Design and Analysis of OCPA Secure Encrypted Databases

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Master of Engineering
IN
COMPUTER SCIENCE AND ENGINEERING

by
Gagandeep Singh

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

JULY 2014
DEDICATED TO

My Family

for continuous support and encouragement
Acknowledgements

I am deeply grateful to Prof. Jayant Haritsa for his unmatched guidance, enthusiasm and supervision. He has always been a source of inspiration for me. I have been extremely lucky to work with him.

I am thankful to Prof. Sanjit Chatterjee for providing the references during the Topics in Cryptology course which were very helpful in my project work. I am also very thankful to Mr. Akshar Kaul, Mr. Ramanujam S Halasipuram and Dr. Prasad M Deshpande for sharing the helpful references and time to time useful discussions that happened during the project work.

My sincere thanks goes to my fellow lab mates for all the help and suggestions. Also I thank my CSA friends who made my stay at IISc pleasant, and for all the fun we had together.

Finally, I am indebted with gratitude to my parents and other family members for their love and inspiration that no amount of thanks can suffice. This project would not have been possible without their constant support and motivation.
Abstract

In this thesis we present the work done in the construction of new Database model eDB under the *Database as a service* (DaaS) framework. The main challenge in the DaaS model is to provide data security from service provider along with efficient computation of queries on the cloud database. To handle both the challenges (data security and efficient computation), we use multiple partial homomorphic encryption schemes where each scheme handles a specific database operator, and the data is encrypted under these multiple encryptions such that cipher texts of these schemes co-exist at the server. For analyzing the security of eDB, we formally define the security model (IND-OCPA) against our system and theoretically layout the arguments to prove the security of the system against the security model. We have also developed the idea of query processing for floating point numbers in encrypted database. A new efficient storage strategy in form of *Encryption Dictionary* is presented in this thesis. The query execution time for queries involving only order predicate clauses improved significantly in Encryption Dictionary model as to comparison with traditional horizontal storage model with a trade off on aggregate queries where it performed slightly bad.
# Contents

Acknowledgements .................................................. i
Abstract .................................................................... ii
Contents .................................................................. iii

1 Introduction ................................................................ 1
  1.1 Background .......................................................... 1
  1.2 Contributions ...................................................... 2

2 Related Work .............................................................. 4

3 eDB Model ................................................................ 6
  3.1 Generic Assumptions ................................................ 6
  3.2 Adversary Model ..................................................... 6
  3.3 Security Model ...................................................... 7
  3.4 Encryption Schemes ................................................ 8
    3.4.1 OPE .............................................................. 8
    3.4.2 ADD ............................................................ 9
    3.4.3 MULT .......................................................... 10
    3.4.4 SEARCH .................................................... 10

4 eDB Design and Analysis ............................................. 12
  4.1 PhantomDB Tools .................................................. 12
    4.1.1 Arithmetic Engine ........................................... 12
    4.1.2 Round Communication .................................... 13
    4.1.3 Hybrid Storage Model ..................................... 13
  4.2 Security Considerations .......................................... 14
Chapter 1

Introduction

1.1 Background

With the emergence of cloud computing in various domains, database systems are not left behind and a paradigm Database-as-a-service (DaaS) has been rigorously explored [14, 15, 19, 22, 23]. Under the DaaS model, the database services are provided to customers over the Internet by the database service provider (DSP). There are many existing commercial vendors in market who are providing cloud services in DaaS model like Amazon Relational Database Service (MySQL) [1], Microsoft SQL Azure (MS SQL) [2] etc. The various benefits provided in DaaS model are as below:

- Client doesn’t have to incur expenses for hardware and software.
- The collective usage of machinery cohesively by many clients brings the total worth for the money spent while purchasing them.
- A lot of manpower is saved as in case of traditional database systems every client would have to hire DBA’s separately for each location where as in DaaS model it is avoided by hiring DBA only at central DSP location.
- Client pay according to their necessity i.e. pay per usage.

