An Efficient Approach for Secure Data Outsourcing using Hybrid Data Partitioning

Sultan Badran
College of Graduate Studies
Palestine Polytechnic University (PPU)
Hebron, Palestaine
176030@ppu.edu.ps

Nabil Arman
Dept. of Computer Science and IT
Palestine Polytechnic University (PPU)
Hebron, Palestaine
narman@ppu.edu

Mousa Farajallah
Dept. of Computer Engineering
Palestine Polytechnic University (PPU)
Hebron, Palestaine
mousa_math@ppu.edu

Abstract— This paper presents an implementation of a novel approach, utilizing hybrid data partitioning, to secure sensitive data and improve query performance. In this novel approach, vertical and horizontal data partitioning are combined together in an approach that called hybrid partitioning and the new approach is implemented using Microsoft SQL server to generate divided/partitioned relations. A group of proposed rules is applied to the query request process using query binning (QB) and Metadata of partitioning. The proposed approach is validated using experiments involving a collection of data evaluated by outcomes of advanced stored procedures. The suggested approach results are satisfactory in achieving the properties of defining the data security: non-linkability and indistinguishability. The results of the proposed approach were satisfactory. The proposed novel approach outperforms a well-known approach called PANDA.

Keywords— hybrid data partitioning implementation, data outsourcing, data sensitivity, cloud.

I. Introduction

Generally, data outsourcing is vulnerable to different types of attacks. Secure and efficient retrieval of outsourced data is still an open challenge. Data sensitivity is one of the most critical security issues that need to be investigated. Generally, the data owner avoids data outsourcing (sensitive and non-sensitive data). Alternatively, they outsource all data in encrypted form to protect sensitive data. In this research, the data partitioning techniques, based on data sensitivity, is considered to secure data outsourcing.

Accordingly, hybrid data partitioning techniques are addressed [1]. However, to improve data security against attacks, data partitioning techniques (PANDA) were proposed in [2], [4], and [5]. These techniques divide a relation into a set of relations based on data sensitivity. While good partitioning techniques prevent data leakage against inference attacks, these techniques must have specific characteristics to be considered secure, such as non-linkability and indistinguishability.

This paper aims to improve query execution time, secure the sensitive data against inference attacks, and prevent data leakage while outsourcing data to a public database for storage by encrypting less amount of data than PANDA. A hybrid data partitioning technique is used to divide relations into vertical and horizontal relations based on data sensitivity. Query binning (QB) techniques [3] are applied for securing sensitive data and enhancing the performance of query execution time.

The remaining sections of this paper are organized as follows: The hybrid data partitioning technique is presented in section II. Section III shows experiments demonstration and evaluation. And in section IV conclude the paper.

II. Hybrid Data Partitioning Technique

A. Hybrid Data Partitioning Model

In this paper, consider that the following two entities in the proposed model:

1) Trusted Database on-premises contains the whole data in plaintext format and executes queries and sends query requests to untrusted DB on the cloud. assume that a relation R has attributes, say $A_1, A_2, A_3 \ldots, A_n$, containing all sensitive and non-sensitive values in tuples $t_1, t_2, t_3 \ldots, t_m$. According to values stored with specific attributes, the DB owner determines which attributes are sensitive or not and lays rules that determine when the tuple is sensitive.

The database (DB) owner divides relation R into several relations based on the tuples' data sensitivity using a hybrid technique that divides the tuple into three tuples at max; each divided tuple is stored in a different relation. The first one contains the total values stored in the attributes marked as sensitive. The second contains the total values stored in the attributes marked as non-Sensitive, and the rest of the values may either include sensitive or non-sensitive values. This means that the third tuple may either be considered sensitive or non-sensitive. The DB owner outsources the relations that contain non-sensitive data to a public cloud in plaintext form. The tuples of the relations that contain sensitive data are encrypted using any existing non-deterministic encryption mechanism before outsourcing to the same public cloud.

In the proposed model setting, the DB owner must store metadata such as a mapping relation that stores the original tuple ID with the new tuple IDs in each of the divided relations. The Metadata will be used for appropriate query formulation using the Query Binning (QB) proposed in [3] and explained in section B.

2) Untrusted Database on Cloud that hosts the databases contains the partitioned relations, executes queries, and provides answers to the trusted DB stored on-premises.

