International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

VOTE: Verifiable Auditing for Outsourced Database with Token Enforced Cloud Storage

Geeta C M, Rashmi B N, Nikhil R C, Rajkumar Buyya, Venugopal K R

Abstract: ***Database deploying is one of the remarkable utilities in cloud computing where the Information Proprietor (IP) assigns the database administration to the Cloud Service Provider (CSP) in order to lower the administration overhead and preservation expenditures of the database. Regardless of its overwhelming advantages, it experiences few security problems such as confidentiality of deployed database and auditability of search outcome. In recent past, survey has been carried out on the auditability of search outcome of deployed database that gives preciseness and intactness of search outcome. But in the prevailing schemes, since there is flow of data between IP and the clients repeatedly, huge communication cost is incurred at the Information Proprietor side. To address this challenge, we introduce Verifiable Auditing of Outsourced Database with Token Enforced Cloud Storage (VOTE) mechanism based on Merkle Hash Tree (MHT), Invertible Bloom Filter(IBF) and Counting Bloom Filter(CBF). The proposed scheme reduces the huge communication cost at the Information Proprietor side and achieves preciseness and intactness of the search outcome. Experimental analysis show that the proposed scheme has totally reduced the huge communication cost at the Information Proprietor side, and simultaneously achieves the preciseness and intactness of search outcome though the semi- trusted CSP deliberately sends a null set.***

Keywords: Cloud Computing, Database Encryption, Integrity Auditing, Invertible Bloom Filter, Outsourcing Computation, Query Auditing.

I. INTRODUCTION

Distributed computing allows adaptable and on-demand network access to a concentrated pool of customizable reckoning resources. It has enough advantages for applications for instance, universal network access, locality independent resource pooling, fast resource adaptable, utilization-based pricing and deploying. In the outsourcing reckoning model, the customers can deploy the extravagant reckoning and repository into the *CSP* and relish the extensive reckoning and repository utilities in a pay-as-you-go fashion [1]. Repository outsourcing is becoming progressively appealing to both commercial enterprise and academic community because of the benefits of reasonable price, high availability and easy distribution.

Revised Manuscript Received on December 30, 2019.
* Correspondence Author
Geeta C Mara*, CSE, University Visvesvaraya College of Engg, Bangalore University, Bangalore, India. Email: geetacmara@gmail.com
Rashmi B N, CSE, University Visvesvaraya College of Engg, Bangalore University, Bangalore, India. Email:rashmikaindian@gmail.com
Nikhil R C, ASE, Jain University, Bengaluru, India. Email:nikhilrc1504@gmail.com
Rajkumar Buyya, CSE, The University of Melbourne, Australia. Email:rbuyya@unimelb.edu.au
Venugopal K R, Bangalore University, Bangalore, India. Email: venugopalkr@gmail.com

As one of the repository deploying forms, distributed repository acquires wide recognition in the last few years. One of the primary utilities of distributed computing is deploying the database. Hacigumus [2] originally implemented the model of database deploying. In the deployed database framework, the *IP* transfers the database administration to the *CSP*, in order to alleviate the huge database preservation cost. Furthermore, the *IP* carry out the database encode process and deploys the encoded database along the equivalent labels to the *CSP*.

Now a day's optical networks [3], [4] have been deployed all over the globe for efficient information communication. The *CSP* is liable for furnishing all imperative resources and utilities to the clients. The clients can generate numerous inquiry requests to the *CSP* and accept the equivalent outcome from the *CSP*. The Database-as-a-Service (*DAS*) framework, presented in [5], is one of the significant database deploying framework. In this framework, the *CSP* is accountable for providing schemes for customers to generate, fetch and modify their deployed information. Anyhow, the database deploying arises with the issue on the reliability of customer's information, and it is necessary to furnish suitable safety means for preserving the deployed information from mischievous external adversary and the *CSP*. Security in database deploying typically involves information secrecy and information integrity.

Initially, to provide the privacy of the information, conventional encode mechanisms can guarantee that the *CSP* is unable to learn about the deployed information. The information integrity involves two features, repository integrity and query integrity. Repository integrity implies to the capability to examine if the deployed information are misplaced or manipulated without fetching it. Numerous prevailing mechanisms like confirmable information possession [6] and proofs of retrievability [7] focus to figure out this problem. Query integrity implies to the capability to examine the preciseness and intactness [8], [9] of the inquiry outcome sent from the *CSP*. Precisely, preciseness implies that the proprietor ought to be proficient to examine the genuineness of the reverted outcome, i.e., the reverted documents are existing in the deployed database and the documents have not been altered. Intactness indicates that the inquiry outcome needs to contain all original documents that satisfy the query conditions. Several confirmable deployed database mechanisms that are implemented utilizing *MHT* [10], [11] are suggested to audit the preciseness of search outcome. Few of them are: Ma *et al*., [11] designed an effective authentication mechanism by creating a *MHT* on every respective tuple.

In order to validate the auditability of search outcome, the *CSP* only needs to yield the equivalent Verification Object (*VO*) with signature of *MHT*. Still, the construction does not assure the entirety of search outcome. Pang *et al*., [12] partially reviewed the challenges of entirety based on the signature aggregation method. The limitation of the mechanism is that the scheme does not determine the misbehavior of the *CSP*.

However, many of these schemes are unable to accomplish intactness validation. To address this challenge, few mechanisms are analyzed in accordance with aggregated signature and signature chaining [11], [13]; these mechanisms decline to examine the case when a null set is sent by the cloud. While, the probabilistic auditing techniques accomplish the preciseness by inserting few spurious documents into the deployed database in advance, anyhow it is necessary that the spurious documents need to be recognized by all genuine customers. Currently, Wang *et al*., [8] designed a confirmable auditing mechanism which is constructed utilizing the Bloom filter.

The customer is able to examine the sincerity of search outcome though the *CSP* knowingly sends a null set. The limitation is that the mechanism has huge reckoning and transmission cost. In [9], authors improved their proposal with a *IBF* and *CBF* so that it functions in the dynamic conditions. The proposed mechanism is enhanced to multi-client framework that can withstand conspiracy assault between the *CSP* and any mischievous customers. The drawback is that it has equivalent reckoning and repository overhead. But in the prevailing schemes [8], since there is a flow of data between *IP* and the clients repeatedly, huge communication cost prevails at the *IP* side. To address this challenge, we introduce Verifiable Auditing of Outsourced Database with Token Enforced Cloud Storage (*VOTE*) scheme.

A. *Motivation*

In the existing scheme, any client can request any data from the cloud. As *CSP* is a semi-trusted entity, he does not bother whether the client is legitimate or not. The *CSP* transmits the information relevant to the query, as requested by the client. The information received by the client is in an encrypted form; hence the client sends ciphertext to the Information Proprietor (*IP*). Further, the *IP* confirms whether the client is legitimate person decrypts and send the plaintext data to the client. This is a tedious procedure and hence there is a necessity to reduce the huge transmission cost caused by the irrelevalent transaction of huge data. In the proposed scheme, we allow the clients to send the token that consists of queries to the *IP*. The *IP* decides whether the client is a legitimate person and then sends the token to the *CSP*. The *CSP* sends the requested tuples to the clients. Thus the proposed scheme reduces the huge communication cost prevailing at the *IP* side and achieves preciseness and intactness of the search outcome.

