JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 40, 1139-1157 (2024)
DOI: 10.6688/JISE.202409_40(5).0013

Blockchain-Based Secure Deduplication using Robust Systematic Cryptographic Methods

S. RUBA+ AND A. M. KALPANA
Department of Computer Science and Engineeing
Government College of Engineering
Tamil Nadu, 636011 India
E-mail: {ruba+; drkalpana}@gcesalem.edu.in

In recent years, people are drastically using the internet. Their daily lives have grown to depend on it, and it now connects to their social lives. In addition, internet usage has increased as a result of the COVID pandemic. Data growth and the associated need for storage space have emerged as major issues in cloud storage management. To improve the efficiency of cloud storage and save bandwidth of network data transmission, the duplication of cloud data has become a research goal, especially in the field of encrypted cloud data storage. Deduplication is a technique for removing data redundancy that was created to reduce redundant data in cloud storage and conserve system storage resources. Nowadays, due to the growth of cloud computing technologies, duplication techniques are increasingly used in cloud service centers. Therefore, many researchers have adopted several deduplication methods to remove redundant data from cloud storage. For secure deduplication, previous works have typically used third-party auditors to verify data integrity, but it may be suffered from data leaks by third-party auditors. Conventional methods also could not face more difficulties in the deduplication of big data to properly consider the two conflicting goals of high deduplication ratio and deduplication efficiency. In this paper, an improved blockchain-based secure data deduplication is presented with efficient cryptographic methods to save cloud storage securely. In the proposed method, an Attribute-based Role Key Generation (ARKG) method is implemented in a hierarchical tree manner to generate a role key when the data owners upload their data to the cloud service provider (CSP) and to allow authorized users to download the data. In the storage system, the smart agreement (agreement between the data owner and CSP) is done using SHA-256 (Secure Hash Algorithm-256) to generate a tamper-proofing ledger for data integrity, in which data is protected from illegal modifications and duplication detection is executed through hash-tag that can be formed by SHA-256. Content Sealed Encryption (CSE) is employed to encrypt data for data uploading by the data owners to the CSP. The experimental results show that the proposed secure deduplication scheme is efficient according to data deduplication, storage costs, computational costs, and computation delay and it can give higher throughput and a low duplicate elimination ratio.

Keywords: secure data deduplication, cloud storage, blockchain, role key generation, SHA-256, CSE, chunking

1. INTRODUCTION

Cloud computing has been employed to store, manage and process the data as alternating storage for data backup with limited local storage space. For data storage, cloud

Received February 7, 2023; revised May 27, 2023; accepted July 27, 2023.
Communicated by Raylin Tso.
+ Corresponding author.

storage is utilized by several big companies. Cloud storage might be suffered from several difficulties such as security issues, storage space issues, *etc*., due to a huge quantity of backup storage constraints. In cloud storage services, one challenge is the vertical amount of duplicate data. Particularly, there could be almost 68% of duplicate data is obtainable on normal file systems, and up to 90-95% of duplicate data is available on backup applications [1]. As a result, issues of storage space are managed through the data deduplication process, in which the storage space effectiveness can be enhanced. De-duplication technology is frequently used by cloud computing services, utilizing carefully chosen hash or string algorithms as labels to prevent uploading the same data repeatedly to reduce administrative expenses and increase storage availability [2]. In the deduplication techniques, the cloud is allowed to store only one copy of duplicate data, and links are given to that copy when required. The deduplication is mostly done through hash values (fingerprints) for the files and chunks representation wherever these values can be compared with others to establish if the chunk or file they symbolize will be a duplicate or not. The fingerprints are stored in an index structure that can be too large to fit in memory. Therefore, it will be stored in an on-disk structure that will direct to a popular difficulty called disk index lookup bottleneck. In general, the data deduplication process is done on the block level or file level. In the block-level data deduplication, the files are segmented into blocks and only a copy of every block is stored. Block-level deduplication can thus eliminate duplicate data blocks in non-identical files and it is used to improve storing efficiency more compared to file-level data deduplication. Thus, we focused on block-level data deduplication in this paper. However, data deduplication can direct the tension between data confidentiality and storage efficiency. Mostly, cloud users have tension about the confidentiality of their sensitive data hosted by cloud servers that could not be trusted.

The rapid growth of cloud computing, an efficient data security system becomes important for cloud computing-based environments. Encryption algorithms not only play an important role in information security systems but also play an important role in many cloud computing-based systems. For the encryption process, Elliptic Curve Cryptography (ECC) is used as public-key cryptography, based on elliptic curves over finite fields.

The main function of ECC is point addition, which can be very computationally expensive and it takes more time for generating a key pair. And also, security threats are possible and energy consumption may have been raised when performance evaluation is performed on different cloud clients [3].

A simple idea for data confidentiality consideration can be to let every cloud data owner outsource encrypted data to the cloud server that does not recognize the decryption key. In the traditional encryption process, every user needs to employ a different key to encrypt the data. Consequently, similar data for different users can appear as completely different ciphertexts. In the data deduplication methods, the cloud server needs to recognize the same data. Thus, natural conflict may have happened between data confidentiality and storage efficiency. As a result, improved data deduplication is presented in this paper with a computing environment for efficient storage in a more secure manner. Now a day, the blockchain-based security method is mostly focused on a distributed computing paradigm for its low cost, high efficiency, high credibility, and high data security. It is a key technology that has been derived from the agreement system that can be an instance of a disseminated computing scheme with higher fault tolerance. In the blockchain technique, point-to-point transactions or collaboration are realized in disseminated systems

with high traceability. For example, decentralized storage systems such as File coin have taken blockchain's benefits, in which security file storage could be achieved using an incentive mechanism and distributed structure. But none of these schemes have attempted to achieve the storage effectiveness that can be achievable through deduplication. In this paper, a cloud storage deduplication system is presented with high fault tolerance to overcome the issues of integrity, confidentiality, and reliability. For data integrity and system confidentiality, blockchain technology can be exploited. The communication frequency can also be reduced between the data owner and cloud servers, where the risk of communications for monitoring and information leakage is reduced. The authors in [4] proposed the basic number that combines the discrete logarithm challenge and the Chinese residual theorem, absolute safety is realized in the proof that attempts to address the confidentiality leak issue. In the proposed system, a smart agreement is done with SHA-256, and the data owner can encrypt their data by CSE scheme to make sure file confidentiality in the case of showing the files information on the blockchain. Moreover, the role key is generated according to the user's attributes hierarchically for data uploading to a cloud storage server, in which authorized users are only allowed to access the data.