Apart from all the benefits of DaaS model there have been many challenges related to the model. One of the main challenges is to provide data security. As in current scenario client has to share all its data with DSP who might not be trust worthy. For many companies like banking, telecom services etc data is considered as their most valuable asset which they refrain from sharing with other party. Due to this lack of trust on DSP they avoid the movement from traditional self-contained database model towards DaaS model.
To overcome the issue of lack of trust the basic solution is to store the data at server after applying some encryption scheme. But by doing so database will lose power of executing many queries. Like for example, if we apply AES [11] encryption over data and store the cipher-text in the database then we will lose the power of performing range queries over the data.

To avoid this, one approach is to use Fully Homomorphic encryption scheme [13]. This scheme allows evaluation of arbitrary functions on cipher text without decrypting them. So for database encrypted using this encryption scheme one can execute any valid SQL query over the database to get valid results in encrypted format which are then decrypted by client. But the problem with this scheme is that arbitrary function evaluation is orders of magnitude time slower \( (10^9) \) as compared to evaluation in plain text domain.

The second approach is to treat DSP servers as data storage units where data is stored after applying some encryption scheme. And to execute some query over the data, client needs to download whole database, decrypt it and then perform query execution over it. But the very idea of downloading all the data at client side and performing query execution negates the advantage of using DaaS.

The third approach is to encrypt same data using multiple partial homomorphic encryption schemes such that each encryption scheme is compatible for handling a specific kind of database query operation. So in this approach multiple encryption values of same data will co-exist at the DSP server. eDB works under this approach using multiple encryption schemes side by side on the same data.

Many constructions have been proposed which vary in usage of above approaches. NetDB2 [14, 15] construction involves mapping of encrypted data on the server to partition buckets. When a query is made the relevant buckets are transferred to the client which are decrypted and further filtered out to generate final result. CryptDB [19] and MONOMI [22] are other constructions which use multiple encryption schemes on the same data where each scheme used has compatibility for specific SQL operation. SDB [23] is recent work which provide set of interoperable operators with help of asymmetric secret sharing scheme. eDB is a model closely related to CryptDB and MONOMI, where both of them did not formally define any security model and the adversary bound for their respective system, we first define the security model and then give an adversary bound in the defined model for our system.

### 1.2 Contributions

The work done under eDB is the extension to the work done in PhantomDB [16]. In this thesis we have worked towards proving security of the system in multiple encryption scheme scenario. We will be proving security of textual and numerical data in Indistinguishability
under Ordered Chosen Plain text Attack (IND-OCPA) model [7] (definition given in Section 3.3) when encrypted in accordance with eDB Framework.

PhantomDB did not consider to handle query processing for floating point numbers. In eDB we have developed an idea of splitting the exponential and mantissa part of floating point number into separate columns and then worked out to show how to execute the queries involving these numbers under various scenarios. In eDB we also present a new storage model *Encryption Dictionary* which is storage efficient providing a trade off where it improves query execution significantly on some of the queries and performs somewhat poorly on some others.

**Organization:** The rest of the thesis is organized as follows. Chapter 2 gives the glimpse of the work that already has been done for solving secure DaaS framework. Chapter 3 describes the basic model assumption, security model and encryption schemes that are used. Chapter 4 describes about tools that are used and the work done in proving the security of eDB. In Chapter 5 we presented the results of the experiment done over eDB. Finally in Chapters 6 and 7 we conclude and portray the future work.
Chapter 2

Related Work

There has been considerable research [14, 15, 19, 22, 23] in the area looking for a solution to provide secure DaaS environment. NetDB2 [14, 15] approach divided the domain of columns into partitions, assigning each partition a unique id and then encrypting those partitions separately and storing the same along with id’s at server. When a query comes against this system, server chooses the partitions satisfying the query and transfers them back to client who further perform filtering of results at his end to evaluate the final result. The main drawback of this approach is that it requires database system operations to perform result evaluation and storage of partition at clients end which nullifies the very advantage of DaaS model where client doesn’t perform query processing and storage of data at its site.

