RESEARCH ARTICLE OPEN ACCESS

Enhancing Performance of Search Over Encrypted Data in Cloud Computing

Nasrin Dalil Ali [1], Ahmed Kayed Ahmed [2]
Department of Computer Science and Information Technology [1]
Sudan University of Science and Technology, Khartoum, Sudan,
Department of Computing and Information Technology [2]
Sohar University, Sohar, Oman,

ABSTRACT
Order Preserving Encryption (OPE) scheme is an appealing method for database encryption as it allows execute sort and range queries without decrypting data. Popa's presented an ideal mutable order preserving encryption (mOPE), but their cost of communications between the client and the server to process queries is very high. This paper introduces an OPE model improved on mOPE model for reducing the rate of requests and responses between the client and the server and enhancing the performance when querying the encrypted database. Our model presents a new indexing mechanism that will help to arrange the encrypted data in the server. However, it uses two types of encryption techniques. The first type applies general encryption technique and the second type follows the Popa's mOPE and replacing the one OPE tree in mOPE scheme with a number of OPE trees in the server. Further, it permits the server to accomplish part of works without revealing any information about original data besides the order. We implement our scheme and evaluate it on simulation programs. We observe that in addition to providing efficiency and security, our scheme achieves better performance than mOPE scheme.
Keywords :- Order preserving encryption, mOPE, base model.

I. INTRODUCTION
Outsourcing data to cloud service offers lower cost by sharing hardware and elastic scaling [1] [2]. A key problem in outsourcing storage is that the sensitive data may be subject to unauthorized access not only from an unknown attacker but also from curious service provider [3]. One possible approach to protect outsourced data is encryption [3] [4]. But this solution requires more decryption operations when there is a need to process data. To face this limitation the cloud server should be able to perform some computation over encrypted data exactly as the same as for unencrypted one. The idea behind this is to use encryption techniques that support processing operations on encrypted data exactly and efficiently as unencrypted one and without knowing any information about it [5].

A practical approach that facilitates querying over encrypted data is order-preserving encryption scheme (OPE) [6] [7]. OPE scheme is a common structure in which for cipher-text values C_1 and C_2 corresponding to plaintext values P_1 and P_2, if $P_1<P_2$ then $C_1<C_2$. OPE scheme permits the untrusted server to process SQL queries over encrypted data especially comparison queries and range queries [8] [9].

There has been significant work on OPE schemes, but mOPE model in [10] is the first one that achieves ideal IND-OCPA security (Indistinguishability under Ordered Chosen-Plaintext Attack). IND-OCPA is the strongest security definition for order-preserving schemes proposed in [6] in which the adversary cannot learn anything about the plaintext values except for their order.

The mOPE functions by building a balanced binary search tree comprising all the encrypted data on the server. In such a tree for each node v, all the nodes in the left subtree are smaller than v and all the nodes in the right subtree are greater than v. Fig. 1 shows the OPE tree which adapted from Popa's mOPE [10].

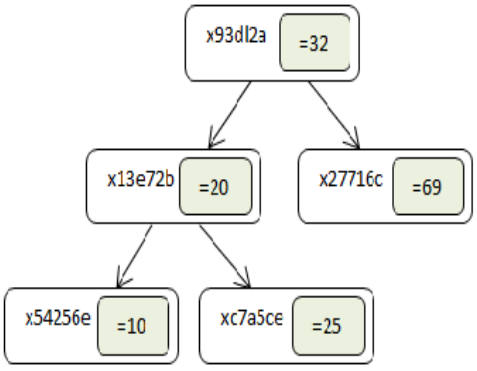

Fig. 1 The OPE tree in mOPE scheme. Each node contains the cipher-text, the gray blocks show the corresponding plaintext value [10].

The following is an example used in [10] to illustrate the structure of mOPE model and how the encrypted data is arranged in the OPE tree. Consider the setting in which the client inserts 55 into the server using the OPE tree shown in Fig. 1In the beginning, the client demands the server for the root node of the OPE tree, and then the server returns the encrypted value x93d12a. Which the client decrypts to 32 and compares 55 with 32 since 55 greater than 32 the client

demands the server for the right child of the root node. The server responds with the encrypted value x27716c. The client decrypts x27716c to 69 and compares 69 with 55 since 55 lesser than 69 then the client demands the server for the left child of the new node. The server responds that there is no such child. This means that the client can insert 55 in this position [10].

One problem of mOPE is that; the server needs the client's help usually to move over the OPE tree. The dependence of the server on the client causes more requests and responses between them to find the right position in the OPE tree. Unfortunately, the more data we store, the more requests and response we generate. Hence, the system performance is a slowdown.

