International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-12, October 2019

Markle Hellman Knapsack Crypto-System Auditing Scheme for Privacy Preserving in Cloud Storage

N. Gowtham Kumar, P. Niranjan, D. Aruna Kumari

Abstract*: Cloud computing is an on demand paradigm which provides a different kind of services to the cloud users. Cloud storage is the most popular service, as the data owners are free from the data management and storage overhead. However, the data owners' concern about the security of the data. In order to address this issue, this paper presents an efficient security with an auditing scheme that guarantees the security of the data and preserve data integrity. In this paper, the cloud storage auditing model used efficient privacy preserving algorithm, namely Markle Hellman Knapsack Crypto-System (MHKCS) algorithm. This algorithm effectively improves the data integrity, confidentiality and security. Moreover, reduces the key generation time, encryption time and decryption time. The performance of MHKCS algorithm is calculated using evaluation metrics like encryption time, decryption time, key generation time and communication cost. The MHKCS algorithm achieved approximately 10% better performance in terms of encryption time than the existing methods RSA, MRSA, and MRSAC.

Index Terms*: Auditing system, Cloud Storage model, Data Integrity, Markle Hellman Knapsack Crypto-System, Rivest-Shamir-Adleman.

I. INTRODUCTION

In present days, cloud computing is an emerging paradigm that rapidly intensifies in the numerous areas such as Internet of Things, e-commerce, scientific research etc. [1]. The major responsibility of cloud is to share the data computations over scalable network nodes, namely user computers, cloud services, and data centers [2]. The cloud platform provides a numerous service such as software-as-a-service (Saas), Platform-as-a-service (PaaS), Infrastructure-as-a-Services. Likewise, cloud storage is a significant service in cloud computing and it facilitates to provide the accessibility, deployment, security, etc. [3]. There is no need to worry about software and hardware failures, but security is the major issue. The malicious users may retrieve, corrupt or steal cloud users' data and eliminate the confidentiality, integrity and availability of data [4].

Revised Manuscript Received on October 30, 2019.
***** Correspondence Author
N Gowtham Kumar*, Department Computer Science and Engineering, KL (Demmed to be University), Vijayawada, India.
P. Niranjan, Department Computer Science and Engineering, Kakatiya Institute of Technology and Science, Waranagal, India.
D. Aruna Kumari, Department of Electronics and Computer Engineering, KL (Deemed to be University), Vijayawada, India.

Although, the security of data integrity in the cloud is a challenging task for cloud users with constrained computing resources. In cloud, users cannot audit their stored data hence, Third Party Auditor (TPA) used for cloud data auditing [5, 6]. An essential property of the auditing process is (i) confidentiality (ii) dynamic auditing and (iii) batch auditing [7]. The data integrity is one kind of significant security problem because CSP may loss the data due to data leakage problem. [8]. To rectify these problems several clouds auditing schemes such as Third Party Medium [9], Identity based data out sourcing [10], Renaissance system model [11], etc. are used.

Recent research works recommends that a main obstacle in cloud is the lack of cloud auditability. The auditability framework performs better in cloud but several issues like data privacy, security, data loss [12], transparency, and portability [13] are present. In this paper, an effective cloud storage-auditing model is proposed. This model consists of three major entities such as data owner, TPA, cloud server. In order to improve the data security, MHKCS algorithm is proposed. This algorithm employs public key for encryption and private key for decryption process. The MHKCS algorithm generates the particular private and public keys for each input data files. As a result, it improves the security of cloud storage data and prevent the data access from unauthorized users.

This paper is composed as follows. Section II survey several recent papers on secure cloud storage auditing strategies. In section III, an effective cloud storage system includes MHKCS algorithm for key generation, encryption and decryption process. Section IV shows comparative experimental result for proposed and existing cryptographic strategies. The conclusion is made in section V.

II. RELATED WORKS

Many research techniques are suggested by researchers in the field of ensuring secure cloud data auditing in cloud storage system. Several recent researches in the respective area are studied in this section.

S. Anbuchelian, et al. [14] presented a secure cryptographic hash algorithm used to encrypt the data and split the data into many chunks of files. The method developed a new algorithm known as Modified RSA Cryptosystem (MRSAC), which was an enhancement of RSA key generation algorithm.

A key was generated and given to the cloud user for the trusted retrieval of the data from the cloud server. The third party auditor checked the integrity of the data using a multilevel hash tree algorithm. The experimental results showed the efficiency of the method. This method suffered from the high cost of processing the encryption.

