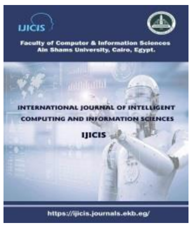
IJICIS, Vol.21, No.1, 104-118
DOI: 10.21608/ijicis.2021.71016.1081

International Journal of Intelligent Computing and Information Sciences

https://ijicis.journals.ekb.eg/



Securing Query Results for Cloud Databases

Mai Rady
Information System Department, Faculty of Computer and Information Science, Ain Shams University,
Cairo, Egypt
mai.mustafa.rady@gmail.com

Tamer Abdelkader
Information System Department, Faculty of Computer and Information Science, Ain Shams University,
Cairo, Egypt
tammabde@cis.asu.edu.eg

Rasha Ismail
Information System Department, Faculty of Computer and Information Science, Ain Shams University,
Cairo, Egypt
rashaismail@cis.asu.edu.eg

Received 2021- 4-4; Revised 2021-5-9; Accepted 2021-5-9

Abstract: *Cloud computing offers several computing service. Among these services is the Database Service, which stores outsourced databases and provides all the database services to users. However, despite, the attractiveness of the database services to users, security issues remain a main concern to database owners, as the data and the execution of database is out of their control. To ensure the outsourced database confidentiality and integrity, which are two of the most important security concerns in data outsourcing, we propose the design and implementation of an efficient Secure scheme to provide Confidentiality and Integrity of Query results for Cloud Databases based on Merkle B-Tree (SCIQ-CD). We develop a technique to convert the Merkle B-Tree into data authentication tables and store it within the relational database in DSP. We have implement query rewrite algorithms for trusted third party to generate the SQL statements to speed up the query processing, which will be sent within the query by the user to DSP. The performance analysis shows that our proposed scheme is secure and efficient for practical deployment. The experimental results show that our scheme imposes a low overhead for queries executions with confidentiality and integrity assurance.*

Keywords: *Cloud Database, Confidentiality, Integrity, Merkle B-Tree.*

1. Introduction

* Corresponding author: Mai Rady

Information System Department, Faculty of Computer and Information Science, Ain Shams University, Cairo, Egypt
E-mail address: mai.mustafa.rady@gmail.com

There is a direct correlation between the amount of stored data on outsourced servers and the attacks that target these servers. These attacks are done through unauthorized access by attackers, who may be the Database Service Providers themselves or other intruders [1].
There are several issues that should be considered when storing a database in cloud service providers (CSPs) [13]. These issues can be listed in the following points: 1) Security issues, the data can be accessed by anyone and from anywhere. 2) Privacy issues, the CSPs enforce their own policies to ensure the security of the database stored in their servers. 3) Application issues, the database monitoring and maintenance should be done by the CSPs. Among the previous issued, the main question here is how we can protect the outsourced database from inside and outside malicious or attackers. Here are the important security issues to be addressed during our work, data confidentiality and data integrity. Data confidentiality is guaranteed by encrypting the database before outsourcing, and only the authorized users can decrypt it [2]. Data integrity of the query results, includes three aspects: correctness, completeness, and freshness. Data integrity is guaranteed by converting the database into authenticated data before storing it in DSP. In this model, we use a Trusted Third Party (TTP) to provide an indirect mutual trust between the data owner/users and the DSP. The TTP checks the data confidentiality and verifies the query result integrity before sending it to the users. As shown in Figure 1, the Database Storage Model (DSM), it consists of four components: the data owner, authorized users, the TTP, and the DSP. We implement and test our scheme on Microsoft Azure Cloud. The performance of this solution is assessed in terms of the response time for data outsourcing, and data retrieving.
The following sections are organized as follows. In Section 2, we present the related work. In Section 3, we propose the secure scheme using MBT-Base Identifier. In Section 4, we show the different data operations on the outsourced database. In section 5, the experimental results and performance analysis are discussed. In Section 6, conclusions are shown.

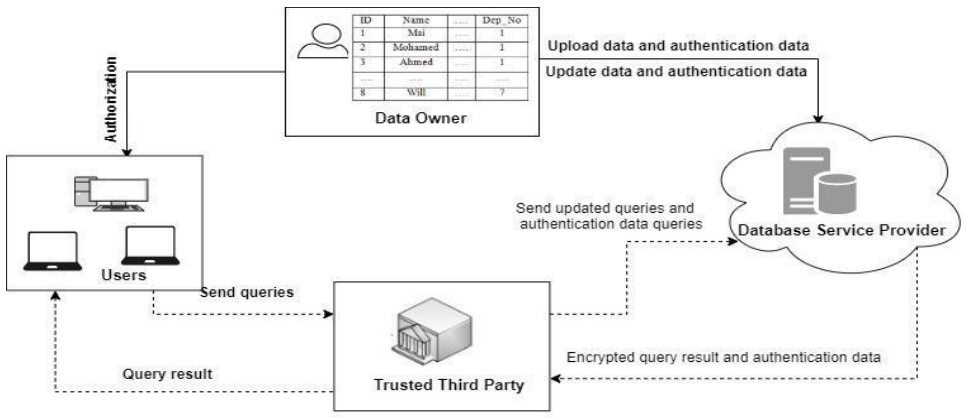

Data Owner
Upload data and authentication data
Update data and authentication data
Authorization
Users
Database Service Provider
Send updated queries and authentication data queries
Send queries
Trusted Third Party
Encrypted query result and authentication data
Query result

Figure 1. Database Storage Model (DSM)

2. Related work

This section briefly presents the most important security concerns, data confidentiality, and data integrity according to the previous related work.