This paper introduces a scheme follows the mOPE scheme proposed in [10]. A key contribution of our scheme lies in reducing the dependence of the server on the client by permitting the server performs part of the search processes without leaking any information about the original data besides the order. More precise, we reduce the number of requests and responses between the client and the server by replacing the one OPE tree in mOPE scheme with the number of OPE trees in the server. Besides that, we employ an indexing mechanism to speed up the process of search operations (section III). To show how our model reduces the communication between client and server, we implement simulation programs to perform insert operations and search operations. We showed that our model provides a higher level of security than mOPE model (section IV). Detailed experiments show that the new model achieves a lower rate of communication across the network compared with mOPE model (section VII). Sequentially it provides higher performance than mOPE model.

The study is organized as follows: Section (II) is dedicated for to reports the related work. Section (III) introduces the model scheme. Section (IV) shows the security evaluation. Section (V) assesses the performance evaluation. Section (VI) is dedicated to implementation. Section (VII) elaborates experiments. Section (VIII) explains the results & discussion and section (IX) concludes the study.

II. ACADEMIC STATE OF THE ART

Our work is mostly related to OPE schema. The notion of OPE scheme was proposed earlier by Agrawal [7]. The first formal study of the concept and its security was performed by [6]. There has been a significant amount of work on OPE schemes which allow querying over encrypted database without any change on the database engine [11] [12] [13] [14] [15] [16]. Despite the large body of works, the prior schemes reveal more information about the plaintext besides the order. The first ideal order-preserving encoding schema that achieves IND-OCPA security presents in mOPE [10]. Since the mOPE leaks order information for plaintext values as the minimum needs for the order-preserving property, [17] discuss approaches to limit the impact of this leakage.

Our work is also related to cryptographic schemes for performing queries over encrypted data. In [3], it presents a general framework for analysing and constructing searchable public-key systems for various families of predicates. Then they construct public-key systems that support comparison queries, general subset queries, and conjunctions. [18] It provides CryptDB, a system to protect data confidentiality against threats by executing SQL queries over an encrypted database. The architecture of CryptDB consists of a database proxy which intercepts all the user's queries and transforms them into an understandable form for the DBMS server. It is also responsible to perform all the encryption and decryption processes. As the system users request order queries, they apply the OPE technique by using AVL binary search trees (an AVL tree is a self-balance tree that has advantages of the binary search tree). Giang and Keong in [19] are investigating certain types of complex queries namely multi-dimensional range queries over encrypted data in cloud platforms (where the plaintexts are multi-numerical attributes). Their solution combines packetization based scheme with order-preserving encryption-based techniques. Further, our work basically related to mOPE introduces in [10]. Our scheme provides ideal-security similar to Popa's mOPE. We improve on their structure and combine it with indexing technique to reduce the amount of communication done globally. There are some studies improve on Popa's mOPE results and also provide ideal-security [20] [21].

III. The DEVELOPED MODEL

The developed model based on Popa's mOPE architecture described in [10]. It consists of two main parts: the client (or an enterprise) who is the owner of data to be outsourced, thus the client is trusted; and the database server which provides storage to the client's data, it is untrusted.
Moreover, we introduce a trusted part at the client-side called "trusted party" which acts as a connector between the client and the database server. Also, we develop a structure called "index table" and allocate it in the trusted party, as illustrated in Fig. 2. The index table information plays a basic role in both storing ordered encrypted data in the server and retrieving exact results to the client.

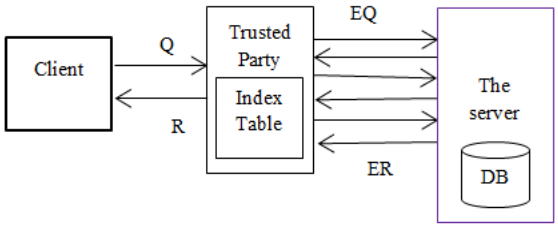

Fig. 2 The architecture of the new model and the data flow between the client, the trusted party, and the server. Q: query. EQ: encrypted query. ER: encrypted result. R: result

In the next subsections, we explain the structure of our model. Mainly, we describe the responsibilities of the trusted party, the structure of the index table, how the encrypted data is arranged in the database server, and how to access the encrypted data in the database server.

A. Trusted Party

It acts as a communication channel between the client and the database server. It allocates either near the client boundaries or inside it. It intercepts all requests and responses between the client and the server. The trusted party is responsible for encrypts the client's queries, rewrites queries in an understandable form to the server, decrypts the returned encrypted results, and gives the client exact results. Also, it applies two different encryption techniques to the client's data as we explain in section III-C. Fig. 2 shows the communication between the client, the trusted party, and the server.