Jia Yu, et al. [15] presented an efficient cloud storage auditing with outsourcing key update strategy. The key updation process was very clear to the cloud users and users know all the information of key updates. This method updates the key from outsourced to TPA very securely. Therefore, key updating process reduces the burden of client side and TPA only holds the encryption secret key while performing the entire task on behalf of the users. The client can only download the secret key from the TPA when uploading new files to the cloud. The security proof and the performance simulation showed that the design was secure and efficient.

T. Xiang, et al. [16] presented an efficient auditing mechanism namely dynamically Adjustable-capacity Cuckoo Filter (DACF) for privacy preserving outsourced database without the aid of TPA. This method sustained flexible data dynamics and partial attribute retrieval and save communication overhead. This proposed method attained high security, but computational complexity was maximized.

D. Kim, et al. [17] presented a public auditing protocol for educational multimedia data outsourced in the cloud storage. The method ensured data privacy in the cloud and the TPA by using random values and a homomorphic hash function. The method is secure against lose attack and temper attack. The fully dynamic auditing was supported by the protocol. The security and performance study results showed that the scheme was secure while guaranteeing minimum extra computational costs. The communication cost between the user and the TPA is high because the protocol is needed to ensure the security.

H. Tian, et al. [18] proposed a novel strategy namely Dynamic Hash Table (DHT) for public auditing that enhance the cloud storage security. This scheme validates the authorized data from CSP to TPA. Moreover, the benefit of DHT algorithm is less communication cost and communication overhead. In this literature, the key generation process was performed by random masking strategy. Through experiment, the proposed scheme proved that it is an efficient cloud auditing process in cloud storage, but while auditing process data may loss.

An efficient encryption algorithm (MHKCS) is implemented to overcome the above-mentioned drawbacks and to improve the auditing process with maximum cloud storage security.

III. PROPOSED METHODOLOGY

Data security in cloud is one of the significant issues with cloud storage facility. All the clients store their data in cloud. This section addresses the proposed cloud storage-auditing model MHKCS algorithm for improving security of the cloud storage. In this design, cloud storage auditing model generates the client's secret key and verify the cloud storage for ensuring proper placement of data by the client. The cloud storage model consist of three major entities such as data owner, TPA, and cloud server. Those are explained in the following sections.

A. Cloud Storage Model

A number of users use cloud storage to store the data to avoid spending more money on local hardware/software deployment and data maintenance. The cloud is accessible at any time and it is more useful for users to store or share the data universally. Moreover, cloud computing suffers from several security issues like, integrity checking for cloud data, keyword search over encrypted cloud data, etc. Hence, cloud storage auditing process is utilized to check the integrity of cloud data. Data owners may perform the cloud storage auditing process, but computation overhead may increase. Therefore, TPA is introduced to rectify this problem and to perform integrity of cloud data. In case TPA started the auditing process without data owner's permission, then computation resources in auditing process increase. In order to eliminate this problem, the proposed cloud storage model is proposed and it is shown in the Fig. 1.

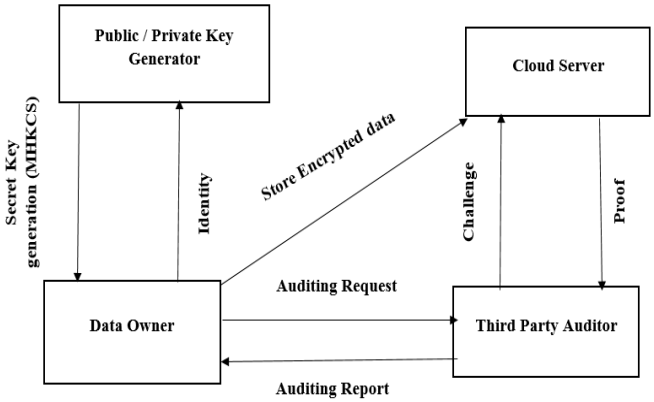

Fig. 1. Proposed architecture of cloud storage model

The proposed cloud storage architecture consists of majorly four entities. Those are Private Key Generator (PKG), Data Owner, TPA, and Cloud Server. Significant entities of these cloud storage model are described in the following sections.

a. Data Owner

Data owner is an individual or enterprise that consists of a large volume of private data. The server configuration of a data owner includes the restricted storage space due to a maximum number of computations and data storing facilities are allocated to the other cloud users. According to the role of data owner in cloud storage model, initially select the file and stored in cloud server. The file split into a number of blocks. The data owner generates different processes such as the key generation, encryption, and decryption using MHKCS algorithm. In data owner side, an input data file is divided into sub-blocks, after that MHKCS algorithm is applied to each data blocks for private key generation, file encryption and decryption. The secure cloud storage data preserving process is shown in the Fig. 2.

