Computers, Materials & Continua
DOI: 10.32604/cmc.2022.021269
Article

Tech Science Press

Cloud Data Encryption and Authentication Based on Enhanced Merkle Hash Tree Method

J. Stanly Jayaprakash[1], Kishore Balasubramanian[2], Rossilawati Sulaiman[3], Mohammad Kamrul Hasan[3,*], B. D. Parameshachari[4] and Celestine Iwendi[5]

[1]Department of CSE, Mahendra Institute of Technology, Namakkal, 637503, India
[2]Department of EEE, Dr. Mahalingam College of Engineering and Technology, Pollachi, 642003, India
[3]Center for Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia (UKM), 43600, Bangi, Malaysia
[4]Department of Telecommunication Engineering, GSSS Institute of Engineering and Technology for Women, Mysuru, India
[5]School of Creative Technologies, University of Bolton, Deane Road, Bolton, United Kingdom
*Corresponding Author: Mohammad Kamrul Hasan. Email: mkhasan@ukm.edu.my
Received: 28 June 2021; Accepted: 03 September 2021

Abstract: Many organizations apply cloud computing to store and effectively process data for various applications. The user uploads the data in the cloud has less security due to the unreliable verification process of data integrity. In this research, an enhanced Merkle hash tree method of effective authentication model is proposed in the multi-owner cloud to increase the security of the cloud data. Merkle Hash tree applies the leaf nodes with a hash tag and the non-leaf node contains the table of hash information of child to encrypt the large data. Merkle Hash tree provides the efficient mapping of data and easily identifies the changes made in the data due to proper structure. The developed model supports privacy-preserving public auditing to provide a secure cloud storage system. The data owners upload the data in the cloud and edit the data using the private key. An enhanced Merkle hash tree method stores the data in the cloud server and splits it into batches. The data files requested by the data owner are audit by a third-party auditor and the multi-owner authentication method is applied during the modification process to authenticate the user. The result shows that the proposed method reduces the encryption and decryption time for cloud data storage by 2–167 ms when compared to the existing Advanced Encryption Standard and Blowfish.

Keywords: Cloud computing; cloud data storage; cloud service provider; merkle hash tree; multi-owner authentication; third-party auditor

1 Introduction

Cloud computing has been adopted in many organizations to store and process data in an efficient manner and any organizations prefer cloud computing due to its advantages of scalability, flexibility

and reliability. Cloud services are the best option for companies for the quick response and the best flexibility [1,2]. User outsources the data in the cloud to store and process the data due to its scalability and flexibility. Cloud Service Provider (CSP) has to preserve the privacy of the sensitive data and the user has the option of encrypting the data before uploading in the cloud [3,4]. Cloud computing provides virtual computing services to extensive, medium, and little industries and services for example Software as a service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Services (IaaS) [5,6]. The improved accessibility and technical advancements are extensively favorable for the deployment of cloud servers for storage and computation. Quality of Services, low cost and stability make a cloud a typical choice for computing-intensive tasks [7,8]. Security is a major problem for the cloud computing system and significant revision of the existing security choices leads to develop modern techniques. Guarantee cloud data security and users access control are the current security issues in the cloud [9,10]. A trusted Third-Party Auditor (TPA) is applied to verify the cloud data to reduce the burden on users, this is called public auditing. However, the TPA may have unnecessary access to private information in the auditing process [11,12].

The receiver node is verified by the sender node in the authentication process. The sender independently generates the encrypted messages in basic authentication and the symmetric key is used in the receiver to match the key [13]. Some of the research involves in apply the complex cryptosystem for authentication in a cloud [14]. OTP-based authentication [15], authentication mechanism and cryptography algorithm [16–20], Multiple factor-based authentication, anonymous node ID assignment are some types of methods in cloud authentication. Security is the important aspect of cloud services and attacks in the cloud tend to lose of personal information of the users and profit to the company. Data integrity is the way to track the data in the cloud and retrieve the old version to retrieve the affected data easily. Searching is another important aspect of encrypted data and the Cloud needs to support the search process in encrypted data to retrieve the user data securely. Therefore, an efficient method is needed in the cloud to manage security, integrity and search on encrypted data. In this research, the enhanced Merkle hash tree method with multi-owner authentication is proposed to secure data in cloud storage. The enhanced Merkle hash tree method is applied to perform a significant data integrity check in cloud storage. Every node in the Merkle hash tree contains the information of position relative to its parent node. The user obtains a public key from the TPA using an authentication request to download the encrypted data. If the user is authorized, the data owner gives the decryption key to the user to decrypt the encrypted data.

This research paper is organized as follows: Section 2 surveys several existing research papers on cloud data storage authentication with problem statements. In Section 3, an effective method called the enhanced Merkle hash tree is presented with a multi-owner authentication technique for cloud data storage authentication. Section 4 explains the performance of the proposed method. The conclusion of this research paper is given in Section 5.

2 Literature Review

This section reviews recent and significant research approaches suggested by researchers in cloud data storage authentication. A brief evaluation of the essential contributions and main gaps in the existing literature are presented to support the presented proposal.

Shajina et al. [21] proposed a dual authentication protocol with two level of authentication with the list of precedence-based access control to improve the security and scalability of the cloud. The triple-DES algorithm with the identity of the user was applied as an extra attribute to improve the security in the cloud. The anonymous of the user was protected in this method to improve the security

and the proposed method for the multi-owner cloud was verified in multiple scenarios. The verification shows that the proposed method has a lower average network time and high security in the cloud. The developed methodology achieves secure and effective data sharing in dynamic cloud storage. However, the developed methodology provides insecure data sharing as a one-to-one solution and lower efficiency in large datasets.