The paper can be organized as follows: the related works are reviewed in Section 2 based on the secure deduplication in cloud storage. In Section 3, the presented secure duplication algorithm is discussed in detail. After that, the presented method is evaluated and compared to other methods applied to deduplication in Section 4. The conclusion can be discussed briefly in Section 5.

2. RELATED WORKS

The Secure Deduplication and Virtual Auditing of Data have been presented in Cloud called (SDVADC) [5], in which integrity auditing and deduplication of information have been realized in the cloud. Secure deduplication of information was supported by this method and effective virtual auditing of the documents was done in the download process. However, a duplicate chunk detection problem may have occurred. A novel method has been presented in [6] using Convergent and Modified Elliptic Curve Cryptography (MECC) algorithms over the cloud and fog environment for secure deduplication system construction. The file encryption was done initially with the help of the Convergent Encryption technique and afterward, re-encryption was done using MECC. However, the data may be obtained by the opponent through the brute-force attack when the data belongs to a knowable set for CE. A Collaborative Edge (CE) network combined is presented in [7] with the CE-D2D framework for solving maximizing video caching with efficient cellular and bandwidths. The proposed system has been using a strategy of caching only the watching chunk videos instead of offloading and caching video by one edge node. The ZEUS (zero-knowledge deduplication response) system was presented in [8]. The authors have developed ZEUS and ZEUS+ as two privacy-aware deduplication protocols: in which weaker privacy guarantees were given by ZEUS as being more capable in the communication cost, as ZEUS guarantees stronger privacy properties, at an enhanced cost of communication. However, low deduplication throughput may have happened during the ZEUS process. A security evaluation model was presented in [9] that consists of API (Application Programming Interface) from different clouds evaluating including video

discovery, security recovery engine, scanning engine, quantifiable evaluation, *etc*. However, energy consumption may have increased during the process of measurable evaluation in various cloud clients. The RSA (Rivest–Shamir–Adleman) based Cross-Domain Secure Deduplication (RCDSD) of synchronization was presented in [10] between dispersed storage managers without enlightening a lot of information about the real data stored by the customers. The synchronization part has been done through an interactive protocol of collaboration between the servers. But computing overhead may have happened in the process of CSPs' distribution of duplication information to the users through the domain managers. The investigation of backup storage setting enterprise of secure chunk-based deduplication method is done in [11]. The authors proposed a randomized key generation based on their inner works of backup service. Conversely, low chunking throughput can happen. Three stages are done in [12] for deduplication such as chunking, fingerprinting, and indexing fingerprints. In chunking data files are divided into chunk boundary is decided by the divisor values. For each one, there may be a unique value of identification that is computed through a hash signature (*i.e*., MD-5 (Message Digit-5), SHA-1, SHA-256), which is called a fingerprint. Finally, fingerprints are stored as an index term to detect replication chunks by comparing the same fingerprints. However, in this method, high-cost computing, and medium-security happened. Two secure data deduplication methods were proposed in [13] according to the Rabin fingerprinting in wireless sensing data of cloud computing. In the first system, deterministic tags and the other one adopts random tags were applied. But low secure deduplication may have happened through Rabin fingerprinting. Secure data deduplication is focused on cloud storage [14]. Compared to the traditional detection approaches, machine learning is a novel and flexible method to detect intrusions in the network, it applies to any network structure. This paper proposed the challenges of anomaly detection in the traditional network, as well as in the next-generation network, and reviews the implementation of machine learning in anomaly detection under different network contexts. However, computing overhead may have occurred during the process of chunking of original data.

In 2022, the authors in [15] have proposed a duplication strategy, this technique was used to reduce the amount of redundant data which was stored in the system. It was suggested to remove duplicate data from the cloud using a revolutionary method that also saves bandwidth and storage. According to the experiment results, the suggested method provides greater security for cloud-stored data while resolving the primary issues with the existing solutions. They developed distributed and single-server storage systems that offered data security and space efficiency. So, the same chunk was always encrypted with the same chunk text, because the chunk data is constantly generating encryption keys. The authors in [16] introduced a novel secure encrypted data deduplication scheme with dynamic auditing, named SecDedup. SecDedup uses homomorphic authentication and designs a multi-functional data tag that supports both duplication and audit simultaneously with lightweight storage overhead. To enhance deduplication security, we develop a Proof of Ownership (PoW) verification algorithm that protects against file ownership spoofing attacks, a tag consistency verification algorithm that protects against duplication, and an auditing mechanism that protects against file tampering attacks. This means that the system can protect against attacks by malicious data owners as well as malicious CSPs. In addition, SecDedup also supports batch auditing, which optimizes computation and data transfer costs for auditing multiple deduplication operations. The authors in

[17] proposed a data-sharing system using the ABE method that further supports deduplication. Blockchain technology is used to allow users to quickly verify the legitimacy of encrypted text. Data is checked to see if it is duplicate ciphertext to improve memory space. In addition, cloud servers use re-randomization technology to improve the efficiency combination of ciphertexts.

In this work, the hash codes have been generated with the help of the Secure Hash Algorithm (SHA-512) [18]. According to this hash code-verifying system, authentication is provided. For secure Data Deduplication, cloud storage, and service providers are permitted to use data with confidentiality. However, high key utilization is done by SHA-512 for deduplication and thus, low throughput can have happened.

3. PROPOSED METHODOLOGY

This research work proposes an enhanced blockchain-based secure data uploading and downloading processes for more secure deduplication in a system with high throughput and a lower deduplicate elimination ratio to save cloud storage given in Figs. 1 and 2. The data uploading and downloading procedure can avoid data duplication by using the following techniques.