B. *Index Table*

It is the essential part of the developed model. We should create it at the initialization stage of the database system. The index table is built based on the expectation of the client's data that would be outsourced to the server. The total number of expected client's data (from the smallest one to the greatest one in sequence) is dividing to groups of equal elements. Actually, we divide the total number of expected client's data by a suggested number. As a result, we obtain a number of groups, each one of them contains number of elements equal to the suggested number that used to divide the expected client's data. Really, every group of elements implements a part of the expected client's data in sequence. Each group of sequence elements is called "range" and the suggested number of elements in the group is called "range value".

Of course, the total number of ranges depends on the length of developed range. Thus, the smaller range value has a shorter range and produces many ranges, whereas the greatest range of value has longer range and produces a few ranges. Note that, we construct our model for numeric data. For simplicity, we implement the expected client's data in forms of integer values.

Moreover, the structure of the index table is based mainly on the ranges of the expected client's data. Once the range value is determined then the ranges of the expected client's data are appearing clearly (also their start value and end value). We use the ranges information to create the index table like the following:

1. Specify the start value and the end value for each range.
2. Give each range a unique id range.
3. Create the index table with three columns: id range (key-value), start value, and an end value.
4. Insert the values of the id range, the start value, and the end value for all ranges in the index table and save them.

As a result, we have the index table in the trusted party with the number of rows equal to the total number of ranges. And each row contains unique values related to a specific range.

Table I shows an example of an index table. It implements the expected client's data in the form of integer values from 1 to 20 and the suggested range value=5. As we see in Table I, the index table has four ranges as the following:

- The first range has an id range=1, a start value=1, and an end value=5.
- The second range has an id range=2, a start value=6, and an end value=10.
- The third range has an id range=3, a start value=11, and an end value=15.
- The forth range has an id range=4, a start value=16, and an end value=20.

Table I. The Index Table for Expected Client's Values from 1 to 20 and the Range Value=5

Id range	Start value	End value
1	1	5
2	6	10
3	11	15
4	16	20

From all the above mentioned we can say that the two factors affect the index table data are the expectation of the client data to be outsourcing and the length of suggested range value. In the next subsection, we explain how the client's data is organized in the database server.

C. Encrypted Data in the Database Server

Mainly, the arrangement of encrypted data in the database server is directly related to the index table. The basic idea is to have the client's data organized at the server in two different parts. One part uses encryption technique doesn't preserve the order of data and the other part uses encryption technique that preserves the order of data. Based on the index table information, the first part which might not preserve the order of encrypted data is for the id ranges, and the second part which preserves the order of encrypted data is for the client's data that reside in the same range (the values bounded from start value up to end value for the exact id range in the index table).

Furthermore, to encrypt the client's data corresponding to the index table in Table I the trusted party will apply the two encryption techniques like the following:

- The id ranges: 1, 2, 3, and 4 are encrypting by using an encryption technique doesn't preserve the order.

- The values from 1 to 5 are encrypting by using an encryption technique that preserves the order of data.
- The values from 6 to 10 are encrypting by using an encryption technique that preserves the order of data.
- The values from 11 to 15 are encrypting by using an encryption technique that preserves the order of data.
- The values from 16 to 20 are encrypting by using an encryption technique that preserves the order of data.

More importantly, all the encrypted id ranges of the index table are storing in the server before sending the client's data. This is because the encrypted id ranges act as the directories for the client's data that will reside in the different ranges. Every one of them is pointing to the part of the client's data that belongs to a specific range as declared in the index table. As we said before, those id ranges are encrypting by using an encryption technique doesn't preserve the order of them. Because of this, they are presenting in the server as unordered. Moreover, we follow Popa's mOPE which described in [10] to order the encrypted data in different ranges. That is, we use the AVL tree to preserve the order of the client's data that resides in the same range. And every encrypted id range is pointing to the root node of an AVL tree as declared in the index table. As a result, we would have many AVL trees in the server based on the index table knowledge. Every AVL tree contains values related to a specific id range (values from the start value to the end value as declared in the index table).

Therefore, in the server, we preserve the order of encrypted data that belongs to the same range, but we don't preserve the sequence of ranges. This provides some aspect of security to user's data in the database server.

Of course, the AVL tree has advantages of the binary search tree, in an AVL tree for each node v, all the left subtree nodes of v are smaller than v and all the right subtree nodes of v are greater than v. Also it is a self-balance tree, so that after balancing the OPE tree some nodes may change their positions. Practically, the nodes of the tree are implementing in the server as pre-ordered traversal (root node, left subtree nodes and right subtree nodes respectively). This allows us to restore the original tree construction from the database to perform operations. Because of the tree balancing and the need for maintaining the pre-order traversal implementation, the server must update any storage related to the balanced AVL tree. Note that, such update affects only the target AVL tree not all the other AVL trees corresponding to the other ranges.