B. Contribution

In this paper, we introduce a Verifiable Auditing for Outsourced Database with Token Enforced Cloud Storage (*VOTE*) mechanism that achieves the preciseness and intactness of search outcome though the semi-trusted *CSP* deliberately sends a null set. Particularly, our contributions can be summarized as follows:

(i) The proposed scheme reduces the huge communication cost prevailing at the *IP* side (in existing scheme) i.e., the flow/interactions with the *IP* and the clients again and again is reduced.

(ii) Clients efficiently checks for the exactness and entirety of the search outcome.

C. Organization

The paper is organized as follows: Initially, we study the related works on confirmable deployed database and the background work gives earlier models and their drawbacks; these are outlined in Section 2. Preliminaries used in the scheme are discussed in Section 3. Problem statement and System architecture are discussed in Section 4. In Section 5, the scheme details of Verifiable Auditing of Outsourced Database with Token Enforced Cloud Storage (*VOTE*) have been proposed. Security analysis is discussed in section 6. Performance evaluations are analyzed in Section 7. Conclusions are presented in Section 8.

II. RELATED WORK

In this section, we study the advanced research corresponding to the verifiable database approaches and we highlight the limitations associated with each approach. Merkle Hash Tree (*MHT*) is utilized to confirm the inquiry integrity [10], [8]. The *MHT* is built for the complete database, in which every leaf node contains an information tuple. The preciseness of inquiry outcome may be checked by re-estimating the signature on the root of the MHT. The limitation is that it has huge amount of transmission and estimation overhead.

Narasimha *et al*., [14] suggested a confirmation of deployed database mechanism that is constructed utilizing signature aggregation and chaining method. As the CSP is unable to differentiate the false tuple from the genuine ones, the customer can randomly accomplish the preciseness of search outcome by auditing if any qualifying false tuple is existing. But it is required that every false tuple need to be distributed by all the approved clients. It indicates that the adversary may know the false tuples by conspiring with the negotiated client. A versatile and provable search technique [9] is constructed utilizing *IBF* to realize confirmability of search outcome. A justifiable search technique is outlined for multi-customer scenario. The scheme realizes required security goals and is economical in both information transformation and depository overhead. C++ is used to carry out simulations [15]. Xiang *et al*., [16] proposed database outsourcing convention Secure *DBS* utilizing the Order-Preserving Encryption (*OPE*) method. The convention bolsters numerous *SQL* statements over encoded database, for e.g., information retrieval, dynamic insertion and deletion. The limitation of the scheme is that the protocol reveals the orders of attribute values to DSP's. Venugopal *et al*., [17] have applied soft computation procedures for data mining applications for repository. Zhang *et al*., [18] introduced a publicly justifiable reckoning mechanism for batch matrix multiplication.

The mechanism is safe under the *co−CDH* presumption. The advantage of the scheme is that it has reduced reckoning cost. The drawback is that it has expensive computation cost in the *KeyGen* phase.

Xiang *et al*., [19] suggested a justifiable examination scheme for deployed database without a public examiner. The scheme can concurrently prove the preciseness and entirety of inquiry outcome, restricting the mischievous *CSP* from sending spurious or partial outcome to the customers. The advantages of the mechanism are that it bolsters flexible data dynamics. The limitation is that the mechanism has high reckoning cost in auditing phase. Miao *et al*., [20] proposed a tangible *VDB* architecture that supports dynamic keyword search. The advantages of the scheme are that it can concurrently carry out provability of search outcome and sincerity of database. The limitation is that the scheme does not bolster for multi-keyword setting.

Shen *et al*., [21] designed a secure provable database system that realizes the provability of database documents in the cloud. The proposed mechanism achieves the features of security, exactness, provability and accountability. The limitation of the mechanism is that the reckoning cost is more. Xiang *et al*., [22] proposed provable verifying scheme for deployed database without a public verifier. The mechanism bolsters partial attribute recovery and extensible data dynamics. The limitation is that the mechanism has high auditing time cost. The comparison of existing schemes for verifiable auditing of outsourced database is shown in Table I

A. Background Work

Wang *et al*., [8] proposed an original provable auditing mechanism for the deployed database that can concurrently accomplish the preciseness and entirety of search outcome though the mischievous CSP intentionally sends a null set. In addition, the scheme achieves the required security features even in the encoded deployed database. The limitation of the mechanism is that it does not strengthen the dynamic database setting.

III. PRELIMINARIES

A. Invertible Bloom Filter

Counting Bloom Filter (*CBF*) [9] is a data structure that can detect a given element in a set *S* and can carry out insert/delete operation. The Invertible Bloom Filter (*IBF*) [23], is an improved variant of *CBF* that can effectively determine a specified component of a set. The *IBF* uses three arbitrary hash functions f_1, f_2, g, f_i: [1, p] → [1, q] and g: [1, p] → [1, p_2]. Assume that (h_1, . . . , h_k) are *k* discrete hash functions, where h_i: (0,1)∗ → [1, q]. Every cell in *IBF* consists of three fields: a count field, an *idSum* field and a *hashSum* field. We construct a complementary Bloom filter (called as *C*), that has the similar configuration as *B*, but utilizes only two functions f_1 and f_2 to map items to its cells.

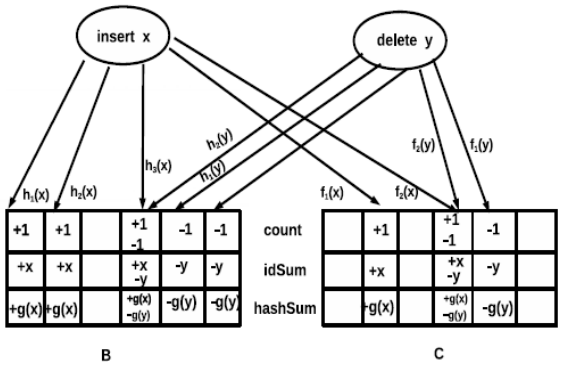

Fig. 1: The Update operations of Invertible Bloom Filter.

The element *x* is inserted into the *IBF* [See Fig. 1.], by choosing *k* cells that *x* map to and then the count field is incremented, next, add x and *g(x)* to the *idSum* and *hashSum* field, respectively. Likewise, the deletion operation can be accomplished by just decrementing the counts and subtract out the corresponding summands.

Merkle Hash Tree

The Merkle Hash Tree (*MHT*) [24] is a binary tree that is utilized to validate information with reduced correspondence
cost. Every inner node holds the hash value of the concatenation of its child nodes and every leaf node holds hash value $h(d_i)$ of the verified information value d_i, where h(·) is a one-way conspiracy-resistant hash function. For example, given a data set $D = (d_1, d_2, d_3, d_4)$, as illustrated in Fig. 2, $h(i) = h(d_i)$, $i \in [1,4]$, $h_{1-2} = h(h(d_1) \| h(d_2))$, where *k* indicates string concatenation. The root of *MHT* is signed by customary public key signature method. The *MHT* is utilized to validate any subset of *D* by employing the Verification Object (*VO*). For example, in order to validate d_1, the *VO* consists of h_2, h_{3-4} and the signature of h_{1-4}. The examiner estimates h'_{1-4} = h(h(h(1) ‖ h(2)) ‖ h_{3-4}) and examines whether h'_{1-4} is equal to h_{1-4}. If so, d_1 is valid; otherwise, it has been tampered with.