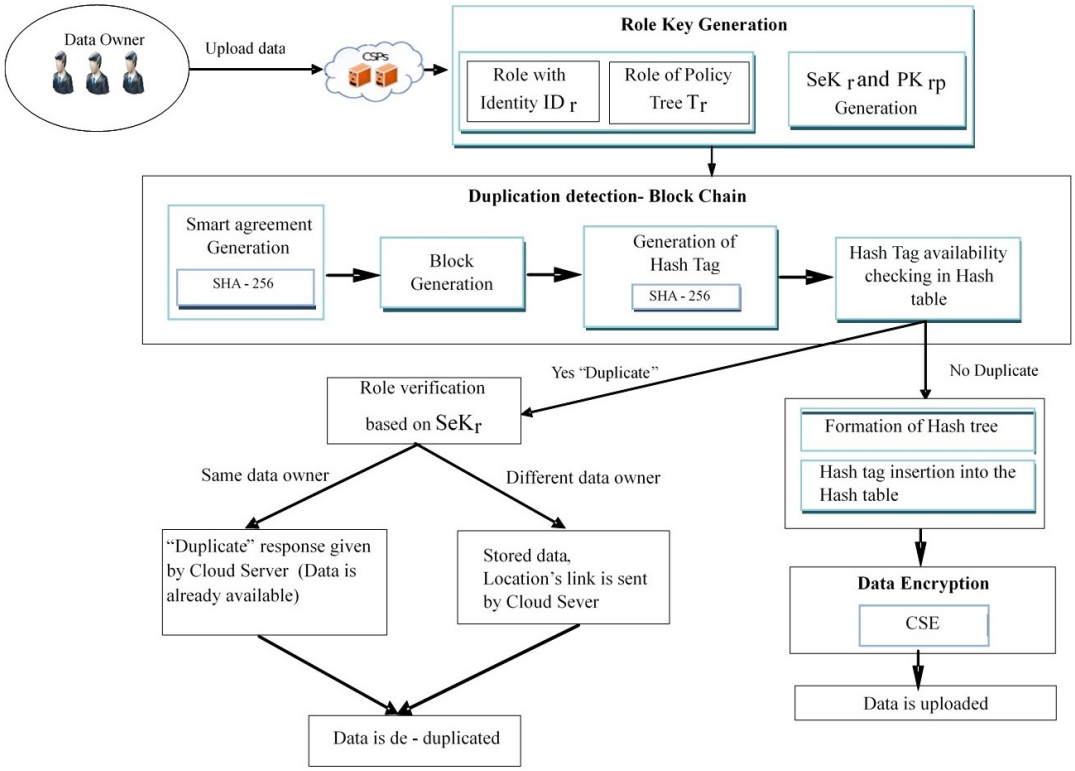

Fig. 1. Proposed data uploading to the Cloud Server (CS).

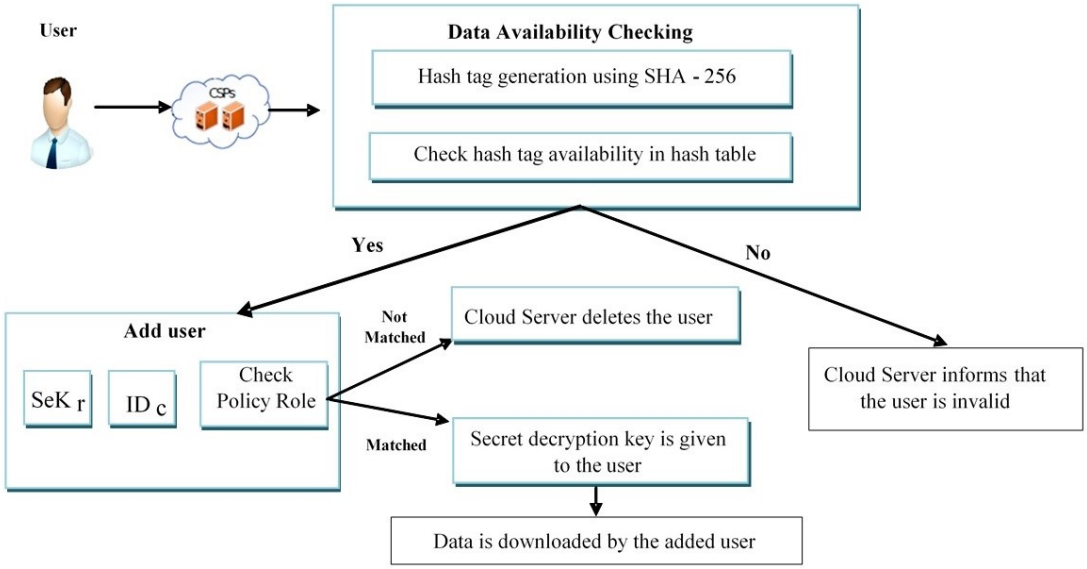

Fig. 2. Data downloading using the proposed scheme.

3.1 Role Key Generation

When a data owner uploads data to a cloud server, the secret role key SeKr and the public key role parameter PKrp are generated following the identity and policy role. Only authorized users can access the data [19].

3.2 Duplicated Data Checking

The SHA-256 method used for hashtag and hash table generation to validate tag availability before data uploading by the data owner is used to reach an agreement between CSP and data owner [20] and can be used to check for duplicate data during the block-chain process. The cloud server can determine whether the same tag is present in the hash table upon receiving data. Here, the cloud server responds to the data owner with "no duplicate" or "duplicate".

3.3 Encryption

If the data owner gets "duplicate" with a file pointer, then the data is not needed to upload. If the data owner gets a "no duplicate" reply from a cloud server, then data will be encrypted by applying the CSE and after that, encrypted data is uploaded to the cloud server with generated secret role keys [21]. If the same data is available in a different data owner storage space, the cloud server sends the stored location link.

3.4 Secure Data Downloading

For secure data downloading, the role is assigned by the cloud server to the user (data consumer) if the user's attributes can satisfy the role policy, in which the role policy is checked through generated identity role IDr and role secret key SeKr [22]. In this process,

the data consumer can be added according to the policy role. Thus, added authorized users can only download the requested data from the cloud server [23]. The proposed phases are described in detail below:

3.4.1 Attribute-based role key generation

In this paper, attribute-based role key generation is done before the uploading of data to a cloud server. If the data owner wants to outsource some data to cloud storage, the data should be encrypted before being outsourced. Consequently, the data owner can be responsible to establish the access policy for each data, establish the roles, encrypting the data under the access policy, and master public and private keys can be generated [24, 25]. In this proposed methodology, a hierarchical tree is applied for the access policy structure representation. Here T is an access tree containing interior nodes that can be utilized as threshold gates such as AND/OR gates as the leaves are connected with attributes. A data consumer will be allocated to a role and as a result, ciphertexts will be decrypted corresponding to that role if and only if there can be an assignment of the attributes from the private key of the data consumer to the tree's leaf nodes such that the tree will be satisfied [26, 27]. For an example of attributes, in the university domain, the staff may be in the position of "Assistant professor," or "lecturer", *etc*., the courses are in the form of "commerce," "medicine", and "computer sciences". The data owner is responsible for the following three processes.