Anand et al. [22] proposed Elliptic Curve Digital Signature Algorithm (ECDSA) and Enhanced Elliptic Curve Diffie-Hellman (EECDH) method for mutual authentication in the multi-owner in the cloud. The proposed EECDH method was applied to exchange the secure key to the owners and eliminate the Man-In-The-Middle (MITM) attack. The proposed method protects the data integrity in the cloud. The identity and attribute-based access policy increases the robustness of the method. The developed cloud computing system still has vulnerabilities, specifically in networks that have several third-party platforms and complex infrastructures.

Deep et al. [23] applied blockchain technology for secure authentication in the cloud to increases security. Blockchain technology makes it easy for an insider to change the login credential details for the authentication process. The correctness and applicability of the method were tested on the cloud. The Scyther formal system was used to test the proposed method against no reply attack, offline guessing, impersonation, and denial of services. The result shows that the proposed method was robust in secure the user information in the cloud. The developed authentication scheme was an open cloud computing system that frequently encountered social engineering and phishing attacks. The integrity check in the model was poor due to the unstructured encryption process in the data.

Badr et al. [24] proposed an Attribute-based encryption method that considers authority, cloud servers, data users and data owners to secure the data in the cloud. The decryption process was delegated by the data owner to reduce the computational complexity. The MAC encryption text associated with the data stored in the cloud was generated after encryption. The attribute property was applied for the encryption and decryption was based on verification. The various attacks have been applied to test the performance of the developed method. If the number of authentication samples is low, the authentication rate will be reduced. The Attribute-based encryption has a higher overhead for the large database and has lower performance in integrity check.

Ge et al. [25] developed Accumulative Authentication Tag (AAT) using symmetric key cryptography for the authentication of keywords. The authentication tag was updated based on dynamic operations in the cloud. The secure index consists of a search table based on the orthogonal list and verification list containing AAT. The method updates are based on connectivity and flexibility. The AAT method has lower overhead and lowers computation time during a keyword search than a secure conjunctive keyword search. An analysis showed that the developed method was secure and efficient and the memory usage of the method needs to be reduced for the authentication process to be effective. The AAT methods store the tag method for the encrypted data for integrity and memory usage for the tag is high for the large dataset.

Zhong et al. [26] developed a mutual authentication and key agreement scheme based on elliptic curve cryptography for the peer-to-peer cloud. The elliptic curve certificate-free cryptography method was used for key generation. This method eliminates the trusted authority and simplifies mathematical operations to increase security. The security correctness of the method was analyzed, and it showed that the method was secure. This proposed method reduces computational and communication costs compared with existing methods. Further development is required to support multi-user access to cloud data securely.

The existing methods have considerable performance in the encryption of data in the cloud for multi-cloud support. The DES algorithm [21] method provides the support multi-cloud with improved security and has lower efficiency in large datasets. The EECDH [22] method provides the data integrity in the cloud and this method fails to perform in the large dataset due to the attribute-based method. Blockchain technology [23] provides higher security in data transmission and this is not suitable for cloud-stored data. Attribute-based methods [24–26] was suitable for cloud storage and lower efficiency in large dataset. Commonly, existing methods have limitations of doesn't support data integrity or lower efficiency in large datasets.

3 Problem Statement

This section describes the problem statement for the Merkle hash tree in cloud data storage authentication and explains how the proposed methodology provides a solution to the described problems. The concerns about the Merkle hash tree approach are detailed as follows:

Searching the i^{th} leaf node in data integrity auditing is computationally complex in the Merkle hash tree approach. If the storage structure is continuous, the number of operations like insertion and deletion are quite high. After performing the deletion and insertion operations, the sequence number of the nodes is likely to be modified, and the height of the tree may become imbalanced [27].

The Merkle hash tree approach has no function to manage data integrity auditing, which is called empty proof [28]. Also, it does not provide any integrity guarantee for the data, because integrity assures that outsourced data is intact. The traditional Merkle hash tree uses either a message digest (MD) or a Secure Hash Algorithm (SHA) as the hash function. Most of the time, these functions do not indicate tree depth, which leads to second preimage attacks.

Solution: To overcome the above-mentioned drawbacks, an enhanced Merkle hash tree is implemented with a multi-owner authentication methodology to improve the performance of cloud data storage authentication. Here, a new hash function of the tiger tree hash function is replaced with an MD or SHA hash function in the Merkle hash tree. The tiger tree hash method is a truncated version with a specific hash size, so there is no need to distinguish the defined values. It contains a total of 24 rounds, with digest sizes of 128, 160, and 192. The tiger hash function uses 2^{44} compression functions in the 16^{th} round to avoid equivalent time complexity and uses 2^{48} compression functions in the 19^{th} round to secure the stored data from a collision attack. A detailed description of multi-owner authentication and an enhanced Merkle hash tree is given in Section 4.

4 Proposed Methodology

The proposed methodology is used for data integrity, load balancing, and multi-owner authentication in the cloud environment. A publicly verifiable methodology of enhanced Merkle hash tree with multi-owner authentication is used to protect the integrity of the cloud data and also to support dynamic maintenance.

4.1 Multi-Owner Authentication

In this method, a highly secure multi-owner authentication methodology is implemented to secure the cloud server database. Initially, the data are uploaded to the cloud server by the data owner in an encrypted format using an enhanced Merkle hash tree approach.