2.1 Data confidentiality:

Data confidentiality is the process of protecting data from illegal access and disclosure from the CSP, intruders and unauthorized users as no one can obtain the plaintext from the outsourced data beyond the query result. This can be guaranteed by encrypting the data and only the authorized users can decrypt it. The authors in [8] added the encryption properties as follows: 1) only the sensitive data is encrypted, 2) using different encryption keys to encrypt different parts of data, 3) during the query execution, only the data that relevant to it is encrypted/ decrypted. These properties are useful while there is another way to ensure data privacy as checking data integrity.

The authors in [9] used vertical fragmentation to protect the confidentiality of data, it hides the users identifies by separating identifier attributes with descriptive attributes. To satisfy privacy constraints, encrypted searching is developed to preserve privacy against adversaries in a cloud computing environment.

In our previous work [12], we have been studied in details how the database confidentiality is guaranteed by encrypting sensitive attributes using the encryption key that we created using AES algorithm, and we create an additional table called, search table (ST), which contains only two columns, the column having the original sensitive attributes, and the corresponding row id from the original table is encrypted using the encryption key. Then, both tables stored in DSP.

2.2 Data integrity:

Data integrity is applied on query results, which is retrieved from the DSP, and includes three aspects:

1. Correctness: the query issuer is able to validate that the returned results do exist.
2. Completeness: the result is complete and no answers have been omitted from it.
3. Freshness: the results are based on the last version of the data.

To check the query result integrity, different solutions have been proposed in literature. The authors in [10], proposed a deterministic private verification integrity check scheme, which uses RSA based accumulator as verification value and this value is stored by the data owner. This scheme limits the computational and storage overhead for both the CSP and the data owner as it prevents data deletion, replacement, and data leakage attacks and detects replay attacks, as the CSP might use an old challenge response to respond to a challenge which is new that matches it.

The authors in [11], proposed a data auditing system that integrates Merkle Tree and block chain methods and uses a third-party auditor (TPA) to verify the data integrity. The core idea is to record each verification result into a block chain as a transaction, and the verification result are time stamps, to let the users check that the verification is performed at the prescribed time.

To check the query result integrity, in our previous work [12], we create the authenticated data structure Merkle B-tree (MBT) over the encrypted database, for each encrypted table, each record is hashed using SHA256. The leaf nodes contain the hash values of the data records. The internal nodes contain a pointer to the leaf nodes. The tree root is signed using secret key that we created using RSA algorithm. Then, we serialize each tree that is converted into a sequence of bits in order to store it within the database in DSP, and to check the data integrity, we de-serialize the tree in the TTP server.

3. The Proposed Security Scheme using MBT Base-Identifier

This section presents the scheme notations that are shown in Table 1, scheme preliminaries, and the discussion of how the proposed scheme guarantees data confidentiality and query results integrity using TBI

The basic notations involved in the proposed scheme that is used in our work are as follows:

Table 1: Scheme notations

Abbreviation/Symbol	**Definition**
DB	Database
ET	Encrypted Table
ST	Search Table
$r_{i..n}$	Record in table ($X_1, X_2, X_3, ..., X_n$)
$d_i = H(r_i)$	Digest, hash output
K	AES data encryption key
EK	Data Encryption
DK	Data Decryption
SK	Secret Key, data owner private key, to sign the data (RSA)
PK	Data owner Public Key to decrypt the signature (RSA)
σ_{ti}	Root signature from root table
l	Level of the MBT
f	Fan-out of the MBT
i	Index of pointer or key in node (0….f)

3.2. Scheme preliminaries:

- Advanced Encryption Standard (AES) [3]: the data owner generates the secret key, K, which is used to encrypt and decrypt a text.
- Hash function (SHA256) [4]: is a function that takes a variable length input r, and outputs a short-fixed length, h(r), called digest, d.
- RSA [7]: the data owner generates two different keys: a public key, Pk, and a private key, Sk. A digital signature is created by a signing function, which takes as input a data value and Sk, and uses them to generate a signature σ. To verify the data value, the verification function is used, which takes as input the signature, σ, together with the public key, Pk.
- Merkle B-Tree (MBT) [5]: is a balanced tree data structure created over each table from the encrypted database. It consists of:
  1. Leaf nodes, which contain the hash values of data records,
  2. Inner nodes, which contain the keys for its child nodes. The associated hash values are computed on the contention of the hash values of their children,

3. The hash of the root, which is signed using Sk.

In the following subsections, we discuss how the proposed scheme guarantees data confidentiality, query results integrity using TBI.

