Received: May 21, 2021. Revised: July 29, 2021.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Dynamic and Secure Public Auditing of User Data in Cloud by Using IRSAC

Shruthi Gangadharaiah[1]* **Purohit Shrinivasacharya[2]**

[1]*Department of Computer Science and Engineering, Siddaganga Institute of Technology, Tumakuru, India*
[2]*Department of Information Science Engineering, Siddaganga Institute of Technology, Tumakuru, India*
* Corresponding author's Email: shruthiindbit@gmail.com

Abstract: Cloud computing is gaining popularity due to its higher performance in evaluation computing such as wider availability, lower cost of service, and scalability. Cloud storage is most used in the application where the users get benefits to maintain the local storage device. However, the data integrity is more complex in cloud storage as the Cloud Service Providers (CSP) will lose their outsourced data from users in some accidents. The Third-Party Auditor (TPA) in public auditing makes an impractical assumption, which has a higher computation capability to tolerate the expensive verifications of overhead. Further, the public key infrastructure-based is audited to verify each owner's public key certification that depends on the mechanism and management of public-key certificates. In this research, proposed Improved Rivest Shamir Adleman Cryptosystem (IRSAC) algorithm for dynamic and secure public auditing of user data in the cloud. The encryption and smaller chunks formation is carried out using the Novel and Secure Cryptography-Hashing (NSCH) algorithm. When data gets uploaded, the private and public key is issued to the users to retrieve the secure data. Both the keys are obtained using the IRSAC algorithm that effectively audits the data files regularly. These experimental results showed that the proposed IRSAC method acquires lesser encryption and decryption time of 1420ms and 240ms for 1000 bits of data size. Whereas, the existing method acquired encryption and decryption time of 2210ms and 250ms respectively.

Keywords: Cloud computing, Cloud storage, Improved rivest shamir adleman cryptosystem, Novel secure cryptography hashing, Third-Party auditor.

1. Introduction

Cloud Computing (CC) has become popular in the commercial as well as academic world which offers resources like storage, computing power, networking infrastructure, and online application. [1]. currently, the data owners of the cloud are present in different types of services using Cloud Service Provider (CSP) [2]. The software is a service that is used to access the available services in the cloud. The platform as a service is used to access the requirements of higher investment. The infrastructure as a service is used by an organization to store a huge amount of data in a database [3]. The cloud storage service provides scalable storage to users or organizations for storing the data at a lower cost, which has now become popular. In cloud storage, the users can create a group and share the data easily in a group at anytime and anywhere via the internet [4]. Cloud storage reduces the burden of local storage maintenance and is currently a popular service for various data owners due to its higher scalability and larger scale storing capability. The data users utilize the higher and open scalable API to store the pictures, files, video, etc. The data users delete their original larger data after outsourcing from their server of local storage to save storage space [5]. The storage of data on cloud servers without possessing the original data will result in many security and privacy issues.

The security and privacy issue tells whether the CSP provides legal services to the user for data integration, which includes twofold reasons [6]. Initially, the data owners will not be able to check the integrity of data through traditional methods used in local storage by losing control of data. Next, the cloud storage infrastructure is highly vulnerable to external and internal security attacks because of the shared and open behavior of the cloud [7].

The private and public auditing protocol-based approaches were developed to overcome the issue of security and privacy of data in the cloud. The private auditing was performed among the users and CSP with the lesser communication. But, the two-party private auditing was not able to provide authenticated outcomes whereas the group of users and CSP will not trust each other [8]. The public auditing utilizes Third Party Auditor (TPA) to check whether the users' data are stored correctly by cloud server and establishes Service Level Agreement (SLA) among the cloud users CSP. However, the TPA makes the impractical assumption that it has a higher computation capability to tolerate the expensive verifications of overhead [9].

In public key infrastructure, the third-party auditor is utilized to check the cloud data integrity that relieved data owners and allowed public integrity of auditing for owners in a privacy-preserving way. But the public key infrastructure-based auditing has to verify every owner's public key certificate that depends on the mechanism and management of public-key certificates [10]. To solve such an issue, proposed an IRSAC algorithm for dynamic and secure public auditing of user data in the cloud [11]. The encryption and smaller chunks formation is carried out by using the NSCH algorithm. To retrieve the data when data gets uploaded, users will be given a private as well as a public key. Both the keys are generated by using the IRSAC algorithm that effectively audits the data files regularly.

The paper is organized as follows, the literature review is described in section 2, the proposed methodology is explained in section 3, the experimental results and the discussion is discussed in section 4, the conclusion of the proposed method is discussed in section 5.

2. Literature review

The existing techniques are developed to provide the data security in CC that is reviewed. Additionally, the benefits of the developed method with its limitations are explained in this section.