To explain query execution in the proposed model, let assume a query σ over the relation R and p is a preposition, denoted by $\sigma_p(R)$. The query is executed on trusted DB with no limitation on number of attributes in WHERE condition clause. The results of the query include four attributes:

- Tuple ID: the original ID for each tuple.
- ID_E: tuple ID representing the primary tuple key in new relation for sensitive data (R_E).
- ID_P: tuple ID representing the primary tuple key in new relation for non-sensitive data (R_P).
- ID_{P_E}: tuple ID representing the primary key of relation R_{P_E} in plaintext or relation R_{E_P} in encrypted form.

After that, the query process splits the execution of σp(R) into four subqueries. As shown in equation 1, each subquery is sent to an untrusted DB to be executed. Then the results of subqueries are returned to the Trusted DB. Then inside the trusted DB, there will be two subqueries ((R_{P_E} and R_{E_P})) that have the same scheme, for which a union operation is performed. Then join operations are performed to join the union result with R_P and R_E. In particular, the query σ on a relation R is executed, as:

$$\sigma_p(R) = \sigma_p(R_E) \bowtie \sigma_p(R_P) \bowtie (\sigma_p(R_{P\_E}) \cup \sigma_p(R_{E\_P})) \qquad (1)$$

Example: Let us illustrate the hybrid data-partitioning model using the relation presented in Table 1.

TABLE 1 EMPLOYEE RELATION

	Attributes No					
	a_1	a_2	a_3	a_4	a_5	a_6
	ID	Name	Department	Salary	Location	Password
t_1	1	Ali	IT	1,000	Jerusalem	*******
t_2	2	Intisar	Marketing	900	Jerusalem	*******
t_3	3	Mahmoud	IT	1,200	Hebron	*******
t_4	4	Susan	Marketing	1,500	Ramallah	*******
t_5	5	Sultan	Marketing	1,450	Bethlehem	*******
t_6	6	Kazem	HR	1,050	Nablus	*******
t_7	7	Alaa	Marketing	1,460	Bethlehem	*******
t_8	8	Ahmad	HR	980	Nablus	*******

Table 1 considers an Employee relation R. Note that the notation a_i ($1 \le i \le 6$) is an attribute in the relation; it indicates the ith attribute. In this relation, note that the notation t_j ($1 \le j \le 8$) of the relation is used this to indicate the jth tuple. In this relation, the DB owner considers that the password attribute values are not outsourced data, and the salary attribute values are sensitive. Moreover, all values in department attribute that meet Department = "Marketing" are sensitive. In such a case, and after applying the Hybrid partitioning, the metadata are generated as shown in Table 2. Metadata includes four attributes as described. It is worth mentioning that ID_E, ID_P, and ID_{P_E} attributes are unique identifiers of 50 characters.

TABLE 2 METADATA TABLE FOR RELATION R

Tuple No	Tuple ID	ID_E	ID_P	ID_{P_E}
t_1	1	848CC055...A	43AACEF7...P	F0D9C43C...R
t_2	2	DF8BC1A8...C	2CF79E45...O	485F36AB...J
t_3	3	03E47A30...E	1AC4E44F...Y	CAF5A05C...Q
t_4	4	5E1A2955...A	990D4BF7...I	17EDA383...8
t_5	5	EF036F92...F	BA921C43...G	F1859688...Y
t_6	6	CB1CCD4D...K	4276A931...K	A03E7373...D
t_7	7	116DB16E...H	10E7C843...U	14C0E88B...X
t_8	8	F2220062...P	892285C5...D	05B4FA48...Z

The Employee relation may be stored on the cloud as:

1) Relation 1, which contains all sensitive values in Salary's attribute and stores values in encrypted form, as shown in TABLE 3.

TABLE 3 RELATION 1

Attributes No	ID_E	a_4
Tuple No	ID	Salary
t1	848CC055...A	E(1000)
t2	DF8BC1A8...C	E(900)
t3	03E47A30...E	E(1200)
t4	5E1A2955...A	E(1500)
t5	EF036F92...F	E(1450)
t6	CB1CCD4D...K	E(1050)
t7	116DB16E...H	E(1460)
t8	F2220062...P	E(980)

2) Relation 2, which contains all non-sensitive values in all attributes marked as ,non-sensitive attributes and store values in plaintext form, as shown in Table 4.