3.4.2 Setup

In this process, the security parameter is taken as input and a master key and a group public key is given as output.

3.4.3 Role creation

In this process, the role with identity IDr and policy tree of role Tr are taken as inputs, the secret key role SeKr is generated and the role's public parameters set PKrp is returned and after that, an empty user list will be provided, in which list users can be listed whose attributes matched with the role policy tree.

3.4.4 Data encryption

In the proposed methodology, the encryption process executed by the data owner after the duplication-checking process using CSE and ciphertext C can be outsourced to the cloud.

Algorithm 1: Attribute-based role key generation

Input: Role with identity IDr, Role with policy tree Tr;
Output: Secret key role SeKr, Public key role parameter PKrp.
Step 1: To state a role with policy tree Tr over an attribute set 'S'
Step 2: Assume thr_a can be the node's threshold value for every node 'a' in the hierarchical tree T_R
Step 3: If the non-leaf node contains a logical gate of "OR"

Then thr_a=1
If the non-leaf node contains a logical gate of "AND" Then thr_a=Number of its children
Step 4: Select a polynomial as Y_a initiating from the tree's root node A with its degree and is given by

$$D_a = thr_a - 1 \tag{1}$$

Step 5: If the node 'A' can be the root
Then, Select a random value of tree $v \in Z_X^*$ and Set $Y_a(0)$=v
Else

$$SetY_a(0) = Y_X(a)(index(a)) \tag{2}$$

where $X(a)$ denotes node a's parent node and a unique index number will be assigned in an $index(a)$ to every node in Tr.
Step 6: To assume P can be a leaf node set in Tr.
Step 7: To generate a secret key role using the following Eq. (3),

$$SeK_r = \left(K_r, \left(\forall_p \in P : C_p = g^{Y_p(0)},\ C_p^{'}\text{=}H(p)^{Y_p(0)v}\right)\right) \tag{3}$$

where $K_r = g^{\frac{1}{s+H_1(ID_r)}}$, C_p: cipher text for leaf node P, g: key generation, H: hierarchical tree manner.
Step 8: To generate public role parameters PK_{rp} using the following Eq. (4),

$$PK_{rp} = \left(h^{(s+H(ID_r))\prod_{k=1}^{n}\left(s+H\left(ID_{r_k}\right)\right)}, \left(h^{(s+H(ID_r))\prod_{k=1}^{n}\left(s+H\left(ID_{r_k}\right)\right)}\right)^m\right) \tag{4}$$

Where $(ID_{r_1}, ID_{r_2}, \ldots, ID_{r_m})$ can be the identities role r's every predecessor roles.

In the above Algorithm 1, the role with identity IDr and policy tree of role Tr are used as inputs, the secret key role SeKr is created and the role's public parameter set PKrp is returned and after that, an empty user list will be provided, in which users can be listed whose attributes S matched with the role policy tree Tr. In this algorithm, a hierarchical tree is applied to the representation of the access policy structure. Here, T is an access tree containing internal nodes that can be used as threshold gates, like AND/OR gates, because the leaves are related to attributes. A data consumer is assigned a role, and as a result, ciphertexts are decrypted according to that role, if and only if the attributes of the data consumer's private key can be assigned to the leaf nodes of the tree such that the tree is satisfied. Thus, the role has been generated in the hierarchical tree manner when the data owners upload their data to cloud service providers, and therefore, authorized data consumers are only allowed for data accessing or downloading.

3.5 Duplicated Data Checking

Before the data uploading, the duplicated data is checked through the blockchain process, in which the smart agreement is done using SHA-256 to ensure the integrity and confidentiality of the data stored on CS (Cloud Server) and to avoid illegal data uploading

and modifications [28]. Compared to SHA-512, SHA-256 can provide shorter outputs that save bandwidth, and thus, we have chosen SHA-256. The SHA-256 hashing function will be utilized in the Bitcoin process of blockchain to generate smart agreements between the data owner and CSP for duplicated data checking process. The SHA-256 hash function has been illustrated in Fig. 3.

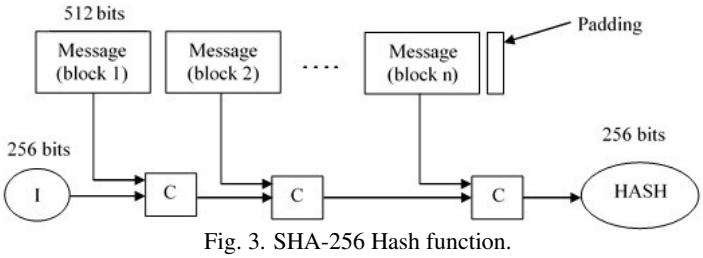

Fig. 3. SHA-256 Hash function.

In this process, the hash code key is generated by the cloud server using SHA-256 when the data owner tries to upload new data. The input file can be added with padding and a fixed 64-bit length field. The input enlarged data (message) is chopped into 512 bits-sized files (messages). These messages are in multiple blocks with extra 64 bits of padding only for the final message. From the input data, the first message and 256 bits are provided to C which can be a compression process, in which 256-bit output is produced that can be an input for a second message that goes through C to produce 256 bits. When continuing this process until the end of every message, the input file's HASH is obtained in 256 bits by the cloud server. Thus, the original input data has been divided into several blocks. Afterward, for each block, the hashtag values are allocated [29]. For a particular block's tag value, a hash code is generated by applying the same SHA-256 algorithm. Here, hashtag values can be shown in the blockchain with transaction information about the file's storage payment. The data owner can check duplication check through a cloud server locally as this is unchangeable information. In the duplicated data-checking process, cloud servers check the hashtag is available in the hash table. If the hashtag is already available in the hash table, the cloud server replies "duplicate" to the data owner, in which role verification is done according to the secret role key SeKr, and thus, data is not required to be uploaded by the same data owner. If the same input files are obtainable in different owner data storage spaces, then the cloud server sends the stored data's location link [30]. For example, when the data owner tries to upload the same data again, the hash value is calculated with the SeKr using SHA-256. Here, the input file is divided into various blocks. In the proposed system, the block sizes 10 MB, 20 MB, 30 MB, and 40 MB are considered. For every divided block, the tag key has been generated. After that, for every tag key, the hash value can be calculated by SHA-256. The hashtag is checked for an appropriate file by the cloud server during the data uploading process. If the input data's hashtag is already in the hash table, the queries asked by the CSP "What is the hash tree's path?". If the data owner gives the correct path, the data owner's SeKr is verified by the CSP. If the SeKr is the same, the data is not stored again by the cloud server. The hash tree path is in the form of $P(HT)$=$RRL, RLR, RRR, etc.$, where, HT: Hash Tree, RRL: Root, Right, Left, RLR: Root, Left, Right, RRR: Root, Right, Right. Therefore, the leaf node is not needed to add to the hash tree. The same data owner tries