Zhang [12] developed the improved and secured fuzzy auditing protocol to store the cloud data. Initially, the owner of the data encodes the outsourced files with other codes and divides the files into blocks. Then, the blocks were computed by utilizing a signature or private key and the user uploads the file to the server of cloud storage. In the next phase, the aggregate messages and tags were returned to the trusted third party or data user. Finally, the data user and third party are verified the data by using a secret key. The developed protocol investigated the efficiency for data sharing with multiple user changes and supporting public integrity checking.

Nayak [13] developed a server-aided data deduplication approach to secure and efficient storage of data. The developed method supported an intra-key server and cross key server to check the duplication. Initially, the two groups satisfied the bilinear property that is initialized. The cloud service generates the key to re-encrypt that are generated by the tags. Then, generated convergent keys are checked for data duplication before uploading the files to the cloud. The major developed scheme was fixed to the size of ciphertext generation irrespective of key server numbers. The developed method was resistant to the inconsistent tag where the brute force is attacked to support the intra and cross key servers. When the number of files or number of key servers increased, the developed schemes are increased by the computation time, which leads to poor performance.

Wu [14] developed a threshold anonymous cloud that audited the scheme with multiple uploaders to preserve privacy in cloud storage. In the developed auditing scheme every user-generated the individual information from the algorithm generation tag. To respond to the challenge from the third-party auditor, the server required tag pieces from the various users among every user. In this threshold approach, the uploader was individually selected his or her key pair and generated the tag piece, therefore it is not required to trust third-party information. The computation and communication overhead of the developed method was higher than the existing techniques.

Han Qiu [15] developed a data protection approach to store in a cloud that is combined to utilize the fragmentation and dispersion process. This developed method was based on invertible discrete wavelet transform to separate agnostic information into three fragments with different level protection. The fragments were separated by various types of storage with a trustworthy approach to protecting end-user's data by stopping the leaks in the cloud. The developed method is optimized by saving cost, private, and utilized cheap storage space for security. The intensive security evaluation was performed to verify the higher level protection of the developed approach where the storage space of the developed method was lesser trustworthy.

Guohua Tian [16] developed a randomized client-side de-duplication approach to secure the data transmission in the cloud. The developed method utilized the de-duplication protocols to overcome the attack of collusive authentication to the prevented

brute force attack that is launched by the adversaries to store outside. The developed method preserved each data according to tag two files to overcome fake duplicate attacks. The improved dynamic key-encryption key tree with the flexible user space was employed to get the management of owners. According to two file tags, the data are stored to overcome the duplicate of fake attacks. Additionally, the developed method used most of the available ownership management and shares the data using a dynamic Key-Encrypting Key tree. However, some major security attacks were not studied in the developed de-duplication method.

Khalid El Makkaoui [17] developed a fast cloud-paillier method to preserve the confidentiality of data in CC. The developed method improved the security levels that introduced different methods is known as cloud paillier. The paillier scheme supported the additive homomorphic that are integrated more where the attacks are confidential. The developed variant showed speed in larger decryption when compared to the cloud paillier algorithm. However, the bit size of data was increased and encryption time also increased which resulted in performance degradation.

3. Proposed methodology

The purpose of storing the data and dynamic as well as secure public auditing consists of 3 entities by a network of cloud. In this research, the users are considered as entities where it maintains larger files of data computations, organization, and individuals. The CSP manages a Cloud Storage Server (CSS) or cloud computing system which includes more space to the resource and storage for maintenance and computation of data with clients and TPA. The request of clients can verify the data integrity which is stored in the remote server based on a trusted approach by exposing and assessing the risk in CSS. When data gets changed, the clients able to detect the changed part in the data and also data integrity checked by verifiers. Finally, the data need to be stored securely and private, privacy is biggest threats in cloud network as the access of stored information is available to administrators.

In the proposed dynamic and public auditing approach, the users choose the file of data that is required to the uploaded server of the cloud. Before uploading, the user requested the private key, as well as the public key, is to download and upload the data. During the phase of key generation, the public and private keys are generated using the IRSAC algorithm. To improve the storage and security utilization, the whole file of data is encrypted which is dynamically divided into various smaller chunk files using the NSCH algorithm. When the cloud servers divide the entire files among chunks of smaller files, the hash value of each smaller file is calculated. The unique name will be assigned to every smaller file in such a way where the attackers, cloud servers, and TPA are not able to obtain unique names for smaller files. The integrity of data is determined by utilizing the Multi-Level Hash Tree (MLHT) algorithm and identifies the integrity of files in various servers in a single time. The system architecture of the proposed IRSAC for dynamic and secure public auditing in the cloud is shown in Fig. 1.