TABLE 4 RELATION 2

Attributes No	ID_P	a_2	a_5
Tuple No	ID	Name	Location
t1	43AACEF7...P	Ali	Jerusalem
t2	2CF79E45...O	Intisar	Jerusalem
t3	1AC4E44F...Y	Mahmoud	Hebron
t4	990D4BF7...I	Susan	Ramallah
t5	BA921C43...G	Sultan	Bethlehem
t6	4276A931...K	Kazem	Nablus
t7	10E7C843...U	Alaa	Bethlehem
t8	892285C5...D	Ahmad	Nablus

3) Relation 3, which contains all tuples that the attributes include sensitive values. In the example, all sensitive values in Department attribute, where Department = Marketing", are stored encrypted as shown in Table 5.

TABLE 5 RELATION 3

Attributes No	ID_{P_E}	a_3
Tuple No	ID	Department
t_2	CAF5A05C...Q	E(Marketing)
t_4	17EDA383...8	E(Marketing)
t_5	A03E7373...D	E(Marketing)
t_7	05B4FA48...Z	E(Marketing)

4) Relation 4, which contains all sensitive values in Name and Location, where Department= "Marketing" and saved as plaintext as shown in Table 6.

TABLE 6 RELATION 4

Attributes No	ID_{P_E}	a_3
Tuple No	ID	Name
t_1	F0D9C43C...R	IT
t_3	485F36AB...J	IT
t_6	F1859688...Y	HR
t_8	14C0E88B...X	HR

Hence, the sensitive data stored in Relation 1 and Relation 3 are encrypted before being outsourced to an untrusted database. In contrast, Relation 2 and Relation 4, including only non-sensitive data, are outsourced in plaintext form. The partitioning is executed on the tuple level, which means every time a tuple insertion, modification, or deletion operation occurred, a trigger is fired and run the partitioning code as presented in Algorithm 1.

```
Algorithm 1 Insert Tuple
      Inputs: t: inserted/updated tuple.
      Variable: Metadata: table to store metadata about t. a[] list of
      attributes. v[] sensitive values list for each attributes. $ID_E$, $ID_P$,
      $ID_{P\_E}$
1     Function Insert Tuple ( t ) begin
2         a[]←Relation attributes v[]←Relation attributes Sensitive
          Values
3         $ID_E$ ← Generate Unique Identifier key
4         $t_E$ ← $ID_E$
5         $t_E$ ← Encrypt all values store in attributes marked as
          sensitive in t
6         Send $t_E$ to $R_E$ in cloud
7         $ID_p$ ← Generate Unique Identifier key
8         $t_P$ ← $ID_P$
9         $t_P$ ← all values store in attributes marked as non-sensitive
                  in t
10        Send $t_P$ to $R_P$ in cloud
11        $ID_{Temp}$ ← Generate Unique Identifier key
12        If the rest of the values in t marked as sensitive values
```

```
13          $t_{E\_P} \leftarrow ID_{Temp}$
14          $t_{E\_P} \leftarrow$ Encrypt all values marked as sensitive in t
15          Send $t_{E\_P}$ to $R_{E\_P}$ in the cloud
16          Else
17          $t_{P\_E} \leftarrow ID_{Temp}$
18          $T_{P\_E} \leftarrow$ all values marked as non-sensitive in t
19          Send $t_{P\_E}$ to $R_{P\_E}$ in the cloud
20      Metadata $\leftarrow$ t.ID, $ID_P$, $ID_E$, $ID_{Temp}$
21      Return
```

To continue with example 1, consider a query σ: SELECT Name, Department from Employee where Location = N'Jerusalem'. In the trusted DB, the query $\sigma_{\text{Location} = \text{N'Jerusalem}}(R)$ is executed on relation R, then as shown in Algorithm 2, the results of the query are joined with metadata relation, after that, they produce four queries that are sent and executed in Untrusted DB:

- $\sigma_{\text{IDe in (query results)}}(R_E)$ executes on R_E relation.
- $\sigma_{\text{IDp in (query results)}}(R_P)$ executes on R_P Relation.
- $\sigma_{\text{IDp\_e in (query results)}}(R_{P\_E})$ executes on R_{P_E} Relation.
- And the last query $\sigma_{\text{IDe\_p in (query results)}}(R_{E\_P})$ executes on R_{E_P} Relation.

