BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **19**, No 1
Sofia • 2019 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2019-0003

Efficient Dynamic Bloom Filter Hashing Fragmentation for Cloud Data Storage

S. Jegadeeswari[1], *P. Dinadayalan*[2], *D. Gnanambigai*[3]

[1]*Bharathiar University, Coimbatore 641046, Tamil Nadu, India*
[2]*Department of Computer Sci., Mahatma Gandhi Government Arts College, Mahe 673311, Kerala, India*
[3]*Department of Computer Sci., Indira Gandhi College of Arts and Science, 605009 Puducherry, India*
E-mails: jega_sathya@yahoo.co.in pdinadayalan@hotmail.com dgnanambigai@hotmail.com

***Abstract*:** *Security is important in cloud data storage while using the cloud services provided by the service provider in the cloud. Most of the research works have been designed for a secure cloud data storage. However, cloud users still have security issues with their outsourced data. In order to overcome such limitations, a Dynamic Bloom Filter Hashing based Cloud Data Storage* (*DBFH-CDS*) *Technique is proposed. The main goal of DBFH-CDS Technique is to improve confidentiality and security of data storage in a cloud environment. The proposed Technique is implemented using data fragmentation model and Bloom filter. The DBFH-CDS Technique uses data fragmentation model for fragmenting the large cloud datasets. After that, Bloom Filter is employed in DBFH-CDS Technique for storing the fragmented sensitive data along with higher security. The DBFH-CDS Technique ensures high data confidentiality and security for cloud data storage with the help of Bloom Filter. The performance of proposed DBFH-CDS Technique is measured in terms of Execution time and Data retrieval efficiency. The experimental results show that the DBFH-CDS Technique is able to improve the cloud data storage security with minimum space complexity as compared to state-of-the-art-works.*

***Keywords*:** *Cloud data storage, Cloud users, security, Confidentiality, fragmented table, unfragmented table, Bloom filter.*

1. Introduction

Cloud Computing is a potential paradigm employed for the deployment of applications on the Internet. Cloud and it is an on-demand computing service that offers a dynamic environment for the users to guarantee Quality of Service (QoS) on data for its secrecy in cloud data centers. Cloud applications utilize large data centers and efficient servers that host web applications and services. In addition, Cloud storage is a model of networked storage system in which data is stored in a pool of storage that are usually hosted through third parties. There are numerous benefits in utilizing cloud storage. The most significant is data accessibility. The data stored in

the cloud can be accessed at any time from any place. The other benefit of cloud storage is data sharing among users.

The cloud users outsource their data to the remote cloud storage for reducing the storing cost. The third-party auditor is a partially trusted and independent entity that assesses the data and arbitrates if necessary. The cloud users interrelate with cloud server for accessing and updating data stored in the cloud. The key issues in cloud data storage is security owing to possible unauthorized access within cloud service providers. Hence, there is a requirement for a new technique for a secure cloud data storage in the cloud computing environment to enhance the data confidentiality.

Recently, many research works have been designed for a secure cloud data storage. For example, Asymmetric Key Fragmentation Scheme (AKFS) was designed in [17] for improving the security of private data in a cloud environment. However, the security level of private data is not sufficient. A two-factor data security protection mechanism was designed in [8] for cloud storage system and to improve the confidentiality of the data in cloud. However, storage complexity was higher.

A Cryptography-Based Secure Data Storage was developed in [15] to provide a secure data exchange in the cloud environment. Ciphertext Policy Attribute-Based Encryption (CP-ABE) scheme was intended in [23] to address the security problem in a cloud environment and to provide a secure and efficient environment for the cloud storage. However, the efficiency of encryption and decryption was not at the required level and therefore CP-ABE scheme had to improve security levels.

A Multi-Replica Dynamic Public Auditing (MuR-DPA) scheme was implemented in [3] to provide a higher security against dishonest cloud service providers. However, supporting secure public auditing of dynamic data and streaming data had remained unaddressed. A systematic design methodology was developed in [24] to achieve the best performance tradeoff between security requirements and data storage costs. However, the security level for data recovery reliability constraint was not considered.

A novel secure data de-duplication scheme was employed in [9] that improve data privacy and confidentiality in cloud storage. However, the data de-duplication scheme increases issues relating to security in the cloud. Secure User Authenticated Cloud RAID model was introduced in [18] that guards user data from unauthorized access by the cloud service provider.

A novel method was designed in [1] for a secure and confidential storage of data in the cloud environment to reduce the computations overhead owing to encryption. A privacy-preserving public auditing system was explained in [4] for data storage security and to carry out auditing with lower communication and computation overhead in cloud computing. However, privacy protection of cloud users' data against external auditors was sufficient.

A public auditing scheme was designed in [7] for a secure cloud storage using a Dynamic Hash Table (DHT) and to reduce the cost of the verification process in cloud computing. However, the data accessing time was more. An efficient data integrity scheme was intended in [11] for protecting outsourced data from the external

adversaries and malicious auditors. However, security of **outsourced** data was not enough.

A Dual-Server Public Key Encryption with Keyword Search (DSPEKS) scheme was used in [16] to resolve the security vulnerability in cloud data storage. However, the DSPEKS scheme suffers from several constraints relating to the security. A probabilistic challenge-response scheme was developed in [20] to present an efficient way to improve the security and reliability of cloud storage. However, computation and communication overhead was poor.

An effective and flexible distribution verification mechanism was implemented in [10] to achieve higher data storage security and to authenticate the correctness of the user's data in cloud data storage. A Data Partitioning Technique was designed in [19] to present an efficient data storage security for cloud service and to offer flexible data access. But, the security level was not at the required level.

An efficient Secure-Channel Free Public key Encryption with Keyword Search (SCF-PEKS) scheme was developed in [13] for securing data form keyword and ciphertext attacks and keyword guessing attacks. However, data privacy rate was not sufficient. A novel scheme was intended in [22] with the help of dual system encryption technique for supporting privacy preserving predicate encryption with fine-grained searchable capability for Cloud storage. However, the encryption and decryption performance was not effective.

Public auditing protocol was intended in [6] to provide secure cloud storage service to cloud users and to verify the data integrity. But, security of the cloud data storage against the pollution attacks was not considered. An enhanced security mechanism based on erasure code was developed in [21] to attain higher data availability, security in cloud data storage.

Cryptographic Role-Based Access Control (RBAC) schemes were employed in [12] by combining the trust models to ensure data security in the cloud. However, the time consumption was higher. A novel data sharing method was designed in [5] based on Oblivious Random Access Memory (ORAM) to preserve the data privacy in cloud storage. The performance of Data retrieval efficiency was not efficient.

Multi-Agent System architecture (MAS) was designed in [2] for increasing security and confidentiality in cloud storage. However, communication overhead was improved. Remote Data Auditing (RDA) approaches were implemented in [14] to improve the integrity of cloud storage system. But, the data confidentiality was not sufficient.

Fragmented-Iterated Bloom Filters (FIBFs) were developed in [25] to efficiently route the events in a sensor network. FIBFs reduce the Execution time but the security was not improved. A mitigation system termed as Distributed Denial of Service attacks (DDoS)-Mitigation System (DDoS-MS) was designed in [26] to resolve the issues due to loss of cloud service availability. However, the security issues were not addressed efficiently.