CryptDB [19] used multiple encryption schemes to encrypt same data and thereafter it stored the encrypted data in a onion model. An onion consists of multi-layered encryption to provide different levels of security. For example a data $m$ in onion environment is encrypted as $E_1(E_2(E_3(m)))$ where $E_1, E_2, E_3$ are encryption schemes with $E_1$ being the strongest and $E_3$ being the weakest of three encryption schemes. The argument of security was if there doesn’t come any query for operation on a column then the security of the column is maintained at the highest that is in our case of encryption $E_1$. And subsequently if some query comes then security of column is lowered to the encryption to which the onion layer have been removed i.e to the layer of encryption which can perform the operation required by the query. In their setting there is no way of bringing back the security to the higher level once peeling of onion layer (strong encryption) is done. Also this peeling process slows down the query execution the first time it is run.

MONOMI [22] is based on CryptDB using split client/server query execution approach where parts of query plan that can be executed over server are executed over encrypted data and rest of the plan where client input is required, is executed by client machine. Many
optimization techniques were also introduced like precomputation, space efficient encryption and grouped homomorphic encryption. Precomputation in real time is not feasible as database designer can’t guess what kind of computations are required by the queries. Both CryptDB and MONOMI worked in multi encryption setting and both didn’t layout arguments on security aspect of the system in multi encryption setting.

**TrustedDB** [5] and **Cipherbase** [4] provides a hardware approach on providing the solution to secure DaaS problem. Both assume the existence of secure co-processor at the server end which can decrypt the data at server and perform query execution over the same in secure co-processor environment. However such co-processor units are expensive. Our work focuses in scenarios where nothing is trusted at server side.

**SDB** [23] performed query processing with set of secure, data-interoperable operators by using asymmetric secret-sharing scheme. They provided protocols to handle queries having across the column computations like for columns A, B, C to compute the expression \((A-B)\times C\) on server. They used costly key update operation which requires whole column modification. However when aggregate queries are to be executed there protocol updates target column values to encrypted values of form \(m_z \times v\) where \(v\) is plain text value and \(m_z\) is column key which can reveal high information about values in term of their ratio. For experiment purposes they assumed that few of the columns of TPCH database are not sensitive and hence they kept them in plaintext which resulted to better results.
Chapter 3

eDB Model

3.1 Generic Assumptions

Various assumptions [16] that are made in eDB are as follows:

1. Server is not trusted. Hence client does not share its secret encryption keys with server.

2. Client is fully trusted and won’t be compromised. If we remove this assumption then security can never be guaranteed since adversary can compromise client and see all the data (client has all the encryption keys).

3. The communication channel between client and server is secure. This can be ensured by using various techniques such as TLS (Transport Layer Security) and SSL (Secure Sockets Layer).

4. All the encryption schemes used by eDB are individually secure.

5. We have large disk space at the server. This assumption is practical since disk is very cheap now-a-days.

6. Client always expect the valid, exact tuples in response to the made query. He doesn’t perform any kind of result filtering at his location.

3.2 Adversary Model

Adversary Model used by eDB is “Honest But Curious” [16] model. Various assumptions of this model are:

- Server does not tamper with the database engine. Hence database engine works correctly as is expected of it. It does not give any false tuples or incomplete answers for any query made to it.
• Server has access to all the encrypted data which client stores in the database.

• To give adversary some power we assume server knows which encryption schemes client has used to encrypt the data but does not know which encryption scheme has been used to encrypt any particular column (server may try to guess it by seeing the length of column values).

• Server does not have any domain knowledge about the data being stored by client.

• Server cannot launch any active attack on the encryption scheme. He can only use brute force attack (or any passive attack specific to an encryption scheme) to break the encryption scheme.

• The goal of adversary is to determine what plain text data client has stored in the database.