The queries' results are sent back to trusted DB, and SQL operation is performed as shown in 1. Algorithm 2 shows how the query request process works. It is worth mentioning that the partitioning computation occurs during the insertion of tuples into R relation. That saves time instead of doing the partitioning of the whole data at once.

```
Algorithm 2 Query Request
    Inputs: SQLstr: Select query statement, Metadata: table to store
    metadata about tuples
    Outputs: Results: Query results
    Variable: T_R: temporary data table to store metadata about SQL
              results, Result1 temporary relation
1   Function run_SQL ( SQLstr ) begin
2       T_R ← Execute( SQLstr ) ⋈ Metadata
3       Result1 ←Execute_on_Cloud($R_{E\_P}$, Domain(T_R.$ID_{E\_P}$))U
                  Execute_on_Cloud($R_{P\_E}$,
                  Domain(T_R.$ID_{P\_E}$))
4       Result1 ← Result1 ⋈ Execute_on_Cloud($R_P$,
        Domain(T_R.$ID_P$))
5       Result1 ← Result1 ⋈
        Execute_on_Cloud($R_E$,Domain(T_R.$ID_E$))
6       Query results← retrieve tuple from Result1 match the original
                  where clause
7       Return Query results
```

B. *Query Binning Technique*

QB involves two steps: first, the creation of the query bins. The second step consists of rewriting the query based on the binning. Could be say that the QB in the base case is a one-to-one relationship between one sensitive tuple and one non-sensitive tuple. Accordingly, this means that both tuples cannot be sensitive or non-sensitive. Before describing QB, present the concept of approximate square factors of a number used to create the bins is needed. As defined in [3], "two numbers, say x and y, are approximately square factors of number n, where n > 0, if x × y = n, and x and y are equal or close to each other. So that the difference between x and y is less than the difference between any two factors, say x′ and y′, of n such that x′ × y′ = n". In this research, the QB uses tuples stored in partitions divided horizontally to create the binning. Continuing with the example in section A, to calculate the approximately square factors, let us consider that n = number of non-sensitive tuple = 4 tuples, according to the definition of Approximately square factors, x = 2 and y = 2, this satisfies the definition of the Approximately square factors. Now two sensitive bins and two non-sensitive bins are created. After creating the bins, filling them with tuples using the algorithm described in [3]. That links between sensitive tuples and non-sensitive tuples. The results of this operation are shown in Fig. 1. In the example shown below, the location attribute in WHERE clause are used and the same data in example1 (Table 1), and the tuples retrieved as follows:

- Retrieve tuples corresponding to employees who work in Location ='Jerusalem'.
- Retrieve tuples corresponding to employees who work in Location ='Hebron',
- And retrieve tuples corresponding to employees who work in Location ='Bethlehem'.

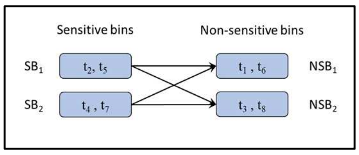

Fig. 1 QB for four sensitive and four non-sensitive tuples.

Adversarial view

Assume that the adversary has access to Untrusted DB and to the transactions log file, which means that when answering a query, the adversary knows the retrieved encrypted tuples and the complete information of the retrieved non-sensitive tuples. This information is known to the adversary as the adversarial view, shown in Table 7. This table contains the retrieved tuples without applying the QB.

TABLE 7 QUERIES RESULTS, WITHOUT APPLY QB

Query value	Returned tuples/Adversarial view			
	Relation 1	Relation 2	Relation 3	Relation 4
Jerusalem	$E(t_1)$, $E(t_2)$	t_2	$E(t_2)$	t_1
Hebron	$E(t_3)$	t_3	Null	t_3
Bethlehem	$E(t_5)$, $E(t_7)$	t_5, t_7	$E(t_5)$, $E(t_7)$	null

To apply the QB bins technique, need to modify the query request performed, so Algorithm 3 shows how to query request work with QB. To understand the QB effects on the query request process, the adversarial view will be changed after Algorithm 3 is applied;

```
Algorithm 3 Query Request with QB
    Inputs: SQLstr: Select query statement, Metadata: table to store
    metadata about tuples
    Outputs: Results: Query results
    Variable:T_R_B: temporary data table with Bins, T_R_B: temporary
             data table without Bins, T_R: temporary data table to
             store metadata about SQL results, Result1 temporary
             relation
1   Function run_SQL ( SQLstr ) begin
2       T_R_W ← Execute( SQLstr )
        T_R_B ← Retrieve_Bins( T_R_W)
        T_R ←T_R_B ⋈ Metadata
3       Result1 ←Execute_on_Cloud($R_{E\_P}$, Domain(T_R.$ID_{E\_P}$))U
                  Execute_on_Cloud($R_{P\_E}$,
                  Domain(T_R.$ID_{P\_E}$))
4       Result1 ← Result1 ⋈ Execute_on_Cloud($R_P$,
        Domain(T_R.$ID_P$))
5       Result1 ← Result1 ⋈ Execute_on_Cloud ($R_E$,
        Domain(T_R.$ID_E$))
6       Query results← retrieve tuple from Result1 match the original
                  where clause
7       Return Query results
```