Privacy and Secure Data Storage Mechanism was discovered in [27] for balancing group members or a community in cloud environment. However, the data privacy level was not improved to the desired level. Secured Document Sharing Using Visual Cryptography (SDSUVC) approach in [28] implemented an efficient

Storage Scheme for storing and recovering a document file with high data confidentiality and integrity. However, computation complexity was increased in SDSUVC approach.

A cloud-oriented two-layer data model was discussed in [29] to design the nested data in representation layer and formation of Component-Attribute-Object (CAO). But, two-layer cloud database management system depended on algebraic operations that were not implemented. A cryptographic access control solution depending on Attribute-Based Encryption (ABE) and Identity-Based Signature (IBS) was introduced in [30] for improving cloud confidentiality against unauthorized users. However, scalable user revocation methods were not designed for achieving better security level.

In order to overcome the drawbacks of the existing data security models, Dynamic Bloom Filter Hashing Based Cloud Data Storage (DBFH-CDS) Technique is proposed. This paper concentrates on the data security in the cloud, which addresses the customers view towards data confidentiality. The main objective of DBFH-CDS Technique is to defend outsourced data from attackers and from curious cloud providers by using cloud data fragmentation model and Bloom filter.

The rest of the paper is organized as follows. In Section 2, the proposed DBFH-CDS Technique is explained with the aid of a neat architecture diagram. In Section 3, the experimental setting is discussed with an exhaustive analysis of results described in Section 4. In Section 5, the concluding remark is presented.

2. Dynamic bloom filter hashing based cloud data storage

Cloud users store their data remotely and attain a higher quality of services on the various cloud applications. The mechanism used to store data in the cloud needs to be efficient and confidential for achieving higher security for outsourced data. In order to overcome such limitation, Dynamic Bloom Filter Hashing Based Cloud Data Storage (DBFH-CDS) Technique is developed. The DBFH-CDS Technique is designed with the aims of achieving confidentiality of cloud data using horizontal and vertical fragmentation. Fragmentation is a technique in which the data can be stored in different cloud data centers by means of fragmenting the whole database into several pieces termed fragments. In DBFH-CDS Technique, Data confidentiality is achieved by fragmenting the relational databases into independent fragments and then processing them into different locations. The architecture of Dynamic Bloom Filter Hashing for Secure Cloud Data Storage DBFH-CDS technique is demonstrated in Fig. 1.

The proposed DBFH-CDS Technique ensures High Confidentiality and Security in the Cloud Data Storage Environment. Generally, Cloud Data Storage model includes data owners, cloud service providers, and data users. Data users access the data from a service provider to verify the information of the data. The data owners ensure that only authorized users access the data for its confidentiality. Therefore, the DBFH-CDS Technique uses Bloom Filter structure for Efficient and Secure Cloud Data Storage. Bloom filter employs hash functions to hash the data and store it.

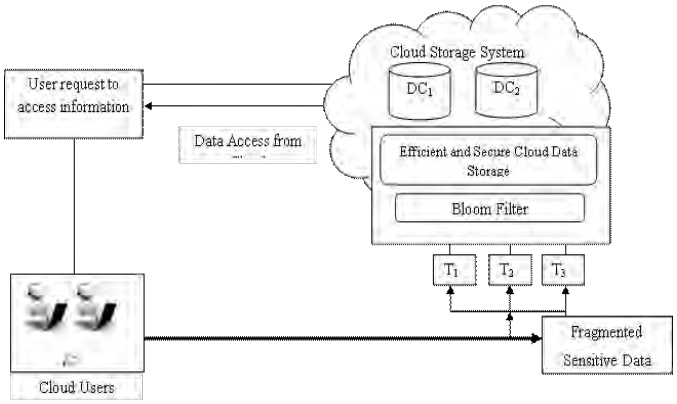

Fig. 1. Dynamic BFH for SCD Storage

In Data Fragmentation Model, the sensitive datasets are partitioned into a number of fragments. These fragmented data are stored in dynamic hashing schemes using Bloom filter. Then stored sensitive data are retrieved by using hash function through mapping. The dynamic hashing allows the cloud users to read, update, insert and modify the data. The block diagram of DBFH-CDS Technique for Secure Cloud Data Storage is shown in Fig. 2 below.

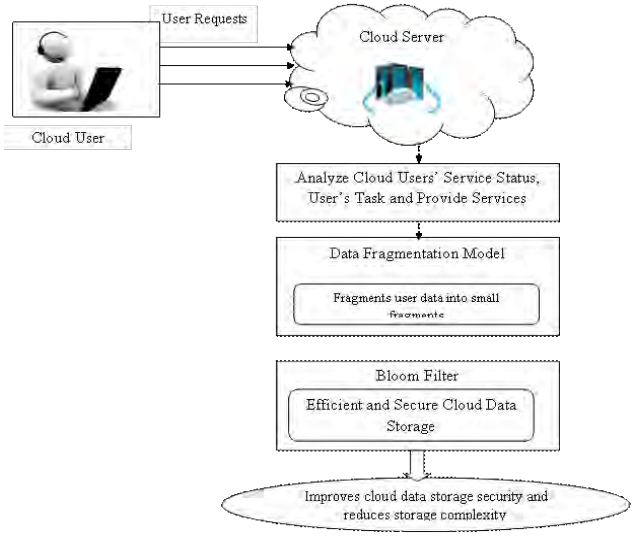

Fig. 2. Block-diagram of DBFH-CDS technique for SCD Storage

As shown in Fig. 2, initially the request is sent from the cloud users to cloud server. Then, the cloud server analyzes users' service status and different user's task for providing services to appropriate users in a cloud environment. After that, the user requested data from the cloud server is fragmented into a number of small fragments like T_1, T_2, T_3 ... by using data fragmentation model such as horizontal and vertical fragmentation. Finally, Fragmented data are efficiently stored in Bloom filter storage in order to reduce the storage complexity and to improve the security of cloud data storage with a higher level of confidentiality.

2.1. Data Fragmentation Model

The Data Fragmentation Model is a dynamic structure and its size is increased or decreased depending on a given data set. The DBFH-CDS Technique uses the Data Fragmentation Model for fragmenting the large cloud datasets. This Data Fragmentation Model is used in DBFH-CDS Technique for the purpose of ensuring data privacy and confidentiality. The DBFH-CDS Technique identifies high confidential, medium confidential, low confidential attributes in a given data set by performing fragmentation. The data, which are kept more secure, is said to have a high confidential attribute. The data that is not secret is said to have low confidential attribute. The data set which are considered to have high or low attributes are considering as medium confidential attributes.

The process of Data Fragmentation Model is explained by using the following steps.

Algorithm 1. Data Fragmentation Model Process

// Data Fragmentation Model process

Input: Bank Marketing Dataset *D*

Output: Fragmented data

Step 1. Begin

Step 2. The Sensitive data is obtained from the cloud database from the various data centers

Step 3. The key and non key attributes are fragmented by the horizontal and vertical fragmented mechanism

Step 4. The fragmented data sets are stored in the Bloom filter storage

Step 5. The hashing function is used to retrieve the sensitive data from the cloud.