to upload the same data is estimated by the following Eq. (5),

$$S_{do} \overset{uploaddata}{\rightarrow} S_d \left(CSP \overset{replies}{\rightarrow} duplicate \right). \tag{5}$$

Where S_{do} denotes the same data owner, S_d represents the same data. If the SeKr is different, the stored particular data location's reference link is sent by the cloud server to the different data owner and it is expressed by the following Eqs. (6)-(8),

$$D_{do} \overset{uploaddata}{\rightarrow} S_d \left(CSP \overset{ask}{\rightarrow} SeK_r \& P(HT) \right), \tag{6}$$

$$(SeK_r \& P(HT))_{matched} \rightarrow S_d(CSP) \overset{send}{\rightarrow} R_l(d), \tag{7}$$

$$(SeK_r \& P(HT))_{notmatched} \rightarrow S_d(CSP \overset{replies}{\rightarrow} Invalid\_user). \tag{8}$$

Where D_{do} denotes the different data owner and $R_l(d)$ represents the reference link of particular stored data. If the hashtag is not available in the hash table, the cloud server replies "no duplicate" to the data owner, and encrypted input data is uploaded to the cloud server. Here, the encryption is done through the CSE method. The data owner first puts the decryption key in the encryption area before encrypting input data. Efficient data confidentiality is provided by the CSE function. The six primitive functions are utilized in this paper for data encryption such as $CSE = (ParGen, KeyGen, Enc, Dec, Equ, Valid)$ working with space of plaintext, ciphertext, keyspace such as $M = \{M_\lambda\}_{\lambda \in N}$, $C = \{C_\lambda\}_{\lambda \in N}$, and $K = \{K_\lambda\}_{\lambda \in N}$. For data encryption, the following processes are done before the data uploading to the cloud server. For data encryption, the following processes are done before the data uploading to the cloud server, with the use of Eqs. (9)-(14).

1. *ParGen* represents the parameter generation algorithm that takes 1^λ as input and returns public parameters pp as output:

$$PP \leftarrow KeyGen(1^\lambda). \tag{9}$$

2. *KeyGen* represents the random key generation algorithm that takes public parameters pp and message M as inputs and returns a key to the message

$$SK_M \leftarrow KeyGen_{pp}. \tag{10}$$

3. *Enc* is the encryption algorithm that takes public parameters PP, a message M, and generated secret key SK_M as inputs. It returns a cipher text:

$$C \leftarrow Enc_{pp}(SK_M, M). \tag{11}$$

4. *Dec* represents the decryption algorithm that takes PP, C, and SK_M as inputs and outputs a message M:

$$M \leftarrow Dec(SK_M, C). \tag{12}$$

5. Equ is the equality algorithm that takes as PP, two cipher texts C_1 and C_2 as inputs, and it gives outputs as 1 when both C_1 and C_2 have been generated from the same message and is given by

$$Equ_{pp}(C_1, C_2) = 1. \tag{13}$$

6. $Valid$ represents the validity-test algorithm that takes PP and C as inputs and gives output as 1 if the C can be valid cipher text:

$$Valid_{pp}(C) = 1. \tag{14}$$

Thus, the encrypted data can be uploaded to the cloud storage server with generated key roles such as SeKr, and IDr that have been generated by the attribute role key generation.

Algorithm 2: File Uploading

Input: Original data;
Output: Deduplication, file uploading to the cloud server
Process:
Step 1: To initialize smart agreement with key K, tag T, message blocks $(MB_1, MB_2, MB_3, \ldots, MB_n)$, and hash tree HT
Step 2: To generate tag key K by applying the SHA-256 for appropriate data 'D'
Step 3: To divide 'D' into several message blocks $(MB_1, MB_2, MB_3, \ldots, MB_n)$
Step 4: To generate hashtag 'T' for the data 'D'
Step 5: To format the hash table using the same SHA-256
Step 6: To check whether the hashtag is obtainable in the hash table through the cloud server
Step 7: To verify the role of the data owner based on the SeKr, if the cloud server replies "duplicate". If the SeKr is the same, data is not needed to upload again. If the SeKr is different, then the reference link of appropriate stored data is sent to the data owner. Thus, data duplication is eliminated.
Step 8: To insert has tag into the hash table and encrypt data using CSE if the cloud server replies "no duplicate" and then encrypted data is uploaded with generated roles in the cloud server.
Step 9: Stop

In Algorithm 2, before the data is uploaded, the duplicated data is verified by the blockchain process where a smart agreement is done using SHA-256. The input data (D) is divided into several blocks $(MB_1, MB_2, MB_3, \ldots, MB_n)$. The cloud server generates a tag key K using SHA-256 for appropriate data 'D' when the data owner tries to upload new data. Afterward, the hashtag value T is generated for each block. In the process of verifying for duplicate data, the cloud servers check that the hashtag is available in the hash table. If the hashtag already exists in the hash table, the cloud server responds "duplicate" to the data owner, where the role is verified according to the secret key role SeKr, and thus, data is not required to be uploaded by the same data owner. If the secret

key role SeKr is different, the cloud server sends a reference link to the location of the specific data stored to the owner of the different data. If the hashtag is not available in the hash table, the cloud server replies "no duplicate" to the data owner, and the encrypted input data is uploaded to the cloud server. Here, the encryption is done through the CSE method.