Table 8 shows the query request result for an adversary using the QB technique. In this example, will use the same conditions in the previous example after applying the QB technique.

TABLE 8 QUERY RESULT USING QB

Query value	Returned tuples/Adversarial view			
	Relation 1	Relation 2	Relation 3	Relation 4
Jerusalem	$E(t_1)$,$E(t_2)$, $E(t_5)$, $E(t_6)$	t_1,t_2,t_5,t_6	$E(t_2)$, $E(t_5$)	t_1,t_6
Hebron	$E(t_2)$,$E(t_3)$, $E(t_5)$, $E(t_8)$	t_2, t_3,t_5,t_8	$E(t_2)$, $E(t_5$)	t_3,t_8
Bethlehem	$E(t_2)$,$E(t_4)$, $E(t_5)$, $E(t_7)$, $E(t_3)$, $E(t_8)$	t_2, t_3, t_4, t_5, t_7, t_8	$E(t_2)$,$E(t_4)$, $E(t_5)$, $E(t_7$)	t_3, t_8

C. Data Partitioning Security

Using a non-deterministic encryption for sensitive data achieves the property of cipher-text indistinguishability (i.e., an adversary cannot distinguish between two cipher-texts) [3]. Hence, the same plaintext values have two different cipher-text values. Furthermore, the non-linkability will be achieved in two positions, first in the public database by using ID for each tuple stored in each divided relation different from the original ID in the private database. Second, in the query request process, this is achieved by using query binning (QB). Fig. 2 illustrates the security context.

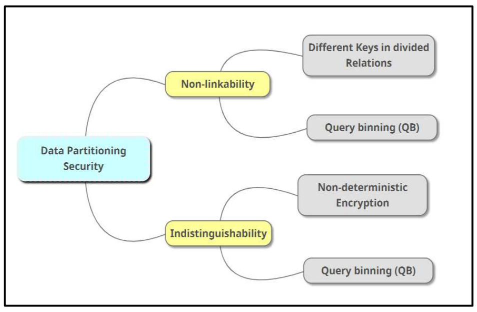
Data Partitioning Security
Non-linkability
Different Keys in divided Relations
Query binning (QB)
Indistinguishability
Non-deterministic Encryption
Query binning (QB)

Fig. 2 Map mind

Adversarial view: the authors want to explain the adversarial view that assumes that the adversary has full access to Untrusted DB and the transactions log file. This means that when answering a query, the adversary can retrieve the all-Select SQL statements, and re-execute these statements and retrieve the encrypted tuples and the complete information of the retrieved non-sensitive tuples. The adversarial view lets the adversary knows this information. Besides, the adversary has no access to Trusted DB.

Based on the adversarial view, proof of data security is needed. For that, should be first explain the notion of partitioned data security used in PANDA [3] that is established when a partitioned computation over sensitive and non-sensitive data does not leak any sensitive information. Note that an adversary may infer sensitive information using the adversarial view that was created during query processing, knowledge of frequency counts, and workload characteristics. In PANDA, they begin by clarifying the concepts of associated values, associated tuples, and the relationship between counts of sensitive values.

The definitions used are the same notation used in [3] with additional notation added to prevent data leakage after hybrid partitioning:

1) t_1, t_2. . . ,t_m are tuples of a sensitive relation, say R_{E_P}. Thus, the relation R_{E_P} stores the encrypted tuples $E(t_1)$, $E(t_2)$, . . . , $E(t_m)$.
2) s_1, s_2, . . . , $s_{m'}$ are values of an attribute, say A, that appears in one of the sensitive tuples of R_{E_P} . Note that $m' \leq m$, since a number of tuples may have an identical value. Additionally, $s_i \in Domain(A)$,$i = 1, 2, . . . ,m'$.
3) $|s (s_i)|$, refer to the number of sensitive tuples with s_i as the value for attribute A. They further define $|s (v)| = 0$, $\forall v \in Domain(A)$, $v < s_1, s_2, . . . , s_{m'}$.
4) t_1, t_2, . . . , t_n are tuples of a non-sensitive relation, say R_{P_E}.
5) ns_1, ns_2, . . . , $ns_{n'}$ are values of the attribute A that appears in one of the non-sensitive tuples of R_{P_E}. In equivalence with the case where the relation is sensitive, $n' \leq n$, and $ns_i \in Domain(A)$, $i = 1, 2, . . . ,n$.