Step 6. The retrieved data is given to corresponding users in the cloud

Step 7. End

2.2. Bloom filter for SDS

Bloom filter is an efficient data structure that is used for storing any set of elements and its hashed value. Hence, the proposed DBFH-CDS Technique has used Bloom filter for storing the fragmented cloud data with the objective of improving the security of data storage in a cloud environment. The Bloom filter is employed in DBFH-CDS Technique for securely storing the fragmented data from the cloud server. The Bloom filter comprises of two sub-blocks such as hashing block and mapping block. Initially, the bloom filter performs insertion operation when it stores fragmented cloud data. After that, a Bloom filter is used for retrieving the stored data. The key advantage of using Bloom filter is saving the computational time since it is independent of the number of cloud data stored. Therefore, the proposed DBFH-CDS Technique reduces the time taken for storing data in an effective manner.

A Bloom filter data structure for storing fragmented data is mathematically represented as follows:

(1) $$\mathrm{BF} = (d_1, d_2, d_3, \ldots, d_n),$$

where BF denotes the Bloom filter structure in which d represents a different number of fragmented data stored in an array of m bits. A Bloom filter structure for CDS is shown in Fig. 3.

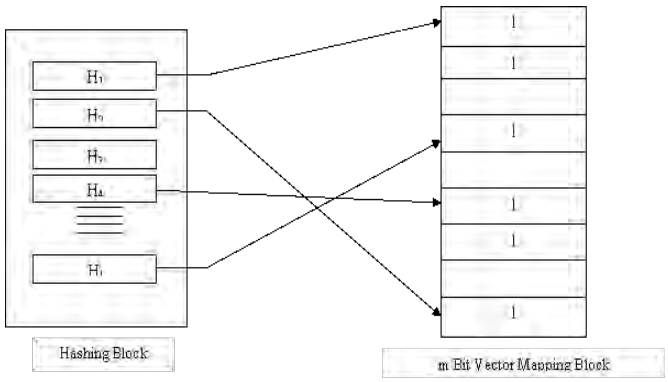

Fig. 3. Bloom filter structure for CDS

As shown in Fig. 3, Bloom filter employs n independent hash functions $(h_1, h_2, \ldots, h_l)$ with range $(1, 2, \ldots, m)$ for storing set of fragmented data. For each cloud data $d \in$ BF, the bit $(h_i(d))$ is set to 1 for $1 < i < l$. In Bloom filter, l different **hash functions** are used for mapping some set of fragmented data to one of the m array positions with a uniform random distribution for retrieving the data stored.

2.2.1. Data insertion operation in Bloom filter

Let us consider a set S of fragmented cloud data like $S = (d_1, d_2, d_3, \ldots, d_n)$. Bloom filters define membership information of set S through making use of a bit vector V. Initially, m bits of array are set to 0. The fragmented cloud data are stored using l hash functions$(h_1, h_2, \ldots, h_l)$ with $h_1 \rightarrow \{1, \ldots, m\}$. Fig. 4 shows insertion operation of fragmented data in Bloom filter.

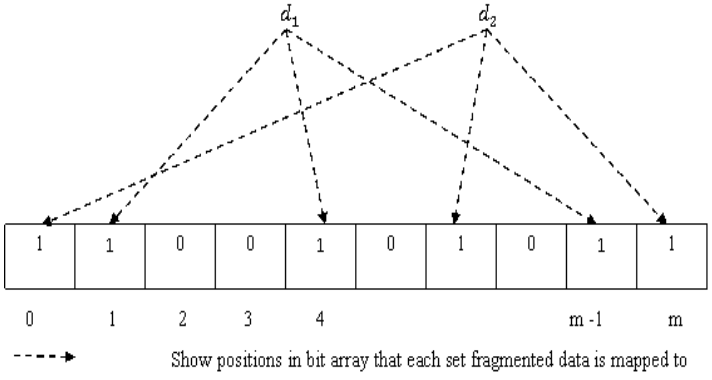

Fig. 4. Insertions of fragmented cloud data d_1 and d_2 into Bloom filter using hash function

Fig. 4 shows the insertion process of fragmented cloud data d_1 and d_2 in bloom filter. The algorithmic process of data insertion is shown in below.

Algorithm 2. Data Insertion Process in Bloom filter
// Data Insertion Process
Input: Set of Fragmented data
Output: Reduced space complexity with minimum time
Step 1. Begin
Step 2. For each Fragmented data d_i
Step 3. if **(**size of d_i< total storage capacity)
Step 4. Insert new data into Bloom filter structure using (Equation 1)
Step 5. Determine the bit array vector V for the respective new data
Step 6. Compute the hash functions values
Step 7. End for
Step 8. End

With the aid of the above algorithmic process, DBFH-CDS Technique efficiently stores the fragmented data from the Cloud Data Fragmentation Mode with minimum space and time. This, in turn, helps for reducing space complexity and Execution time for secured cloud data storage.

2.2.2. Data retrieval from Bloom filter

The blooming filter is a probabilistic data structure that is used for retrieving the data stored in a set with minimum false positive rates. Bloom filters utilize lesser space and constant time to respond to the queries for set membership. The blooming filter concept efficiently reduced the false positive probability. The Bloom filter contains the locations of the bit corresponding to existing entries. The outsourced message field $F = \{m_1, m_2, \ldots, m_n\}$ included n set of entries arranged into a membership function (M) of bit vector V length n. The hash functions $H = \{h_1, h_2, \ldots, h_l\}$ with $h_1 \rightarrow \{1, \ldots, m\}$ were determined initially. The Data retrieval process from bloom filter is shown in Fig. 5.

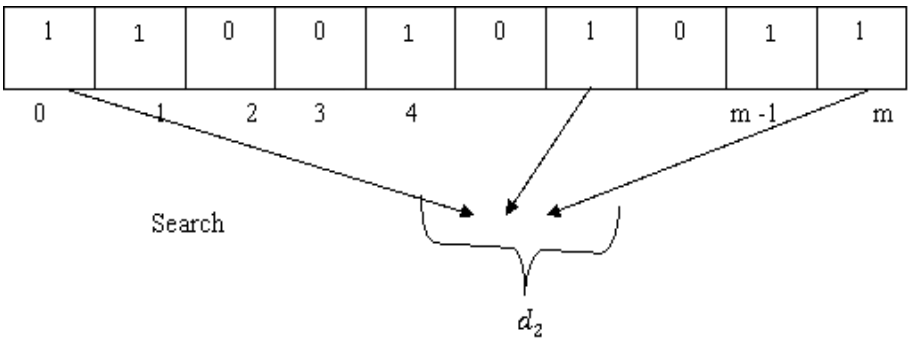

Fig. 5. Searching data d_2 in Bloom filter

The algorithmic process of Data retrieval from Bloom filter is shown in below.
Algorithm 3. Data retrieval process
// Data Retrieval Process
Input: Cloud User query request
Output: Improved Data retrieval efficiency with higher security

Step 1. Begin
Step 2. For Cloud User query request
Step 3. Searching is performed through Hash mapping and data file is prepared for the response
Step 4. The prepared data file is outputted to the appropriate users in Cloud environment
Step 5. End for
Step 6. End

With the help of the above algorithmic process, DBFH-CDS Technique securely retrieves the cloud user requested query data, which is stored in the Bloom filter by means of performing the hash mapping. Therefore, DBFH-CDS Technique improves the Data retrieval efficiency and cloud data security. This, in turn, helps for achieving high data confidentiality and security for sensitive data in a cloud environment.