3.3. Data Confidentiality:

By encrypting the sensitive attributes, data confidentiality can be guaranteed. Using the secret key k, the sensitive data from the table are encrypted (Encrypted Table ET), and another table is created. The new table contains two columns, the column having the original sensitive data, while the corresponding record id from the original table is encrypted (Search Table ST). To decrypt the data, the decryption key is stored in the TTP.

3.4. Data Integrity:

Query result integrity is guaranteed using the MBT through the following steps:
To store a tree, we use MBT-Base Identifier (TBI) technique: convert the tree into authentication tables and store it within the DB in DSP.

- Each pointer of the internal node and each key of the leaf node is identified with numbers based on any number equal to the fan-out of the MBT and it depends on the node level and position [6]. TBI is computed by using the following equation.

$$\textbf{\textit{Eq. (1):}}\ TBI = TBI_{parent} * base + \mathrm{i},$$

- Compute the hash digest of each record in the ET for the employee table.

In figure 2, the MBT is shown for data table within the TBI, which are in ternary system, and to store TBIs in authentication tables, we convert it to decimal system, as shown in Table 1, Table 2, and Table 3. The TBI of the root always equal to the index. Each level of MBT is stored in an individual table except the leaf level, as the data table will be extended by adding two columns TBI and hash to store authenticated data record. Each record in data table represents a key from a leaf node of the MBT. Each record in EMP_1 and EMP_0 tables represents a pointer from an internal node. The ID column in EMP_1 and EMP_0 tables, are the keys from the MBT, adding an extra pointer -1 because the number of pointers is always one more than the number of keys. The Hash column holds the hashes associated with the pointers in internal nodes and keys in leaf nodes. In the EMP_0 table, the extra pointer -1 hash value is signed using RSA SK.
A Metadata table is created and stored in TTP server that is used to check data integrity, in which consists of 5 columns (ID, Table_name, PK, SK, Version_No),

-PK: the key used to decrypt the root signature,

-SK: initially store the root signature in SK, and when an update is done in DB, the MBT is updated too, so the signature SK will be changed.

-Version_No: contains a counter that increments each time an update is done in the table.

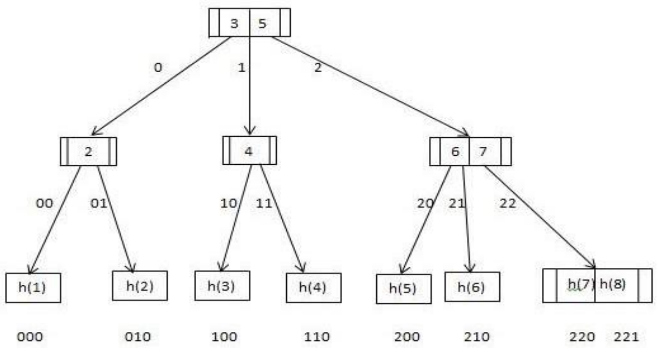


Figure 2: MBT-Base Identifier (TBI)

Table 2: Emp_0

ID	TBI	Hash
-1	0	8z9cxqrUMfP+Q
3	1	2efae55
5	2	634c277

Table 3: Emp_1

ID	TBI	Hash
-1	0	3788957
2	1	ace31427
-1	3	70df12da
4	4	09f55fb6
-1	6	c287feb8
6	7	1f05b76
7	8	5b56746

Table 4: data authentication table

ID	Name	JOB	Salary	Dept_No	TBI	Hash
1	Mai	Engineer	3000	1	0	b569052f48

2	Mohamed	Engineer	3000	1	3	7a9bad81750
3	Ahmed	Engineer	3000	1	9	fa286ace226
4	Heba	Engineer	3000	1	12	38f8dee21a
5	Huda	Specialist	2500	3	18	e8ca57f3f27
6	Alaa	Specialist	2500	3	21	0787f6b5e53
7	John	Technician	2000	7	24	2e84cfd88d3
8	Will	Technician	2000	7	25	3b21ac01353

3.4. Query Assurance and Processing using TBI

The procedure of user query, query result and assurance processes are shown in the following figure 3:

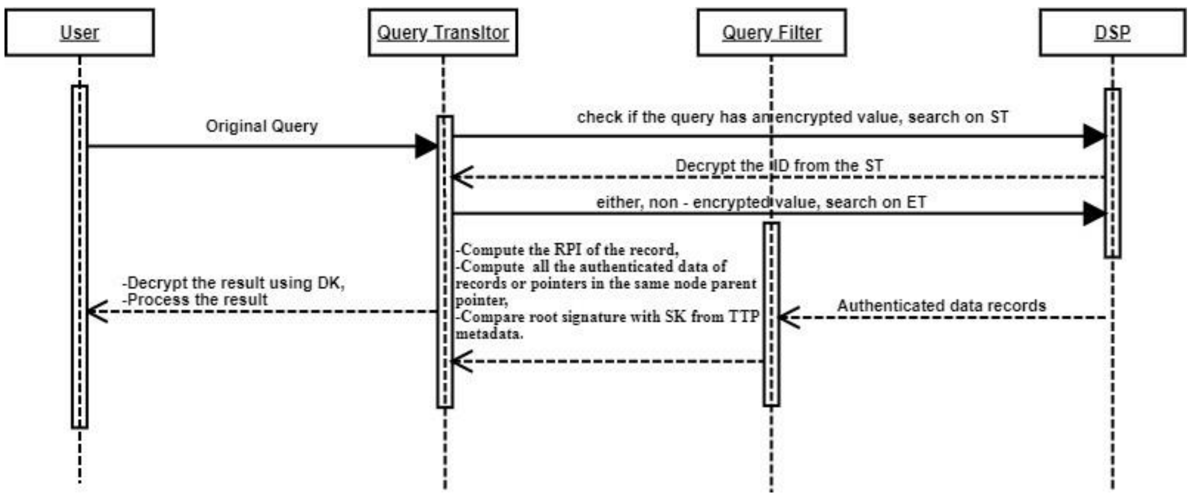

User
Query Translitor
Query Filter
DSP
Original Query
check if the query has an encrypted value, search on ST
Decrypt the ID from the ST
either, non - encrypted value, search on ET
-Compute the RPI of the record,
-Compute all the authenticated data of records or pointers in the same node parent pointer,
-Compare root signature with SK from TTP metadata.
-Decrypt the result using DK,
-Process the result
Authenticated data records

Figure 3: Query assurance and processing using TBI

The original query before process to DSP, process first to the TTP, which has two parts: Query Translator and Query Filter. The TTP processes the query to the DSP to get the result and check the result confidentiality and integrity.

a- <u>Query Translator:</u>

1. Checks if the original query has an encrypted value, TTP will process the query on the ST instead of the ET at DSP.

2. When the requested values are found, the DSP sends the values and record_ids to query translator, which decrypts the record_id for each value using DK, and uses the decrypted ids, to get the records from DSP and process it to query filter, to check the integrity and decrypt the result.
3. After the query is verified by the query filter, the result is decrypted using DK, and process it to the user.

b- Query filter:

1. Receives the data records, which contains the encrypted result and the TBI from DSP.
2. Compute the TBI of its parent pointer, using the TBI of record.
3. To ensure the data correctness and data completeness, all the authenticated data of records or pointers in the same node, which have the same parent TBI can be computed by using any of their TBI. Then, Compute the root hash and then the root signature, by using the SK from the Metadata Table.
4. Compare between the new computed root signature and the SK, if both are identical the data freshness is ensured, and then the root signature is decrypted using PK from the Metadata table.

4. Data Operations on the Outsourced Data using TBI

This section presents the details of handling queries, including static operations such as select, and dynamic operations such as insert, update and delete.

4.1 Insert Operation

Inserting a new record to a table is complicated process, as to insert a record; this may cause a change to the MBT structure, while inserting new leaf node may cause splitting the parent nodes. In this paper we will handle inserting a new leaf node without split the parent node as shown in figure 4.
There is available TBI for the new record between sibling nodes in MBT.

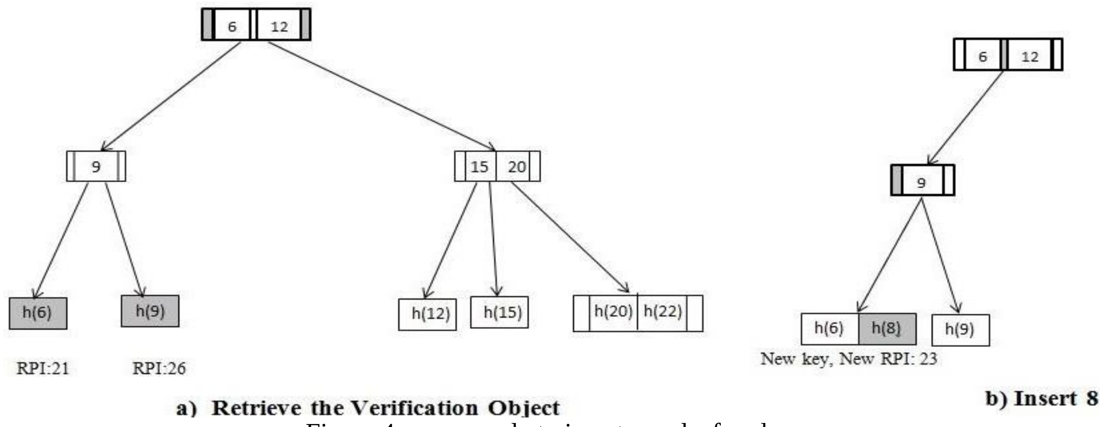


Figure 4: an example to insert new leaf nodes

The insertion process doesn't include only the new data, but also the new authentication data records. We have implement query rewrite algorithms for trusted third party to generate the insert SQL

statements to insert the authentication data, which will be sent within the insert query by the user to DSP. We assume that the new record always contains sensitive data.
This process involves 3 network round trips between user, TTP and DSP.