3.1. Key generation

The RSA cryptosystem includes three types such as generation of encryption, decryption, and prime key. The RSA algorithm will not provide security for the attacks such as brute force and security of RSA dependent on larger prime numbers because of difficulty in breaking. Therefore, the improved RSA is proposed securely to generate the key as well to establish a private and public key, which is known as the IRSAC algorithm.

In the proposed IRSAC algorithm, the generate keys are preserved in the file which utilizing the base64 encoding approach where the values are stored securely in the cloud. The proposed IRSAC key generation includes the utilization of two random numbers and two prime number. The encryption values are dependent on values of public key components (N) and the decryption values are dependent on a private key component (M). The value of N is equal to the product of two prime and random numbers. The value of M is equal to the product of two prime numbers. The process of encryption and decryption is carried out using modulus values such as (e, N) which is required to encrypt the data.

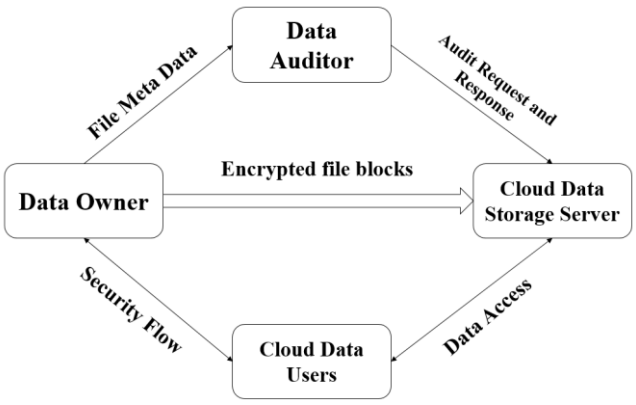

Figure. 1 The system architecture of proposed IRSAC method for dynamic and secure public auditing in cloud

The M value is not dependent on the N value where the attackers know the details of N that are not able to identify the prime number and cannot determine the(d, M)values of decryption. The size of prime numbers includes similar sizes and the keys that are generated by utilizing base64 encoding to store it securely. By using components of public-key encryption is carried out and a private key is used to decrypt the data.

Encryption:

1. In the process of encryption, it obtains the components of a public key(n, e).
2. Representing the plain texts of a message as a positive integer m.
3. Computed the ciphertext $c = m^e \ mod \ n$, m^e is encrypted message.
4. Forwarded the ciphertext c to a user.

Decryption:

1. In the process of decryption that utilizes the user's private key (n, d).
2. Computed the message $m = c^d \ mod \ n$.
3. Extracted the plain texts from the represented message m.

The algorithm of the proposed IRSAC for key generation, encryption and decryption is explained below.

The IRSAC algorithm for key generation:

Input

Two prime numbers such as p, q and $p \neq q, p, q > 3$

Where, $p \ and \ q$ are the prime numbers considered as input.

Output

The components of public key $\{e, N\}$

The components of private key $\{d, M\}$

Where, $\{e, N\}$ and $\{d, M\}$ are the encrypted and decrypted components of public and private key.

Procedure

$N \leftarrow p \times q \times (p-1) \times (q-1)$

$M \leftarrow p \times q$

The random integers are selected $r, p > 2^r < q$

Compute $\emptyset$ value of N

$\emptyset(N) \leftarrow \frac{(p-1)\times(q-1)\times(p-2^r)\times(q-2^r)}{2^r}$

Selecting a random number e, in such a way that $1 < e < \emptyset(N)$ and $\gcd\big(e, \emptyset(N)\big) = 1$

Computing the random number d, in such a way that

$d * e = 1 * mod(\emptyset(N))$

Where N and M are the public and private key components.

The IRSAC algorithm for encryption

Input

Enter a plain text message, $P < N$

Enter a public key components: {e,N}

Output

Cipher text, C

Procedure

Cipher text, $C \leftarrow P^e \ mod \ N$

Where, P is plain text, {e,N} is encrypted public key components, C is cipher text, P^e is encrypted plain text.

The IRSAC algorithm for decryption

Input

Enter a cipher text message,C

Enter a private key components: $\{d, M\}$

Output

Decrypted plain text,P

Procedure

$$P \leftarrow C^d \ mod$$

Where, C is cipher text, $\{d, M\}$ is the private key components, C^d is the decrypted cipher text.

3.2. Novel secure cryptography hashing (NSCH)

The cryptography hashing algorithm is used for creating the message that digests the fixed lengths of bit strings for arbitrary lengths of data. The whole file of data is encrypted which is dynamically divided into various smaller chunk files using the NSCH algorithm. The hash functions are taken as tools for digital signature, time-stamping and checking the integrity of a message. The outsourced data are assured to know whether the information is modified by irrelevant users. The NSCH algorithm is utilized for the encryption of encrypted data files of data are split into smaller chunk files. After encrypting the data, it assigns and creates the name for every block, which is created from the efficient and optimal encrypted files of data. The name is assigned to the blocks in such a way that the users, cloud server and attackers are not able to identify the assigned name. The algorithm of NSCH is explained as follows:

The NSCH algorithm,

Input

Messages with arbitrary length

Output

The input message with a hash value of 128 bit.