3. Experimental settings

The Dynamic Bloom Filter Hashing Based Cloud Data Storage (DBFH-CDS) Technique is implemented in Java Language using bank marketing dataset from UCI machine learning repository. The CloudSim simulator toolkit has been employed as a simulation platform with 8 GB of RAM and 1 TB of storage space. The bank marketing dataset includes of 45211 instances and 17 attributes. The bank marketing dataset is interrelated with direct marketing campaigns of a Portuguese banking institution. The performance of DBFH-CDS Technique is compared against with exiting methods such as Asymmetric Key Fragmentation Scheme (AKFS) [17], two-factor data security protection mechanism [8] and Fragmented-Iterated Bloom Filters (FIBFs) [25]. The experimental evaluation using DBFH-CDS Technique is conducted on different factors such as Execution time, and Data retrieval efficiency.

4. Results and discussions

The effectiveness of DBFH-CDS Technique is compared against with existing methods namely AKFS [17], two-factor data security protection mechanism [8] and FIBFs [25], respectively. The performance of DBFH-CDS Technique is evaluated along with the following metrics.

4.1. Measurement of Execution time

In DBFH-CDS Technique, Execution time measures the amount of time taken for storing the cloud data. The Execution time is evaluated in terms of milliseconds (ms) and formulated as

$$\text{Execution time} = \text{time (cloud data storage)}, \tag{2}$$

where time taken for storing the cloud data is obtained. While the Execution time is lower, the method is said to be more efficient.

Table 1. Tabulation for Scalar query vs Execution time

Number of Scalar queries	Execution time (ms)							
	Unfragmented table				Fragmented table			
	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique	FIBFs	AKFS	Two-factor data security protection	DBFH-CDS technique
50	73.2	69.3	60.4	55.5	52.4	48.2	44.1	40.2
100	76.1	74.2	65.1	58.2	56.1	54.6	49.5	45.3
150	80.3	78.6	68.1	62.3	59.3	56.3	51.2	47.1
200	82.4	78.9	70.3	64.5	63.4	58.6	52.8	46.5
250	84.5	80.5	75.3	67.8	65.2	62.3	59.2	45.6
300	85.6	83.6	78.2	74.2	68.5	64.1	61.4	49.2
350	88.2	85.3	79.2	73.4	70.1	66.5	62.8	52.8
400	92.6	89.5	80.3	75.6	72.3	68.4	66.3	54.1
450	94.1	92.3	84.6	77.3	74.6	72.3	64.3	56.3
500	95.3	93.6	85.7	79.5	77.3	74.2	68.2	58.4

Table 1 demonstrates the comparative results analysis of data storage time taken for unfragmented table and fragmented table using four methods based on the different number of Scalar queries. The performance of Execution time using the proposed DBFH-CDS Technique is compared with existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25]. From the table values, it is illustrative that the Scalar query Execution time of fragmented table using proposed DBFH-CDS Technique is lower as compared to other existing methods and unfragmented table.

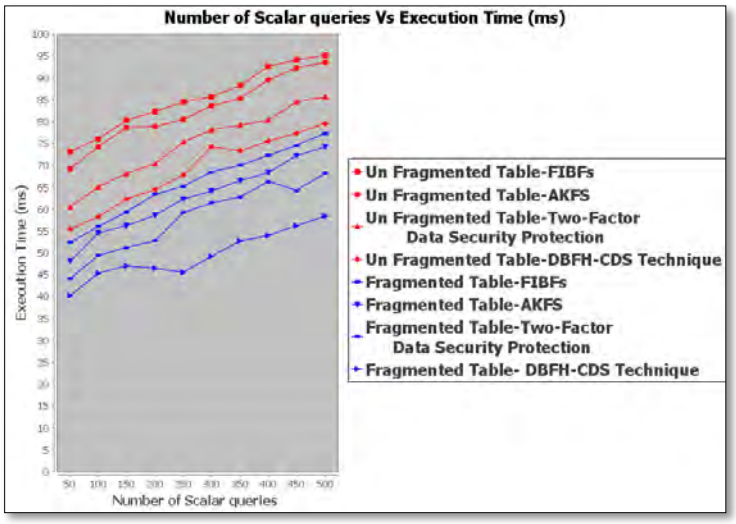

Fig. 6. Measure of Scalar query vs Execution time

Fig. 6 shows the impact of Execution time for unfragmented table and fragmented table using four methods with respect to a Scalar query type. From the figure, red line indicates the results of data storage time taken for unfragmented table whereas the blue line represents data storage time taken for the fragmented table using four methods. As shown in the figure, the Execution time for the Scalar query using proposed DBFH-CDS Technique is lower in the fragmented table as compared to other existing methods namely AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25].

From the experiments done for unfragmented table, the proposed DBFH-CDS Technique takes 75.6 ms of data storage time while considering number of scalar queries as 400 whereas existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], and Fragmented–Iterated Bloom Filters (FIBFs) [25] take 89.5 ms, 80.3 ms, 92.6 ms, respectively.

Table 2. Tabulation for Range query vs Execution time

Number of Range queries	Execution time (ms)							
	Unfragmented table				Fragmented table			
	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique
50	83.4	81.6	74.1	63.1	62.3	59.4	50.3	46.2
100	86.2	83.5	75.3	66.5	63.1	60.5	55.2	47.3
150	91.2	89.2	81.6	67.4	65.2	63.3	54.4	48.4
200	90.3	88.6	78.2	67.3	65.9	64.8	58.2	50.3
250	91.5	89.1	80.6	72.6	69.2	68.7	64.3	49.5
300	92.3	88.8	84.2	74.7	72.5	71.4	65.3	52.2
350	93.1	91.2	83.4	76.4	74.6	73.6	68.6	56.3
400	92.2	90.6	84.4	77.3	75.6	74.7	61.2	53.6
450	94.2	91.4	83.6	80.5	78.2	75.3	67.4	57.7
500	94.6	92.9	86.2	83.7	83.1	82.4	70.5	61.2

Similarly, the proposed DBFH-CDS Technique takes 54.1 ms, existing AKFS [17] takes 68.4, Two-Factor Data Security Protection Mechanism [8] takes 66.3 ms, FIBFs [25] takes 72.3 ms while considering a number of scalar queries as 400 in the fragmented table. Thus in turn, the proposed DBFH-CDS Technique reduces the Scalar query Execution time when compared to existing methods. Therefore, the proposed DBFH-CDS Technique reduces the Execution time using scalar queries in unfragmented table by 17% ,8%, and 19% when compared to AKFS [17], Two-Factor Data Security Protection Mechanism [8] and FIBFs [25]. In addition, Execution time is reduced by 21%, 14%, and 25% through considering scalar queries using a fragmented table when compared to existing AKFS [17], Two-Factor Data Security Protection Mechanism [8] and FIBFs [25] methods.

