent works have a fitness value lesser than the BM-BAT for the differing number iterations.

## 5. Conclusion

Users who store their data in the cloud tend to consider Cloud security as a vital aspect. They believe that their data will be safer on their local servers wherein they have more control over the data. However, CC holds a host of security risks. This paper proposes an effectual modified cryptographic for the distributed cloud storage to avert the data as of attacks and security threats. Here, 10 - 50 records are regarded for the experimental assessment of the proposed work. Contrasted with the prevailing works, the proposed one attains a better SL. The proposed BM-BAT is contrasted with the SFLA, GSO, CSO, together with BAT grounded on the assortment of distributed CS and Fitness vs Iteration. Contrary to the prevailing methods, the BM-BAT performs much better. Regarding ET, DT, SL, MU on encryption, together with MU on decryption, the XOR-MECC is examined with the Diffie-Hellman, RSA, and ECC. The XOR-MECC attains the SL of 96.325678%, which is better when weighing against the existent methods. In the future, data duplication can be commenced to lessen memory use, and a novel CS portability is commenced to shun the inconvenience of data access.

## References

[1] Mohammad HassanAmeri, MahshidDelavar, JavadMohajeri, and Mahmoud Salmasizadeh, “A key-policy attribute-based temporary keyword search scheme for secure cloud storage”, IEEE Transactions on Cloud Computing, vol.8, no. 3, pp. 660-671,2018.

[2] Anjia Yang, JiaXu, JianWeng, Jianying Zhou, and Duncan S. Wong, “Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage”, IEEE Transactions on Cloud Computing, vol. 9, no. 1, pp. 212 – 225, 2018.

[3] HuiTian, Zhaoyi Chen, Chin-Chen Chang, Yongfeng Huang, Tian Wang, Zheng-an Huang, YiqiaoCai, and Yonghong Chen, “Public audit for operation behavior logs with error locating in cloud storage”, Soft Computing, vol. 23, no. 11, pp. 3779-3792, 2019.

[4] ShengminXu, Guomin Yang, Yi Mu, and Robert H. Deng, “Secure fine-grained access control and data sharing for dynamic groups in the cloud”, IEEE Transactions on Information Forensics and Security, vol. 13, no. 8, pp. 2101-2113, 2018.

[5] Ping Li, Jin Li, Zhengan Huang, Chong-ZhiGao, Wen-Bin Chen, and Kai Chen, “Privacy-preserving outsourced classification in cloud computing”, Cluster Computing, vol. 21, no. 1, pp. 277-286, 2018.

[6] YangYang, Ximeng Liu, XianghanZheng, ChunmingRong, and WenzhongGuo, “Efficient traceable authorization search system for secure cloud storage”, IEEE Transactions on Cloud Computing, vol. 8, no. 3, pp. 819-832, 2018.

[7] WeiLuo, and Wenping Ma, “Secure and efficient data sharing scheme based on certificateless hybrid signcryption for cloud storage”, Electronics, vol. 8, no. 5, pp. 590, 2019.

[8] Bharat SRawal, VijayakumarV, GunasekaranManogaran, VaratharajanR, and Naveen Chilamkurti, “Secure disintegration protocol for privacy preserving cloud storage”, Wireless personal communications, vol. 103, no. 2, pp. 1161-1177, 2018.

[9] LibingWu, Biwen Chen, SheraliZeadally, and Debiao He, “An efficient and secure searchable public key encryption scheme with privacy protection for cloud storage”, Soft Computing, vol. 22, no. 23, pp. 7685-7696, 2018.

[10] JiantingNing, Zhenfu Cao, Xiaolei Dong, Kaitai Liang, Lifei Wei, and Kim-Kwang Raymond Choo, “Cryptcloud+: Secure and expressive data access control for cloud storage”, IEEE Transactions on Services Computing, vol. 4, no. 1, pp. 111 – 124, 2018.

[11] Ximeng Liu, Robert H. Deng, Kim-Kwang Raymond Choo, Yang Yang, and HweeHwa Pang, “Privacy-preserving outsourced calculation toolkit in the cloud”, IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 5, pp. 898-911, 2018.