- The TTP encrypts any sensitive attribute value for the new record, from the insertion query by user.
- Retrieve the verification object from the DSP, which contains pointers and hashes within the tree range if left or right according to the insertion value.
- In one transaction at the DSP, the SQL statements have been sent for Insert the new record which contains, insert the new record in ST and ET, the record is hashed, compute the TBI and update the authenticated data records.

Based on the example in figure 4, the following example shows the generated update and insert SQL statements that is used.

Example 1:
Insert a new record to employee table, assume this record id is 8
INSERT into employee values ('Mai', engineer, 3000, 3)

Insert node 8, and the update queries for authentication data on DSP
****insert the new record**
Insert into [Employee_ST] ([ID],[Salary]) values('DteXTR0NG8z9cxqrUMfP+Q==',3000);
Update [Employee_ET] set [Name]='Mai', [Job]='engineer', [Salary]='lkeo9ijeijxqrUMfP+Q==o', [Department_No] = 3; [TBId] = 23, [hash]= 'b569052rik' where [id]=8;

****update the parent and the root node hash values**
Update [emp_1] set [hash] = 'y6Mg68L32gcxtMC+XOlV4w*=' where [TBId] = 3;
Update [emp_0] set [hash] = "+XOlV4w*='5aKaSct3o' where [TBId] =0;
Update the metadata table on TTP
Update [Metadata] set [SK] = '"+XOlV4w*='5aKaSct3o', [Version_NO] +=1;

4.2. Select operation:

Assume that the selection value is always encrypted, the selection process involves 4 network round TBIs between user, TTP and DSP.

- Retrieve the encrypted record ids that relevant to the selection value from the ST table to TTP.
- The TTP decrypts the record ids, and searches with the decrypted record ids from DSP.
- Retrieve the selected records that are relevant to the selection value within the verification object to the TTP.
- The TTP checks the data integrity, and sends the results to the user.

To retrieve the authenticated data records to the TTP, we have implement query rewrite algorithms for trusted third party to generate the select SQL statements to select the authentication data, relevant to the selected record, as shown in example 2.

Example 2:

Select a record from employee table
*SELECT * FROM employee WHERE id=24*

****To find the tbid of the data record with id 24**
```
Declare @recordTBId AS int;
Select top1 Value @recordTBId =id, From [ Employee_ET] Where id=24;
```

****Retrieve the verification object for for all the tree levels**
```
Select [TBId], [hash] From [Employee_ET] Where TBId<=(@ recordTBId /3)*3 And TBId<(@ recordTBId /3)*3 +3;
Select [TBId], [hash] From [emp_1] Where TBId<=(@ recordTBId/3*3)*3 And TBId<(@ recordTBId/3*3)*3+3;
Select [TBId], [hash] From [emp_0] Where TBId<=(@ recordTBId/ 3*3*3)*3 And TBId<(@ recordTBId/3*3*3)*3+3;
```

4.3 Update Operation

Updating a record in a table doesn't change anything in the MBT structure, but requires an update on the authentication data. So, when we update a record from ET table, ST Table, Emp_0, and Emp_1 tables will be updated also. We have implement query rewrite algorithm for trusted third party to generate the update SQL statements to update the authentication data, which will be sent within the update query by the user to DSP, but first we retrieve the verification object of the data record to be updated from the DSP. We assume that the new record always contains sensitive data.
This process involves 3 network round trips between user, TTP and DSP.

- The TTP encrypts any sensitive attribute value for the record, from the modification query by user.
- Retrieve the verification object from the DSP, which contains pointers and hashes within the tree range if left or right according to the modification value from the update query.
- In one transaction at the DSP, the SQL statements have been sent for update the existing record which contains, update the record in ST and ET tables, the new record hash, the root signature.

An example to the record modification query is as shown in example 3.

Example 3:
Update single encrypted value based on selection value
UPDATE employee.Salary = 2500 WHERE id=9

The update queries for data and authentication data
```
Update [Employee_ST] set [Salary]= (2500) where [id]= 'DteXkmklm;lmxqrUMfP+Q==';
Update [Employee_ET] set [hash]= 'b569052,m;k;iujjirik' where [TBId]=26;
Update [emp_1] set [hash]= 'y6Mg68L32gcxtMC+XOlV4w*' where [TBId]= 3;
Update [emp_0] set [hash] = 'XOlV4dctgaSct3o' where [TBId] =0;
```

Update the metadata table on TTP
```
Update [Metadata] set [SK] = 'XOlV4dctgaSct3o', [Version_NO] +=1;
```

4.4 Delete Operation

Record deletion process is complicated process as record insertion, which may cause a change to the MBT structure, as merging nodes together. But, we will handle deleting a leaf node a leaf node without causes of anything to the parent as shown in figure 5.