D. Access of Encrypted Data in the Database Server

The power of the index table is appearing clearly when we need to access the data in the database server. The following points describe the general algorithm that used to process the encrypted data in the server:

1. The client sends v to the trusted party (to insert or search for).
2. The trusted party determines the exact id range for v from the index table.
3. The trusted party encrypt the id rang and sends the Enc (id) to the server.
4. The server obtains the Enc (id), then:
    a) It starts a sequential search for matching value by performing equality checks.
    b) When finding the matching value it tells the trusted party.
5. The trusted party asks the server for the encrypted value c^{-} at the root node of the AVL tree which the Enc(id) point to.
6. The trusted party decrypts c^{-} and obtains v^{-}.
7. Compare the decrypted value v^{-} with the client value v:
    a) If $v < v^{-}$ the trusted party tells the server go left.
    b) If $v > v^{-}$, the trusted party tells the server go right.
    c) If $v = v^{-}$ the trusted party tells the server it is found.
8. The server returns the next encrypted value c" at the tree node based on the trusted party's information, and goes back to step 6.
9. The algorithm stops when v is found, or when the server achieves an empty node in the tree.

From the previous algorithm, we observe that to process encrypted data in the OPE storage we require two search operations into two different directions. Vertically, the first search operation for the suitable id range to reach the target AVL tree. Horizontally, the second search operation for the appropriate location in the target AVL tree. More important, the process of search for the encrypted id range is performing locally in the server, whereas the search over the AVL tree is performing globally across the network with the help of the trusted party. So, instead of return to the trusted party in all cases we just return when the encryption technique preserves the order.

For instance, suppose the actual client's values are: 1, 3, 4, 13, 14, 15, 16, and 19 using the index table shown in Table I. Table II illustrates the structure of the previous client's data in the database server. The grey blocks show the encrypted id range, while the white one shows the encrypted client's data. Also, we use the label (") to denote the part of data encrypted by encryption technique that does not preserve the order of data. And we use the label ($^{-}$) to denote the part of data which encrypted by encryption technique that preserves the order of data.

Table II. The Client Data in the Database

2"					

4“	16 $^-$	19 $^-$			
1“	3 $^-$	1 $^-$	4 $^-$		
3“	14 $^-$	13 $^-$	15 $^-$		

As we see in Table II, the encrypted client's values are arranging in their suitable location as specified before in the index table. Every encrypted id range is referencing to an AVL tree that contains data of a specific range. And all of the AVL trees are implementing in the form of pre-ordered traversal tree.

For example, assume the client wants to insert 2 using the database shown in Table II. Whenever the trusted party receives the value it looks in the index table for the suitable id range (the related index table is shown in Table I). It finds that 2 is in the id range=1. Next, it encrypts 1 by encryption technique doesn't preserve the order (the same one used before storing all id ranges in the server) and obtains **1**“. Then, it asks the server to search for **1**“. The server makes the first check: 2“doesn't match **1**“, the second check: 4“doesn't match **1**“, and the third check: **1**“matches **1**“. At this point, the server reaches the target AVL tree.

After that, the server needs the help of the trusted party to move over the target AVL tree to find an empty node. Fig. 3(a) shows the target AVL tree. Firstly, the trusted party requests the root node of the AVL tree, and the server returns 3^-, which the trusted party decrypts to 3. Since 2 < 3, the trusted party requests the left child of the root node, and the server responds with 1^-, which the trusted party decrypts to 1. 2 >1, so the trusted party requests the right child of the last node requested, and the server responds that there is no such child. This means that the trusted party can insert 2^- in this position. Fig. 3(b) displays the AVL tree after inserting 2^-. Lastly, there is a need to update the part of the database where the AVL tree is storing. This is because we have to maintain the pre-order traversal tree in the server. Fig. 3(c) shows the impact of insertion on the database.

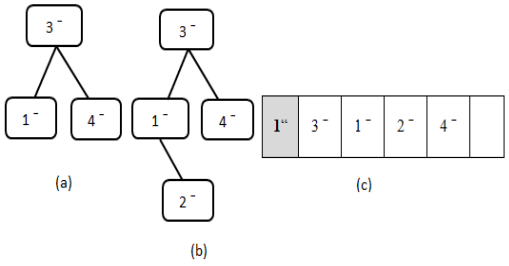

Fig. 3 The AVL tree pointed by the id range=**1**“ in the database server. (a) The target AVL tree. (b) The AVL tree after inserting 2^-. (c) The part of the database contains the target AVL tree after inserting 2.