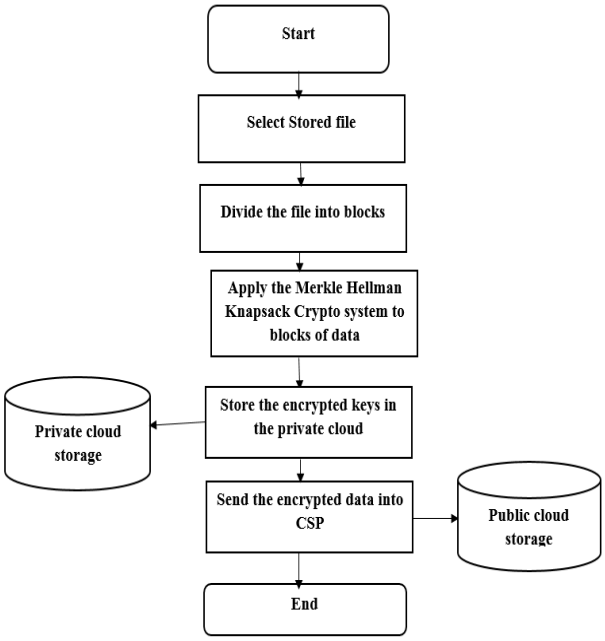

Fig. 2. Flowchart of proposed methodologies

According to the Fig. 2, the proposed cloud storage model majorly focused on three steps such as (i) data owner divide the input file into several blocks then encrypted data stored in cloud server. (ii) For each block MHKCS algorithm is applied for secret key generation, encryption and decryption. (iii) The encrypted keys are stored in the private cloud and encrypted data are stored in the public cloud. The major process of key generation, encryption and decryption process is mathematically described in the following sections.

b. MHKCS Key Generation

The key generation process performs two major operations; those are encryption and decryption of cloud storage data. This PKG produces the private key based on the user's identity information. In this phase, MHKC algorithm consists of two knapsacks such as hard knapsack and easy knapsack. Let us consider the variable x is a public key, which is the hard knapsack, and variable is the private key that is the easy knapsack. Additionally, both those knapsacks combination knows as modulus and it's represented as p. The modulus and multipliers are used for conversion of easy knapsack to hard knapsack and to solve the subset-sum problem. The pseudocode for MHKC key generation is shown below.

1) MHKCS key generation

1. Input: Message length n bits.
2. Output: Public key x_i & private key b_i, p, r
3. Method: $MHKC_{keygen(n)}$
4. for every n-bit message,
5. Selected a super increasing vector,
6. $b_i : \{b_i, b_2, ..., b_n\} - n$ nonzero natural numbers
7. Select a number p such that $p > \sum_{i=1}^{n} b_i$, / / p is known as modulus.
8. Select a number r and p are coprime (i.e.) $\gcd(r, p) = 1$,
9. Compute the vector $x_i = (x_1, x_2, ..., x_n)$ such that $x_i = r.b_i \bmod(p), 0 \le x_i < p$
10. end for
11. Public Key: x_i
12. Private Key: (b_i, p, r)

According to the key generation pseudocode, MHKC algorithm generates the private key and public key. The modulus p is selected to maintain the unique content of the cipher text, otherwise two or more plaintext results in the same cipher text. The p value is higher than the b and sums are not matching to mod, so private keys sums are not equal.

c. MHKCS Encryption

In encryption process, the hard knapsack is represented as x subset and its selection depends on the plaintext length. Each entity in the subset of public key corresponds to the bit 1 in the plaintext is an element of the subset A and the entity corresponds to the bit 0 are ignored. Then, the elements in the subset are added together to compute the sum as cipher text. The encrypted result is obtained in Eq. (1).

1) Pseudocode for MHKCS Encryption

1. Input: Message $m-n$ bits, public key: x_i
2. Output: Cipher text C
3. Method: $MHKC\_enc(m, x_i)$
4. n bit message,
5. $m_i : \{m_1, m_{2,.....} m_n\}$
6. Public key, $x_i : \{X_1, X_2, ... X_n\}$
7. Encrypted message,
8. $C = \sum_{i=1}^{n} m_i, x_i, where\, 0 \le C << p$ (1)

The MHKC highly concentrated on data confidentiality because the public key is used for encryption and private key is employed for decryption process. At first, input parameters for encryption process is message m and public key x_i. The encryption process is mathematically shown in the Eq. (1).

d. MHKCS Decryption

In decryption phase, the encrypted data are input to the decryption operation and it performed using MHKC algorithm. The decryption process includes the modulus p and multiplier p used to convert easy knapsack to public key and cipher text to super-increasing knapsack. In order to decrypt cipher text, greedy algorithm is used and the easy knapsack is solved using $O(n)$ arithmetic operations. The final decryption result is obtained in Eq. (2). The pseudocode for MHKC decryption algorithm is described below.