We choose this adversary model for eDB because it is a very practical model in which very limited trust is put on the server.

### 3.3 Security Model

The ideal security model that should be targeted by a system looking for data security is IND-CPA (indistinguishability under chosen plaintext attack explained in section 3.4.2) but to leverage the advantages of indexing in DBMS the order information is required to be revealed to server by encrypted data. Thus after revealing the order information to server IND-CPA guarantee can not hold, and the ideal security guarantee that should be targeted then is IND-OCPA. The IND-OCPA term was first refereed in [7] in context with security related to OPE scheme(Order Preserving Encryption scheme discussed in section 3.4.1). The IND-OCPA security model for an encryption scheme $E$ can be defined by the game between challenger $C$ and adversary $A$ as below:

1. $C$ generates a secret key sk s.t sk = KeyGen(k) and choses a random $b \in \{0, 1\}$. Here KeyGen is key generation algorithm of encryption scheme $E$.

2. Polynomial number of adaptive challenge rounds occur between $A$ and $C$ as below:
   
   In round $i$
   
   • $A$ gives $C$ two data entries $d_i^0$ and $d_i^1$
   • $C$ sends back the encryption $E(d_i^b)$ of data entry $d_i^b$ to $A$. 

7
Here polynomial is considered w.r.t. security parameter \( k \). And the sequence of \( d_i^0, d_i^1 \) over the polynomial rounds \( i \) should satisfy the property such that iff \( d_i^0 < d_j^0 \) then \( d_i^1 < d_j^1 \) for any two rounds \( i, j \).

3. \( A \) outputs his guess \( b' \)

\( A \) wins the game if \( b = b' \). And a scheme is \((t, \epsilon)\) IND-OCPA secure if there doesn’t exist an adversary \( A \) with running time at most \( t \) and advantage atleast \( \epsilon \) who wins the above game. Advantage of \( A \) is defined as below:

\[
Adv_A = |Pr[b = b'] - \frac{1}{2}|
\]

### 3.4 Encryption Schemes

eDB uses multiple encryption schemes so as to support various SQL operations. The four basic operations that are mostly used in SQL are aggregation, multiplication, equality checking and order comparison. Till now there doesn’t exist a single practical encryption scheme that supports all these operations. So we used multiple encryption schemes to support the various operations of SQL. These schemes are discussed below:

#### 3.4.1 OPE

OPE stands for Order Preserving Encryption scheme. It is a deterministic encryption scheme. For a given order on some data values the encryption of those values under OPE scheme maintains the same order as that of original data. For example, consider two data entries \( x, y \) such that \( x < y \) then the encryption of \( x, y \) under OPE will maintain the same order i.e \( E_{OPE}(x) < E_{OPE}(y) \). This kind of scheme will be useful in execution of range queries over encrypted database like the one given below.

\[
\text{select name from account where salary} > 100000
\]

The ideal security model targeted by any OPE scheme is IND-OCPA (defined in Section 3.3). Initial proposal for the scheme was made by Boldyreva et.al [7, 8]. There they have proved that it is impossible to obtain an ideal IND-OCPA security for OPE encryption scheme unless the size of cipher-text space is exponential in size of plain-text space. Subsequently in [18] it was proven that IND-OCPA guarantee is possible under the assumption of mutability of cipher text space. They also provided a new OPE scheme called mOPE (mutable OPE) which satisfied the IND-OCPA guarantee with a condition that on OPE encryption of new data value, existing some (log scale) of the cipher values for other data items are changed.
eDB uses mOPE which is till date the strongest known OPE scheme.