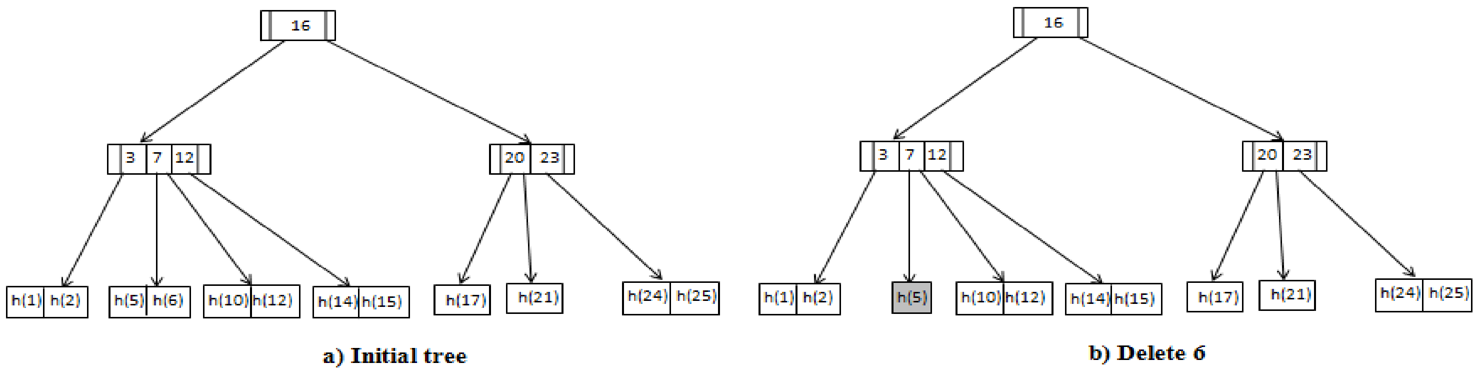


Figure 5: delete leaf node

We have implement query rewrite algorithms for trusted third party to generate the delete SQL statements to delete the authentication data, which will be sent within the delete query by the user to DSP.

This process involves 2 network round trips between user, TTP and DSP.

- Retrieve the encrypted id from the DSP, which is relevant to the record to be deleted. Moreover, the verification object from the DSP, which contains pointers and hashes within the tree range if left or right according to the deletion value.
- In one transaction at the DSP, the SQL statements have been sent for delete the record which contains, delete the record in ST and ET, compute the TBI and update the authenticated data records.

Based on the examples in figure 5, the following example shows the generated update and delete SQL statements that is used.

Example 4:

Delete a record from employee table, assume this record id is 6
DELETE from employee WHERE id=6

Delete node 6, and the update queries for authentication data on DSP

****Delete the record**

Delete From [Employee_ET] where [id]=6;
Delete From [Employee_ST] where [id]=DteXTR0NG8z9cxqrUMfP+Q;

****Update the parent and the root node hash values**

Update [emp_1] set [hash] = 'y6Mg68L32gcxtMC+XOlV4w*=' where [TBId] = 3;
Update [emp_0] set [hash] = 'jkunll,nuuiooolekmj', where [TBId] =0;

Update the metadata table on TTP

```
Update [Metadata] set [SK] = ‘jkunll,nuuiooolekmj’, [Version_NO] +=1;
```

5. Experimental Evaluation

Scheme implementation: Our implementation of the proposed scheme consists of four modules: Owner module: runs on the owner side, DSP module: runs on Microsoft Azure cloud platforms, user module: runs on the authorized users’ side, and TTP module: runs on an extended trusted server. We have implemented the Merkle B-tree to be stored within encrypted database in DSP, as each level of the tree is stored as table. We also implement the query rewrite algorithms for users in TTP server, which is the core of generating select, insert, update, and delete SQL statements based on a data table in a DB according to the users’ query. Our scheme retrieves within the query result, and the authentication data for integrity checking. Our implementation is based on C# and SQL Server 2012, by using Microsoft Visual Studio 2013. For efficiency analysis, we have compare our result with the MBT-Serialization [8].

Experiment setup: We use a database that consists of two tables to test our scheme, employee table and department table. The employee table contains 10,000 records, the salary column is encrypted, and this table contains the hash value of each record and the MBT-Base Identifier and there is a search table for it contains the actual employee salary that kept non-encrypted, while the id is encrypted. The record id is used as a key in the MBT, which is a pointer to record hash value. There are two extended tables to DB, contains the parent and the root levels of the tree that used to check the query result integrity and confidentiality. We upload the database and the authentication data to an Azure cloud service provider, which deploys the SQL Server 2012 as a database service, and run experiments from the owner and the user using a laptop with Intel(R) Core(TM) 2.2GHz processor and 6GB RAM running Windows 10 through a home network with 75 Mbps download and 20 Mbps upload. Upload the TTP metadata to small database by using SQL server 2012. We test our scheme using different queries as: insert, select, update, and delete to evaluate the performance overhead of query result confidentiality and integrity verification and the efficiency of the proposed mechanisms.