3.6 Secure Data Downloading

For more secure data downloading by the data consumers, the tag value and generated roles are utilized in the proposed methodology. Initially, the data consumer sends particular data's tag value to the cloud server. After that, the hashtag value can be generated by applying the SHA-256 algorithm. Consequently, the cloud server verifies whether the tag value available in the hash table. If the tag value is not available in the hash table, then the cloud server considers that data consumer as an invalid user [31]. If the tag value is available in the hash table, then the following processes are done to download the data securely:

3.6.1 Add data consumer

This process can be done through the generated key roles that have been given in the above section (a). Cloud server takes SeKr (that contains access policy) and identity of data consumer (user) IDC as inputs. The cloud server can check whether the consumer attributes satisfy the role's access policy. If the attributes of the data consumer satisfy the role's access policy, then the data consumer is added and a secret decryption key is given to that added data consumer to download the data securely.

3.6.2 Decryption

This process can be done by the added consumer to decrypt the encrypted data outsourced in the CSP using CSE in Eq. (12). In this task, the data consumer's secret decryption key SK_M is taken as input and the plain text M is returned as output if the user or data consumer received permission from the cloud server to access the data otherwise, it gets fails.

3.6.3 Delete data consumer

In this process, SeKr and IDC are taken as input, the user is removed from the data consumer role's list RL, and the public key role parameter PKrp is updated. Thus, the data can be downloaded by authorized users only to avoid data duplication.

Algorithm 3: Data downloading

Input: Tag value, SeKr, IDC;
Output: Secure data downloading
Process:
Step 1: To initialize original data, hashtag value, SeKr, IDC
Step 2: To generate hashtag value for all tags by applying the SHA-256
Step 3: To check whether the hashtag value is obtainable in the hash table through the cloud server

Step 4: If the cloud server informs hash value is obtainable in the hash table, the data consumer is permitted to access the data for data downloading, otherwise, the cloud server informs that the consumer is invalid.
Step 5: To add data consumer or user if satisfies the role's access policy according to the SeKr, IDC, and to provide the secret decryption key to the added user.
Step 6: To decrypt the ciphertext C using CSE in Eq. (12)
Step 7: Data is downloaded by the added user securely
Step 8: Stop the algorithm

In Algorithm 3, for secure data downloading, the cloud server takes SeKr (that contains access policy) and the identity of data consumer (user) IDC as inputs. The hashtag value can then be generated using the SHA-256 algorithm. The cloud server checks whether the hashtag value is available in the hash table. If so, the data consumer is allowed to access the data for data downloading otherwise, the cloud server informs that the consumer is invalid. To add a data owner or user, the cloud server verifies that the data consumer's attributes satisfy the role's access policy. If the attributes of the data consumer satisfy the access policies of the role, the data consumer is added according to the SeKr, IDC, and provides the secret decryption key to the added user.

4. RESULTS AND DISCUSSIONS

In this section, the proposed deduplication system with an encryption and decryption process using key generation algorithms is evaluated. The experiment was conducted in Python and executed in the system environment of the Intel core I3 processor with 4 GB RAM, and Windows 10 operating system. In this experiment, initially, a user needs to register with the corresponding server. AES encryption techniques are used for user registration. While registering a unique role key is generated. Table 1 shows the parameters used for role key generation. After registration, the admin grants permission to the user with the registration confirmation email. After successful registration, the user can upload any form file format like text, image, video, audio, *etc*. Each file is attached with the generated role key.

Table 1. Role key generation.

User Name	Password	Role Identity, IDr	Policy Tree, Tr	Secret Key, SeKr	Public Key, PKrp

To encrypt the file in the uploading process, a blockchain algorithm is applied. The user selects a file to upload and the hash value of the file is generated. Table 2 shows the parameters used for the file-uploading process. It calculates the hashtag value and verifies the match. If the match is found, the file has been uploaded to the server as a successful process. Table 3 shows the parameters of the file-downloading process. Every upload and download process utilizes these table attributes for a flexible and secure process. It shows that the content-based deduplication process has more efficient. The performance of the proposed secure data deduplication and existing data deduplication methods such as SDVADC [5], ZEUS [8], and RCDSD [10] are simulated and compared using TensorFlow

Tool in terms of data deduplication rate, deduplication throughput, time complexity, and communication overhead.

Table 2. File uploading.

Original Data	Smart Agreement Generation	Block Generation	Hash Tag value Generation (SHA256)	Hash Table	File Uploading

Table 3. File downloading.

Hash Tag value	Hash Table	Cloud Server	Secret Key, SeKr	IDc	Verify Policy Role	Cloud Server

4.1 Data Deduplication Rate

The comparison of the data deduplication rate has been illustrated in Fig. 4. From the blow comparison chart, the proposed method improved blockchain-based deduplication has taken a high data deduplication rate compared to existing methods SDVADC [5], ZEUS [8], and RCDSD [10]. The proposed method has taken a 28% data duplication rate for the 5MB file while the existing methods SDVADC [5], ZEUS [8], and RCDSD [10] gave 23%, 22%, and 26%. The proposed method has given a 26.7% data duplication rate for the 25MB file while the existing methods SDVADC [5], ZEUS [8], and RCDSD [10] gave 20.9%, 20%, and 24.5%. Likewise, for other file sizes 10MB, 15 MB, and 20 MB, the proposed secure deduplication scheme has provided the best results compared to SDVADC [5], ZEUS [8], and RCDSD [10].

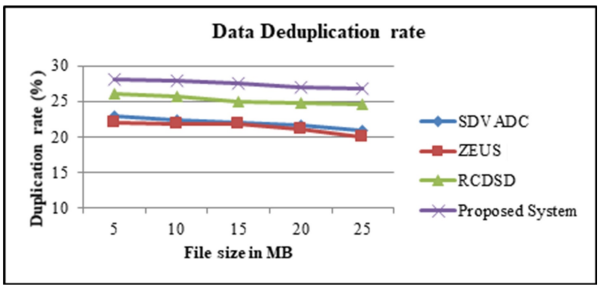

Fig. 4. Data deduplication rate.

4.2 Deduplication Throughput

Deduplication throughput is the rate at which blocks, hashtags, and keys are generated successfully. Deduplication throughput has been demonstrated in Fig. 5. In Fig. 5, X-axis contains the files in MB size and Y-axis contains the throughput rate in MB/S (megabit per second). From Fig. 5, it is clearly stated that the proposed system has taken high throughput which means it has generated the blocks, keys, and hashtags at faster rates for the deduplication checking process compared to existing methods such as SDVADC [5], ZEUS [8], and RCDSD [10].