3.4.2 ADD

ADD stands for ADDition homomorphic encryption scheme. Encrypting the values under this scheme will provide a method to compute the encryption of summation of these values directly from cipher values and public key of the encryption scheme. Like for example we are given encryption of two values under ADD $x, y$ as $E(x)$ and $E(y)$. Then there exist some method that computes $E(x + y)$.

i.e. $E(x + y) = E(x) *_k E(y)$

where $*_k$ is some binary operation under public key $k$. In eDB we used Paillier Encryption scheme[17] as ADD. Paillier encryption scheme is a non deterministic encryption scheme. In a non deterministic encryption scheme, if we encrypt the same number again then the probability of getting the same cipher value is negligible, which implies that server can’t perform equality checks, less than and greater than operations using only non deterministic scheme. It is due to the involvement of randomness factor while encryption of values. The binary operation involved in this scheme is multiplication. This scheme is useful in queries involving aggregation operations. For example:

```
select sum(acc_bal) from account
```

Paillier encryption is secure under IND-CPA security model. IND-CPA security model for public key encryption scheme $\mathcal{E}$ is defined by the game between challenger $C$ and adversary $A$ as below:

1. $C$ generates a secret key $sk$ and public key $pk$ s.t $(sk, pk) = \text{KeyGen}(k)$ and shares $pk$ with $A$. Here KeyGen is key generation algorithm of encryption scheme $\mathcal{E}$.

2. $A$ gives $C$ two challenge messages $m_0$ and $m_1$

3. $C$ sends back the encryption $\mathcal{E}(m_b)$ of message $m_b$ to $A$ where $b \epsilon \{0,1\}$.

4. $A$ outputs his guess $b'$

$A$ wins the game if $b = b'$. And a scheme is $(t, \epsilon)$ IND-CPA secure if there doesn’t exist an adversary $A$ with running time at most $t$ and advantage atleast $\epsilon$ who wins the above game. Advantage of $A$ is defined as below:

$$Adv_A = \left| Pr[b = b'] - \frac{1}{2} \right|$$
3.4.3 MULT

MULT stands for MULTiplicative homomorphic encryption scheme. As ADD, MULT also provide method for computing multiplication from the cipher values. For example, consider two values $x, y$ encrypted under MULT scheme as $E(x)$ and $E(y)$. Then there exist some method that computes $E(x \cdot y)$.

\[ \text{i.e. } E(x \cdot y) = E(x) \cdot_k E(y) \]

where $\cdot_k$ is some binary operation under public key $k$. There exist many schemes that provide MULT scheme requirement like RSA encryption scheme [20] and variant of IBE scheme given by Boneh and Franklin [10]. In eDB we used IND-CPA secure Elgamal Encryption scheme [12] as MULT. The binary operation involved in this scheme is multiplication. This scheme is useful in queries involving multiplication of values. Eg:

select (cost * discount) as netsaving from parts

As ADD, MULT used in eDB is also non deterministic and secure under IND-CPA security model.

3.4.4 SEARCH

SEARCH encryption scheme is scheme using which we can perform the search operation of words on encrypted data when given some trapdoors for those words. This encryption scheme allows only full word searches on encrypted data. In asymmetric setting there has been a scheme proposed by Boneh et al [9] which uses bilinear maps for searching words when given a trapdoor for those words. As the scheme involves the costly computation of bilinear mapping operations, we use the scheme proposed by Song et al [21]. This scheme encrypts the textual data using a sequence generated with the help of a secure pseudo random generator and secure pseudo random function. This scheme allows only full word search on the encrypted data. For example if the phrase “I’m adaptable” is encrypted then we can search the word “adaptable” but not “table” in the same phrase.

The security of SEARCH scheme is defined in terms of the crypto tools (Pseudo random generator $\mathcal{G}$ and Pseudo random Function $\mathcal{F}$) used underneath to encrypt the textual data. Basically a new Pseudo random sequence $\mathcal{T}$ is generated using these tools. It has been proven in [21] that iff the cryptographic tools $\mathcal{G}$ and $\mathcal{F}$ used to generate $\mathcal{T}$ are secure then the Pseudo random sequence $\mathcal{T}$ is also secure and thus the encrypted cipher text is semantically secure. A security definition for Pseudo random sequence is as below:
A pseudo random sequence $T$ is said to be $(t, \epsilon)$ secure if there doesn’t exist an algorithm $A$ which takes at most $t$ time and distinguishes $T$ from truly random sequence $R$ with advantage more than $\epsilon$. The advantage of $A$ is defined as below:

$$Adv_A = |Pr[A(T) = 1] - Pr[A(R) = 1]|$$
Chapter 4

eDB Design and Analysis

Like CryptDB [19], MONOMI [22], PhantomDB [16] eDB also uses multiple encryption schemes on single data item where each encryption scheme supports some special characteristic to perform some special database operation. It uses OPE, MULT, ADD and SEARCH encryption schemes (discussed in Section 3.4). It encrypts numeric data with OPE, MULT and ADD and textual data with OPE and SEARCH. SQL query coming from end user is converted to map the schema definition for database that is stored at DSP servers, and all the constant terms in the clauses are encrypted using encryption scheme that are compatible with the operation being performed in that clause. There were many tools that were present in PhantomDB [16] and are used in eDB which are described in the subsequent subsection.

4.1 PhantomDB Tools

4.1.1 Arithmetic Engine

To perform the arithmetic operations like (/,*,+,- etc) that cannot be computed at server due to reason like underlying encryption schemes on data doesn’t support these operations, the client system is given powerful Arithmetic Engine to perform these operations. Like for example consider a query:

\[
select \text{avg(price)} \text{ from part}
\]

where we want to compute the average value of price from part table. Here we need to perform division on Sum(Price) and Count where Sum is encrypted under ADD scheme, and count is a integer and there is no way of performing division between them. So in this case server will transfer the data, Sum as ADD cipher and Count as integer to Client who will decrypt the Sum and will perform the division using the Arithmetic Engine at its side.
4.1.2 Round Communication

Giving support of Arithmetic engine alone is not enough. Consider for example the query:

\[
\text{select } i1.\text{name from item } i1 \text{ where } i1.\text{price } > (\text{select } \text{avg}(i2.\text{price}) \text{ from item } i2 \text{ where } i2.\text{type } = i1.\text{type})
\]

In this query every tuple extracted from item table has to perform comparison of price with average price of same type. Comparison is performed using OPE scheme, and average is calculated using Arithmetic Engine at client side. So to perform comparison operation, input (OPE encryption of average) from client side is required which is done by giving server/client a power of round communication.

The Architecture of PhantomDB is shown in the figure below:

![Figure 4.1: Architecture [16]](image)

4.1.3 Hybrid Storage Model

A concept of using hybrid storage model for storing encrypted data was introduced in PhantomDB. Horizontal and Vertical storage models store table data in form of rows and columns respectively on disk blocks of hard disk. In hybrid model, mixture of both techniques is used. Data encrypted under OPE and SEARCH schemes is stored using Horizontal storage model and data encrypted under MULT, ADD in vertical storage model.
4.2 Security Considerations

The interesting question to address is security of the system when same data is stored in different columns after encrypting it with different encryption schemes. Usually the encryption schemes are based on some underlying hard problem. And a security argument for that scheme models around the assumption that if we break the encryption scheme then we are able to solve the same underlying hard problem.

In our context, we are encrypting textual data using SEARCH and OPE encryption schemes, and numeric data with ADD, MULT, OPE schemes. Therefore the security argument for our system will involve proving the security of two models:

- Proving the security of system involving encryption of textual data with OPE and SEARCH scheme.
- Proving the security of system involving encryption of numeric data with ADD, MULT and OPE scheme.

4.2.1 Security of Textual data

As mentioned earlier textual data is encrypted under OPE and SEARCH encryption schemes. And for both the encryption schemes the security notions are different as discussed in Section 3.4.

For security analysis let us assume eDB uses:

- \((t_1, \epsilon_1)\) secure pseudo random sequence \(T\) for encrypting data under SEARCH encryption scheme.
- \((t_2, \epsilon_2)\) IND-OCPA secure OPE encryption scheme.