Procedure

Step1: To enter the input string in general form, the random strings are generated.

((a-z) + (A-Z) + (0-9) + (!@#$%^&*()_ + \– = \[\]{};: \\|, . <> \/?) + (Sunday-Saturday) + (0-9))

Where, + indicates the positions in random strings

Step2: Select and count the total alphabet in random strings

Step3: Identify the secret values by considering below assumptions in order to enhance the randomness.

$$if\ alphabet\ count = 10, secret\ value = ascii\ value\ of\ uppercase\ letter\ for\ first\ position\ of\ character + 1.$$

$$if\ alphabet\ count > 10\ and\ < 20, secret\ value = \begin{pmatrix} ascii\ value\ of\ uppercase\ letter\ for \\ first\ position\ of\ character - \\ ascii\ value\ of\ A \end{pmatrix} + \begin{pmatrix} ascii\ value\ of\ uppercase\ letter\ for\ second \\ position\ of\ character - ascii\ value\ of\ A \end{pmatrix} + 2.$$

$$if\ alphabet\ count \geq 20\ and\ < 30, secret\ value = \begin{pmatrix} ascii\ value\ of\ uppercase\ letter\ for \\ first\ position\ of\ character - \\ ascii\ value\ of\ A \end{pmatrix} + \begin{pmatrix} ascii\ value\ of\ uppercase\ letter\ for\ third \\ position\ of\ character - ascii\ value\ of\ A \end{pmatrix} + 2.$$

$$if\ alphabet\ count \geq 30, secret\ value = \begin{pmatrix} ascii\ value\ of\ uppercase\ letter\ for \\ second\ position\ of\ character - \\ ascii\ value\ of\ A \end{pmatrix} + \begin{pmatrix} ascii\ value\ of\ uppercase\ letter\ for \\ fourth\ position\ of\ character - \\ ascii\ value\ of\ A \end{pmatrix} + 2.$$

Step4: Look Table need to be considered for finding the position of alphabet as shown in Table1,

a. To choose each character from the input strings.
b. The present position = Identifying the positions from Table1 +Secret value %26.
c. The present alphabet= The present positions equal value from Table1.

Step 5: a) To calculate the binary value of strings for each character by using characters ASCII value.

b) Evaluate the binary value of string for special characters and numbers by utilizing ASCII value.

Step 6: According to the random strings sequence following conditions need to be satisfied:

To transform the string length into multiple 128 bits and if it is not strung append 0.

To scan the binary strings from step 5 the append 64 bits beginning from the location.

Step 7: a) The output strings from the previous step is converted into a block of 128 bit.

b) The binary key with 128 bits is generated by utilizing a random key generator.

c) To perform the bitwise operation like XOR, OR, AND, left-shift, right-shift, etc, for 12 bit of block string and 128 random key strings.

d) Finally, the output saved from bitwise operations

Step 8: Transform the output of step 7 into the character of values and preserve the final version of 128 block

Step 9: The final block messages and random keys of 128 bits are combined by utilizing bitwise operation to obtain the final digest message.

The user is required to provide the name of the file where the number of smaller chunks created with correct values to download the files. By utilizing the file name and smaller chunks values, the meta tags are created and forwarded to the data owner which can download the files from the cloud server using the values of meta tags. The users forward the meta-tags and name of a file to TPA for the process of public auditing. To check the respective data files, the TPA gets file names and smaller chunks of a file are verified by the data owners. By utilizing the meta tags values, TPA verifies the data file without the actual content of the files in the data. The smaller chunks of data file storage decrease the issue of fragmentation which enhances the resource utilization efficiency in a cloud server. The CSP handles the process of storing and fetching the data, but users are not able to identify the difficulty of the mechanism. The secured hash function describes the name to a smaller chunk of files and avoids CSP as well as irrelevant users to know the details about the smaller chunk files. The secured hash function is calculated based on random manners which confuse the irrelevant users as the CSP avoiding them to identify the blocks.

Table 1. The look up Table for alphabet positions

A	0	G	6	M	12	S	18	Y	24
B	1	H	7	N	13	T	19	Z	25
C	2	I	8	O	14	U	20		
D	3	J	9	P	15	V	21		
E	4	K	10	Q	16	W	22		
F	5	L	11	R	17	X	23		

3.3. Multi-Level hash tree

The auditing of data is carried out by utilizing the MLHT algorithm, which helps the Merkle Hash Tree (MHT) algorithm to efficiently identify the integrity of data files among the multiple servers. The MHT is a binary tree in which every leaf node stores the data files and the data item in the node. The collision resistance hash function h_{CR} is utilized to label the intermediate nodes in a tree. Every output of h_{CR} with different input includes a binary string with the length of $O(\lambda)$. The labels of intermediate node v to the output h_{CR} are calculated based on the labels of children nodes v. The Merkle hash tree is efficiently utilized as a standard tool to check the memory. Fig. 2 shows the Merkle hash tree which contains the data items [18].