The user is provided with a public key for viewing and downloading the data, and the data owner verifies the user (authorized or unauthorized) using the public key. If the user is authorized, then a

decryption key is provided by the data owner to the user for decrypting the data. A load balancing concept is also implemented for processing the user-requested job. Finally, the user request is passed to the cloud server. If the user is authenticated, the cloud server replies to the user query. The general design of the cloud data storage is denoted in Fig. 1. A brief evaluation of the proposed technique is determined as follows:

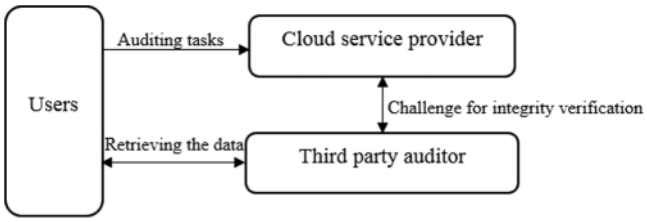

Figure 1: Cloud data storage architecture

4.2 System Model for Cloud Data Storage

This sub-section describes the system model shown in Fig. 1. Generally, the integrity of cloud data architecture involves the following parties [29–31].

Data owner

First, the data owner needs to register on the cloud service provider's server. After registration, the cloud service provider generates private and public keys and sends the keys to the data owner. The respective data are encrypted based on the Enhanced Merkle Hash tree method and uploaded to the cloud server. The sender encrypts the data using a public key, and only the holder has a private key to decrypt the respective data.

User

To access the network, the user needs to initially create an account. After creating the account, the user logs into the account and requests auditing from the cloud service provider. Based on the user request, the cloud service provider will process the task. The network uses programming languages like .NET or Java for communicating with the cloud server. The user can access the requested data by sending a request to the cloud service provider. The enhanced Merkle Hash tree given private key is used by the user to access the cloud data.

Cloud Service Provider

The cloud service provider makes a combination of hardware and software resources available and delivers flexible online data storage and computing. The cloud service provider stores a large amount of data and also manages the authentication of the data owner and users. To process the user-requested job, the cloud server redirects the job to any queue. In the queue, virtual machines are utilized to process user requests.

Third-Party Auditor

The TPA is performed on the encrypted cloud data to check the integrity of the data stored in the cloud. Based on the data owner's request, the TPA audits the data, which are uploaded by the data owner. The TPA needs to register with the cloud server to audit user-requested data. The TPA sends a request to cloud storage to audit the data present in the cloud. The cloud performs a bilinear mapping on the user data and sends the proof to the TPA. The TPA performs a comparison between

the user-provided data and the cloud-provided data. The encrypted data is used to protect the privacy of the file.

Bilinear Mapping

The bilinear mapping is the preliminary step of encryption to represent the data in mapping form. The cyclic group G of the data is given as input and e is the bilinear mapped output. Consider a group G as a gap Diffie-Hellman group with prime order p. A bilinear map is developed as $e{:}G \times G \rightarrow G_T$, where G_T is a prime order, multiplicative cyclic group. A useful e has the following properties:

Bilinearity-$\forall m, n \in G \Rightarrow e(m^a, n^b) = e(m, n)^{ab}$;

Non-degeneracy—$\forall m \in G, m \neq 0 \Rightarrow e(m, m) \neq 1$;

Computability—e should be efficiently computable.

Bilinearity is the property of representing the data with a two-dimensional vector, non-degeneracy is the property of data that is degenerated to the original data, and computability is the ability to solve a problem in an effective manner where a and b are real random numbers.

4.3 Merkle Hash Tree Approach for Key Generation

The Merkle Hash tree method is applied to encrypt the data before uploading to the cloud and the Merkle Hash tree method generated private is used to decrypt the data in the cloud. Merkle Hash Tree method stores the table of hash values from the encrypted data in the non-leaf node and leaf node consists of encrypted data with a labeled hash value. This structure of encryption helps to identify the changes in the data related to the leaf node that helps to increases the integrity. Merkle Hash tree method stores the hash value instead of duplicate the whole data in the leaf node and this helps to encrypt the large dataset with less memory. The Merkle hash tree method is a binary tree data structure with leaf nodes integrated at each node. The Merkle hash tree root node is present as the top node, and the leaf nodes consist of data hashes. Root node authentication provides access to the leaf nodes' integrity declaration. A Merkle hash tree with eight leaf nodes is shown in Fig. 2 [28]. Bilinearly mapped cloud data is used for encryption based on the Merkle hash tree method. The advantage of using the Merkle Hash tree is that it effectively verifies the data in the distributed system using hash values. In the Merkle Hash tree, a node is denoted, $N_{i,j}$ were i and j are the i^{th} level and j^{th} level of the node. The cryptographic variable $H_{i,j}$ stores node $N_{i,j}$, and the nodes at level 0 are called "leaves." The leaves describe the data stored in the tree. In the case of revocation, the leaves that have been revoked are represented by a certificate set Φ that has been revoked, as shown in Eq. (1).

$$\Phi = \{c_0, c_1, \ldots, c_j, \ldots, c_n\} \tag{1}$$

where, c_j is the data stored in leaf $N_{0,j}$ and $H_{0,j}$ is computed as in Eq. (2).

$$H_{0,j} = h(c_j) \tag{2}$$

where, h is a one-way hash function.

To build the Merkle hash tree, adjacent nodes set t at a given level i $(N_{i,j}, N_{i,j+1}, \ldots, N_{i,j+t-1})$ are combined into one node at the upper level that is denoted by $N_{i+1,k}$. Then, $H_{i+1,k}$ is measured by applying h to the concatenation of the t cryptographic variables, as in Eq. (3).

$$H_{i+1,k} = h(H_{i,j}|H_{i,j+1}|\ldots|H_{i,j+t-1}) \tag{3}$$

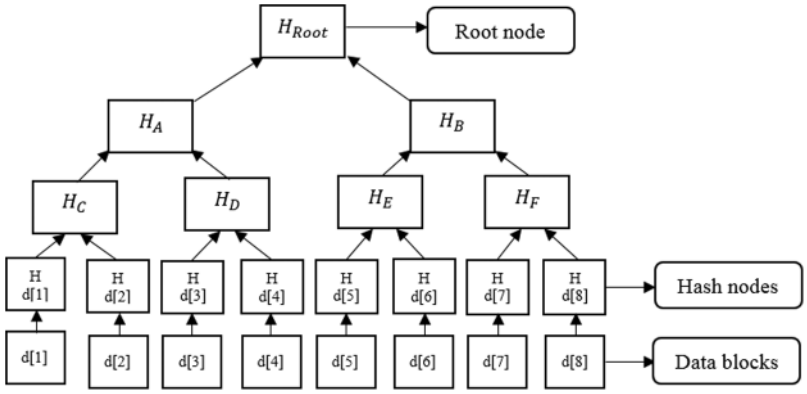

Figure 2: Eight leaf nodes Merkel hash tree

At the top level, one node is present and is denoted as "root"; H_{root} is a digest for all the data stored in the Merkle hash tree.