In eDB the cipher text from these two encryption scheme exist side by side. Therefore a given word \(w\) the underlying security argument will involve two element \(X\) and \(Y\) where \(X\) is sequence of bits from pseudo random sequence \(T\) and \(Y\) is the respective OPE of \(w\).

**Theorem 1:** If \(T\) is \((t_1, \epsilon_1)\) secure pseudo random sequence used in SEARCH scheme and underlying OPE scheme is \((t_2, \epsilon_2)\) IND-OCPA secure then the underlying system \(S\) which uses these encryption schemes side by side in multiple encryption setting is \((t_3, \epsilon_3)\) IND-OCPA secure where \(\epsilon_3 = 2\epsilon_1 + \epsilon_2\) and \(t_3 \leq Min(t_1, t_2)\).
Proof: Before going into the proof let us define two sequences as follow:

\[ \alpha = \langle X_1, X_2, ..., X_n \rangle \]
\[ \beta = \langle U_1, U_2, ..., U_n \rangle \]

Here sequence \( \alpha, \beta \) represents the sequence of bits generated by pseudo random sequence \( \mathcal{I} \) and truly random sequence \( \mathcal{R} \) respectively used to encrypt the sequence of words \( \langle W_1, W_2, ..., W_n \rangle \) under SEARCH encryption scheme.

Now we define two games:

Game 1: This game consists of two players, challenger \( C_1 \) and adversary \( A_1 \). \( C_1 \) takes an instance of sequence \( \alpha \) i.e. \( \alpha_1 \) and the instance of OPE scheme. Then \( A_1 \) starts playing IND-OCPA game with \( C_1 \) as defined in Section 3.3. The data entries \( d_i \) defined in the definition (Section 3.3) are the sequence of words \( W_i \) to be encrypted in this particular game instance and the encryption value returned after each round by \( C_1 \) consist of two encryption values one encryption of word \( W_i^b \) using sequence \( \alpha \) and other encryption using OPE scheme. Let us limit the IND-OCPA advantage of \( A_1 \) by \( e_1 \) as below:

\[ Adv_{A_1} < e_1 \]  
(4.1)

Game 2: In similar fashion to Game 1 let us define the Game 2 which is played between challenger \( C_2 \) and adversary \( A_2 \). \( C_2 \) takes an instance of sequence \( \beta \) i.e \( \beta_1 \) and the instance of OPE scheme. Then \( A_2 \) starts playing similar kind of IND-OCPA game as played in Game 1 with \( C_2 \) with here now \( C_2 \) using sequence \( \beta \) instead of \( \alpha \) for encrypting the word \( W_i^b \). Let us limit the IND-OCPA advantage of \( A_2 \) by \( e_2 \) as below:

\[ Adv_{A_2} < e_2 \]  
(4.2)

Claim: Assuming \( e_1 > e_2 \) we say

\[ e_1 - e_2 < 2\epsilon_1 \]  
(4.3)

Proof of Claim: If \( e_1 - e_2 > 2\epsilon_1 \) then we can construct the adversary \( A_3 \) who can distinguish between sequence \( \alpha \) and \( \beta \) with non negligible advantage (\( > \epsilon_1 \)) as against the assumption on \( \mathcal{I} \) as below:

\( A_3 \) will choose an instance of OPE scheme and will start playing game with the adversary of Game 1 or Game 2 as depending upon the sequence given at his hand (\( \alpha \) or \( \beta \)). \( A_3 \) will output sequence \( \alpha \) if respective adversary of Game 1 or 2 wins the game and \( \beta \) if he lose the game.
Let $I$ denote the event that sequence $\alpha$ is given to $A_3$.
Let $W_1$ be event in game 1 when $A_1$ wins Game 1.
Let $W_2$ be event in game 2 when $A_2$ wins Game 2.
As defined in Section 3.4.4 the advantage of $A_3$ is as follow:

$$Adv_{A_3} = |Pr[A_3(I) = 1] - Pr[A_3(\bar{I}) = 1]|$$

$$= |Pr[I], Pr[W_1|I] - Pr[\bar{I}], Pr[W_2|\bar{I}]|$$

Assuming $Pr[I] = Pr[\bar{I}] = 1/2$ we get

$$Adv_{A_3} = 1/2(1/2 + e_1) - 1/2(1/2 + e_2)$$

$$= 1/2(e_1 - e_2)$$

$$> \epsilon_1$$

Which is against the assumption and hence proved the claim by contradiction.