The MHT contains the data items such as $\{d_1, d_2, \ldots, d_8\}$ that stored in leaf nodes. Similarly, the labels of intermediate nodes are calculated by utilizing h_{CR} hash functions where the hash values of a node are known as root digest. This proof shows that data files d is present in a tree which includes d data files of nodes with the associated paths such as a sequence of siblings in the nodes that contains data files d. Example: The d_3 is present in the tree of $\{d_3, (d_4, ID, IC)\}$, where, d_4,ID and IC are the labels of K, D and C nodes. The verifier calculates the value of hash for root and the output will be accepted if the calculated value of hash value matches with root, otherwise, the root will be rejected.

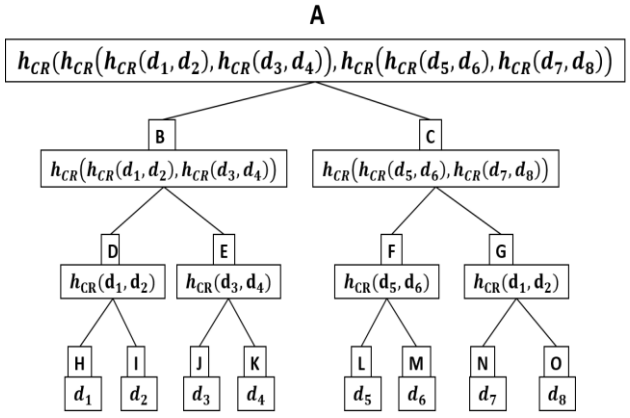

Figure. 2 The data items are included in the Merkle hash tree [18]

The proof size will be the logarithmic numbers to the data files stored in the leaf node of a tree. The property of collision resistance of h_{CR} is infeasible to add or change the data files in the MHT without modifying the root.

To avoid privacy leakage of user data, the advance calculation on the blocks of user data is carried out without affecting the features of TPA. It provides the construction of a novel efficient method that blinds every user block of data in the prior stages to prevent privacy leakage in the process of public auditing and to provide better privacy protection than existing methods. It generates the random parameters in the generation of key stage and utilizes the blind metadata which includes blocks of data. The users forward the parameters to the server when TPA sends a request message to the data auditing to a server, the server estimates the proof by using metadata, blocks of data, and random parameters which returns the proof for TPA. The proof received and public key generated by users are used by TPA to auditing the integrity of data. This simple and effective approach solidifies the privacy protection of users and reduces the privacy leakage probability as much as possible during the process of auditing. The reduction in the process of auditing and parameters effectively stops the TPA to know about the data users. The developed method has the advantage to overcome the problem of privacy leakage and maintains the required TPA auditing capability without incurring computational time.

4. Experimental result and discussion

The experimental results of the proposed IRSAC method for dynamic and secure public auditing in the cloud are described in this section. The validation of the proposed IRSAC method is carried out using the larger data files that are required to be stored in the servers of the cloud. The input plain text needs to be given as a console input or dataset and the proposed IRSAC algorithm calculate the output of message digest. In this work, the proposed work is compared with the benchmark models to secure the public auditing of the data cloud. This proposed IRSAC

method is simulate using python 3.6 software in an anaconda navigator with system requirements such as the windows 10 operating system, RAM of 128GB, Intel Core i9 processor, and hard disk of 4 TB. The parameters are considered to evaluate the proposed IRSAC method to the dynamic and secured public auditing which is explained in this section. The quantitative and comparative analysis of the proposed IRSAC method with the existing method is also described in this section.

4.1. Performance metrics

The proposed IRSAC method for dynamic and secure public auditing of user data in the cloud is evaluated and compared with the existing methods. The parameters such as encryption, decryption, and execution time are considered to analyze the effectiveness of the proposed ISRAC approach with existing methods that are explained as follows:

- **Encryption time:** The encryption time is utilized to estimate the throughput of an encryption model that indicates the speed of encryption. The throughput of the encryption model is calculated using encrypted of overall plaintext in bytes which is divided by the encryption time.
- **Decryption time:** The time taken to convert the encrypted data into its original format is called decryption time. Generally, the reverse process of encryption which decodes the encrypted data that authorized the user will be able to decrypt using a secret key.
- **Execution time:** The execution time is also called Central Processing Unit (CPU) time that is spent by the system to execute the task or it is considered as how much time it took for the program to complete or terminate.