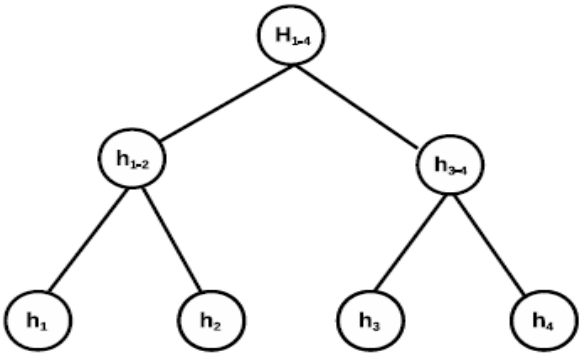

Fig. 2: An Example of Merkle Hash Tree.

Tuple-MHT: It refers to a *MHT* constructed on an

Table-I:Comparison of schemes for verifiable auditing of outsourced database

Authors	Concept	Performance	Advantages	Disadvantages
Miao *et al.*, 2019 [20]	Publicly provable database mechanism with keyword search.	Computation overhead is more expensive.	Simultaneously achieve verifiability of search outcome and sincerity of database.	Does not support for multi-keyword setting.
Xiang *et al.*,, 2018 [19]	Realizing provable, dynamic and adequate inspecting for deployed database in cloud.	Time cost of integrity verification is more.	Supports information dynamics and fractional attribute retrieval.	More computational cost in correctness verification.
Xiang *et al.*,, 2018 [22]	Confirmable examining mechanism for deployed database without a public verifier.	Communication cost is low.	Supports partial attribute recovery and extensible data dynamics.	Time cost of integrity verification is more.
Zhang *et al.*,, 2017 [18]	Publicly verifiable computation scheme for batch matrix multiplication.	Remarkable time savings on the key construction and the estimation phases.	Supports public verification and public delegation.	Expensive computation overhead in the *KeyGen* phase.
Shen *et* 2017 [21]	Provable database mechanism supporting effective dynamic operations in cloud	The update operation is more efficient	Achieves the properties of security, exactness, verifiability and accountability.	The reckoning cost at the server side is more
Xiang *et al.*,, 2016 [16]	Processing safe, provable and effective *SQL* statements over outsourced database.	Less reconstruction cost for range queries.	Supports *SQL* statements over encoded database.	The orders of attribute values are revealed to DSPs.
Our's scheme	Verifiable Auditing of Outsourced Database with Token Enforced Cloud Storage.	Huge communication cost is reduced prevailing at the data owner side.	Achieves the exactness and entirety of search results.	Scheme does not support multi-user setting.

isolated tuple. In particular, every attribute of tuple is identified by a leaf node in the *Tuple-MHT* [9] creation. Analogous to the creation of *MHT*, the value of every inner node is obtained from the hash of the concatenation of its two child nodes. The procedure is carried out repeatedly till the root value is allocated. Subsequent to that, the signature of the root is created by the *IP*. The confirmation procedure of *Tuple MHT* is identical to the *MHT*.

IV. PROBLEM DEFINITION

Given the Information Proprietor (*IP*) encrypts the database and outsource to the *CSP* and group of users queries the tuples to the *IP*, the main objectives are:

- The proposed scheme reduces the huge communication cost prevailing at the *IP* side (in existing scheme) i.e., the flow/interactions with the *IP* and the clients is reduced.
- Clients efficiently checks for the preciseness and intactness of the search results.

A. System Model

The system framework of verifiable search for deployed database is depicted in Fig. 3. The framework comprises of four objects: Information Proprietor, Group of customers, Cloud Service Provider *(CSP)* and Arbitration Center (*AC*). The *IP* initially encodes the database and deploys to the *CSP*, and creates corresponding confirmation format that validates the sincerity of the deployed database. Any customer (from the group of user's) who is interested to query the tuples in the database, sends token which consists of user *id* and search query request to the *IP*. The *IP* decides whether to pass this token to *CSP* or decline the request or if the user is not reliable, user is revoked.

Once the *CSP* accepts the token from the *IP*, *CSP* sends the encrypted tuples as search results to the clients. The client accomplishes confirmation on retrieved ciphertext tuples. If verification is successful then the client decrypts the ciphertext tuples with the key sent by the *IP*. Otherwise, the client transmits the conflict index r_q to the *AC*. The *AC* checks r_q with the *IBF* and suggests the client whether to accept or reject the r_q. In the case, when the outcome that is sent by the *CSP* is a null set, *AC* resolves the issue and suggests the clients. In the existing system, huge communication cost prevails due to the decryption happening at the *IP* side.

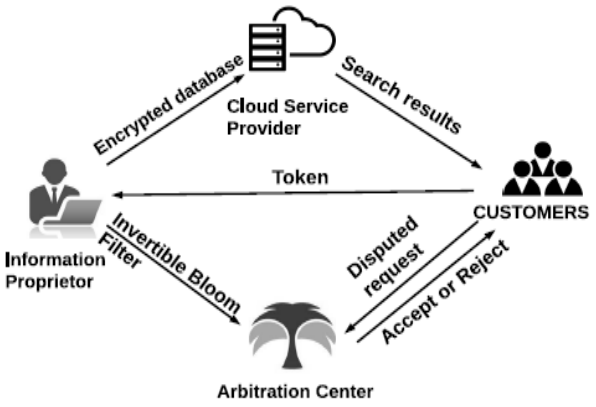

Fig. 3: System framework of verifiable search for outsourced database

But in the proposed system, the users decodes their respective tuples by utilizing the key forwarded by the *IP*, thus we have reduced the irrelevant transactions of huge data.

B. Threat Model

In this paper, we assume that the *CSP* is semi-trusted entity but inquisitive, i.e., it might not precisely act in accordance to the proposed convention and sends preciseness and intactness search outcome. Hence, we consider two types of attackers:

- An outside adversary is an object that targets to acquire a bit of information of the database *via* public channels. This type of adversary involves the repudiated customer and illegitimate customer.
- An internal adversary might possess a bit of information of database (i.e., the *CSP*). Their objective is to furnish a spurious search outcome without being identified.

V. THE ALGORITHM

A. System Setup Phase

Π= (K(1^{λ}), *Enc*, *Dec*) and Π^0 = (K^0 (1^{λ}), Enc^0, Dec^0) be *IND-CPA* reliable symmetric encode mechanisms. Define a pseudo-random permutation $P: K\times M\rightarrow K$, where K and M have the same length. The *IP* constructs the master key $\widehat{\kappa}$ = (κ_0, κ_1, κ_2), where $\kappa_0 \xleftarrow{R}$ K(1), κ_1, $\kappa_2 \xleftarrow{R} K^0(1^{\lambda}) : \kappa^0$ is the key of Π, κ_1, κ_2 are the keys of *P*. The *IP* allocates (κ_1, κ_2) with all genuine customers. *IP* initializes the *IBF* with κ hash functions that can map integer from [1, 2,..., *N*] to κ distinct cells from *IBF*. Then all specific attributes a_i and their indexes κ_i are utilized as a form of (key, value) pair and entered it into *IBF*. The *CBF* is initialized with *k* hash function. Further, all attribute values are embedded into *CBF*. *IP* also generates key for the root of *Tuple-MHT*, i.e., the *IP* constructs *Tuple-MHT* for every clients search request for their respective tuples. Then *IP* shares these keys with the respective clients.