For a file block m_i, a leaf node is used to compute $h_i = H(m_i)$, and a parent node of N_1 and N_2 is constructed as $N_p = \{H(h_1||h_2)\}$. A leaf node of $m_i's$ Auxiliary Authentication Information (AAI) Ω_i is a set of hash values selected from each of its upper levels so that the root value R can be computed $\{m_i, \Omega_i\}$.

For example, the data owner asks the auditor for data integrity verification at one position. The auxiliary information $AI(d[1])$:$\{(H_D, R), H(d[1]), (H(d[2]), R), (H_B, R)\}$ is provided by the data owner to the auditor. Then, the root H_R is generated by the auditor as follows,

Measure $H_D \leftarrow (H(d[1])||H(d[2]))$;

Measure $H_A \leftarrow (H_C||H_D)$;

Determine the root $H_{Root} \leftarrow (H_A||H_B)$.

The root node authenticity is verified to automatically authenticate all the blocks. Each node consists of two pieces of information, such as the hash value and the relative index. The Merkle hash tree method consists of three stages, namely the set-up stage, challenge-prove stage, and update stage. An explanation of each stage is given as follows:

Set-up Stage

The set-up stage involves system initialization at the client end. This stage consists of three functions.

Key generation (1^{key}): The probabilistic key generation method $\{skey, pkey\}$ is used and the input is the *key* security parameter. The input is the private key *skey*, and the output is the public key *pkey*. The public key *pkey* is published by the data owner and the private key *skey* is kept safe for decrypting the respective data.

Tag generation $skey, pkey, m \rightarrow meta\ data$: The client document input metadata is, $skey, pkey, m$ and the client stores locally the output metadata tags δ. The client remotely stores the respective documents m and tag δ at the server end.

Block-sig-generation: Once the tag documents δ are generated, the data owner uses the cryptographic hash function to generate signatures for each data block. Generally, either MD or SHA is used as the hash function.

Challenge-Prove Stage

The challenge-prove stage is an interaction between client and server. The client identifies the problem and sends challenges to the server. The server analyzes the respective issues and sends output to the client. The challenge-prove stage involves three processes.

Gen-challenge$(C) \rightarrow \{chal\}$: The input for this process is the client's private parameter c, and the output is challenge *chal* for future queries.

Gen-proof $((p_s, m, \delta, chal) \rightarrow \{G\}$: The input is a document m, metadata δ, public parameters p_s, and challenge *chal*, and the output G is given to the user for server verification.

Check-proof $(pkey, chal, meta, data, G) \rightarrow \{$"*reject*", "*accept*"$\}$: This process is used for target document server possession check. The input is the user public key *pkey*, metadata δ, challenge *chal*, and evidence G. The evidence G is analyzed, and the function returns "accept" or "reject."

Updated stage

In the Merkle hash tree, the update stage is a verification process for supporting dynamic data operations. The function of this stage is explained as follows:

Perform-update $(pkey, m, \delta, update) \rightarrow \{m', \delta', G_{update}\}$: After the server receives the *update* requirement from the client, this operation is performed. The input for this process is a document m, metadata δ, public key *pkey*, and the update. The output for future checking consists of a new tag δ', updated document m', and updated evidence G_{update}.

Verify-update $(pkey, update, metadata, G_{update}) \rightarrow \{(metadata', "accept"), "reject"\}$: This process checks the correctness of the dynamic operation at the server end, and verifies the client. The update triggers the verification function, and the input is metadata δ, public key *pkey*, and G_{update}. If the update evaluates correctly, it returns "accept" and retains the metadata locally, or the process returns "reject." The Merkle hash tree protocol supports the data dynamic process and the public auditing of data. Still, a few major flaws in the Merkle hash tree protocol are detailed in Section 3. To overcome these issues, an enhanced Merkle hash tree approach is developed in this research study.

4.4 Enhanced Merkle Hash Tree Approach

This sub-section describes the enhanced Merkle hash tree approach. In this proposed technique, a new hash function (tiger tree hash function) is used in a Merkle hash tree instead of the MD or SHA hash function. In the tiger hash function, the one-way compression function operates on 64-bit words, where it processes 8 words of data and maintains 3 words of state. In total, the tiger hash function includes 24 rounds, using a combination of operations like S-box lookups, rotate, and XOR (addition and subtraction). An effective, intricate scheduling key technique is used to derive keys in 24 rounds from the 8 input words. This operation makes the implementation easy in a microcontroller and other hardware. A sample binary tiger hash tree is shown in Fig. 3.

Finally, the TPA audits the data requested by the data owner using the enhanced Merkle hash tree approach. The TPA updates the auditing information at a certain time, so the security of the data is ensured. Also, the auditor updates the data owner if any change occurs while the data are being audited.