1) Pseudocode for MHKCS Decryption

1. Input: Cipher text C, private key : (b_i, p, r)
2. Output: Message m_n bits
3. Method: $MHKC\_dec(C, b_i, p, r)$
4. Using extended Euclidean algorithm,

5. Compute $r^{-1} \bmod p$
6. Compute $C' = \mathrm{C}.\mathrm{r}^{-1} \bmod(\mathrm{p})$
7. $\Rightarrow C' = \sum_{i=1}^{n} m_i x_i r^{-1} \bmod(\mathrm{p})$
8. Applying $x_i = r.b_i \bmod(\mathrm{p}), 0 \le \mathrm{x}_i < p$
9. $\Rightarrow C' = \sum_{i=1}^{n} m_i x_i r \bmod(\mathrm{p})$
10. Since, $p > \sum_{i=1}^{n} b_i$, $m_i \in \{0,1\}$ and $\sum_{i=1}^{n} m_i x_i < p$
11. $\Rightarrow C' = \sum_{i=1}^{n} m_i b_i$
12. Now, receiver has to solver the subset problem in $O(\mathrm{n})$
13. Assign $S = C'$
14. for $i = n$ down to 1
15. {
16. if $S \ge b_i$ then
17. {
18. $m_i = 1;$
19. $S = S - b_i;$
20. }
21. else
22. $m_i = 0;$
23. }
24. **Result in message:** $\{m_1, m_2, \ldots\ldots \mathrm{m}_n\}$ (2)

In decryption process, MHKCS algorithm considered input as cipher text C and private key $(\mathrm{b}_i, \mathrm{p}, \mathrm{r})$. At first, calculate the modular multiplicative as inverse $r^{-1} \bmod p$ using extended Euclidean algorithm. In the second step, multiply every element of the encrypted message (cipher text) C with $r^{-1} \bmod p$ and it's denoted as C'. However, if the set of numbers (knapsack) is super-increasing then every element of the set is greater than the sum of all the numbers in the set lesser than it. Finally, the decrypted message should be matched to the original input.

B. Cloud Server

Generally, data owners store all the data in cloud server but its semi trusted because attackers try to access the stored data in different manners. If authorize users view or modify the stored data, then cloud storage security decrease. In order to rectify these problems, data should be encrypted before storing at a remote location, which increases the level of stored data security.

C. Third Party Auditor

The TPA check the integrity of data that is stored in the cloud and perform the auditing process to improve the vulnerability of user's data privacy. The major responsibility of TPA in cloud storage is to monitor or manage outsourced data under the delegation of the data owner. Whenever TPA receives the data integrity verification request from the data owner, immediately TPA send the challenge request to the cloud server. The TPA receives the proper response from cloud auditing task and sends the result back to the user. Moreover, hash values of the files are stored at TPA. In this work, auditing phase verifies the data integrity by performing four tasks such as (i) Auditing Request, (ii Challenge, (iii) Proof Generation, and (iv) Proof Verification.

Auditing Request: The data owners send the request to the TPA to check the data integrity. After receiving the data owner request TPA starts to audit the data integrity process.

Challenge: When user monitor any modification happened in the stored blocks of data or files without user authentication TPA will send a challenge to CSP. The challenge process considers the information of message m i.e. total number of blocks, file identity, version number, etc. and indicated as M_{info}. It chooses some data blocks to generate the challenge set Q and generates a random coefficient $v_i \in Z_p^*$ for each chosen data block $m_i(\mathrm{i} \in Q)$ and output of this process is represented as $Chal$ shown in Eq. (3).

$$Chal = Challenge\{(\mathrm{i}, \mathrm{v}_i)\}_{i \in Q} \tag{3}$$

Proof Generation: In this step, encrypted data C and $Chal$ is the input for proof generation. After receives the challenge from TPA, the cloud server send the proof P. This proof indicates the user's confidential data stored securely in cloud. The proof generation equation is shown in the Eq. (4).

$$P = \prod_{j=1}^{s} e(\mathrm{u}_j, \mathrm{R})^{MP_j} \tag{4}$$

Whereas, the u_j is indicated as set of cloud server parameters. In order to generate the data proof initially computes the sector linear combination of all the challenged data blocks is indicated as MP_j for each $j \in [1, \mathrm{s}]$.

Proof Verification: In this step, data proof P consider as inputs and system public parameters, and returns "success" if the proof is valid; or "failure". The proof generation is mathematically shown in Eq. (5),

$$\text{Proof Verification} = \mathrm{P}.\mathrm{e}(\mathrm{chal}, \mathrm{x}_i) \tag{5}$$

Whereas, verification process considers the input as challenge as $Chal$, proof as P, public key x_i.