Associated values. Let us say $e_i = E(t_i)[A]$ is the encrypted demonstration of an attribute value of A in a sensitive tuple of the relation R_{E_P}, and ns_j is a value of the attribute A for some tuple of the relation R_{P_E}. They said that e_i is associated with ns_j (denoted by $\overset{a}{=}$) if the plaintext value of e_i is identical to the value ns_j. Because hybrid data partitioning used, this association applies only to tuples divided horizontally.

Associated tuples. Let us say t_i is a sensitive tuple of the relation R_{E_P} (i.e., R_{E_P} stores encrypted representation of t_i) and t_j is a non-sensitive tuple of the relation R_{P_E}. the authors state that t_i is associated with t_j (for an attribute, say A) if the value of the attribute A in t_i is associated with the value of the attribute A in t_j (i.e., $t_i[A] \overset{a}{=} t_j[A]$). Note that this is the same as stating that the two values of attribute A are equal for both tuples.

Relationship between counts of sensitive values. Let v_i and v_j be two different values in Domain(A). They denote the relationship between the counts of sensitive tuples with these A values (i.e., $|s (v_i)|$ (or $|s (v_j)|$)) by $v_i \overset{r}{\sim} v_j$.

Note that $\overset{r}{\sim}$ can be one of <, =, or > relationships. Such as, in example, the $t_2 \overset{r}{\sim} t_4$ corresponds to =, since both values have exactly one sensitive tuple in relation divided horizontally R_{E_P} (see Table 5).

Given the above definitions, the authors formally state the security requirements needed for selecting SQL queries over sensitive (encrypted values) and non-sensitive (plaintext values) data so that it does not leak any information. Before that, it is worthy of mentioning the security definition in the context. The inference attack in partitioned computing can be considered under the known-plaintext attack (KPA) category. The adversary could know some plaintext data hidden in a set of cipher-text. The adversary's goal in KPA is to designate cipher-text data that are related to a given plaintext, i.e., define a mapping between cipher-text and the corresponding plaintext data representing the same value. In the adversarial view, non-sensitive values are visible to the adversary in plaintext. However, the attacks are different since, unlike the case of KPA, in the proposed setup, the cipher-text data might not contain any data value that is the same as some non-sensitive data visible to the adversary in plaintext. That means by assuming the useing of the non-deterministic encryption to encrypt the sensitive data, the adversary cannot launch the chosen plaintext attack (CPA) and the chosen-ciphertext attack (CCA). It is not subject to the cipher-text only attack (COA).

D. *Encryption Technique*

In the proposed solution, the authors create an Microsoft .NET Framework common language runtime (CLR) functions to encrypt and decrypt data. CLR function is created as a database object inside an instance of SQL Server as a programmed assembly. CLR function is built using Microsoft visual studio 2015 with C# language, and the encryption implemented using the AES encryption technique. Algorithm 4 shows how the encryption is applied.

```
Algorithm 4 Encryption
     Inputs: Tuple_ID, Attributes_Value
     Outputs: Cipher-Text
     Variable: Encryption_Key
1    Function Encryption ( Tuple_ID, Attributes_Value ) begin
2        | Encryption_Key ← GenerqateKey (Tuple_ID)
3        | Cipher-Text ← AES_Encryption (Attributes_Value,
         |                          Encryption_Key)
4        |___ Return Cipher-Text
```

III. Implementation, Results, and Discussion

This Section presents the implementation's practical approach by applying Hybrid Partitioning and QB to the trusted and untrusted Databases, respectively.

To demonstrate the proposed approach effectiveness, the proposed approach is tested against inference attacks. The inference attacks are applied with and without adopting the proposed approach. The results are discussed at the end of this Section.