B. Data Outsourcing

Assume that the *IP* uploads a relational database $D = (A_1, A_2, \ldots, A_n)$ to the CSP. The *IP* encodes information tuple by tuple in the manner of [8]. The details are outlined as follows:

- The *IP* estimates $h(a_i)$ for every attribute value a_i, where $h(\cdot)$ is a traditional cryptographic hash function, for e.g., *SHA-1*. Next, all $h(a_i)$ are viewed as a leaf node to build a *MHT*. The root is denoted as h(r) [See Algorithm 1, Phase I].
- The *IP* encodes every attribute value a_i with its index r_i by employing an *IND-CPA* reliable symmetric encryption: $c_i = Enc\ k_0 (r_i \| a_i)$.
- The *IP* computes tag generating key $k_{si} = P_{k1}(a_i)$ and intermediate ciphertext $s_i = P_{k2}(a_i)$, then generates search tag $t_i = Enc^0_{ksi}(s_i)$. The ciphertext tuple is represented as $r^E = (t_1, c_1), \ldots, (t_n, c_n) \| h(r) \| Sig(h(r))$. *CBF* represents the first level of *IBF*. Lastly, the *IP* outsources r^E and *CBF* to the *CSP*. Simultaneously, the *IBF* is transmitted to the *AC* [See Fig. 4.].

C. Data Retrieving

Assume that a client desires to search information tuple fulfilling the subsequent search condition $A_q = a_q$. The process is carried out as follows:

- The client primarily produces a token, $T = (id_i, R) = (id_i, q, k_q, A_q^E) = (id_i, P_{k2}(a_q), P_{k1}(a_q), A_q^E)$ and sends it to the IP [See Fig. 5].
- Upon receiving the token, the *IP* decides whether to pass this *T* to *CSP* or decline the request or if client is not reliable, the client is revoked. Once the *IP* confirms that the client is an authorized person, then he sends the token to the *CSP*.
- After obtaining the token consisting of search request *R*, the *CSP* examines if, $Dec_{kq}(t_i) = q$ is satisfied for every tuple (t_i, c_i) of attribute A_q^E. Then, the information tuples fulfilling the equality and *CBF* are sent to the client otherwise, the *CSP* sends *CBF* as proof to the client.
- Further, the client decrypts c_i with the key sent by the *IP*. In the existing scheme [8] huge communication cost prevails as the client sends the ciphertext to the *IP* and the *IP* performs decryption and sends the plaintext to the client. In our proposed scheme, irrelevant transactions of huge data i.e., communication cost is reduced [See Algorithm 1, Phase II].

D. Verifying

The client examines the legitimacy of search outcome in terms of both preciseness and intactness [9] as follows:

Case 1: The outcome is a null set:

The client verifies the preciseness of the accepted tuples by utilizing the *CBF*. If all the retrieved tuples are existing in the *CBF*, the client accepts the tuples; otherwise the conflict index r_q is transmitted to the *AC*. The *AC* categorizes all the items of *IBF*. The *AC* examines if r_q is incorporated in the IBF or not and suggests the client to accept or decline [See Fig. 6].

Case 2: The outcome is not a null set:

The client examines the preciseness of the search outcome, by re-estimating *h(r)* for the retrieved tuples using the Verfication Object (*VO*). The client verifies the legitimacy of *h(r)* by examining its equivalent *Sig(h(r))*. If it succeeds, the preciseness of the tuples is confirmed and the client accepts the tuples. If the least counter of all hashing positions $h_i(r_q)$ is equal to the number of the accepted information tuples, the preciseness of the tuples is successful and the client accepts the tuples. When the client finds that if the retrieved tuples do not satisfy the preciseness and intactness property, the client transmits the conflicted index r_q to the *AC*. The *AC* categorizes all the existing items of the *IBF*. The client inspects the counter value of the pure cell of r_q is equal to the tuple number in the search outcome. If it is successful, then the preciseness and intactness of the search outcome is accomplished and suggests the clients to accept otherwise decline will be submitted [See Algorithm 1, Phase III]. The summary of notations used in the algorithm is shown in Table II.

VI. SECURITY ANALYSIS

Theorem 1: A proposed provable mechanism supports privacy in deployed database.

Proof: With the aim of accomplishing information privacy, the *IP* encodes the database that permits the *CSP* to carry out relational operations on