In this study, to improve the data integrity, confidentiality, and privacy in cloud storage data MHKCS algorithm is proposed. This algorithm is applied in major three steps such as key generation, encryption, and decryption. Those processes are performed in minimum time. This algorithm converts the super-increasing sequence to a high-density knapsack sequence. An experimental performance of MHKCS algorithm is shown in the following sections.

IV. EXPERIMENTAL RESULT AND DISCUSSION

For experimental simulation, CloudSim 3.0 PlanetLab was employed on PC with 3.2 GHz i5 processor. The experimental data were taken from Enron Email Dataset, which consists of total 200, 399 messages belong to 158 users. Different number of emails were randomly selected from the Enron Email to build an experimental dataset. Each set of input keywords was randomly generated through the user.

After that, the cloud server searched the data from database and extracted the qualified files. In order to estimate the efficiency of proposed algorithm the performance of several metrics such as, key generation time, encryption time, decryption time, and computation overhead used in this research work.

A. Performance Measure

In this section, the experimental result of both existing and proposed cloud storage data auditing mechanisms is evaluated using various performance measures, which include key generation time, encryption time, and decryption time. These efficient parameters are described in the below sections.

- **Key generation time:** The (KGT) process is defined as the amount of time that MHKCS algorithm taken for generating the key of the particular data and it is described in the Eq. (6),

$$KGT = File\,transfering\,time - Execution\,time \quad (6)$$

- **Encryption Time (Uploading Time):** It is defined as the amount of time taken by the data owner to encrypt the original data into encrypted data and it's derived in the Eq. (7),

$$ET = Ending\,time - Starting\,Time \quad (7)$$

- **Decryption time (Downloading Time):** DT time is described as the amount of time taken by the data owner to decrypt the encrypted data that is expressed in terms of milli-seconds (ms) and derived in the Eq. (8).

$$DT = Ending\,Time - Starting\,Time \quad (8)$$

- **Communication Cost:** It is defined as calculate the communication cost between the user and TPA likewise cloud and TPA. It is derived in the Eq. (9),

$$Communication\,\text{cost} = \text{c}.(|s| + |p|) \quad (9)$$

Whereas, c is indicated as number of selected blocks, $|s|$ is the size of the file set, i.e. $\{s_1, s_2, ...s_c\}$ and $|p|$ is represented as the size of an element in the file set.

B. Quantitative analysis using key generation encryption and decryption time

In this section, the performance evaluation of the existing methodologies RSA, MRSA, MRSAC and the proposed approach MHKCS are evaluated by means of key generation, encryption and decryption time. For the experiment, seven different key lengths are considered. Those are 100 bit, 128bit, 256bit, 512 bit, 1024bit, 2048bit and 4096bit. The Table 1 shows the KGT with respect to different key lengths. The key generation process generates the private and public key for improving the cloud data security. The proposed MHKCS method shows better results in contrast with existing methods. The graphical representation of the KGT is shown in the Fig. 3.

Table I. Key Generation Time

Key length (in bit)	Key Generation Time (msec)			
	RSA [14]	MRSA [14]	MRSAC [14]	Proposed (MHKCS)
100	72	110	158	47
128	92	144	192	62
256	140	172	244	110
512	469	484	584	394
1024	596	625	1625	540
2048	2453	8125	8925	1171
4096	91542	93899	123899	25203
Average	13623.42	14794.14	19375.28	3932.42

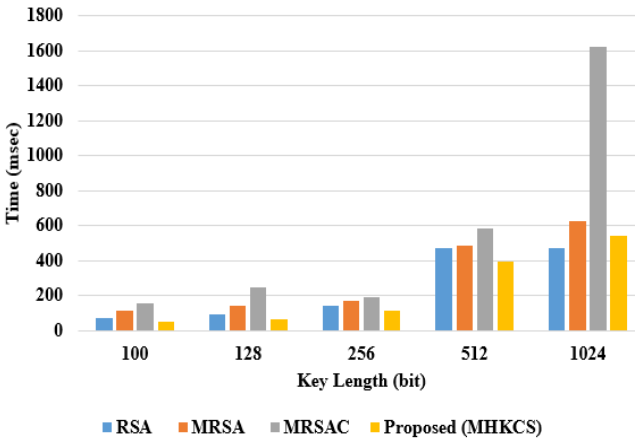

Fig. 3. Key generation Time

The Fig. 3 represents the existing cryptographic algorithms and proposed algorithm's performance of KGT with respect to different key lengths. If number of key length increase, then KGT also increase simultaneously. However, the proposed method shows average KGT for maximum key lengths.