From the above example, we observe that the server can perform the search for the right id range without any trusted party involvement, while it needs the help of the trusted party to find the suitable location in the AVL tree where to insert 2^-. Also, we see that the server makes three checks till it finds the exact id range, after that, there are 6 requests and 6 responses between the trusted party and the server before reach an empty node to insert 2^-. Thus, the server is responsible to perform all the tasks related to the encryption technique that does not preserve the order independently. More importantly, the trusted party told the server order information only for the exact AVL tree and the interaction between them doesn't reveal anything else besides the order.

IV. SECURITY EVALUATION

Since the developed scheme is order-preserving encryption, it reveals nothing in addition to order. The justification of why the new model is secure can prove from the base model which is mOPE. In the Popa's mOPE, they prove that their OPE model is ideal and it achieves IND-OCPA security where the data is ordered in one AVL tree. Further, the server knows nothing besides the order of data. Indifference to theirs, ours order the data in more than one AVL tree based on the length of the range. So, the server reveals only the order of data for the target AVL tree not for all others AVL trees. An important facet of the new model is that, when the length of range is equivalent to the total number of expected client's data then the data is ordered in one AVL tree. This case is the same structure as the mOPE model and has the same level of security as it. Therefore, if we decrease the range value then the depth of the AVL tree becomes smaller and the information leakage becomes lower.

This means, the length of range mainly affects the level of security provided by the model. We can conclude the following:

1. The improved model is ideal and achieves IND-OCPA security.
2. The smaller range value offers a better level of security than the greater one.
3. The increasing of the range value results in decreasing the level of security.
4. We reach the lowest level of security when all the data is organizing in one AVL tree. This state provides the same level of security as the Popa's mOPE.

V. PERFORMANCE EVALUATION

Practically, we measure the performance of our model in terms of calls between the trusted party and the server during performing operations. We count the requests and responses between the trusted party and the server to perform both of the insert operation and the search operation. Besides that, we compute the number of checks done by the server to reach the target AVL tree. In order to make these two computations we characterize two counters to compute in two different directions. The first counter works in the area that doesn't preserve the order of id ranges. It concerns the checks

performed locally in the server. And it counts how many times the server checks the encrypted id ranges to find the appropriate one; this counter called "counter1". The second counter works in the direction that preserves the order of data. It follows the calls performed globally throw network. And it computes how many times the server asks the trusted party to reach the right position in the AVL tree and receives answers; this counter called "counter2".

To evaluate the performance of our model we aim to answer the following questions in the coming sections:

- Does the developed model enhance the performance comparing with the Popa's mOPE model?
- Is it better to apply small or great range value to provide good performance?
- How much does the length of range affect the outcomes of counter1 and countet2?

VI. IMPLEMENTATION

We implement our model including a simulation program in C code to implement the three parts of the model (client, trusted party, and server). We use user-defined functions to describe all the scheme activities. Also, we implement the server-side by adding Database environment.

VII. EXPERIMENTS

In order to answer the questions mentioned in section V, we consider two scenarios. In the first scenario, the client sends his data to store in the database server. In the second scenario, the client retrieves his data from the database server. The structure of the developed model and the based model are implementing in both of the scenarios.

Furthermore, we compare the outcome of "counter1" and "counter2" in the developed model with the Popa's mOPE one using the same arguments (counter1 and counter2 were defined in section V). Unlike the proposed model, Popa's model involves only one counter to compute the numbers of requests and responses between the client and the server. This is because the server completely depends on the client to perform its tasks [10]. Of course, the ones in the base model has the same tasks as "counter2" in the new model and both of them are work in the same area. Because of this, we use them in order to make a comparison between the two models.

A. Insert Operation

In this set, we implement the structures of the developed model and the base model in case of performing insert operations. For simplicity, we use the client's data in forms of integer values in the two models. We generate random distinct numbers from 1 to 100 for every experiment separately. We don't take into consideration the sequence of the generated numbers for all experiments. More important, despite we use 100 numbers as client's data, but we obtain the same results if we use more.

1) Insert in the New Model: We perform eleven experiments for the new model using a different range of values vary from small to great. Our strategy is to start with a small range of value and increasing it every time. In the beginning, we choose 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 to be the range values for our eleven experiments. Then, every range value is used to create an index table for a separate experiment. Of course, all of the experiments have expected client's data as numbers from 1 to 100.