Table 2 describes the performance analysis of Execution time with respect to a different number of range queries using unfragmented table and fragmented table. The performance of range query Execution time using proposed technique DBFH-CDS is compared with existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25]. From Table 2, the Execution time of range query using fragmented table is minimum when compared to unfragmented table. Based on the table value the graph is plotted in Fig. 7.

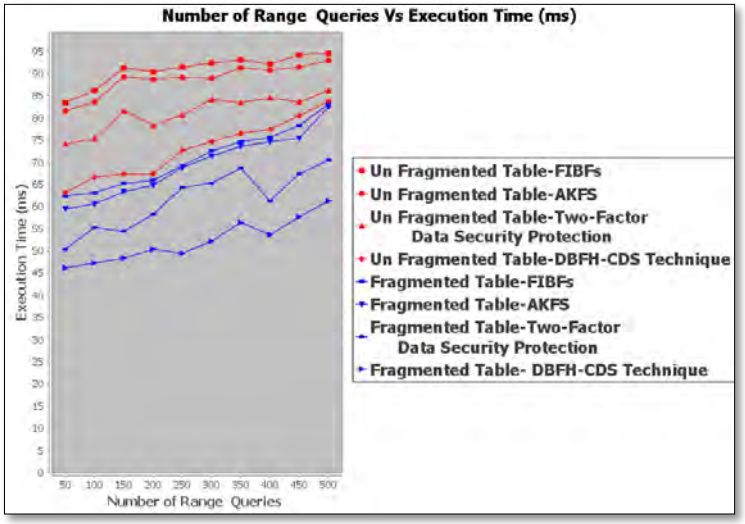

Fig. 7. Measure of Range query vs Execution time

Fig. 7 depicts the analysis of range query Execution time for unfragmented table and fragmented table using four different methods with respect to different range queries. The number of range query is varied from 50-500 for conducting experiments. As shown in the above figure, the proposed DBFH-CDS Technique consumes minimum time for storing cloud data (i.e., range queries) than the existing methods.

From the experiment's results, the range query using fragmented table takes 53.6 ms data storage time in proposed DBFH-CDS Technique while considering a number of range queries as 400, whereas existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25] take 74.7 ms, 61.2 ms, 75.6 ms, respectively. In the same manner, the proposed DBFH-CDS Technique takes 77.3 ms data storage time for considering 400 range queries, whereas existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25] take 90.6 ms and 84.4 ms, 92.2 ms respectively in unfragmented table. Therefore, the proposed DBFH-CDS Technique reduces the Execution time for data storage in unfragmented table by 18%, 10%, and 20% when compared to existing AKFS [17], Two-Factor Data Security Protection Mechanism [8] and FIBFs [25] methods. The proposed DBFH-CDS Technique minimizes also the Execution time in the fragmented table by 21%, 14%, 26% when compared to state-of-the-art methods.

Table 3. Tabulation for Nested query vs Execution time

Number of Nested queries	Execution time (ms)							
	Unfragmented table				Fragmented table			
	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique
50	90.1	88.8	75.6	72.3	69.2	66.7	64.3	47.5
100	94.2	93.3	88.3	74.4	72.1	68.2	58.6	49.2
150	95.6	94.4	84.1	73.5	72.6	70.4	68.3	46.6
200	98.2	97.2	90.4	75.4	73.4	71.5	62.2	53.5
250	95.6	90.3	84.2	78.9	77.4	76.3	69.3	57.2
300	94.2	92.5	86.7	82.6	79.5	74.6	69.4	62.1
350	98.5	97.2	94.5	79.6	77.2	75.7	72.1	58.7
400	98.2	96.4	90.3	81.7	79.1	77.2	73.3	59.3
450	97.7	96.9	94.3	84.2	83.4	82.7	78.5	61.2
500	98.5	97.4	95.3	85.4	83.5	80.9	79.6	63.1

Table 3 describes the result analysis of Execution time by considering a different number of nested queries in a cloud environment. It is clear from Table 2 that the proposed DBFH-CDS Technique reduces the data storage time using a fragmented table. Besides, experiments accomplished for the nested query using a fragmented table, the proposed DBFH-CDS Technique takes 59.3 ms data storage time whereas existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25] take 77.2 ms, 73.3 ms, 79.1 ms, respectively. For a nested query using a unfragmented table, the proposed DBFH-CDS Technique takes 81.7 ms data storage time whereas existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25] take 96.4 ms, 90.3 ms, 98.2 ms, respectively. Among these results obtained for Execution time, proposed DBFH-CDS Technique using fragmented table provides better performance for secure cloud data storage as compared to unfragmented table.

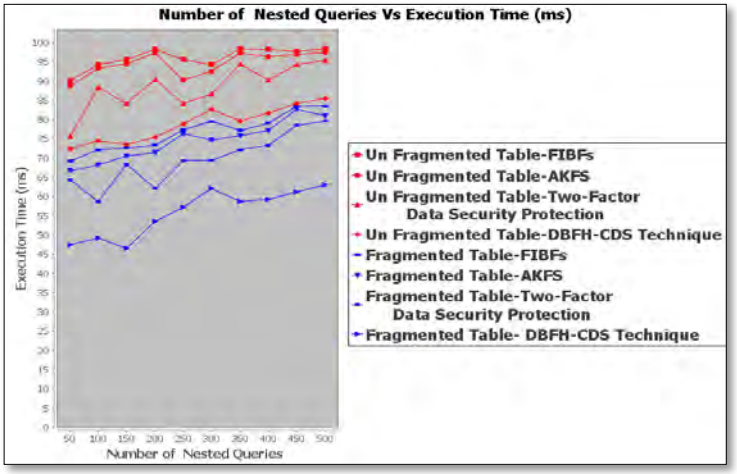

Fig. 8. Measure of Nested query vs Execution time

Fig. 8 shows the graphical representation of Execution time with 500 nested queries. From the figure, the time for data storage is better minimized in the proposed technique than the other methods. This efficient reduction on Execution time is achieved using Bloom filter in proposed DBFH-CDS Technique. Bloom filters have minimum storage requirement and rapid membership analyzing properties. The main aim of a Bloom filter is to reduce the space complexity and Execution time while storing the fragmented data on the cloud. Besides, the insertion operation is carried out in a Bloom filter to store the fragmented cloud data with hash value. In addition, the stored data is efficiently retrieved using a Bloom filter in the cloud environment. As a result, the proposed DBFH-CDS Technique minimizes the Execution time taken for storing different types of queries in a significant manner. Therefore, proposed DBFH-CDS Technique reduces the Execution time while considering nested queries by 17%, 11%, and 18% using unfragmented table when compared to existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], and FIBFs [25] method, respectively. Moreover, the proposed DBFH-CDS Technique decreases the time for data storage by 21%, 20%, and 27% when compared to state-of-the-art methods.