The rest of this Section is organized as follows; Section A introduces the tools used for implementing the proposed approach. Section B describes the steps of implementing the proposed approach. Finally, section C presents Experiment Results and Discussion.

A. *Experimental Tools*

This section introduces the tools used to implement and test the proposed solution; the authors used Microsoft SQL server 2014 installed on Windows Server 2012 R2 to store the database and build the proposed solution. Besides, they used a stored procedure as a tool to log the performance of query requests. Besides, they used Microsoft Visual studio 2015 to write SQL assembly files for encryption and decryption of data.

The experiment environment specification used to evaluate the proposed approach: includes Processor Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz (2 CPUs), Installed Memory 32 GB RAM, Hard Disk 512 GB, Microsoft SQL Server 2014, and Windows Server 2012 R2 Standard 64-bit.

B. *Implementation of Proposed Approach*

This section explains the practical approach and the implementation of the Hybrid partitioning technique used. Fig. 3 shows a general overview of the proposed approach architecture where the two database servers host the trusted and untrusted databases, respectively. The first database server is connected to the internet and private network and hosts the trusted database. The second database server is connected to the internet and is hosting the untrusted database. The Client's devices are connected to the private network.

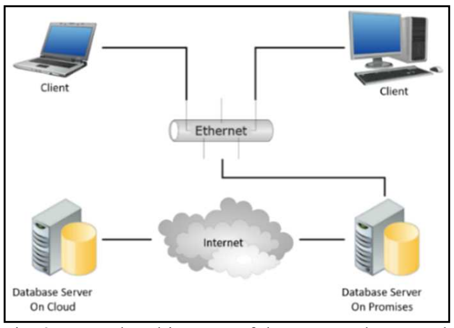

Fig. 3 General architecture of the proposed approach

The rest of this section describes how the proposed approach is implemented.

1) *Data Partitioning*

A stored procedure in SQL server is built to implement the Hybrid data partitioning.

2) *Data Encryption*

Data encryption is implemented by writing two functions in C# language using Microsoft Visual Studio 2015 and Advanced Encryption Standard (AES).

C. *Experiment Results and Discussion*

To evaluate the proposed approach, experiments are conducted with a different number of tuples retrieved from the database, starting from 2000 tuples, and then the number is increased by 2000 until it reaches 20,000 tuples, and in each experiment, the number of attributes that contain sensitive values is gradually increased from 1 to 10. Besides, these attributes contain 50% of the sensitive values only. The following subsections describe the results of the trials in more detail.

1) *Security Proof and Experiments*

In [3], the authors proved that QB is secure and satisfies the definition of partitioned data security and proved that all the sensitive bins are associated with all the non-sensitive by proving that equations mentioned in [3]. have satisfied the data security properties (non-linkability and indistinguishability). Furthermore, after using Hybrid partitioning, a new security gap is raised: the adversary can learn and link the encrypted values (sensitive attributes) to values that are not encrypted (non-sensitive attributes) in the same tuple. This gap does not satisfy new equation in [1]. However, using different keys in the public database for each tuple ensures that equation in [1] is satisfied. It is worth mentioning that the adversary cannot learn anything from the encrypted data since the DB owner is the only party who knows the keys and the Metadata. The Metadata relation is hidden from the adversary [1].

All experiments show that the result of equation 1 to retrieve the original relation from divided relations on public database has 0 tuples all the time. This satisfies equation in [1]and therefore satisfies the data security property (non-linkability).

The first experiment that is being discussed is a query to retrieve 2000 tuples. Table 9 shows the experimental results of the comparison between the proposed approach and PANDA. Experiment results of query execution performance to retrieve 2000 tuple assume %50 of values in sensitive attributes are sensitive.

TABLE 9 QUERY EXECUTION EXPERIMENT RESULTS FOR 2000 TUPLES, 50% OF VALUES ARE SENSITIVE IN EACH ATTRIBUTE

# of Sensitive attributes	Technique		Enhancement percentage
	Proposed approach	PANDA	
1	2.89	9.08	68%
2	4.00	9.08	56%
3	4.97	9.08	45%
4	5.54	9.08	39%
5	6.10	9.08	33%
6	7.09	9.08	22%
7	7.52	9.08	17%
8	8.33	9.08	8%
9	9.49	9.08	~0%
10	9.88	9.08	~0%

Equation 2 shows how the enhancement percentage (EP) of query execution times is calculated.

$$EP = (1 - \frac{Our\ approach\ time}{PANDA\ time}) \times 100\% \qquad (2)$$

Fig. 4 illustrates the performance of query execution time for ten different partitioned relations according to the number of sensitive attributes in the original relation, 2000 tuples and 50% of tuples contain sensitive values for the proposed approach and PANDA. Unites are measured in seconds.