In Table 2, the performance evaluation of the proposed technique and existing approaches are evaluated in terms of encryption and decryption time. The average encryption time of the proposed technique, MHKCS achieved 334.5 msec. The existing methodologies RSA, MRSA, MRSAC attained 485.83 msec, 2029.16 msec and 2194.83 msec of average ET. The graphical representation of encryption time is denoted in the Fig. 4.

Table II. Performance of encryption and decryption time

Key Length (in bit)	Encryption Time				Decryption Time			
	RSA	MRSA	MRSAC	Proposed MHKCS	RSA	MRSA	MRSAC	Proposed MHKCS
100	80	222	188	63	62	107	212	60
128	101	205	305	78	88	122	188	85
256	109	329	409	94	188	156	203	110
512	188	672	762	172	218	968	688	182
1024	484	856	625	334	1453	6938	7038	1245
2048	2953	9891	10880	1266	15203	13609	16709	12271
Average	485.83	2029.16	2194.83	334.5	2868.66	10316.66	4173	2325.5

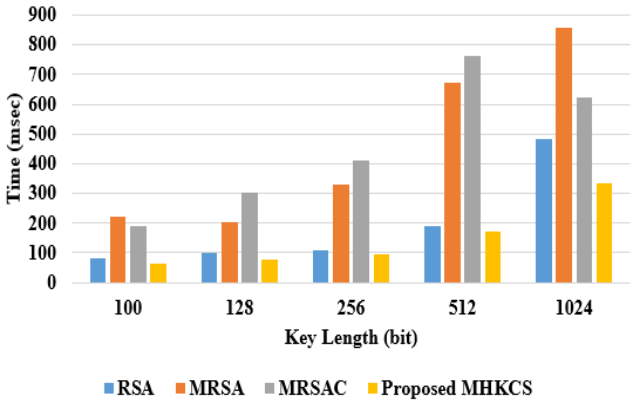

Fig. 4. Encryption Time

Table 2 shows the performance evaluation of existing methodologies and the proposed method. The evaluation metrics (encryption and decryption time) confirms that the proposed scheme performs significantly in cloud data auditing process securely in contrast with previous methods.

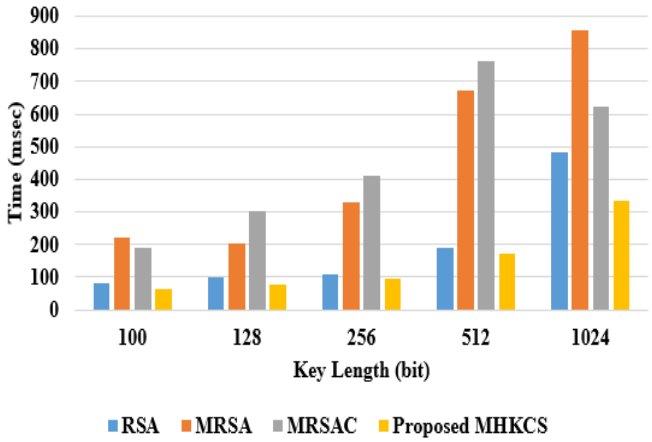

Fig. 5. Decryption time

The Fig. 5 shows the decryption time of the different cryptographic methods in the cloud storage auditing process. The proposed MHKCS methodology achieved 2325.5 ms of DT in contrast with the traditional decryption methods. Hence, MHKCS method shows better results than other methods.

C. Quantitative analysis using communication cost

In this section, the comparative study of existing and proposed work is carried-out by using the performance measure namely communication cost. The MHKCS approach measure the communication cost between two phases such as Challenge and proof generation in cloud. Those two phases communication cost performance is analyzed using existing RSA, Identity based Cloud Data Integrity Checking (ID-CDIC) and proposed MHKCS.

Table III. Communication Cost of various Auditing Phases

Different Auditing Phases	RSA [19]	ID-CDIC [20]	Proposed MHKCS
Challenge Phase	1056	2080	899
Proof Generation Phase	9248	8224	11895

Inspecting the Table 3, the proposed MHKCS methodology and existing algorithms performance of communication cost is evaluated. The MHKCS achieved 899 bits and 11895 bits with respect to challenge phase and proof generation phase respectively. It indicates the minimum challenges with high proof. The existing RSA attained 1056 bits of challenge phase and 9248 bits of proof generation. The ID-CDIC method attained 2080 bits of challenge and 8224 bits of proof generation. Hence, existing methods show the maximum communication cost in terms of challenge phase and proof generation phase. The communication cost is graphically represented in Fig. 6.