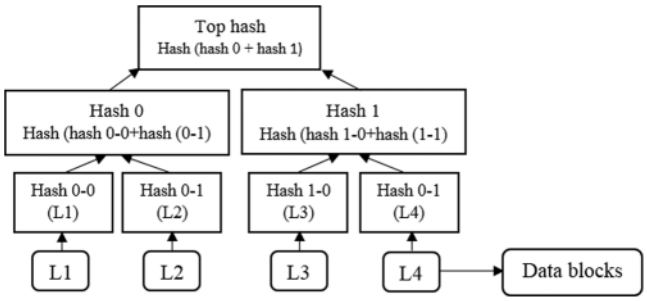

Figure 3: Sample binary tiger hash tree structure

5 Experimental Result and Discussion

This section provides a detailed description of the experimental set-up of the proposed method, and a measure of its performance. The performance is analyzed with comparative and qualitative analyses.

5.1 Experimental Set-Up

The proposed approach was experimentally tested using Net-Beans (version 8.2) with 4 GB RAM, a 3.0 GHz Intel i3 processor, a 500 GB hard disc, and MY-SQL server as a cloud service provider. To estimate the effectiveness of the proposed methodology, the proposed approach of enhanced Merkle hash tree with multi-owner authentication performance was compared with several existing approaches: Advanced Encryption Standard (AES) and Blowfish [6]. In this study, the performance of the proposed approach was compared in terms of encryption time, memory usage, and decryption time.

5.2 Discussion

This section discusses the comparison between the proposed approach and other key assignment approaches in terms of a few characteristics, and it details the compression ratio for dissimilar tree heights and delegation ratios. Tab. 1 presents a comparison of five key assignment approaches with different properties such as decryption key size, cipher-text size, encryption type, and file classification relationship. Tab. 1 shows that most of the encryption types of the existing methods are based on the public key, decryption key size is constant, cipher size is constant and irrelevant to the data. Constant cipher size is vulnerable to attacks and data is easily accessed with common attacks. Encryption of the existing methods is irrelevant to the File classification and integrity check for the data is difficult. Therefore, the proposed Merkle Hash tree method cipher size is inconstant to make it difficult for attacks and integrity check is carried out easily due to the encryption is based on File classification.

Generally, the communication overhead and information leakage risk increase with a greater number of delegation decryption keys. In existing key assignment schemes (symmetric-key encryption, key-aggregate encryption [26], etc.), the decryption key generation depends on previous classification files. When a new file class is uploaded to the cloud server, the whole classification structure must be changed in these methods. The proposed method generates an inconstant ciphertext size and a constant decryption key size. This is relevant to file classification and supports the incessant updating of files.

Table 1: Characteristics of a proposed and existing method

Methods	Encryption type	Decryption key size	Ciphertext size	File classification
Symmetric key encryption with compact key	Symmetric key	Constant	Constant	Irrelevant
Tree-based key assignment method	Public key	Non-constant	Constant	Irrelevant
Key aggregate encryption	Public key	Constant	Constant	Relevant
key-aggregate authentication cryptosystem [15,20,21]	Public key	Constant	Constant	Irrelevant
Proposed approach	Private key	Constant	Inconstant	Relevant

The proposed method is evaluated with dissimilar key sizes and the delegation ratio in Tab. 2 and it clearly shows that the delegation key D_{key} increases with the increase in the delegation ratio.

Table 2: Delegation ratio and dissimilar tree height comparison

Key size	Delegation ratio	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
8	*Dkey*	36.7	38.9	43.2	54.44	55.78	67.98	70	78.42	87.66
	Dkey/*N* (%)	14.33	15.19	16.8	21.26	21.78	26.55	27.34	30.63	34.24
12	*Dkey*	809	834	865.23	899.33	936.47	995.41	1096	1145	1199
	Dkey/*N* (%)	19.75	20.36	21.12	21.96	22.86	24.30	26.75	27.95	29.27
16	*Dkey*	12568	13245	13946	14524	14888	16127	17002	17894	18324
	Dkey/*N* (%)	19.17	20.21	21.27	22.16	22.71	24.60	25.94	27.30	27.96

5.3 Encryption and Decryption Time Analysis

The encryption and decryption of the proposed method are analyzed in this section. For experimental analysis, ten different file sizes are considered, such as 83.3 KB, 108 KB, 249 KB, 333 KB, 416 KB, 1370 KB, 2740 KB, 5480 KB, 10003 KB, and 15483 KB. The existing methods carried out the encryption to encrypt the data with tags in the unstructured format that reduces the capacity to handle the large data. The Merkle has a tree encryption process involves in creating the hash for the data, store the hash value in the table and apply labels to encrypted data. This structured process decreases the encryption time of the Merkle Hash Tree method and the hash value in the table reduces the decryption time in a large dataset. The encryption and decryption times of the proposed and existing methods are shown in Tab. 4. The average encryption time of the proposed method (enhanced Merkle hash tree with multi-owner authentication) is 132.9 msec. The existing methodologies (AES and

Blowfish [6]) achieved average encryption times of 300.4 msec and 215.4 msec. The graphical comparison of encryption time is shown in Fig. 4.

Table 3: Proposed approach evaluation using encryption and decryption time

File size (in KB)	Encryption time (m-sec)			Decryption time (m-sec)		
	AES [6]	Blowfish [6]	Proposed	AES [6]	Blowfish [6]	Proposed
83.3	625	8	56	24	17	12
108	31	16	74	15	10	12
249	468	47	22	16	15	10
333	32	281	63	16	16	8
416	46	343	199	10	17	11
1370	141	78	32	31	62	47
2740	62	93	8	47	78	42
5480	78	156	96	31	156	33
10003	723	531	278	62	234	51
15483	798	601	501	88	305	99
Average	300.4	215.4	132.9	34.4	91	32.5

Table 4: Proposed method memory usage analysis

File size (in KB)	Memory usage (KB)		
	AES [6]	Blowfish [6]	Proposed
83.3	11014064	580824	259080
108	985312	1142800	209716
249	11244008	10968120	497720
333	2230250	10791776	971689
416	4186640	3010032	124168
1370	7063848	85711216	3672418
2740	6361832	17228432	1097168
5480	34660616	16506648	12031900
10003	60697448	59702576	20971687
15483	407289063	76209224	31241680
Average	54573308.1	28185164.8	7107722.6

Correspondingly, the average decryption time of the proposed methodology is 32.5 msec; for the existing methodologies, the average encryption times are 34.4 msec and 91 msec. The comparison of decryption time is shown in Fig. 5 and the performance analysis of the proposed and existing methods

is shown in Tab. 3. The encryption and decryption time show that the proposed method achieves higher performance for cloud data storage authentication than previous methods.