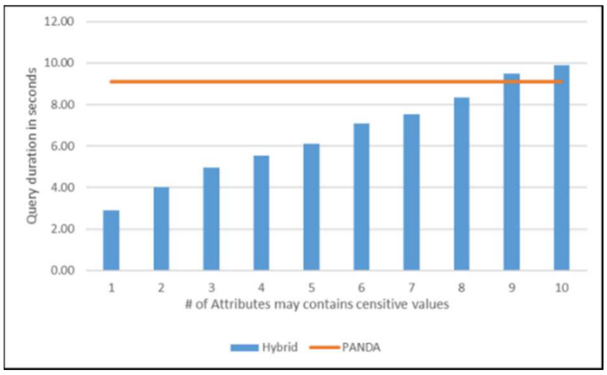

Fig. 4 Query execution experiment results for 2000 tuples, 50% values are sensitive in each attribute.

Overall, the PANDA technique takes more query execution time than the proposed approach in the given attributes range. Both PANDA and the proposed approach spend most of their query execution times when all attributes contain sensitive values.

Finally, the rest of experiments retrieved 4000, 6000, 8000, 10000, 12000, 14000, 16000, 18000, 20,000 tuples. For each experiment, a different number of tuples and ten different partitioned relations according to the original relation's number of sensitive attributes. The sensitive attributes have 50% of tuples with sensitive PANDA values, and the proposed approach, Unites, is measured in seconds.

Overall, the PANDA technique takes more query execution time than the proposed approach in each experiment's attributes range. Both PANDA and the proposed approach spend most of their query execution times in each experiment when sensitive attributes are nine. Furthermore, the most significant difference in performance between the two techniques is when the number of sensitive attributes is one. On the other hand, the proposed approach increases slightly on the number of attributes 10.

TABLE 10 AVERAGE OF PERFORMANCE ENHANCEMENT OF PROPOSED APPROACH

# of Sensitive Attributes	Enhancement Percentage
1	82
2	72
3	62
4	54
5	44
6	35
7	27
8	16
9	6
10	~0

In General, according to Table 10, there is an enhancement in the performance of query execution time. It is worth mentioning that most of the relations do not fully contain sensitive values.

IV. CONCLUSION

In this paper, a Hybrid approach for data partitioning aimed to secure sensitive data when outsourcing data is presented. The proposed approach is essential to secure the sensitive data that is outsourced to a public database. The proposed approach has the main advantage of improving query performance and securing sensitive data against inference attacks. The proposed approach has been evaluated using a set of experiments of partitioning data in an untrusted database. Besides, comparisons of the results with the PANDA technique are presented. The results of the proposed approach were satisfactory in which the properties of defining the data security satisfy the non-linkability and indistinguishable. Furthermore, the proposed approach results are satisfactory, where the performance of query execution is better than the results of PANDA performance.

References

[1] S. K. Badran, N. Arman and M. Farajallah, "Towards a Hybrid Data Partitioning Technique for Secure Data Outsourcing," in *The International Arab Conference on Information Technology ACIT*, Cairo, 2020.

[2] S. Mehrotra, S. Sharma, J. D. Ullman and A. Mishra, "Partitioned Data Security on Outsourced Sensitive and Non-Sensitive Data," *2019 IEEE 35th International Conference on Data Engineering (ICDE),* pp. 650-661, 2019.

[3] S. MEHROTRA, S. SHARMA, J. D. ULLMAN, D. GHOSH and P. GUPTA, "PANDA: Partitioned Data Security on Outsourced Sensitive and Non-sensitive Data," *ACM Transactions on Management Information Systems,* 05 2020.

[4] M. A. Abdelraheem, T. Andersson, C. Gehrmann and C. Glackin, "Practical Attacks on Relational Databases Protected via Searchable Encryption," *International Conference on Information Security,* pp. 171-191, 9 September 2018.

[5] S. Mehrotra, Y. O. Kerim and S. Shantanu, "Exploiting Data Sensitivity on Partitioned Data," *From Database to Cyber Security,* pp. 274-299, 2018.