The obtained results of the proposed IRSAC method for dynamic and secure public auditing of data in the cloud in terms of encryption time, decryption time and execution time is tabulated in Table 2. It depicts the amount of time required to perform the task considering the amount of data in a millisecond.

Table 1. The quantitative analysis of the proposed IRSAC method in terms of encryption-decryption and execution time.

Data size(in bits)	Encryption-Time (ms)	Decryption-Time (ms)	Execution-Time (ms)
100	1050	180	1250
200	1070	190ss	1290
300	1100	195	1310
400	1150	205	1370
500	1200	215	1430
600	1250	220	1500
700	1300	230	1530
800	1350	235	1600
900	1400	240	1670
1000	1420	240	1690

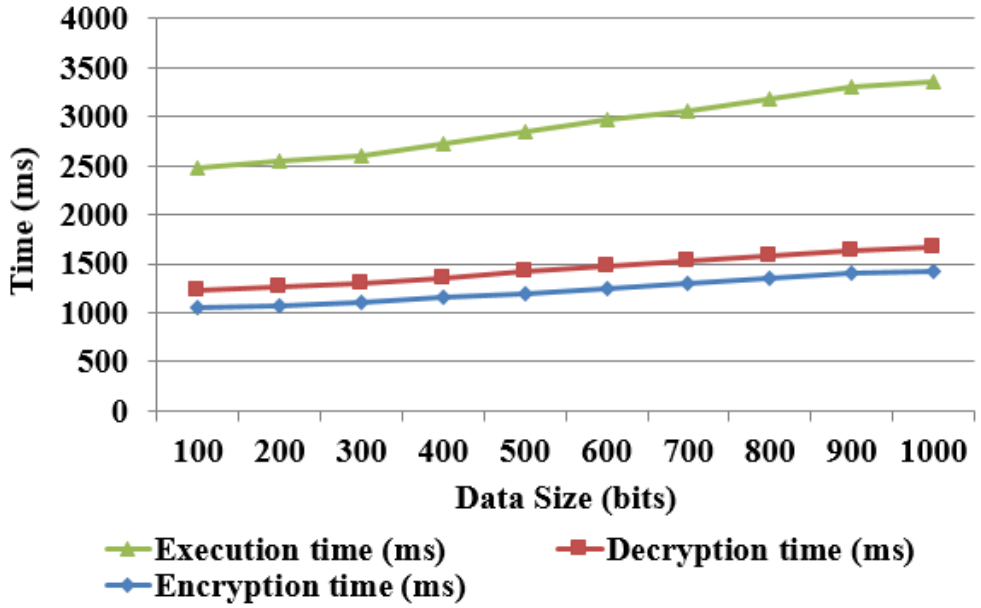

Figure. 3 The graph of the proposed IRSAC method for encryption, decryption and execution time.

Table 2 shows the quantitative analysis of the proposed IRSAC method in terms of encryption time, decryption time, and execution time for dynamic and secured public auditing of user data in the cloud. These outcomes are obtained for 10 executions with the different data sizes from 100 to 1000 in bits. The time taken by the proposed IRSAC method for encrypting the data files is 1420ms for 1000 data size in bits. The time taken by the proposed IRSAC method is decrypting the encrypted data is 240ms for 1000 data size in bits. The time taken by the proposed IRSAC method for executing the complete task is 1690 ms for 1000 data size in bits. The pictorial representation of the proposed IRSAC method in terms of encryption time, decryption time and execution time are shown in Fig. 3.

4.2. Comparative analysis

The obtained results of the proposed IRSAC method are compared with the existing method such as Fast-Cloud-Paillier [16] and Cloud-Paillier [17]. The dynamic and secure public auditing of user data in the cloud is carried out with different sizes from 100 to 100 in bits in terms of encryption time and decryption time. The encryption time comparison with existing methods is shown in Table 3.

Table 3 shows the encryption time of the proposed IRSAC method with the existing Cloud-Paillier and Fast-Cloud-Paillier for public auditing the data in the cloud. The encryption time of the secure public auditing of user data files in cloud approaches increases as the size of data bits increases. The proposed method acquired lesser time to encrypt the data than the existing methods, which is suitable for providing the confidentiality of users' sensitive data. The proposed method acquired an encryption time of 1420 ms for the data size of 1000 in bits. Whereas, the existing Cloud-Paillier and Fast-Cloud-Paillier acquired 1500 ms and 2210 ms for the data size of 100 in bits.

Table 3. The comparison of encryption time with existing methods for different data sizes.