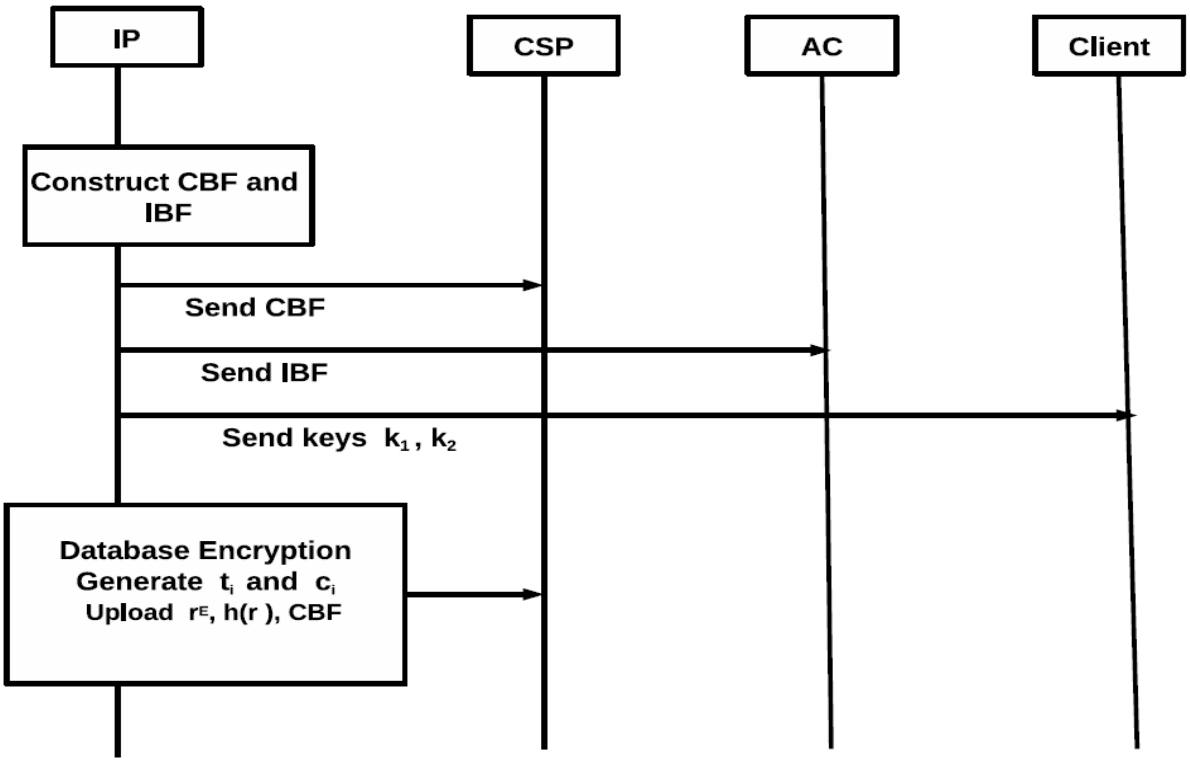

Fig. 4. Overview of data outsourcing phase

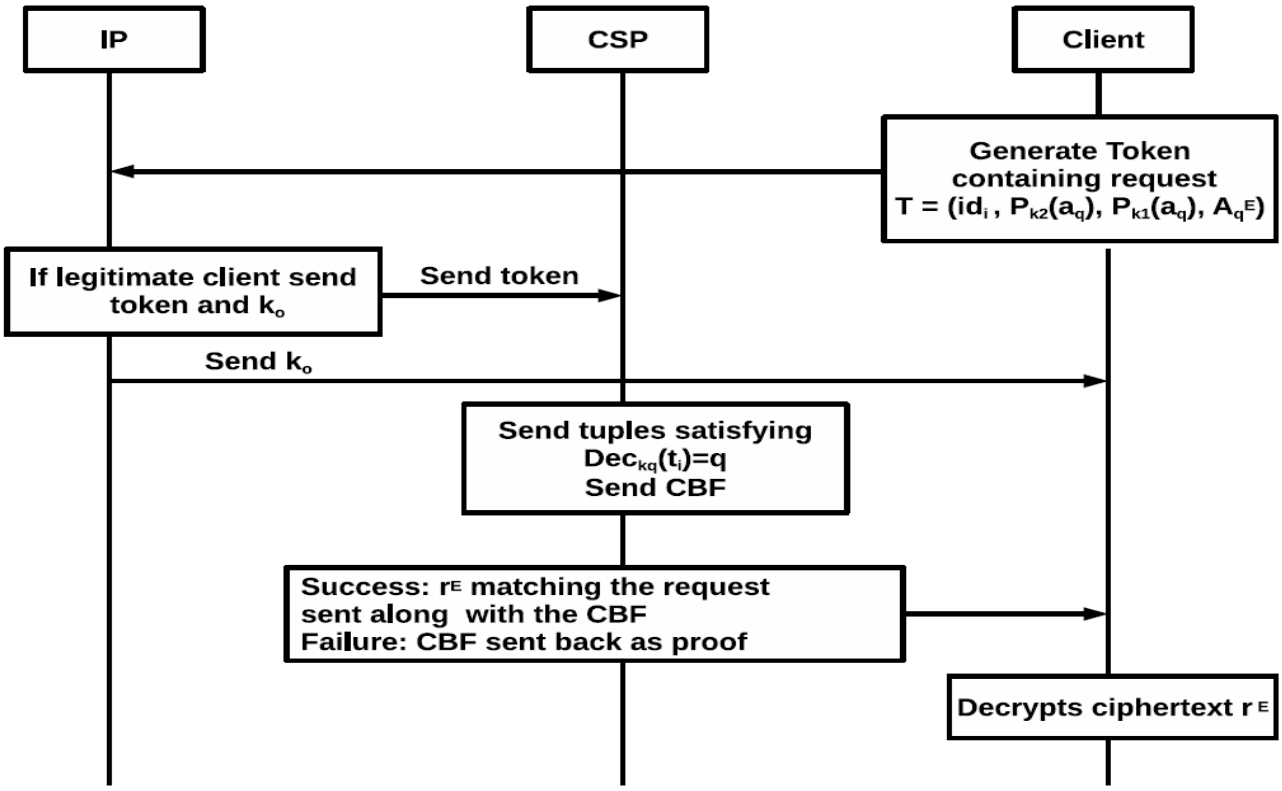

Fig. 5. Overview of data retrieving phase

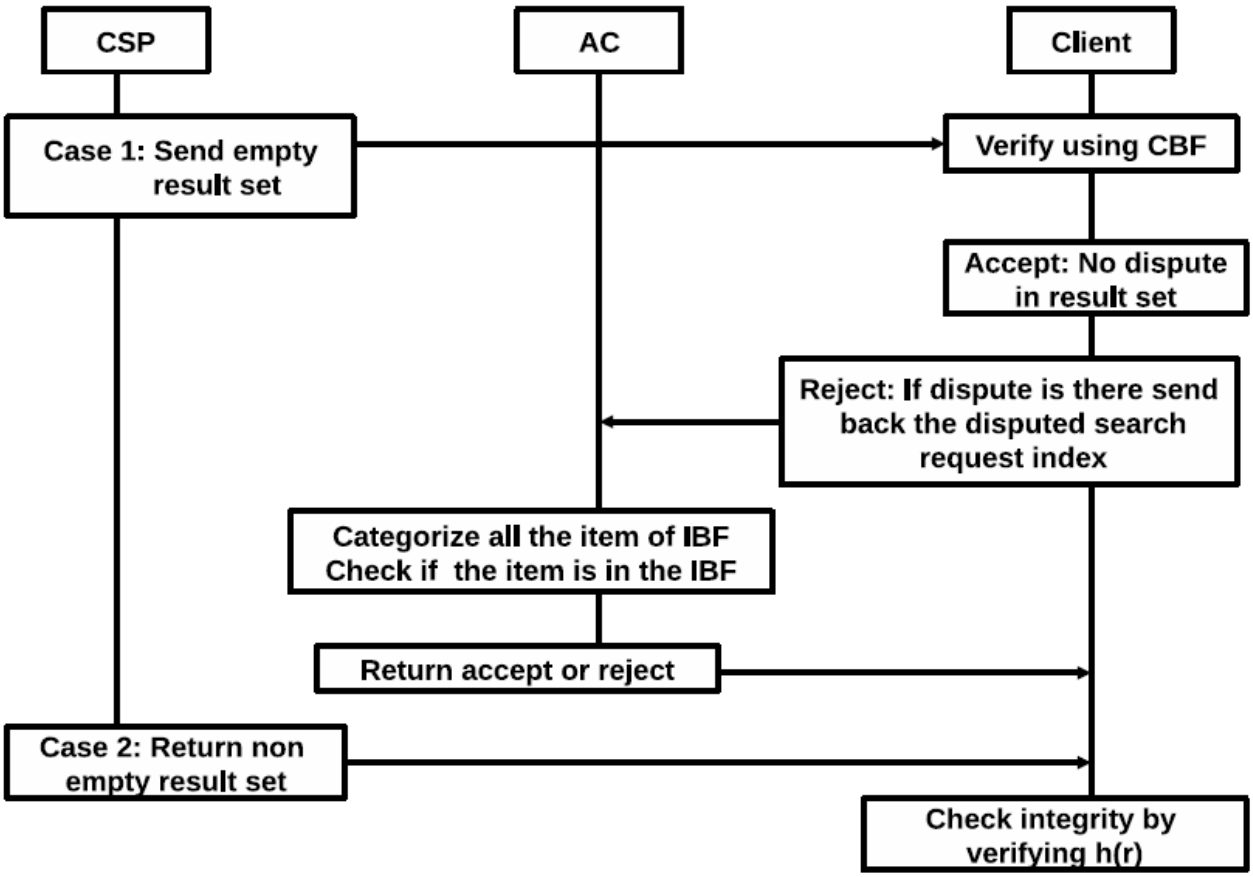

Fig. 6. Overview of verification phase

encoded database without decoding it as follows:

Assume that the client desires to fetch tuples that fulfills $A_i=a_i$. Then the client can transmit tuple $R_q = (t_q, A_i^E) = (H_{k1}(a_i), A_i^E)$ to the *CSP*. After accepting tuple R_q the *CSP* verifies (t_i,c_i) of attribute A_i^E tuple for equality $t_q = t_i$ in tuple by tuple method. If it succeeds, then *CSP* transmits every fulfilling tuples to the client. Tuple R_q does not consist of any plain text. Thus, the *CSP* can achieve relational operation on the tuples in *ODB* without decoding it. However, *CBF* and *IBF* does not disclose the data as arbitrary index is created for every attribute a_i to cache in *IBF* and *CBF*. No person can locate correlation between indexes and attribute without decoding it. The assaulter and unapproved client cannot be able to retrieve plaintext without encode mechanism. Therefore, encode key is required to maintain safely by the client. We are presuming the approval amid client and the *IP* is accurately done. It is also assumed that the assaulter or repudiated client cannot obtain legitimate encode key. Thus, we can realize information privacy.