[12] MahdiGhafoorian, DariushAbbasinezhad-Mood, and Hassan Shakeri, “A thorough trust and reputation based RBAC model for secure data storage in the cloud”, IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 4, pp. 778-788, 2018.

[13] AndreiTchernykh, Mikhail Babenko, NikolayChervyakov, Vanessa Miranda-López, Viktor Kuchukov, Jorge M. Cortés-Mendoza, Maxim Deryabin, NikolayKucherov, GlebRadchenko, and ArutyunAvetisyan, "AC-RRNS: Anti-collusion secured data sharing scheme for cloud storage", International Journal of Approximate Reasoning, vol. 102, pp. 60-73, 2018.

[14] Weikai Wang, LihongRen, Lei Chen, and Yongsheng Ding, "Intrusion detection and security calculation in industrial cloud storage based on an improved dynamic immune algorithm", Information Sciences, vol. 501, pp. 543-557, 2019.

[15] Yannan Li, Yong Yu, Bo Yang, Geyong Min, and Huai Wu, "Privacy preserving cloud data auditing with efficient key update", Future Generation Computer Systems, vol. 78, pp. 789-798, 2018.

[16] Imad El Ghoubach, Rachid Ben Abbou, and FatihaMrabti, "A secure and efficient remote data auditing scheme for cloud storage", Journal of King Saud University-Computer and Information Sciences, vol. 33, no. 3, pp. 1-7, 2019.

[17] Xingbing Fu, XuyunNie, Ting Wu, and Fagen Li, "Large universe attribute based access control with efficient decryption in cloud storage system", Journal of Systems and Software, vol. 135, pp. 157-164, 2018.

[18] HanQiu, Hassan Noura, MeikangQiu, Zhong Ming, and Gerard Memmi, "A user-centric data protection method for cloud storage based on invertible DWT", IEEE Transactions on Cloud Computing, vol. 7, no. 4, pp. 1-12, 2019.

[19] Sajay K. R, SuvanamSasidharBabu, and YellepeddiVijayalakshmi, "Enhancing the security of cloud data using hybrid encryption algorithm", Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 12, pp. 1-10, 2019.

[20] IrfanMohiuddin, Ahmad Almogren, Mohammed Al Qurishi, Mohammad Mehedi Hassan, Iehab Al Rassan, and Giancarlo Fortino, "Secure distributed adaptive bin packing algorithm for cloud storage", Future Generation Computer Systems, vol. 90, pp. 307-316, 2019.

[21] ManojTyagi, Manish Manoria, and Bharat Mishra, "A framework for data storage security with efficient computing in cloud", In International conference on advanced computing networking and informatics,Advances in Intelligent Systems and Computing, Springer, Singapore, vol. 870, pp. 109-116, 11-13 March 2016, New Delhi, India 2019.

[22] AbdulatifAlabdulatif, Ibrahim Khalil, and Xun Yi, "Towards secure big data analytic for cloud-enabled applications with fully homomorphic encryption", Journal of Parallel and Distributed Computing, vol. 137, pp. 192-204, 2020.

[23] Sivaram M, KaliappanM, JeyaShobanaS, VijuPrakashM, PorkodiV, VijayalakshmiK, VimalS, and SureshA, "Secure storage allocation scheme using fuzzy based heuristic algorithm for cloud", Journal of Ambient Intelligence and Humanized Computing, vol. 11, no. 5, pp. 1-9, 2020.

[24] Yongkai Fan, Xiaodong Lin, Gang Tan, Yuqing Zhang, Wei Dong, and Jing Lei, "One secure data integrity verification scheme for cloud storage", Future Generation Computer Systems, vol. 96, pp. 376-385, 2019.

[25] WentingShen, Jing Qin, Jia Yu, RongHao, Jiankun Hu, and Jixin Ma, "Data integrity auditing without private key storage for secure cloud storage", IEEE Transactions on Cloud Computing, vol. 7, no. 4, pp. 1-15, 2019.

.