4.2. Measurement of Data retrieval efficiency

In DBFH-CDS Technique, Data retrieval efficiency measures the ratio of a number of correctly retrieved queries from the cloud storage to the total number of user queries. The Data retrieval efficiency is measured in terms of percentages (%) and mathematically expressed as

$$\text{Data retrieval efficiency} = \frac{\text{Number of correctly retrieved user queriesfrom the cloud storage}}{\text{Total number of user queries}} \times 100, \tag{3}$$

where Data retrieval efficiency is obtained. When the Data retrieval efficiency is higher, the method is said to be more efficient.

Table 4. Tabulation for Scalar query vs Data retrieval efficiency

No of Scalar queries	Data retrieval efficiency (%)							
	Unfragmented table				Fragmented table			
	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique
50	45.2	48.7	59.6	67.5	54.1	57.2	69.4	78.2
100	48.4	52.2	59.5	69.9	55.6	58.6	72.3	80.5
150	50.1	53.3	64.1	72.3	58.4	62.4	74.6	81.2
200	52.6	56.4	64.6	69.2	59.7	60.2	71.2	85.6
250	53.1	55.1	62.4	75.3	57.1	59.3	78.5	84.2
300	55.4	58.6	66.3	77.6	60.2	64.5	79.2	83.4
350	48.2	52.3	67.1	80.3	62.4	65.2	83.3	88.9
400	49.2	54.1	64.2	76.2	54.3	57.3	77.3	89.1
450	51.3	55.6	67.3	82.6	58.9	66.2	84.4	90.4
500	53.5	59.7	69.3	84.5	62.1	67.8	86.6	91.3

The impact of Data retrieval efficiency for unfragmented table and fragmented table using four methods based on the number of scalar queries is illustrated in Table 4. The performance of Data retrieval efficiency using proposed DBFH-CDS Technique is compared with three existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25]. From the table values, it is clear that the Data retrieval efficiency of the fragmented table using proposed DBFH-CDS Technique is higher as compared to other existing methods and unfragmented table. This is due to the application of a Bloom filter in proposed DBFH-CDS Technique. In order to check the presence of fragmented data inside a Bloom filter, hash functions is computed to check the related positions inside the array. By using DBFH-CDS Technique, the user requested query data is retrieved securely with the aid of performing hash mapping in the Bloom filter. In addition, the retrieved data is considered as the output to cloud user with maximum security. Therefore, the Data retrieval efficiency is highly improved in proposed DBFH-CDS Technique.

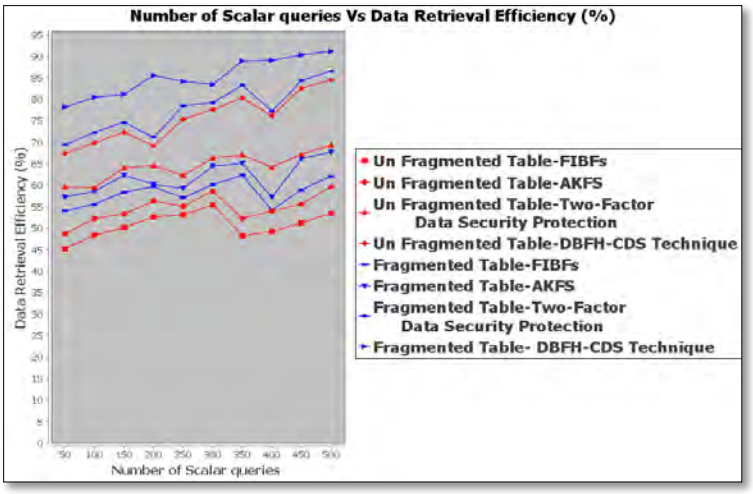

Fig. 9. Measure of Scalar query vs Data retrieval efficiency

Fig. 9 exhibits the impact of Data retrieval efficiency for unfragmented table and fragmented table using four methods with respect to number of scalar queries. From the figure, the red line shows the results of Data retrieval efficiency for unfragmented table whereas the blue line designates Data retrieval efficiency for the fragmented table using four methods. As demonstrated in the figure, Data retrieval efficiency of the fragmented table using proposed DBFH-CDS Technique is higher as compared to other existing methods namely AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25].

From the results of experiments obtained for 500 number of scalar queries using unfragmented table, the proposed DBFH-CDS Technique attains 84.5% Data retrieval efficiency whereas existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25] achieve 59.7%, 69.3%, 53.5%, respectively. Similarly, the fragmented table obtains the 91.3% of Data retrieval efficiency while considering 500 number of scalar queries in proposed DBFH-CDS Technique

whereas existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25] obtain 67.8% and 86.6%, 62.1%, respectively. Therefore, proposed DBFH-CDS Technique improves the Data retrieval efficiency up to 38%, 17% and 49% using unfragmented table when compared to existing methods. Moreover, DBFH-CDS Technique increases the Data retrieval efficiency using fragmented table by 38%, 10%, and 46% when compared to existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], and FIBFs [25] method, respectively.

Table 5. Tabulation for Range query vs Data retrieval efficiency

Number of Range queries	Data retrieval efficiency (%)							
	Unfragmented table				Fragmented table			
	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique
50	48.2	52.2	60.5	72.3	55.2	59.3	74.2	80.5
100	50.1	54.3	63.2	66.2	57.3	61.8	68.3	83.2
150	53.3	59.5	65.8	75.6	58.2	62.5	78.4	85.4
200	54.4	55.7	66.7	75.9	59.1	63.1	77.3	82.3
250	54.2	56.2	67.5	77.3	60.4	64.3	79.5	88.1
300	57.3	58.6	69	78.6	62.3	68.2	80.4	85.5
350	58.1	60.5	69.2	81.5	63.1	65.2	84.6	90.8
400	54.3	57.5	70.6	78.3	62.5	67.5	79.4	91.5
450	59.4	62.4	71.2	84.3	65.4	70.2	86.6	92.6
500	60.2	63.6	75.4	86.1	69.3	74.3	88.2	93.1

Table 5 illustrates the result analysis of Data retrieval efficiency with respect to a different number of range queries using unfragmented table and fragmented table. From the table, the Data retrieval efficiency using proposed DBFH-CDS technique is effectively improved than the other methods.