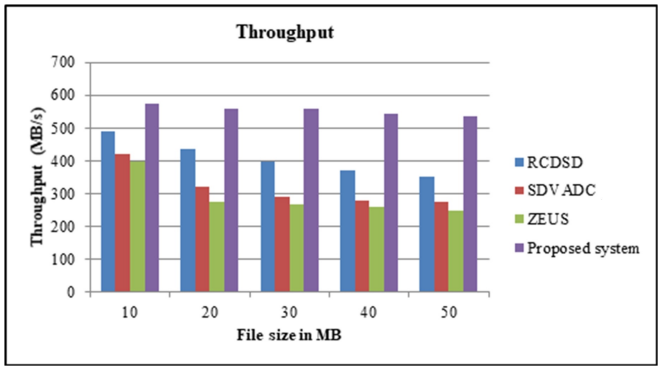

Fig. 5. De-duplication throughput.

4.3 Time Complexity

The comparison chart of time complexity has been demonstrated in Fig. 6. In this kind of estimation, the execution time of key role generation, block generation, the hash-tag generation is evaluated on a given dataset. The execution time is computed in milliseconds. The below comparison Fig. 6 has demonstrated that the proposed secure deduplication method has taken only a small amount of time for deduplication detection and elimination. As a result, this presented deduplication method can take low time complexity to detect and eliminate duplicated data compared to existing deduplication methods SDVADC [5], ZEUS [8], and RCDSD [10].

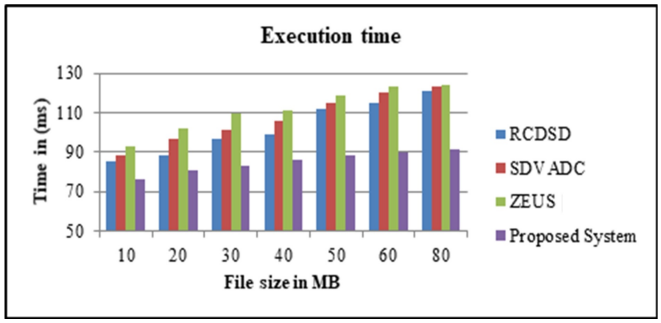

Fig. 6. Comparison of time complexity.

4.4 Communication Overhead

The communication overhead of the proposed and existing methods has been illustrated in Fig. 7. In Fig. 7, X-axis contains the number of data files used for the performance of the proposed and existing deduplication model in terms of communication overhead's percentage level. As observed from Fig. 7, the proposed deduplication system has taken less communication overhead for secure data deduplication process in the cloud storage than existing methods SDVADC [5], ZEUS [8], and RCDSD [10].

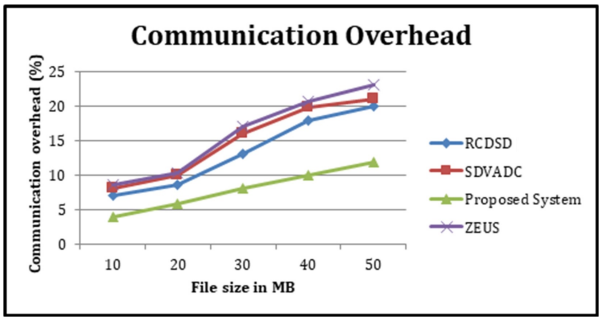

Fig. 7. Communication overhead comparison.

5. CONCLUSION

This research work discusses an improved block-chain based secure data deduplication method that has been developed to save cloud storage space efficiently. In this work, ARKG was implemented to create key roles before uploading data so that authorized users can access the data. The duplicated data has been verified by the blockchain process, in which the smart agreement was done with SHA-256 to verify the integrity and confidentiality of the users. For the duplicated data verification, the hash key and the hashtag were generated using SHA-256. The cloud server verifies the duplicated data through hashtags available in the hash table. Data will not be uploaded to the cloud storage server if it is received as a "duplicate". The cloud storage receives a "no duplicate" message from the cloud server before being uploaded. Then the data is effectively encrypted using the Content Sealed Encryption method. Next, the ciphertext has been uploaded by the data owner. For secure downloading, the generated roles are verified based on the data consumer's attributes. If roles are matched, then authorized consumers can securely download the data using the decryption key. Finally, the performance of the proposed deduplication method was estimated and compared with existing deduplication methods SDVADC [5], ZEUS [8], and RCDSD [10]. This proposed method has given a high deduplication rate and high deduplication throughput and has taken less time complexity and less communication overhead. The experiment result shows that the recommended system is extremely secure and effective for data deduplication for an integrated cloud environment. In the future, this proposed system can be extended to all types of Internet of Things (IoT) applications and can be used in building cyber-physical systems, exploring different use cases with different payloads and modified data formats.

REFERENCES

1. W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang, and Y. Zhou, "A comprehensive study of the past, present, and future of data deduplication," *Proceedings of the IEEE*, Vol. 104, 2016, pp. 1681-1710.
2. S. T. Ahmed and L. E. George, "Lightweight hash-based de-duplication system using the self detection of most repeated patterns as chunks divisors," *Journal of King Saud*