Table III shows the index tables that we are created for the eleven experiments. For each experiment, we explain the selected range value, the specified id ranges, and the start values and end values related to the id ranges. As we see in Table III, in experiment no.1: the range value equal to 5. The index table has twenty ranges and twenty id ranges from 1 to 20 related to them. The start values and the end values corresponding to those id ranges are: 1-5, 6-10, 11-15, 16-20, 21-25, 26-30, 31-35, 36-40, 41-45, 46-50, 51-55, 56-60, 61-65, 66-70, 71-75, 76-80, 81-85, 86-90, 91-95, and 96-100 respectively. In experiment no.2: the range value is equal to 10. The index table has ten ranges and ten id ranges from 1 to 10 corresponds to those ranges. The start values and the end values for the ten id ranges are: 1-10, 11-20, 21-30, 31-40, 41-50, 51-60, 61-70, 71-80, 81-90, and 91-100 respectively. Table III describes the details of the index tables for each experiment separately.

Table III. The Index Tables for the Eleven Experiments

Experiment No.	Range value	Id range	Start value	End value
1	5	1	1	5
		2	6	10
		3	11	15
		4	16	20
		5	21	25
		6	26	30
		7	31	35
		8	36	40
		9	41	45
		10	46	50
		11	51	55
		12	56	60
		13	61	65
		14	66	70
		15	71	75
		16	76	80
		17	81	85
		18	86	90
		19	91	95
		20	96	100
2	10	1	1	10
		2	11	20

		3	21	30
		4	31	40
		5	41	50
		6	51	60
		7	61	70
		8	71	80
		9	81	90
		10	91	100
3	20	1	1	20
		2	21	40
		3	41	60
		4	61	80
		5	81	100
4	30	1	1	30
		2	31	60
		3	61	90
		4	91	100
5	40	1	1	40
		2	41	80
		3	81	100
6	50	1	1	50
		2	51	100
7	60	1	1	60
		2	61	100
8	70	1	1	70
		2	71	100
9	80	1	1	80
		2	81	100
10	90	1	1	90
		2	91	100
11	100	1	1	100

At the end of each experiment, we achieve the number of local checks in the server for the right id ranges (counter1) and the number of calls between the trusted party and the server across the network (counter2). The final results of "counter1" and "counter2" for all experiments are illustrating in Table IV.

Table IV. The Results of "counter1" and "counter2" after Inserting Random Distinct Numbers from 1 to 100 in the New Model

The New Model			
Insert random distinct numbers from 1 to 100			
Exper. No.	Range value	Counter1	Counter2
1	5	1050	486
2	10	550	646
3	20	300	836
4	30	260	914
5	40	200	972
6	50	150	1066
7	60	160	1076
8	70	170	1104
9	80	180	1136
10	90	190	1176
11	100	100	1272

2) Insert in the Base Model: On the other hand, we implement the structure of Popa's mOPE model to insert data in the database server. We use similar data to that used for the insert in the new model (random distinct numbers from 1 to 100).

As we know before, this model organizes all the encrypted data in one AVL tree and the server totally depends on the client to move over the tree. Because of these two reasons, we establish only one counter called "counter" to count the calls between the client and the server during the insertion of data. Table V shows the total requests and responses between the client and the server after inserting the 100 numbers in the base model.

Table V. The Results of "counter" after Inserting Random Distinct Numbers from 1 to 100 in the Base Model

The Base Model	
Insert random distinct numbers from 1 to 100	
counter	1272

B. *Search Operation*

In this set, we examine our model in case of retrieving data from the database server. We aim to search for the same values with a fixed sequence in the two models (the new model and the base model). Hence, we select random twenty numbers between 1 and 100 to search for them in all of the search experiments. The twenty numbers are: 95, 93, 40, 64, 26, 66, 19, 46, 3, 47, 91, 38, 12, 16, 8, 53, 56, 52, 99, and 24 sequentially. We use the databases those results from the

insert operations in the new model to search for the twenty numbers in the two models.

1) *Search in the New Model:* In order to evaluate the performance of the search operations in the new model, we perform the search operations on the databases that resulted from the insert experiments in the new model. That is, the resulting ordered database from each insert experiment in a specific range value is using to search for the twenty numbers in the same range value. Practically, the first experiment of the search operations uses the database resulted from the first experiment of the insert operations, the second experiment of the search operations uses the database resulted from the second experiment of the insert operations, ……., the eleventh experiment of the search operations uses the database resulted from the eleventh experiment of the insert operations.

Table VI shows the eleven experiments of search operations in the new model. Each row describes details about an experiment. The first column for the experiment numbers and the second one for the range values that are used in the insert experiments before. "Counter1" illustrates the total numbers of checks for the right id ranges in the server. And "counter2" shows the total numbers of requests and responses between the trusted party and the server through the network.