Data Size(bits)	Encryption comparison time(ms)		
	Cloud–Paillier [17]	Fast Cloud–Paillier [16]	Proposed IRSAC Method
100	1150	1300	1050
200	1210	1380	1070
300	1250	1420	1100
400	1300	1500	1150
500	1310	1580	1200
600	1360	1620	1250
700	1400	1800	1300
800	1420	1900	1350
900	1450	2180	1400
1000	1500	2210	1420

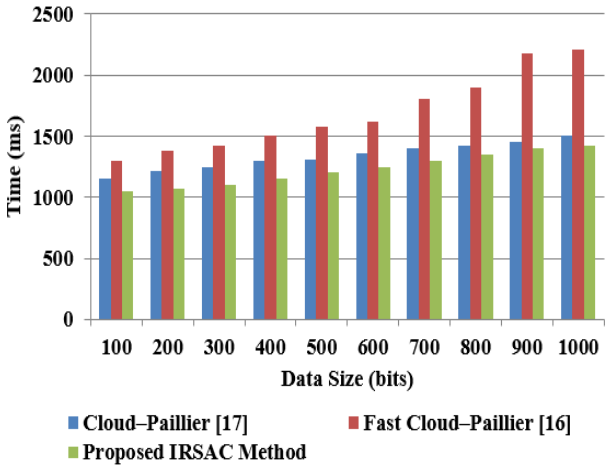

Figure. 4 The comparison graph of the proposed IRSAC method with existing method in terms of encryption time.

Table 4. The comparison of decryption time with existing methods for different data sizes

Data Size (bits)	Decryption comparison time (ms)		
	Cloud–Paillier [17]	Fast Cloud–Paillier [16]	Proposed IRSAC Method
100	2200	250	180
200	2000	250	190
300	2200	250	195
400	2200	250	205
500	2100	250	215
600	2000	250	220
700	2100	250	230
800	2100	250	235
900	2200	250	240
1000	2300	260	240

Table 5. The decryption comparison time with key generation and key size for proposed IRSAC method

Decryption comparison time(ms)		
Key size	**Key generation time**	**Proposed IRSAC Method**
8	126	190
16	130	190
32	140	190
64	148	195
128	162	205
256	168	210
512	174	225
1024	195	240
2048	210	255

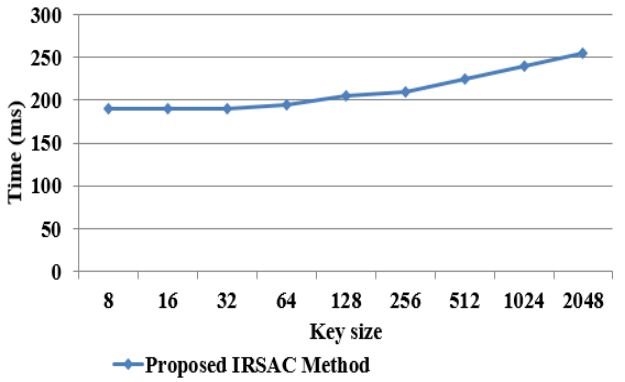

Figure. 5 The graphical representation of proposed IRSAC decryption time with key size

In existing Cloud-Paillier [17], the bit size of data was increased and encryption time also increased which resulted in performance degradation. Further, the computation and communication overhead of the developed method was higher. The proposed IRSAC method solved the problem of existing methods by achieving higher performance. The graphical representation of the comparison of encryption time for the proposed IRSAC method with the existing method is shown in Fig. 4.

The decryption time of the secure public auditing of user data files in cloud approaches increases as the size of data bits increases. The proposed method acquired lesser time to decrypt the encrypted data than the existing methods is suitable to provide the confidentiality of users' sensitive data. The proposed method acquired the decryption time of 240ms to the data size of 1000 in bits. Whereas, the existing Cloud-Paillier and Fast-Cloud-Paillier acquired 2300 ms and 260 ms for the data size of 100 in bits. The graphical representation and comparison of decryption time for the proposed IRSAC method with the existing method is shown in Fig. 5.

5. Conclusion

To provide dynamic and secure public auditing of user data in the cloud, an IRSAC algorithm is proposed by encrypting and decrypting the data securely. The public and private keys are generated using the IRSAC algorithm to enhance the utilization of security and storage. The entire file of data is encrypted and divided into various chunks of files using the NSCH algorithm. When the cloud servers divide the entire files among chunks of smaller files, the hash value of each smaller file is calculated. The unique name will be assigned to every smaller file in such a way that the attackers, cloud servers, and TPA are not able to determine the name for smaller files. The integrity of data is calculated by utilizing the MLHT algorithm and checks the integrity of files in multiple servers in a single time. The proposed method ensures that it is secure for public auditing of user data in the cloud and consumes lesser time for decrypting the encrypted data. The proposed IRSAC method acquired lesser encryption and decryption time of 1420ms and 240ms for 1000 bits of data size. But, existing method showed encryption and decryption time of 2210ms and 250ms. In future work, the batch wise auditing can be carried out which reduces the processing cost and provides interface for user in order to view the details of file.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

The paper background work, conceptualization, methodology, dataset collection, implementation, result analysis and comparison, preparing and editing draft, visualization have been done by first author. The supervision, review of work and project administration, have been done by second author.