University-Computer and Information Sciences, Vol. 34, 2022, pp. 4669-4678.
3. B. Rasina Begum and P. Chitra, "ECC-CRT: an elliptical curve cryptographic encryption and Chinese remainder theorem based deduplication in cloud," *Wireless Personal Communications*, Vol. 116, 2021, pp. 1683-1702.
4. S. E. Ebinazer, N. Savarimuthu, and S. M. S. Bhanu, "ESKEA: enhanced symmetric key encryption algorithm based secure data storage in cloud networks with data deduplication," *Wireless Personal Communications*, Vol. 117, 2021, pp. 3309-3325.
5. C. M. Geeta, R. G. S. Raju, S. Raghavendra, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, "SDVADC: Secure deduplication and virtual auditing of data in cloud," *Procedia Computer Science*, Vol. 171, 2020, pp. 2225-2234.
6. P. G. Shynu, R. K. Nadesh, V. G. Menon, P. Venu, M. Abbasi, and M. Khosravi *et al.*, "A secure data deduplication system for integrated cloud-edge networks," *Journal of Cloud Computing*, Vol. 9, 2020, pp. 1-12.
7. E. Baccour, A. Erbad, A. Mohamed, M. Guizani, and M. Hamdi, "CE-D2D: Collaborative and popularity-aware proactive chunks caching in edge networks," in *Proceedings of IEEE International Conference on Wireless Communications and Mobile Computing*, 2020, pp. 1770-1776.
8. C.-M. Yu, S. P. Gochhayat, M. Conti, and C.-S. Lu, "Privacy aware data deduplication for side channel in cloud storage," *IEEE Transactions on Cloud Computing*, Vol. 8, 2018, pp. 597-609.
9. A. Sun, G. Gao, T. Ji, and X. Tu, "One quantifiable security evaluation model for cloud computing platform," in *Proceedings of IEEE 6th International Conference on Advanced Cloud and Big Data*, 2018, pp. 197-201.
10. S. Mishra, S. Singh, and S. T. Ali, "RCDSD: RSA based cross domain secure deduplication on cloud storage," in *Proceedings of IEEE 9th International Conference on Computing, Communication and Networking Technologies*, 2018, pp. 1-7.
11. W. Sun, N. Zhang, W. Lou, and Y. T. Hou, "Tapping the potential: Secure chunk-based deduplication of encrypted data for cloud backup," in *Proceedings of IEEE Conference on Communications and Network Security*, 2018, pp. 1-9.
12. S. Singhal, A. Kaushik, and P. Sharma, "A novel approach of data deduplication for distributed storage," *International Journal of Engineering and Technology*, Vol. 7, 2018, pp. 46-52.
13. Y. Zhang, H. Su, M. Yang, D. Zheng, F. Ren, and Q. Zhao, "Secure deduplication based on rabin fingerprinting over wireless sensing data in cloud computing," *Security and Communication Networks*, Vol. 2018, 2018, pp. 1-12.
14. D. Praveena and P. Rangarajan, "A machine learning application for reducing the security risks in hybrid cloud networks," *Multimedia Tools and Applications*, Vol. 79, 2020, pp. 5161-5173.
15. R. Vignesh and J. Preethi, "Secure data deduplication system with efficient and reliable multi-key management in cloud storage," *Journal of Internet Technology*, Vol. 23, 2022, pp. 811-825.
16. L. Peng, Z. Yan, X. Liang, and X. Yu, "Secdedup: Secure data deduplication with dynamic auditing in the cloud," *Information Sciences*, 2023, p. 119279.
17. T. Zhang, C. Wang, and M. I. U. Chandrasena, "Blockchain-assisted data sharing supports deduplication for cloud storage," *Connection Science*, Vol. 35, 2023, p. 2174081.

18. R. Misal and B. Perumal, "Data deduplication for efficient cloud storage and retrieval," *International Arab Journal of Information Technology*, Vol. 16, 2019, pp. 922-927.
19. X. Yang, R. Lu, K. K. R. Choo, F. Yin, and X. Tang, "Achieving efficient and privacy-preserving cross-domain big data deduplication in cloud," *IEEE Transactions on Big Data*, Vol. 8, 2017, pp. 73-84.
20. G. Xu, B. Tang, H. Lu, Q. Yu, and C. W. Sung, "LIPA: A learning-based indexing and prefetching approach for data deduplication," in *Proceedings of IEEE 35th Symposium on Mass Storage Systems and Technologies*, 2019, pp. 299-310.
21. Y. Fu, N. Xiao, H. Jiang, G. Hu, and W. Chen, "Application-aware big data deduplication in cloud environment," *IEEE Transactions on Cloud Computing*, Vol. 7, 2017, pp. 921-934.
22. Z. Yan, W. Ding, X. Yu, H. Zhu, and R. H. Deng, "Deduplication on encrypted big data in cloud," *IEEE Transactions on Big Data*, Vol. 2, 2016, pp. 138-150.
23. H. Kaur and P. Mann, "An improved hybrid re-encryption scheme for mobile cloud computing environment," *International Journal of Computer Applications*, Vol. 162, 2017, pp. 12-16.
24. N. P. Kawtikwar, M. Joshi *et al.*, "Data deduplication in cloud environment using file-level and block-level techniques," *Imperial Journal of Interdisciplinary Research*, Vol. 3, 2017, pp. 1294-1298.
25. J. Li, C. Qin, P. P. Lee, and X. Zhang, "Information leakage in encrypted deduplication via frequency analysis," in *Proceedings of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2017, pp. 1-12.
26. J. Periasamy and B. Latha, "An enhanced secure content de-duplication identification and prevention (ESDIP) algorithm in cloud environment," *Neural Computing and Applications*, Vol. 32, 2020, pp. 485-494.
27. S. Abirami and S. Ramanathan, "Linear scheduling strategy for resource allocation in cloud environment," *International Journal on Cloud Computing: Services and Architecture*, Vol. 2, 2012, pp. 9-17.
28. P. Prajapati and P. Shah, "A review on secure data deduplication: Cloud storage security issue," *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, 2020, pp. 3996-4007.
29. M. Yoon, "A constant-time chunking algorithm for packet-level deduplication," *ICT Express*, Vol. 5, 2019, pp. 131-135.
30. J. Paulo and J. Pereira, "A survey and classification of storage deduplication systems," *ACM Computing Surveys*, Vol. 47, 2014, pp. 1-30.
31. S. Ramanathan, S. Goel, and S. Alagumalai, "Comparison of cloud database: Amazon's simpledb and google's bigtable," in *Proceedings of International Conference on Recent Trends in Information Systems*, 2011, pp. 165-168.

S. Ruba is working as an Assistant Professor in the Department of Computer Science and Engineering at the Government College of Engineering, Salem, Tamil Nadu, India. She has teaching experience for ten years. She completed her B.E degree at J.J College of Engineering, Trichy, Tamil Nadu and then she received her M.E degree at Government College of Technology, Coimbatore. Currently, she pursuits her Ph.D. research at Anna University, Chennai. She has many international journal publications. Her research interests include cloud computing, network security, and machine learning.

A. M. Kalpana is currently the Professor and Head of the Department of Computer Science and Engineering in the Government College of Engineering, Salem, Tamil Nadu, India. She has teaching experience for twenthy two years. She received her doctorate from Anna University, Chennai. She is a member of IEEE, ISTE, and CSI. She has pub- lished more than 32 papers in International Journals and 45 papers in national and international conferences. Her research interests include software engineering, software testing, and data mining.