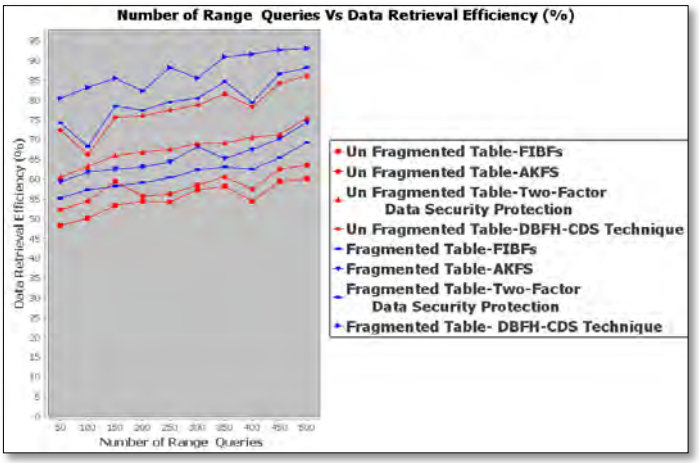

Fig. 10. Measure of Range query vs Data retrieval efficiency

Fig. 10 shows the performance of Data retrieval efficiency with respect to different number of range queries (i.e., 50-500) in cloud environment. From Fig. 10, the Data retrieval efficiency is highly increased in proposed DBFH-CDS Technique using fragmented table. While considering number of range queries is 500, the Data retrieval efficiency in proposed DBFH-CDS Technique attains 93.1% using fragmented table whereas, the existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25] attain 74.3%, 88.2%, 69.3%, respectively. In addition, proposed DBFH-CDS Technique, existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25] attains 86.1%, 63.6%, 75.4%, 60.2% of Data retrieval efficiency using unfragmented table, respectively. As a result, proposed DBFH-CDS Technique provides 34%, 14% and 41% of Data retrieval efficiency using unfragmented table when compared to existing methods. Proposed DBFH-CDS Technique also increases the Data retrieval efficiency by 33%, 10%, 43% when compared to existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25], respectively.

Table 6. Tabulation for Nested query vs Data retrieval efficiency

Number of Nested queries	Data retrieval efficiency (%)							
	Unfragmented table				Fragmented table			
	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique	FIBFs	AKFS	Two-factor data security protection mechanism	DBFH-CDS technique
50	48.2	53.3	63.1	69.3	56.4	62.3	75.5	82.3
100	47.2	51.6	65.2	70.2	59.2	64.9	72.6	85.6
150	50.3	55.8	64.3	69.5	57.3	60.4	80.1	87.8
200	52.4	56.7	66.4	74.8	59.2	62.5	77.3	88.3
250	55.4	58.3	70.5	75.1	62.6	68.7	79.3	89.6
300	56.8	59.1	71.1	75.6	64.5	67.5	77.8	87.3
350	60.5	64.2	72.5	78.4	65.2	69.3	86.8	92.5
400	57.3	61.4	73.6	83.6	67.4	71.5	85.6	91.8
450	55.2	60.5	75.2	82.6	70.6	73.4	88.3	94.7
500	59.6	64.2	76.3	86.3	71.5	75.6	89.4	95.2

Table 6 illustrates the impact of Data retrieval efficeincy using four different methods with respect to number of nested queries in cloud environement. The number of nested query is varied from 50 up to 500 for conducting experiments. From the table values, the Data retrieval efficiency is highly improved in proposed DBFH-CDS Technique compared to the other methods.

Fig. 11 depicts the performance of Data retrieval efficiency using proposed and exisitng methods. From the figure, unfragmented table using the proposed DBFH-CDS Technique obtains 86.3% Data retrieval efficiency while considering 500 number of nested queries whereas existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25] achieve 64.2%, 76.3%, 59.6%, respectively. Likewise, the proposed DBFH-CDS Technique acquires 95.2% Data retrieval

efficiency for nested query whereas existing AKFS [17], Two-Factor Data Security Protection Mechanism [8] and FIBFs [25] attains 75.6 % and 89.4%, 71.5%, respectively.

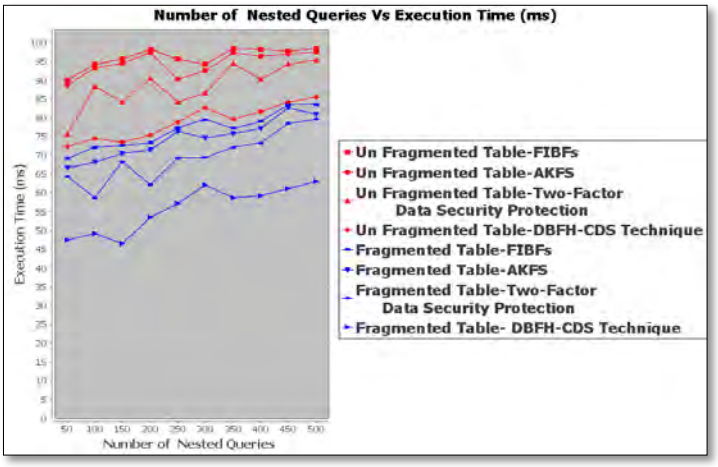

Fig. 11. Measure of Nested query vs Data retrieval efficiency

Furthermore, results of experiments obtained for the nested query using an unfragmented table, the proposed DBFH-CDS Technique improves the Data retrieval efficiency up to 31%, 10%, 41% when compared to existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25]. Similarly, nested query using fragmented table increases the Data retrieval efficiency up to 33%, 10%, 42% when compared to existing AKFS [17], Two-Factor Data Security Protection Mechanism [8], FIBFs [25], respectively. Among these results obtained for Data retrieval efficiency, proposed DBFH-CDS Technique using fragmented table provides better performance when compared to the unfragmented table.

5. Conclusions

An effective Dynamic Bloom Filter Hashing Based Cloud Data Storage (DBFH-CDS) Technique is designed for enhancing the confidentiality and security of data storage in a cloud environment. The DBFH-CDS Technique protects outsourced data from attackers, unauthorized users with the support of data fragmentation model and Bloom filter. At first, DBFH-CDS Technique divides the larger bank marketing dataset into numerous data fragments by using data fragmentation model through performing horizontal and vertical fragmentation. Next, DBFH-CDS Technique uses Bloom filer for storing fragmented sensitive data with less space complexity and minimum time. After that, the stored cloud data are retrieved from Bloom filter storage using the hash function by means of performing a hash mapping with higher Data retrieval efficiency. Finally, the retrieved data is output to the corresponding user in the cloud with higher security. The efficiency of DBFH-CDS Technique is tested with the metrics such as Execution time, and Data retrieval efficiency. With the experiments conducted for DBFH-CDS Technique, it is expressive that the cloud

data storage capacity provides more accurate results when compared to state-of-the-art works. The security of DBFH-CDS Technique can be further improved with encrypting the fragmented using different cryptography techniques.