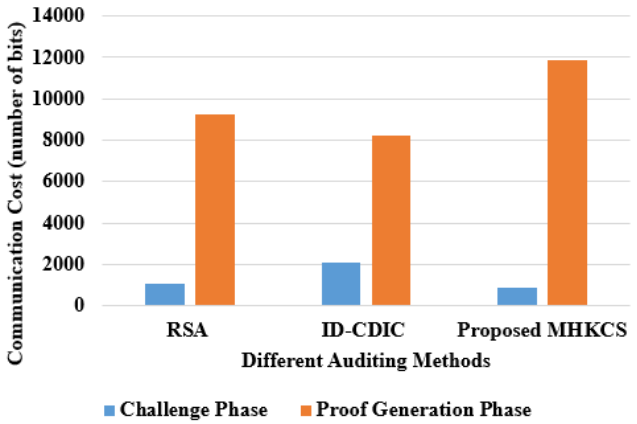

Fig. 6. Communication Cost

D. Comparative study

This section describes the various cryptographic existing methods of cloud storage security such as MRSA [11], Blowfish Hybridized Weighted Attribute-based Encryption [21], Advanced Encryption Standard and Data Encryption Standard [22].

The comparative study of these algorithm performances was measured with respect to KGT, ET, and DT that is shown in the Table 4. The MRSA algorithm achieved 188ms, 212ms and 158ms in terms of ET, DT and KGT respectively for 100 bit of key length. The chunk file storage certainly reduces the fragmentation issues and increase the efficiency of resource utilization in the cloud server [11]. The BH-WABE algorithm achieved 99ms of KGT, 118ms of ET and 212ms of DT with respect specific key length (100bit). This algorithm provided maximum security for cloud storage data but more time complexity [21]. AES and DES algorithm achieved almost similar DT and ET but high time complexity because its processing speed is slow [22]. The proposed MHKCS methodology shows the better results than other methods with high security and data privacy in cloud storage.

Table IV. Study of 100bit of key length of different methods

Author Name	Methodology	Key Generation Time (ms)	Encryption Time (ms)	Decryption Time (ms)
S. Anbuchelian, et al. [14] (length 100)	MRSA	158	188	212
Ghosh, S. and Karar, V., [21]	BH-WABE	99	118	113
AbdElminaam, D.S., [22]	AES	-	173	134
	DES	-	170	134
Proposed	MHKCS	47	63	60

V. CONCLUSION

Recently, the cloud-computing paradigm has become popular because of its huge storage and flexible computation abilities. To utilize these advantages, more data owners tend to outsource their data and further data analysis operations (e.g., data queries, data insertion, modifying and so on) in cloud. For security purposes, a data owner may choose to encrypt its data before outsourcing. In this paper, an efficient auditing cloud storage model based MHKCS algorithm is proposed for improving the security and privacy among the cloud storage and TPA. The MHKCS algorithm performed efficient key generation, encryption and decryption process in order to improve the cloud storage security. The experimental evaluation of proposed algorithm performance is measured by using KGT, encryption time, decryption time, and communication cost. The proposed MHKCS methodology achieved 10% of enhancement than the traditional methods RSA, MRSA, and MRSAC, with respect to encryption and decryption time complexity reduction. In future work, an efficient technique can be employed for improving the confidentiality and integrity of cloud storage data.