References

[1] H. Tian, F. Nan, H. Jiang, C.C. Chang, J. Ning, and Y. Huang, “Public auditing for shared cloud data with efficient and secure group management”, *Information Sciences*, Vol. 472, pp. 107-125, 2019.

[2] E. Daniel, and N.A. Vasanthi, “LDAP: a lightweight deduplication and auditing protocol for secure data storage in cloud environment”, *Cluster Computing*, Vol. 22, No. 1, pp. 1247-1258, 2019.

[3] A. Kalaivani, B. Ananthi, and S. Sangeetha, “Enhanced hierarchical attribute based encryption with modular padding for improved public auditing in cloud computing using semantic ontology”, *Cluster Computing*, Vol. 22, No. 2, pp. 3783-3790, 2019.

[4] J. Xue, C. Xu, J. Zhao, and J. Ma, "Identity-based public auditing for cloud storage systems against malicious auditors via blockchain", *Science China Information Sciences*, Vol. 62, No. 3, pp. 32104, 2019.

[5] H. Yu, X. Lu, and Z. Pan, "An Authorized Public Auditing Scheme for Dynamic Big Data Storage in Cloud Computing", *IEEE Access*, Vol. 8, pp. 151465-151473, 2020.

[6] J. R. Gudeme, S. K. Pasupuleti, and R. Kandukuri, "Attribute-based public integrity auditing for shared data with efficient user revocation in cloud storage", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 12, No. 2, pp. 1-14, 2020.

[7] K. Fan, M. Liu, G. Dong, and W. Shi, "Enhancing cloud storage security against a new replay attack with an efficient public auditing scheme", *The Journal of Supercomputing*, Vol. 76, No. 7, pp. 4857-4883, 2020.

[8] R. Mishra, D. Ramesh, and D. R. Edla, "BB-tree based secure and dynamic public auditing convergence for cloud storage", *The Journal of Supercomputing*, pp. 1-40, Vol. 77, No. 5, 2020.

[9] X. Zhang, H. Wang, and C. Xu, "Identity-based key-exposure resilient cloud storage public auditing scheme from lattices", *Information Sciences*, Vol. 472, pp. 223-234, 2019.

[10] J. Zhao, C. Xu, and K. Chen, "Detailed analysis and improvement of an efficient and secure identity-based public auditing for dynamic outsourced data with proxy", *Journal of Information Security and Applications*, Vol. 47, pp. 39-49, 2019.

[11] M. K. A. Venkataramaiah, and N. A. N. Achar, "Twitter Sentiment Analysis using Aspect-based Bidirectional Gated Recurrent Unit with Self-Attention Mechanism", *International Journal of Intelligent Engineering and Systems*, Vol. 13, No. 5, pp. 97-110, 2020.

[12] J. Zhang, B. Wang, D. He, and X. A. Wang, "Improved secure fuzzy auditin[g protocol for cloud data storage". Soft Computing, vol. 23, no. 10, pp. 3411-3422, 2019.

[13] S. K. Nayak, and S. Tripathy, "SEDS: secure and efficient server-aided data deduplication scheme for cloud storage", *International Journal of Information Security*, Vol. 19, No. 2, pp. 229-240, 2020.

[14] G. Wu, Y. Mu, W. Susilo, F. Guo, and F. Zhang, "Threshold privacy-preserving cloud auditing with multiple uploaders", *International Journal of Information Security*, Vol. 18, No. 3, pp. 321-331, 2019.

[15] H. Qiu, H. Noura, M. Qiu, Z. Ming, and G. Memmi, "A user-centric data protection method for cloud storage based on invertible DWT", *IEEE Transactions on Cloud Computing*, 2019.

[16] G. Tian, H. Ma, Y. Xie, and Z. Liu, "Randomized deduplication with ownership management and data sharing in cloud storage", *Journal of Information Security and Applications*, Vol. 51, pp. 102432, 2020.

[17] K. El Makkaoui, A. Ezzati, A. Beni-Hssane, and S. Ouhmad, "Fast Cloud–Paillier homomorphic schemes for protecting confidentiality of sensitive data in cloud computing", *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-10, 2019.

[18] N. Garg, and S. Bawa, "RITS-MHT: relative indexed and time stamped Merkle hash tree based data auditing protocol for cloud computing", *Journal of Network and Computer Applications*, Vol. 84, pp. 1-13, 2017.