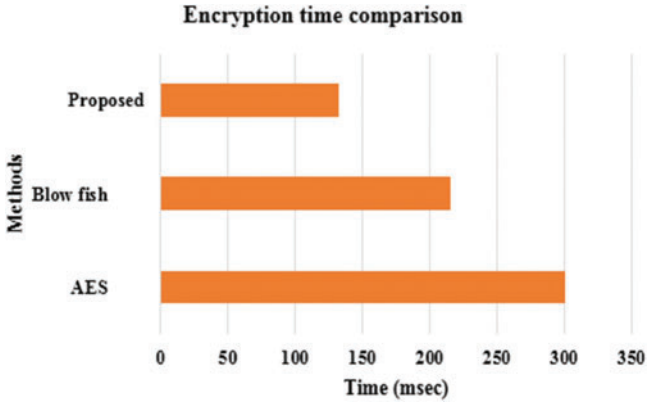

Figure 4: Encryption time of the proposed method

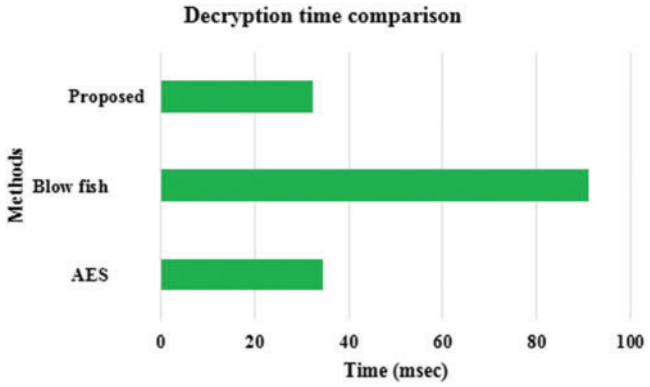

Figure 5: Graphical representation of decryption time comparison

AES [32] is a symmetric Feistel block cipher protocol that utilizes a similar key for both the decryption and encryption processes. This protocol contains fourteen rounds and only accepts a block size of 128 bits. By choice, it contains another two keys with a length of 192 and 256 bits and variable rounds of 10 and 12, respectively.

Blowfish [33] is another Feistel symmetric structure algorithm that consists of a 64-bit block size, which varies from 32 to 448 bits with 16 rounds. Also, the Blowfish algorithm contains a large key-dependent box for encryption, and it uses 4 S-boxes for decryption. In this algorithm, the level of security depends on the size of the key. The Blowfish algorithm is easily affected by the different key attacks because several rounds are used as a masker key that makes the algorithm infeasible [34,35].

5.4 Quantitative Analysis for Memory Usage

The memory usage of the proposed and existing methods is analyzed in this section. In Tab. 4, the proposed approach (enhanced Merkle hash tree with multi-owner authentication) outperforms existing methods with an average memory usage of 7,107,722.6 KB. The existing methods duplicate the encrypted data with the tag that increases the computation time and memory of the cloud. The

proposed Merkle Hash Tree method stores the hash value in the data for the search process and encrypted data is applied with a label. The hash value and label require less memory than unstructured encrypted data. The memory usage averages of the existing methods AES and Blowfish [6] are 54,573,308.1 KB and 28,185,164.8 KB. Tab. 4 shows that the proposed enhanced Merkle hash tree with multi-owner authentication method performed effectively compared with the existing methods. The memory usage graphical comparison is represented in Fig. 6.

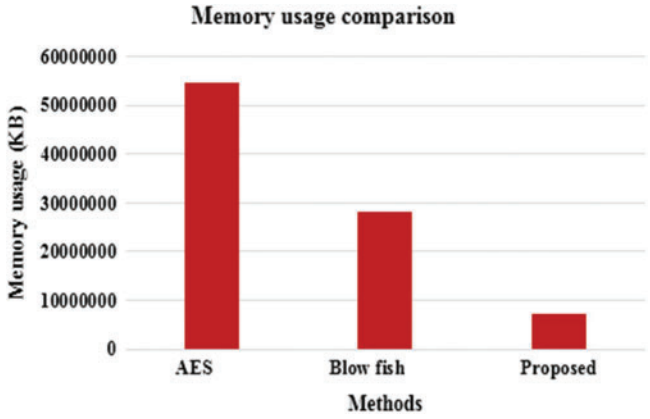

Figure 6: Graphical representation of memory usage comparison

After inspecting Tabs. 3 and 4, it was determined that the tiger hash function-based Merkle hash tree approach shows better performance in terms of memory usage, encryption time, and decryption time, whereas Blowfish is the second-best algorithm for cloud data storage authentication.

The running times of the proposed and existing methods [25,26] are shown in Fig. 7. The AAT [25] requires more computational time than the enhanced Merkle hash tree with a multi-owner authentication method due to the generation of tags in the method, and the elliptic curve cryptography method [26] requires a greater computational time. The proposed method requires less computation due to the tiger hash method.