Table VI. The Results of "counter1" and "counter2" after Searching for the Twenty Numbers in the New Model

The New Model			
Search for the 20 numbers			
Exper. No	Range value	Counter1	Counter2
1	5	170	88
2	10	131	116
3	20	62	146
4	30	59	146
5	40	42	172
6	50	31	204
7	60	34	200
8	70	36	220
9	80	36	204
10	90	36	190
11	100	20	224

2) *Search in the Base Model:* On the contrast, we perform the search operations for the twenty numbers in the base model. We use the database that resulted from the insert experiment in the base model. As we mentioned before, this model orders all the data in one AVL tree and the server needs the client's help in all cases. Because of these reasons, we initialize only one counter called "counter" to compute the total requests and responses between the client and the server while performing the search operations.

Table VII shows the total requests and responses between the client and the server to perform the search operation in the base model.

Table VII. The Result of "counter" after Searching for the Twenty Numbers in the Base Model

The Base Model	
Search for the 20 numbers	
Counter	246

VIII. RESULTS and DISCUSSIONS

Back to Table IV (the results of the insert in the new model) we observe that our scheme behaves better for the small range values than the greater one. We see that the increase of the range value produces significant increasing in "counter2" besides decreasing in "counter1". This is not surprising since the great range value generates a few ranges in the index table. Consequently, the size of the AVL tree is growing and hence its height. For the same reasons, we observe that those experiments of the range values are greater than 50 have a lower change in "counter1" and "counter2" than those of the range values are less than or equal to 50. We reach the greatest value of "counter2" when the range value is equivalent to the client's data expectation (experiment no 11). In this case, all the data organized in one AVL tree, "counter1" records the lowest result, and "counter2" records the highest result. This case is the worst case of our model since it records the highest communications rate throw the network.

Return to Table V (the results of the insert in the base model), we observe that the insertion of data costs the system 1272 requests and responses between the client and the server. Remember that, this model is the same as experiment no 11 of the insert in the new model since both of them are organizing the encrypted data in one AVL tree. Moreover, both the base

model and the last experiment of the insert in the new model are executing 1272 calls throw the network to complete the insert operations. Recall from the previous mentioned that, the order of encrypted data in one AVL tree represents the worst case of our model. This means, our model gets the same result as the Popa's mOPE model in it is worst-case otherwise it behaves better than the Popa's mOPE model.

Moreover, in Table VI (the results of the search in the new model) we observe that the greater range value produces few checks in the server (counter1) and more communications between the trusted party and the server (counter2) than the smaller one to retrieve data. The more range value decreases, the higher rate "counter1" gets and lower rate "counter2" obtains. We attribute this to the fact that the small range value produces more ranges in the index table and hence lower tree depth in the server than the greater range value. This is appearing clearly in Table VI in experiments no 1, 2, 3, 4, 5, and 6. They record significant decreasing in "counter1" and significant increases in "counter2" due to the range value increasing. Whereas in experiments no 7, 8, 9, 10, and 11 we see that: "counter1" and "counter2" record a few changes despite the increase of the range value. Experiment no 11 registers the lowest value in "counter1" and the highest value in "counter2".

In Table VII (the results of the search in the base model) we observe that the search operations for the target numbers cost the system 246 requests and responses between the client and the server. Remember that, this experiment is the same as experiment no 11 in Table VI since both of them search for the same elements in the same database structure (search over one AVL tree). Despite the similarity between the two experiments, there are differences in their results. This is because the data was generated randomly for the two experiments not in a fixed sequence.

Generally, in the developed model the small range values result in fewer calls globally but produce higher overload at the server. Of course, the greater range value results in the opposite. We concern about the calls across the network (requests and responses between the trusted party and the server) because they face more problems than performing tasks locally in the server (checks for the right id ranges). Further, our suggestion is to use a small range of values. The overload in the server can be solved by providing equipment at the server side (high processers).

From all the above mentioned we can say that, the developed model succeeds in providing better performance than the Popa's mOPE. Moreover, the model behaves better when using small range value to speed up the system performance. Take into account, our choice for the range value should respect the balance between the communication accomplished across the network and the tasks performed in the server.

IX. CONCLUSION

We present an efficient order-preserving encryption model improved from Popa's mOPE model. Our model succeeds in reducing the rate of requests and responses throw the network to perform queries over an encrypted database. We show that the use of small range values reflects significant performance enhancement. Moreover, the use of small range value has a positive impact on the model security compared with mOPE. Also, the model doesn't only achieve IND-OCPA security, but more than that it reveals nothing except for the order of data in the target AVL tree.