Performance analysis: We evaluate the performance overhead against the MBT-Serialization scheme [8], as measuring the average latencies of query result according to different operations: insert, select, update, and delete with confidentiality and integrity verification support.
In our scheme the query latency is dominated by the computation cost in the DSP side and the TTP side, as the computation cost in the DSP side is the execution of encrypted queries and authentication data retrieval, and in TTP side is rewriting query, verifying data integrity and decrypt the result.
In MBT-Serialization scheme, also, the query latencies are dominated by the computation cost in the DSP side and the TTP side, as the computation cost in the DSP side is the execution of encrypted queries, and in TTP side is de-serialize the tree, rewriting query, verifying data integrity,
The communication cost in both schemes includes the execution of additional queries to return the result.

Data operations: We run experiments to explore how the latencies overhead changes when the number of records to be retrieved is increased.

Insert: we run an experiment to evaluate the performance of insert a new record in a table with integrity protection. We assume that always the inserted record is at the end of the table. In our work, the experiment was executed for insert query and to insert single record into employee table, we generate insertion queries for both data and authentication data and send them to the DSP to be executed. Suppose the insertion process doesn't affect the nodes in the MBT. The average overhead of our scheme for insert a new record is 1.4 sec, while MBT-Serialization scheme is 1.6 sec. The major performance overhead comes from the number of update statements to be executed in the TTP side.

Select: we run experiments to evaluate the performance of select queries from a table with integrity protection by two different select cases: select a single record or multiple records according to specific value within one query. Assume that the selection value is always encrypted. Figure 6 shows that the latencies overhead increases when the number of records increases. Overall, the overhead in our scheme is less than in MBT-Serialization scheme.

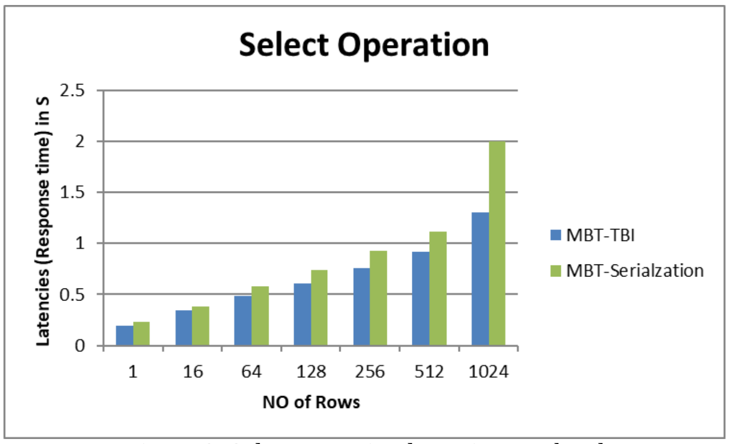


Figure 6: Select operation latencies overhead

Update: we evaluate the performance overhead caused by update queries. The data to be updated is first retrieved to verify its data integrity and then, we generate update queries for both data and authentication data and send them to the DSP to be executed. Figure 7 shows the overhead against the number of rows to be updated. The results show that when we update a few records, the overhead is high in both schemes, because of the additional round trip to verify the data integrity, but when the numbers of rows to be updated are increase, the overhead of update decreases. In both of schemes, if the number of record increases, the overhead decrease, as in the MBT update multiple records is much better than update one record as updating the leaf nodes and the inner nodes just once. The result as shown in Figure 7 shows that the overhead in our scheme is less than in MBT-Serialization scheme.

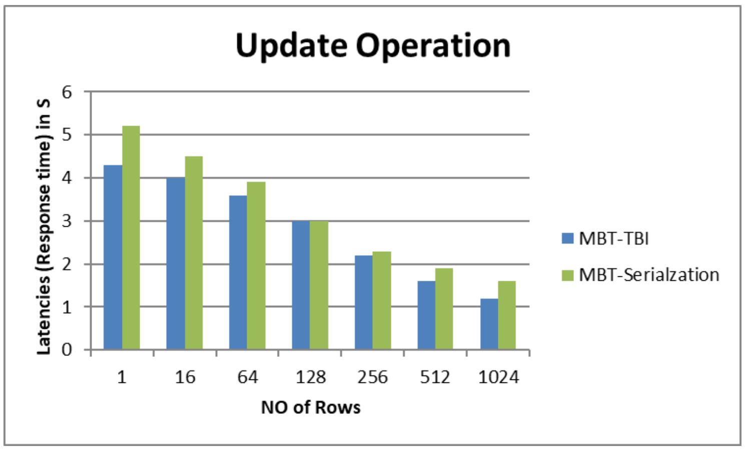


Figure 7: Update operation latencies overhead

Delete: we evaluate the performance overhead caused by delete queries by two different delete cases: delete a single record or multiple records from a table with integrity protection. The data to be deleted is first retrieved to verify its data integrity and then, we generate delete and update queries for both data and authentication data and send them to the DSP to be executed. In both of schemes, if the number of record to be deleted are increases, the overhead increases, because of the update that is done on the MBT. But in overall the average overhead of our scheme for delete query is less than in a MBT-Serialization scheme as shown in figure 8.