REFERENCES

1. X. Liu, B. Qin, R. H.Deng and Y. Li. (2017). An efficient privacy-preserving outsourced computation over public data. *IEEE Transactions on Services Computing, 10(5),* pp. 756-770.
2. V. Muthurajan and B. Narayanasamy. (2016). An elliptic curve based schnorr cloud security model in distributed environment, *The Scientific World Journal.*
3. K. Xue, Y. Xue, J. Hong, W. Li, H. Yue, D. S. Wei and P. Hong. (2017). RAAC: Robust and Auditable Access Control with Multiple Attribute Authorities for Public Cloud Storage, *IEEE Trans. Information Forensics and Security, 12(4),* pp. 953-967.
4. J. Shen, J. Shen, X. Chen, X. Huang and W. Susilo. An efficient public auditing protocol with novel dynamic structure for cloud data, *IEEE Transactions on Information Forensics and Security, 12(10),* pp. 2402-2415.
5. C. Wang, S. S. Chow, Q. Wang, K. Ren and W. Lou. (2013). Privacy-preserving public auditing for secure cloud storage, *IEEE Transactions on computers, 62(2),* pp. 362-375.
6. Y. Luo, M. Xu, K. Huang, D. Wang and S. Fu. (2018). Efficient auditing for shared data in the cloud with secure user revocation and computations outsourcing, *Computers & Security, 73,* pp. 492-506.
7. K. Yang and X. Jia. (2013). An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing. *IEEE Trans. Parallel Distrib. Syst., 24(9),* pp. 1717-1726.
8. D. He, H. Wang, J. Zhang and L. Wang. (2017). Insecurity of an identity-based public auditing protocol for the outsourced data in cloud storage, I*nformation Sciences, 375*, pp. 48-53.
9. W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu and R. Hao. (2017). Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium, *Journal of Network and Computer Applications, 82*, pp. 56-64.
10. Y. Wang, Q. Wu, B. Qin, W. Shi, R. H. Deng and J. Hu. (2017). Identity-based data outsourcing with comprehensive auditing in clouds, *IEEE Transactions on Information Forensics and Security, 12(4),* pp. 940-952.
11. K. Loheswaran and J. Premalatha. (2016). Renaissance system model improving security and third party auditing in cloud computing, *Wireless Personal Communications, 90(2),* pp. 1051-1066.
12. Y. Li, Y. Yu, B. Yang, G. Min and H. Wu. (2018). Privacy preserving cloud data auditing with efficient key update, *Future Generation Computer Systems, 78*, pp. 789-798.
13. S. Rizvi, J. Ryoo, J. Kissell, W. Aiken and Y. Liu. (2017). A security evaluation framework for cloud security auditing, *The Journal of Supercomputing,* pp. 1-23.
14. S. Anbuchelian, C. M. Sowmya and C. Ramesh. (2017). Efficient and secure auditing scheme for privacy preserving data storage in cloud, *Cluster Computing,* pp. 1-9, 2017.
15. J. Yu, K. Ren and C. Wang. (2016). Enabling cloud storage auditing with verifiable outsourcing of key updates, *IEEE Transactions on Information Forensics and Security, 11(6),* pp. 1362-1375.
16. T. Xiang, X. Li, F. Chen, Y. Yang and S. Zhang. (2018). Achieving verifiable, dynamic and efficient auditing for outsourced database in cloud, *Journal of Parallel and Distributed Computing, 112*, pp. 97-107.
17. D. Kim, H. Kwon, C. Hahn and J. Hur. (2016). Privacy-preserving public auditing for educational multimedia data in cloud computing, *Multimedia Tools and Applications, 75(21),* pp. 13077-13091.
18. H. Tian, Y. Chen, C. C. Chang, H. Jiang, Y. Huang, Y. Chen and J. Liu. (2017) Dynamic-hash-table based public auditing for secure cloud storage, *IEEE Transactions on Services Computing, 10(5),* pp. 701-714.
19. Z. Xu, L. Wu, M. K. Khan, K. K. R. Choo and D. He. (2017). A secure and efficient public auditing scheme using RSA algorithm for cloud storage, *The Journal of Supercomputing, 73(12),* pp. 5285-5309.
20. Y. Yu, L. Xue, M. H. Au, W. Susilo, J. Ni, Y.Zhang, A. V. Vasilakos and J. Shen, Cloud data integrity checking with an identity-based auditing mechanism from RSA, *Future Generation Computer Systems, 62*, pp. 85-91, 2016.
21. S. Ghosh and V. Karar. (2018). Blowfish Hybridized Weighted Attribute-Based Encryption for Secure and Efficient Data Collaboration in Cloud Computing, *Applied Sciences, 8(7),* pp. 1119, 2018.
22. D. S. AbdElminaam. (2018). Improving the security of cloud computing by building new hybrid cryptography algorithms, *International Journal of Electronics and Information Engineering, 8(1),* pp. 40-48.

AUTHORS PROFILE

Mr N Gowtham Kumar is Research Scholar in KL (Deemed to University). Currently he is working as an Assistant Professor in CSE department at KITS Singapur. He has 12 years of teaching experience. His research area is Cloud Computing. He has published more than ten papers and attended several conferences, Workshops and Faculty development programs. His other interested areas include Computer networks, Mobile computing, Semantic Web and Social Networks.

Dr. Niranjan Reddy Polala working as a Professor & Head Dept. of CSE, KITS, Warangal having more than 26-year experience in teaching. He is expertise in Software Engineering. Research work done in Representation of software reusable components. His research areas include Software Engineering, Software Reuse, Information Retrieval, Soft Computing Techniques, Data Clustering. He has published more than 30 research articles and organized several workshops and conferences.

Dr. D. Aruna Kumari, received the Ph.D. degree from the K L University, Vaddeswaram, Andhra Pradesh. Currently, she is working as a Professor in VJIT, Hyderabad. Her teaching and research areas include in data mining and published papers on privacy preserving in data mining. She has published more than 70 research articles, Young Scientist awardee and on more research project funded by DST-SERB, and also organized several national Conferences/Workshops.