REFERENCES

[1] B. Sosinsky, Cloud Computing Bible, John Wiley & Sons, 2010.

[2] A. B. a. S. (. M. Rahman, "AN OVERVIEW OF THE SECURITY CONCERNS IN ENTERPRISE CLOUD COMPUTING," vol. 3, 2011.

[3] Dan Boneh, Brent Waters, "Conjunctive, Subset, and Range Queries on Encrypted Data," in *In Theory of Cryptography*, 2007.

[4] Manpreet Kaur, Rajbir Singh, "Implementing Encryption Algorithms to Enhance Data Security of Cloud in Cloud Computing," *International Journal of Computer Applications*, vol. 70, no. 18, p. 0975 – 8887, 2013.

[5] O. G. Sergei Evdokimov, "Encryption Techniques for Secure Database Outsourcing," in *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2007, pp. 327-342.

[6] Alexandra Boldyreva, Nathan Chenette, Younho Lee and Adam O'Neill, "Order-Preserving Symmetric Encryption," in *Annual International Conference on the Theory and Application of Cryptographic Techniques*, Cologne, Germany, 2009.

[7] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, Yirong Xu, "Order Preserving Encryption for Numeric Data," in *SIGMOD*, Paris, France, 2004.

[8] Hakan Hacıg¨um¨us, Bala Iyer, Chen Li, Sharad Mehrotra, "Executing SQL over Encrypted Data in the DatabaseServiceProvider," in *ACM SIGMOND*, 2002.

[9] Hasan KADHEM, Toshiyuki AMAGASA, Hiroyuki KITAGAWA, "Optimization Techniques for Range Queries in the Multivalued-Partial Order Preserving Encryption Scheme," in *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, 2010.

[10] Raluca Ada Popa, Frank H. Li, Nickolai Zeldovich, "An Ideal-Security Protocol for Order-Preserving Encoding," in *IEEE Symposium on Security and Privacy*, 2013.

[11] Hasan KADHEM, Toshiyuki AMAGASA, Hiroyuki KITAGAWA, "MV-OPES: Multivalued-Order Preserving Encryption Scheme: A Novel Scheme for Encrypting Integer Value to Many Different Values," *IEICE TRANS. INF. & SYST*, vol. E93–D , no. 9, 2010.

[12] H. Kadhem, "A SECURE AND EFFICIENT ORDER PRESERVING ENCRYPTION SCHEME FOR RELATIONAL DATABASES," in *International Conference on Knowledge Management and Information Sharing*, 2010.

[13] Seungmin LEE, Tae-Jun PARK, Donghyeok LEE, Taekyong NAM, Nonmembers, Sehun KIM, "Chaotic order preserving encryption for efficient and secure queries on databases," *IEICE Trans. on Info. and Systems*, vol. E92.D(11), 2009.

[14] Gultekin Ozsoyoglu, David A. Singer, Sun S. Chung, "Anti-Tamper Databases: Querying Encrypted Databases," in *IFIP Advances in Information and Communication Technology*, 2003.

[15] Liangliang Xiao, I-Ling Yen, Dung T. Huynh, "Extending Order Preserving Encryption for Multi-User Systems," Cryptology ePrint Archive, Report 2012/192, 2012.

[16] Liangliang Xiao, I-Ling Yen, "A Note for the Ideal Order-Preserving Encryption Object and Generalized Order-Preserving Encryption," Cryptology ePrint Archive, Report 2012/350, 2012.

[17] Vladimir Kolesnikov, Abdullatif Shikfa, "On The Limits of Privacy Provided by Order- Preserving Encryption," Bell Labs Technical Journal 17(3), 2012.

[18] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing," in ACM SOSP, 2011.

[19] Do Hoang Giang, Ng Wee Keong, "Multi-dimensional Range Query on Outsourced Database with Strong Privacy Guarantee," I. J. Computer Network and Information Security, no. 10, pp. 13-23, 2017.

[20] Florian Kerschbaum, Axel Schröpfer, "Optimal Average-Complexity Ideal-Security Order-Preserving Encryption," in ACM, 2014.

[21] K. Srinivasa Reddy, S. Ramachandram, "A New Randomized Order Preserving Encryption Scheme," International Journal of Computer Applications, vol. 108, no. 12, p. 0975 – 8887, 2014.

[22] Satish, Karuturi S R V, and M Swamy Das. "Review of Cloud Computing and Data Security." IJAEMA (The International Journal of Analytical and Experimental Modal Analysis) 10, no. 3 (2018): 1-8, 2018.