Theorem 2: Proposed provable mechanism ensures preciseness of search outcome in deployed database.

Proof: When the outcome is a null set, then the client verifies preciseness of search outcome by utilizing the *CBF*. Then, the client verifies *CBF* $(r_q) = 0$. If this condition succeeds, then r_q prevails in *CBF*. Otherwise, the client transmits r_q to the *AC*. The *AC* categorizes all the existing items of the *IBF*. Then it examines if r_q is there in *IBF* or not. If *IBF* contains r_q, then decline is transmitted to the client, otherwise *AC* transmits accept to the client. An additional case is while the outcome is not a null set, then the client verifies the preciseness of the search outcome by reconstructing *h'(r)* utilizing the acknowledged *VO* and examines it with restored *Sig(h(r))*. If this condition succeeds, then every tuple of the search outcome has not been altered. If the CSP updates earlier ciphertext c_i to c'_i then based on conspiracy resistance of the hash function, the original false *h'(r)* is discarded with overwhelmed probability. Hence, we can verify preciseness of search outcome in deployed database.

Theorem 3: Proposed provable mechanism assures entirety of search outcome in deployed database.

Proof: When the *CSP* does not send a null set, then the client can examine entirety of the search outcome by inspecting the least counter of all hashing locations of $h_i(r_q)$ which is equivalent to the sum of tuples that is sent by the *CSP*. If it is successful, then it outputs accept and the procedure declines; otherwise, the client transmits conflict index r_q to the *AC*. *AC* makes a checklist of every items of the *IBF*. Then, the *AC* verifies the counter value of pure cells of r_q with sum of retrieved tuples. If this condition succeeds, the results are accepted otherwise, outputs decline and procedure is terminated. Thus, we can realize the entirety of the search outcome in the deployed database.

Algorithm 1: VOTE: Verifiable Auditing for Outsourced Database with Token Enforced Cloud Storage

---

Input: $D = (A_1, A_2, \ldots, A_n)$, data tuple=$(a_1, \ldots, a_n)$, R, T
Output: r^E, CBF, IBF

Phase I: Data Outsourcing:

- Assume the *IP* upload a relational database $D = (A_1, A_2, \ldots, A_n)$ to *CSP*.
- For every attribute value a_i, the *IP* randomly chooses an integer $r_i \leftarrow Z_N$ and inserts r_i into the IBF_{Ai}
- The *IP* encodes information, tuple by tuple as follows:
- Step 1: The *IP* estimates $h(a_i)$ for each attribute value a_i. Then, all $h(a_i)$ is viewed as a leaf node to design a *MHT* and h(r) denotes the root, hash of tuple.
- Step 2: The *IP* encodes every attribute value a_i along with its index r_i: $c_i = Enc\ k_0(r_i \| a_i)$.
- Step 3: The *IP* computes $k_{si} = P_{k1}(a_i)$ and $s_i = P_{k2}(a_i)$, then generates search tag $t_i = Enc^0_{ksi}(s_i)$. The ciphertext tuple is: $r^E = (t_1, c_1), \ldots, (t_n, c_n) \| h(r) \| Sig(h(r))$.
- Step 4: Finally, the *IP* outsources r^E and *CBF* to the *CSP*. *IBF* is transmitted to the *AC*.

Phase II: Data Retrieving:

- Considering that a client desires to search information tuple fulfilling the search condition $A_q = a_q$. The process will be carried out as follows:
- Step 1: The client primarily creates a token, $T = (id_i, R) = (idi, q, k_q, A_q^E) = (id_i, P_{k2}(a_q), P_{k1}(a_q), A_q^E)$ and sends it to the *IP*.
- Step 2: Upon receiving *T*, the *IP* decides whether to pass *T* to *CSP* or decline the request or if client is not reliable, the client is revoked. Once the *IP* confirms that the client is an authorized person, then he sends T to the *CSP*.
- Step 3: Upon receiving *T* comprising of search request *R*, the *CSP* verifies if $Dec_{kq}(t_i) = q$ holds for every tuple (t_i, c_i) of attribute A_q^E in a tuple by tuple method. Then the information tuples fulfilling the equality and *CBF* are sent back to the client. If no corresponding tuple is identified, the *CSP* only sends *CBF* as proof to the client. Then the client decrypts c_i with the key sent by the *IP*.

Phase III: Verifying:

- The client examines the legitimacy of search outcome in terms of both preciseness and intactness as follows:
- **Case 1:** When the outcome is a null set:
- By utilizing *CBF*, the client verifies the preciseness of the accepted tuples. The client inspects for the condition $CBF(r_q)=0$, if success, the client accepts. Otherwise disputed index r_q is transmitted to the *AC*. The *AC* categorizes all the items of *IBF* and examines if r_q is incorporated in the *IBF* or not and suggests the client to accept or decline.
- **Case 2:** When the outcome is not a null set:

Step 1: The client examines the preciseness of the search outcome, by re-estimating *h(r)* using *VO*.

- Next, the client verifies the legitimacy of *h(r)* by examining its equivalent *Sig(h(r))*, if it fails, the procedure ends otherwise, it goes to *Step 2*.
- *Step 2:* The client examines the entirety of the search outcome, by inspecting if the least counter of all hashing positions $h_i(r_q)$ is equal to the number of the accepted information tuples, if success, the procedure ends and outputs accept else, it goes to *Step 3*.
- *Step 3:* The client transmits the conflicted index r_q to *AC*. The *AC* categorizes all the existing items of the *IBF*. The client inspects the counter value of the pure cell of r_q is equal to the tuple number in search outcome. If success, the preciseness and intactness of the search outcome is accomplished and inputs accept otherwise decline is submitted.

Table- II: Summary of the Notations used in the Algorithm

Notation	Description
$D=(A_1, A_2, \ldots A_n)$	Relational database
$A_1, A_2, \ldots A_n$	Attribute column
$a_1, a_2 \ldots\ldots a_n$	Data tuple
k_{si}	Tag generating key
s_i	Intermediate ciphertext
t_i	Search tag
r^E	Ciphertext tuple
c_i	Ciphertext
Sig(h(r))	Signature of h(r)
h(r)	Root of hash of tuple
T	Token
A_q^E	Ciphertext of attribute column of A_q
VO	Verification object
r_q	Tuple index
Id	Client identifier
R	Request query sent by client to the *IP*

VII. PERFORMANCE EVALUATION

In this section, we present an experimental evaluation of our proposed mechanism. All experiments are carried out on the same machine with Intel(*R*) Core(*TM*) *i5-6500U CPU @ 2.50 GHz 2.59GHz* and *8G RAM* Memory. We fix a number of hash functions of *IBF k=3*, the false positive of *IBF* is 2^{-20}. We implement hashing functions by utilizing *hashlib* library. We implement *AES* encode operation and signature certificate by utilizing *OpenSSL*. We have fixed the number of attributes for every tuple as 8.