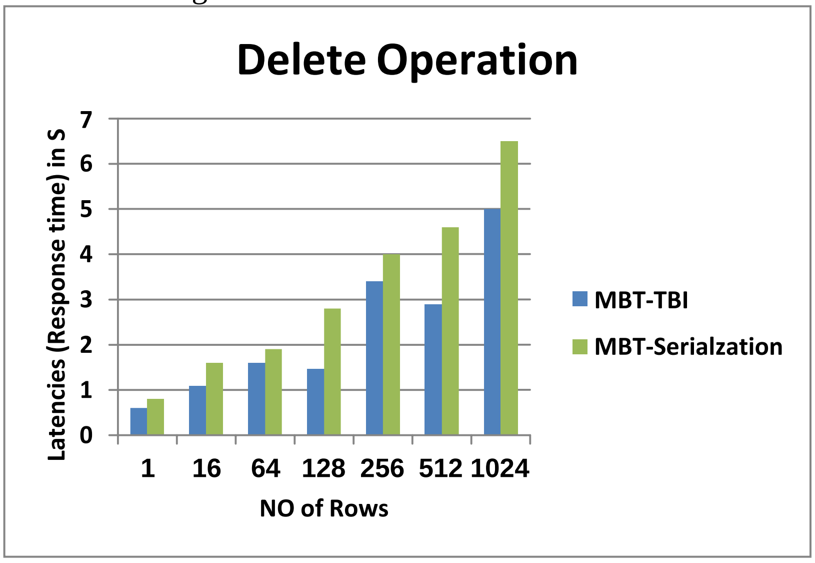


Figure 8: Delete operation latencies overhead

6. Conclusion

In our paper, we design and implement an efficient scheme, using trusted third party (TTP), which enables the mutual trust to store a database in the database service provider (DSP). Our scheme shows that it guarantees the most security concerns, data confidentiality and query result integrity. We run experiments to explore the efficiency of different operations, such as Insert, Update, Delete, and Select, to retrieve the result from DSP. The performance analysis shows that our scheme succeeded to achieve the security goals with a small overhead for the different operations.

References

1. Rana M Pir, Data integrity verification in cloud storage without using trusted third party auditor, In Proc. of IJEDR, 2014.
2. E.Shmueli,R.Vaisenberg, E.Gudes, and Y.Elovici, Implementing a database encryption solution, design and implementation is-sues, In Proc. of Elsevier, 2014.
3. Advanced Encryption Standard, NIST. FIPS PUB 197. (2001)
4. K.Raghuvanshi, P.Khurana and P.Bindal, Study and Comparative Analysis of Different Hash Algorithm, Journal of Engineering Computers and Applied Sciences (JECAS), VOL. 3, 2014.
5. F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin, Dynamic Authenticated Index Structures for Outsourced Databases, In Proc. of ACM Management of Data (SIGMOD), USA, 2006.
6. S. Singh, S.K. Maakar and Dr. S. Kumar, A Performance Analysis of DES and RSA Cryptography, International Journal of Emerging Trends and Technology in Computer Science (IJETTCS), 2013.
7. W. Wei and T. Yu, Integrity Assurance for Outsourced Databases without DBMS Modification, International Federation for Information Processing, 2014.
8. E.Shmueli ,R.Vaisenberg , E.Gudes, and Y.Elovici, Implementing a database encryption solution, design and implementation issues, Elsevier, 2014.
9. S. Vimercati, S. Foresti, G. Livraga, S. Paraboschi and P. Samarati, Confidentiality Protection in Large Databases, Springer, 2018.
10. W. I. Khedr, H. M. Khater and E. R. Mohamed, Cryptographic Accumulator-Based Scheme for Critical Data Integrity Verification in Cloud Storage, IEEE Access, vol. 7, pp. 65635-65651, 2019.
11. A. P. Mohan, M. Asfak and A. Gladston, Merkle Tree and Block Chain Based Cloud Data Auditing, International Journal of Cloud Applications and Computing, Vol. 10, 2020.
12. M. Rady, T. Abdelkader, and R. Ismail, SCIQ-CD: A Secure Scheme to Provide Confidentiality and Integrity of Query results for Cloud Databases, 14th International Computer Engineering Conference (ICENCO), December 2018.
13. Y. An, Z. Zaaba and N. Samsudin, Reviews on Security Issues and Challenges in Cloud Computing, International Engineering Research and Innovation Symposium (IRIS), 2016.