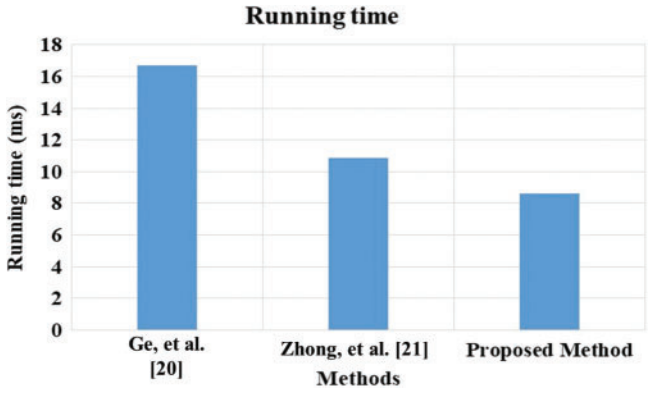

Figure 7: The user side running time of the proposed method

The server-side runtime of the enhanced Merkle hash tree is compared with the existing methods, as shown in Fig. 8. The AAT [25] requires greater computational time on the server-side to analyze the tags, and the elliptic curve cryptography method [26] requires more time for the decryption process. The

tiger hash tree in the Merkle method reduces the decryption process on the server-side. The proposed enhanced Merkle hash tree method has a server-side runtime of 8.14 ms, and the AAT [26] method requires 10.88 ms of runtime. This shows that the proposed enhanced Merkle hash tree method has a lower computation time than the existing methods.

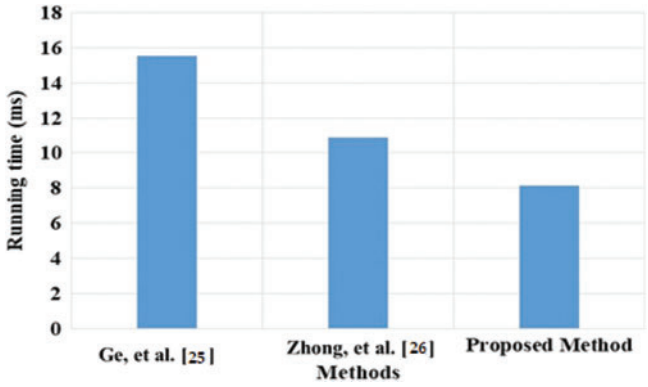

Figure 8: The server-side runtime of the proposed Method

6 Conclusion

In the current decade, the cloud computing paradigm has become standard for computer services due to its flexible computational abilities and high storage capacity. In this article, a new authentication methodology is presented to further ensure the security of cloud data storage. An enhanced Merkle hash tree with a multi-owner authentication technique is used for cloud data security. The enhanced Merkle hash tree approach algorithm encrypts user data and stores it in the cloud. The stored data is retrieved using a decryption function based on a user query. The proposed multi-owner cloud authentication method delivers an effective performance compared with other approaches for cloud data storage authentication. The experimental analysis shows that the proposed enhanced Merkle hash tree method has lower memory usage and lower encryption and decryption times. The proposed methodology saves between 2 and 167 msec of encryption and decryption time over existing methods (AES and Blowfish). In future work, a secure relevant data retrieval approach based on elliptic curve cryptography is incorporated in the cloud data storage application to increase the resilience against various attacks.

Author Contributions: The paper investigation, resources, data curation, writing—original draft preparation, writing—review and editing, and visualization were done by J.S. and K.B. The paper conceptualization, software, validation, and formal analysis were done by C.I., S.C.N, R.S, M.K.H. Methodology, supervision, project administration, and final approval of the version to be published were conducted by A.S. C:I and B.D.P. All authors have read and agreed to the published version of the manuscript.

Funding Statement: The Universiti Kebangsaan Malaysia (UKM) Research Grant Scheme FRGS/1/2020/ICT03/UKM/02/6 and GGPM-2020-028 funded this research.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Abdel-Basset, M. Mohamed and V. Chang, "NMCDA: A framework for evaluating cloud computing services," *Future Generation Computer Systems*, vol. 86, pp. 12–29, 2018.

[2] S. Roy, S. Chatterjee, A. K. Das, S. Chattopadhyay, N. Kumar *et al.,* "On the design of provably secure lightweight remote user authentication scheme for mobile cloud computing services," *IEEE Access*, vol. 5, pp. 25808–25825, 2017.

[3] H. Li, Y. Yang, Y. Dai, S. Yu and Y. Xiang, "Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 484–494, 2017.

[4] V. K. A. Sandor, Y. Lin, X. Li, F. Lin and S. Zhang, "Efficient decentralized multi-authority attribute based encryption for mobile cloud data storage," *Journal of Network and Computer Applications*, vol. 129, pp. 25–36, 2019.

[5] S. A. Butt, M. I. Tariq, T. Jamal, A. Ali, J. L. D. Martinez *et al.,* "Predictive variables for agile development merging cloud computing services," *IEEE Access*, vol. 7, pp. 99273–99282, 2019.

[6] V. Odelu, A. K. Das, S. Kumari, X. Huang and M. Wazid, "Provably secure authenticated key agreement scheme for distributed mobile cloud computing services," *Future Generation Computer Systems*, vol. 68, pp. 74–88, 2017.

[7] A. E. Eshratifar, M. S. Abrishami and M. Pedram, "Joint DNN: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 565–576, 2019.

[8] H. Hassan, A. I. El-Desouky, A. Ibrahim, E. S. M. El-Kenawy and R. Arnous, "Enhanced QoS-based model for trust assessment in cloud computing environment," *IEEE Access*, vol. 8, pp. 43752–43763, 2020.

[9] K. Riad, R. Hamza and H. Yan, "Sensitive and energetic IoT access control for managing cloud electronic health records," *IEEE Access*, vol. 7, pp. 86384–86393, 2019.