System setup phase mainly consists of creation of two main data structures i.e. *CBF* and *IBF*. The comparison of these two data structures is shown in Fig. 7. The existing schemes utilized the Bloom filters to identify the presence of data. In this paper, we have used the improved version of Bloom filter which are *CBF* and *IBF*. These data structures support insertion and deletion of the data. The probability of the false positives occurring when using these data structures is greatly reduced when compared to the Bloom filter.

To find the cost of implementing these data structures, an experiment has been conducted by fixing the hash function *k=3*. The number of tuples taken is in the range 1000 to 8000, where each tuple consists of 8 attributes. From Fig. 7, it is clear that construction of *CBF* is almost constant and negligible even when the number of tuples is increased. The time cost of implementing the *IBF* is increasing linearly with increase in the number of tuples. This is because three fields are managed for *IBF*.

During data deploying process, three main processes are involved. First, the *IP* performs encryption of the data and then signature creation and tag generation is carried out on the search data. As shown in Fig. 8, experiment has been carried on tuples from 1000 to 8000. We can observe that the time needed for signature generation is lower compared to ciphertext and label construction time. The time cost for tag generation and cipher text creation is increasing linearly with increase in tuples.

Fig. 9 shows the time taken by the clients to retrieve the data tuples from the *CSP*. It is observed that the time taken by *VAODC* is more compared to *VOTE* scheme. In the *V AODC* scheme, the clients sends the ciphertext tuples to the *IP*, where it checks if the client is an authorized person; then he decrypts and sends the plaintext to the clients and the time taken is more. But, in the proposed scheme *VOTE*, after receiving their requested ciphertext tuples, the client's checks for the preciseness and intactness of the search results, and decrypt with the key sent by the *IP*. The key sent by the *IP* is the key of the root of the Tuple *MHT* constructed for the requested tuple. In the *VAODC* scheme,

huge communication cost prevails due to the decryption happening at the *IP* side. But in the *VOTE* scheme, by utilizing the key transmitted by the *IP*, the users decode the

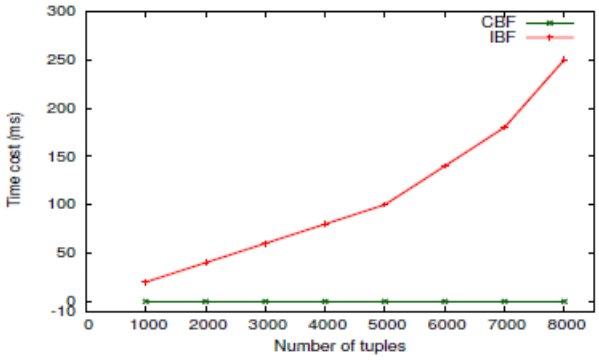

Fig. 7. Construction of *CBF* and *IBF*

tuples; hence we have reduced the irrelevant transactions of huge data.

After receiving information tuples from the *CSP*, the client checks the search outcome in terms of preciseness and intactness as follows: (i) When the search outcome is a null set, the *CSP* send *CBF* as a proof to the client. The client can examine the sincerity of the *CSP* by verifying the *CBF*.

From Fig. 10., it can be seen that computation overhead for verifying signature is almost constant. (ii) When some result set is fetched by the *CSP*, in this case, the client examines the integrity of the information by recreating the hash values and signature certification. The main reckoning cost is auditing the signature of the root hash value of *MHT* and regeneration of hash value. Fig. 10., shows the time needed for verification is linearly increasing with the number of the information tuples of r_q with sum of retrieved tuples. If this condition succeeds, the results are accepted otherwise, outputs decline and procedure is terminated. Thus, we can realize the entirety of the search outcome in the deployed database.

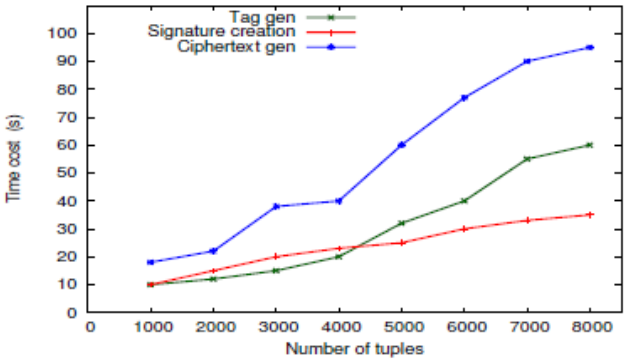

Fig. 8. Time cost of data outsourcing.

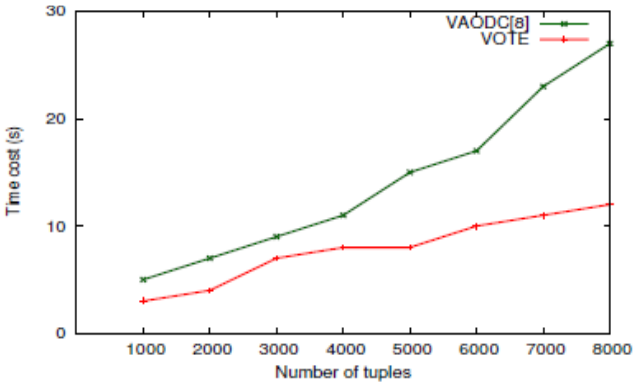

Fig. 9. Time cost of data retrieving.

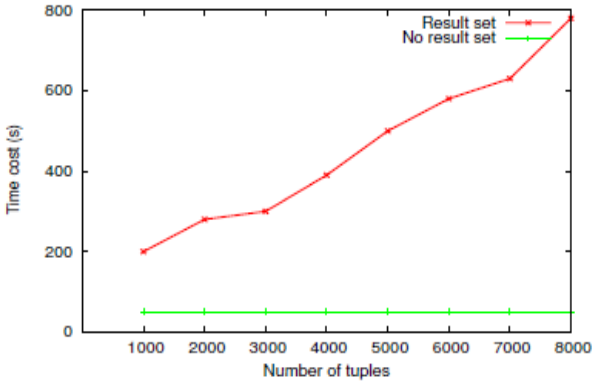

Fig. 10. Time cost of verification process.

VIII. CONCLUSIONS AND FUTURE WORKS

We proposed Verifiable Auditing for Outsourced Database with Token Enforced Cloud Storage (*VOTE*) scheme based on *MHT*, *IBF* and *CBF*. The proposed mechanism accomplishes confidentiality of Outsourced Database (*ODB)* by encoding information before outsourcing to the cloud. We also accomplished preciseness and intactness of the search outcome in *ODB* through *IBF* and *CBF*. The proposed mechanism supports effective data update and can be applicable to dynamic deployed database scenario. The security analysis illustrates that our mechanism accomplishes the intended security goals. The performance analysis demonstrates that our mechanism has totally reduced the huge communication cost prevailing at the information proprietor side, and simultaneously achieves the preciseness and intactness of search outcome though the mischievous *CSP* deliberately returns a null set. In future work, we can enhance our mechanism to multi-client setting.