References

1. Hudic, A., S. Islam, P. Kieseberg, S. Rennert, E. R. Weippl. Data Confidentiality Using Fragmentation in Cloud Computing. – International Journal of Pervasive Computing and Communications, Vol. **9**, March 2013, Issue 1, pp. 37-51
2. Talib, A. M. Ensuring Security, Confidentiality and Fine-Grained Data Access Control of Cloud Data Storage Implementation Environment. – Journal of Information Security, 2015, Issue 6, pp. 118-130.
3. Liu, C., R. Ranjan, C. Yang, X. Zhang, L. Wang, J. Chen. MuR-DPA: Top-Down Levelled Multi-Replica Merkle Hash Tree Based Secure Public Auditing for Dynamic Big Data Storage on Cloud. – IEEE Transactions on Computers, Vol. **64**, 2015, Issue 9, pp. 2609-2622.
4. Wang, C., S. S. M. Chow, Q. Wang, K. Ren, W. Lou. Privacy-Preserving Public Auditing for Secure Cloud Storage. – IEEE Transactions on Computers, Vol. **62**, 2013, Issue 2, pp. 362-375.
5. Yuan, D., X. Song, Q. Xu, M. Zhao, X. Wei, H. Wang, H. Jiang. An ORAM-Based Privacy Preserving Data Sharing Scheme for Cloud Storage. – Journal of Information Security and Applications, Elsevier, Vol. **39**, 2018, pp. 1-9.
6. Liu, H., P. Zhang, J. Liu. Public Data Integrity Verification for Secure Cloud Storage. – Journal of Networks, Vol. **8**, February 2013, Issue 2, pp. 373-380.
7. Tian, H., Y. Chen, C.-C. Chang, H. Jiang, Y. F. Huang, Y. Chen, J. Liu. Dynamic-Hash-Table Based Public Auditing for Secure Cloud Storage. – IEEE Transactions on Services Computing, Vol. **PP**, 2016, Issue 99, pp. 1-14.
8. Liu, J. K., K. Liang, W. Susilo, J. Liu, Y. Xiang. Two-Factor Data Security Protection Mechanism for Cloud Storage System. – IEEE Transactions on Computers, Vol. **65**, 2016, Issue 6, pp. 1992-2004.
9. Hur, J., D. Koo, Y. Shin, K. Kang. Secure Data Deduplication with Dynamic Ownership Management in Cloud Storage. – IEEE Transactions on Knowledge and Data Engineering, Vol. **28**, 2016, Issue 11, pp. 3113-3125.
10. Batra, K., C. Sunitha, S. Kumar. An Effective Data Storage Security Scheme for Cloud Computing. – International Journal of Innovative Research in Computer and Communication Engineering, Vol. **1**, June 2013, Issue 4, pp. 808-815.
11. Han, K., Q. Li, Z. Deng. Security and Efficiency Data Sharing Scheme for Cloud Storage. – Chaos, Solitons and Fractals, Vol. **86**, May 2016, pp. 107-116.
12. Zhou, L., V. Varadharajan, M. Hitchens. Trust Enhanced Cryptographic Role-Based Access Control for Secure Cloud Data Storage. – IEEE Transactions on Information Forensics and Security, Vol. **10**, 2015, Issue 11, pp. 2381-2395.
13. Guo, L., W.-C. Yau. Efficient Secure-Channel Free Public Key Encryption with Keyword Search for EMRs In Cloud Storage. – Journal of Medical Systems, Springer, Vol. **39**, February 2015, Issue 11, pp. 1-11.
14. Sookhak, M., A. Gania, M. K. Khan, R. Buyya. Dynamic Remote Data Auditing for Securing Big Data Storage In Cloud Computing. – Information Sciences, Elsevier, Vol. **380**, 2017, pp. 101-116.
15. Usman, M., M. A. Jan, X. He. Cryptography-Based Secure Data Storage and Sharing Using HEVC and Public Clouds. – Information Sciences, Elsevier, 2016.
16. Chen, R., Y. Mu, G. Yang, F. Guo, X. Wang. Dual-Server Public-Key Encryption with Keyword Search for Secure Cloud Storage. – IEEE Transactions on Information Forensics and Security, Vol. **11**, 2016, Issue 4, pp. 789-798.

17. Y o o, S. M., J. K i m, J. H. P a r k, J. N a m, J. C. R y o u. Ownership-Guaranteed Security Framework for the Private Data in the Entrusted Management Environment. – Cluster Computing, Springer, Vol. **18**, September 2015, Issue 3, pp. 1251-1261.
18. S a h a n a, S., R. B o s e, D. S a r d d a r. Harnessing RAID Mechanism for Enhancement of Data Storage and Security on Cloud. – Brazilian Journal of Science and Technology, Springer, December 2016.
19. K h e d k a r, S. V., A. D. G a w a n d e. Data Partitioning Technique to Improve Cloud Data Storage Security. – International Journal of Computer Science and Information Technologies, Vol. **5**, 2014, Issue 3, pp. 3347-3350.
20. J i a n g, T., X. C h e n a, J. L i, D. S. W o n g c, J. M a a, J. K. L i u. Towards Secure and Reliable Cloud Storage against Data Re-Outsourcing. – Future Generation Computer Systems, Elsevier, Vol. **52**, November 2015, pp. 86-94.
21. W a n g, W., P. L i, L. H a n, S. H u a n g, K. X u, C. Y u, J. L e i l. An Enhanced Erasure Code-Based Security Mechanism for Cloud Storage. – Mathematical Problems in Engineering, Hindawi Publishing Corporation, Vol. **2014**, 2014, Article ID 293214, pp. 1- 8.
22. W a n g, X. A., F. X h a f a, W. C a i, J. M a, F. W e i. Efficient Privacy Preserving Predicate Encryption with Fine-Grained Searchable Capability for Cloud Storage. – Computers & Electrical Engineering, Elsevier, Vol. **56**, November 2016, pp. 871-883.
23. Z h a n g, Y., C. X u, H. L i, X. L i a n g. Cryptographic Public Verification of Data Integrity for Cloud Storage Systems. – IEEE Cloud Computing, Vol. **3**, 2016, Issue 5, pp. 44-52.
24. C h e n, Y.-J., L.-C. W a n g, C.-H. L i a o. Eavesdropping Prevention for Network Coding Encrypted Cloud Storage Systems. – IEEE Transactions on Parallel and Distributed Systems, Vol. **27**, 2016, Issue 8, pp. 2261-2273.
25. M u n o z, C., P. L e o n e. Fragmented-Iterated Bloom Filters for Routing in Distributed Event-Based Sensor Networks. – In: International Conference on Internet and Distributed Computing Systems, Springer, Cham, 2015, pp. 248-261.
26. A l o s a i m i, W., M. Z a k, K. A l-B e g a i n, R. A l r o o b a e a, M. M a s u d. Mitigation of Distributed Denial of Service Attacks in the Cloud. – Cybernetics and Information Technologies, Vol. **17**, 2017, Issue 4, pp. 32-51.
27. G o v i n d a, K., E. S a t h i y a m o o r t h y. Privacy Preservation of a Group and Secure Data Storage in Cloud Environment. – Cybernetics and Information Technologies, Vol. **15**, 2015, Issue 1, pp. 46-54.
28. B r i n d h a, K., N. J e y a n t h i. Secured Document Sharing Using Visual Cryptography in Cloud Data Storage. – Cybernetics and Information Technologies, Vol. **15**, 2015, Issue 4, pp. 111-123.
29. L i, Y., B. D o n g. The Algebraic Operations and Their Implementation Based on a Two-Layer Cloud Data Model. – Cybernetics and Information Technologies, Vol. **16**, 2016, Issue 6, pp. 5-26.
30. T u, S.-S., S.-Z. N i u, M.-J. L i. An Efficient Access Control Scheme for Cloud Environment. – Cybernetics and Information Technologies, Vol. **13**, 2013, Issue 3, pp. 77-90.

Received: 11.08.2018; Second Version: 03.02.2019; Accepted: 14.02.2019