[10] S. Namasudra, "An improved attribute-based encryption technique towards the data security in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 3, pp. e4364, 2019.

[11] A. Fu, S. Yu, Y. Zhang, H. Wang and C. Huang, "NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Transaction on Big Data*, 2017.

[12] S. M. Hezavehi and R. Rahmani, "An anomaly-based framework for mitigating effects of DDoS attacks using a third party auditor in cloud computing environments," *Cluster Computing*, vol. 23, no. 4, pp. 2609–2627, 2020.

[13] A. Mitra, A. Kundu, M. Chattopadhyay and S. Chattopadhyay, "A cost-efficient one time password-based authentication in cloud environment using equal length cellular automata," *Journal of Industrial Information Integration*, vol. 5, pp. 17–25, 2017.

[14] C. Guo, N. Luo, M. Z. A. Bhuiyan, Y. Jie, Y. Chen *et al.,* "Key-aggregate authentication cryptosystem for data sharing in dynamic cloud storage," *Future Generation Computer Systems*, vol. 84, pp. 190–199, 2018.

[15] A. A. Ahmed, K. Wendy, M. N. Kabir and A. S. Sadiq, "Dynamic reciprocal authentication protocol for mobile cloud computing," *IEEE Systems Journal*, vol. 15, no. 1, pp. 727–737, 2020.

[16] M. K. Hasan, S. Islam, R. Sulaiman, S. Khan, A. H. Hashim *et al.,* "Lightweight encryption technique to enhance medical image security on internet of medical things applications," *IEEE Access*, vol. 24, no. 9, pp. 47731–47742, 2021.

[17] I. Memon, R. A. Shaikh, M. Hasan, R. Hassan, A. Haq *et al.,* "Protect mobile travelers information in sensitive region based on fuzzy logic in technology," *Security and Communication Networks*, vol. 2020, pp. 1–12, 2020.

[18] M. M. Saeed, M. K. Hasan, R. Hassan and R. Mokhtar, "Preserving privacy of user identity based on pseudonym variable in 5 g," *Computers Material and Continua*, vol. 70, no. 3, pp. 5551–5568, 2022.

[19] M. K. Hasan, M. Ahmed, A. H. Hashim, A. Razzaque, S. Islam *et al.,* "A novel artificial intelligence based timing synchronization scheme for smart grid applications," *Wireless Personal Communications*, vol. 114, no. 2, pp. 1067–1084, 2020.

[20] M. K. Hasan, M. Shafiq, S. Islam, B. Pandey, B. E. Ya *et al.,* "Lightweight cryptographic algorithms for guessing attack protection in complex internet of things applications," *Complexity*, vol. 5, no. 5, pp. 1–13, 2021.
[21] A. R. Shajina and P. Varalakshmi, "A novel dual authentication protocol (DAP) for multi-owners in cloud computing," *Cluster Computing*, vol. 20, no. 1, pp. 507–523, 2017.
[22] S. Anand and V. Perumal, "EECDH to prevent MITM attack in cloud computing," *Digital Communications and Networks*, vol. 5, no. 4, pp. 276–287, 2019.
[23] G. Deep, R. Mohana, A. Nayyar, P. Sanjeevikumar and E. Hossain, "Authentication protocol for cloud databases using blockchain mechanism," *Sensors*, vol. 19, no. 20, pp. 4444, 2019.
[24] A. M. Badr, Y. Zhang and H. G. Ahmad Umar, "Dual authentication-based encryption with a delegation system to protect medical data in cloud computing," *Electronics*, vol. 8, no. 2, pp. 171, 2019.
[25] X. Ge, J. Yu, H. Zhang, C. Hu, Z. Li *et al.,* "Towards achieving keyword search over dynamic encrypted cloud data with symmetric-key based verification," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 490–504, 2019.
[26] H. Zhong, C. Zhang, Y. Xu and L. Liu, "Authentication and key agreement based on anonymous identity for peer-to-peer cloud," *IEEE Transactions on Cloud Computing*, 2020.
[27] J. Wang, X. Chen, X. Huang, I. You and Y. Xiang, "Verifiable auditing for outsourced database in cloud computing," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3293–3303, 2015.
[28] 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.
[29] C. C. Erway, A. Küpçü, C. Papamanthou and R. Tamassia, "Dynamic provable data possession," *ACM Transactions on Information and System Security*, vol. 17, no. 4, pp. 15, 2015.
[30] J. Mao, Y. Zhang, P. Li, T. Li, Q. Wu *et al.,* "A position-aware Merkle tree for dynamic cloud data integrity verification," *Soft Computing*, vol. 21, no. 8, pp. 2151–2164, 2017.
[31] C. K. Chu, S. S. Chow, W. G. Tzeng, J. Zhou and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *EEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 468–477, 2014.
[32] N. Shimbre and P. Deshpande, "Enhancing distributed data storage security for cloud computing using TPA and AES algorithm," in *Proc. Int. Conf. on Computing Communication Control and Automation*, Pune, India, pp. 35–39, 2015.
[33] S. Mudepalli, V. S. Rao and R. K. Kumar, "An efficient data retrieval approach using blowfish encryption on cloud ciphertext retrieval in cloud computing," in *Proc. Int. Conference on Intelligent Computing and Control Systems*, Madurai, India, pp. 267–271, 2017.
[34] C. O. Iwendi and A. R. Allen, "Enhanced security technique for wireless sensor network nodes," in *Proc. IET Conference on Wireless Sensor Systems (WSS 2012)*, Madurai, India, pp. 1–5, 2012.
[35] C. Iwendi, Z. Zhang and X. Du, "ACO based key management routing mechanism for WSN security and data collection," in *Proc. IEEE Int. Conference on Industrial Technology (ICIT)*, Lyon, France, pp. 1935–1939, 2018.