REFERENCES

1. M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E.E.Spafford, “Secure Outsourcing of Scientific Computations,” vol. 54, pp. 215–272, 2002.
2. H. Hacigumus, B. Iyer, and S. Mehrotra, “Providing Database as a Service,” in Proceedings 18th International Conference on Data Engineering. IEEE, pp. 29–38, 2002.
3. K. R. Venugopal, E. E. Rajan, and P. S. Kumar, “Performance Analysis of Wavelength Converters in WDM Wavelength Routed Optical Networks,” in Proceedings Fifth International Conference on High Performance Computing (Cat. No. 98EX238). IEEE, pp. 239–246, 1998.
4. K. R. Venugopal, E. E. Rajan, and P. S. Kumar, “Impact of Wavelength Converters in Wavelength Routed All-Optical Networks,” Computer Communications, vol. 22, no. 3, pp. 244–257, 1999.
5. H. Hacig¨um¨us¸, B. Iyer, C. Li, and S. Mehrotra, “Executing SQL over Encrypted Data in the Database-Service-Provider Model,” in Proceedings of the ACM SIGMOD International Conference on Management of Data. ACM, pp. 216–227, 2002.
6. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable Data Possession at Untrusted Stores,” in Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 598–609, 2007.
7. H. Shacham and B. Waters, “Compact Proofs of Retrievability,” Journal of Cryptology, vol. 26, no. 3, pp. 442–483, 2013.
8. J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, “Verifiable Auditing for Outsourced Database in Cloud Computing,” IEEE Transactions on Computers, vol. 64, no. 11, pp. 3293–3303, 2015.
9. J. Wang, X. Chen, J. Li, J. Zhao, and J. Shen, “Towards Achieving Flexible and Verifiable Search for Outsourced Database in Cloud Computing,” Future Generation Computer Systems, 2016.
10. P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine, “Authentic Third-Party Data Publication,” in Data and Application Security. Springer, pp. 101–112, 2002.
11. D. Ma, R. H. Deng, H. Pang, and J. Zhou, “Authenticating Query Results in Data Publishing,” in International Conference on Information and Communications Security. Springer, pp. 376–388,2005.
12. H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan, “Verifying Completeness of Relational Query Results in Data Publishing,” pp. 407–418, 2005.
13. E. Mykletun, M. Narasimha, and G. Tsudik, “Authentication and Integrity in Outsourced Databases,” ACM Transactions on Storage (TOS), vol. 2, no. 2, pp. 107–138, 2006.
14. M. Narasimha and G. Tsudik, “Authentication of Outsourced Databases using Signature Aggregation and Chaining,”in International Conference on Database Systems for Advanced Applications. Springer, pp. 420– 436, 2006.
15. K. R. Venugopal and R. Buyya, “Mastering C++,” Tata McGraw-Hill Education, 2013.
16. T. Xiang, X. Li, F. Chen, S. Guo, and Y. Yang, “Processing Secure, Verifiable and Efficient SQL over Outsourced Database,” Information Sciences, vol. 348, pp. 163–178, 2016.
17. K. R. Venugopal, K. G. Srinivasa, and L. M. Patnaik, “Soft Computing for Data Mining Applications,” Springer, 2009.
18. X. Zhang, T. Jiang, K.-C. Li, A. Castiglione, and X. Chen, “New Publicly Verifiable Computation for Batch Matrix Multiplication,” Information Sciences, 2017.
19. T. Xiang, X. Li, F. Chen, Y. Yang, and S. Zhang, “Achieving Verifiable, Dynamic and Efficient Auditing for Outsourced Database in Cloud,” Journal of Parallel and Distributed Computing, vol. 112, pp. 97–107, 2018.
20. M. Miao, J. Wang, S. Wen, and J. Ma, “Publicly Verifiable Database Scheme with Efficient Keyword Search,”InformationSciences,vol.475, pp. 18–28, 2019.
21. J. Shen, D. Liu, M. Z. A. Bhuiyan, J. Shen, X. Sun, and A. Castiglione, “Secure Verifiable Database Supporting Efficient Dynamic Operations in Cloud Computing,” IEEE Transactions on Emerging Topics in Computing, 2017.
22. T. Xiang, X. Li, F. Chen, Y. Yang, and S. Zhang, “Achieving Verifiable, Dynamic and Efficient Auditing for Outsourced Database in Cloud,” Journal of Parallel and Distributed Computing, vol. 112, pp. 97–107, 2018.
23. D. Eppstein and M. T. Goodrich, “Straggler Identification in RoundTrip Data Streams via Newton’s Identities and Invertible bloom Filters,” IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 2, pp. 297–306, 2010.
24. R. C. Merkle, “Protocols for Public Key Cryptosystems,” in 1980 IEEE Symposium on Security and Privacy. IEEE, pp. 122–122, 1980.

AUTHORS PROFILE

Geeta C M, is a research scholar in the department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bengaluru. She has received B.E. degree in Electronics and Communication from Basaveshwara College of Engineering, Bagalkot, Karnataka, India and M.E degree in Information Technology, from Bangalore University, Bengaluru, Karnataka, India. Her areas of interest are Cloud Computing and Wireless Sensor networks. She is a student member of the IEEE.

Rashmi B N, is a PG student in the department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bengaluru. She has re ceived B.E. degree in Computer Science and Engineering from East-West Institute of Technology, Bengaluru, Karnataka, India and M.E degree in Computer Science and Engineering from , Bangalore University, Bengaluru, Karnataka, India. Her areas of interest are Cloud Computing and Internet of Thinks(IoT).

Nikhil R C, is a UG student in the department of Aerospace Engineering, International Institute for Aerospace Engineering and Management, Jain University, Bengaluru, India. He is currently pursuing B. Tech degree in Aerospace Engineering. He has carried out projects under Ministry of Defence organization such as DRDO and CSIR-NAL during the course of his bachelor’s degree. His areas of interest are Cloud Computing, Aerodynamics, Control Systems and Navigation Systems.

Rajkumar Buyya, is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He has authored over 625 publications and seven text books including Mastering Cloud Computing published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. He is one of the highly cited authors in Computer Science and Software Engineering worldwide (h-index=121, 78,000+citations).

Software technologies for Cloud Computing developed under his leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 40 countries around the world. Dr. Buyya is recognized as a Web of Science Highly Cited Researcher in 2016, 2017 and 2018 by Thomson Reuters, a Fellow of IEEE, and Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud Computing.

Venugopal K R, is currently the Vice Chancellor, Bangalore University, Bengaluru. He obtained his Bachelor of Engineering from University Visvesvaraya College of Engineering. He received his Masters degree in Computer Science and Automation from Indian Institute of Science Bangalore. He was awarded Ph.D in Economics from Bangalore University and Ph.D in Computer Science from Indian Institute of Technology, Madras. He has a distinguished academic career and has degrees in Electronics, Economics, Law, Business Finance, Public Relations, Communications, Industrial Relations, Computer Science and Journalism. He has authored and edited 64 books on Computer Science and Economics, which include Petrodollar and the World Economy, C Aptitude, Mastering C, Microprocessor Programming, Mastering C++ and Digital Circuits and Systems etc., He has filed 101 patents. During his three decades of service at UVCE he has over 640 research papers to his credit. His research interests include Computer Networks, Wireless Sensor Networks, Parallel and Distributed Systems, Digital Signal Processing and Data Mining. He is a Fellow of IEEE and ACM Distinguished Educator.