Also in Game 2 since we are using truly random sequence the advantage gain by adversary $A_2$ w.r.t. the adversary against the OPE scheme should be zero as truly random sequence acts as one time pad where the knowledge gain is zero. Therefore,

$$Adv_{A_2} < \epsilon_2$$ (4.4)

from (1),(2),(3),(4)

$$Adv_{A_1} < 2\epsilon_1 + \epsilon_2$$

Hence proved the overall security of textual data.

4.2.2 Security of Numeric Data

Numerical data is encrypted under ADD, MULT and OPE encryption schemes. Their security notions have already been discussed in the previous sections (Section 3.3, 3.4).

For security analysis let us assume eDB uses:

- $(t_1, \epsilon_1)$ IND-OCPA secure OPE encryption scheme.
- $(t_2, \epsilon_2)$ IND-CPA secure ADD encryption scheme.
- $(t_3, \epsilon_3)$ IND-CPA secure MULT encryption scheme.

The proof outline for numeric data is as follows in two parts:
• First we lay down the argument on security of System where IND-CPA ADD and IND-CPA MULT encryption co-exist side by side.

• Then we comment on the security of system where one encryption is IND-CPA and the other co-existing encryption is IND-OCPA OPE.

### Part 1

**Theorem 2:** If $E_1$ is $(t_1, \epsilon_1)$ IND-CPA secure ADD encryption scheme and $E_2$ is $(t_2, \epsilon_2)$ IND-CPA secure MULT encryption scheme then the underlying system $S$ which uses these encryption schemes side by side in multiple encryption setting is $(t_3, \epsilon_3)$ IND-CPA secure where $\epsilon_3 = \epsilon_1 + \epsilon_2$ and $t_3 \leq \text{Min}(t_1, t_2)$.

**Proof:** We prove the theorem by contradiction with the assumption of some adversary say $A_3$ against challenger $C_3$ of system $S$ who breaks the IND-CPA claim of $S$ with some non negligible advantage $> \epsilon_1 + \epsilon_2$. With this adversary at hand we will break the IND-CPA claim of either $E_1$ or $E_2$ which would be against the assumption of $E_1$ or $E_2$.

**IND-CPA game between $A_3$ and $C_3$**

1. $C_3$ gets the instances of encryption schemes $E_1$ and $E_2$, runs the keyGen algorithm to generate pk, sk of both the schemes and then shares pk of both the schemes with $A_3$.

2. $A_3$ gives $C_3$ two numerical values $v_0$ and $v_1$ as challenge.

3. $C_3$ sends back two encryption values $E_1(v_b), E_2(v_b)$ to $A_3$ where $b \in \{0, 1\}$

4. $A_3$ returns his guess $b_3$

$A_3$ wins the game if $b_1 = b_3$ and according to assumption.

$$Adv_{A_3} = |Pr[b_1 = b_3] - \frac{1}{2}| > \epsilon_1 + \epsilon_2$$  \hspace{1cm} (4.5)

Now given this $A_3$ we construct new adversary $A_1$ against challenger $C_1$ who tries to break IND-CPA assumption on $E_1$.

**Construction of $A_1$**

1. $C_1$ sends pk of $E_1$ to $A_1$. $A_1$ chooses the instance of $E_2$ and runs its keyGen to get its sk.pk of $E_1$. $A_1$ then sends pk of both $E_1$ and $E_2$ to $A_3$. 
2. $A_3$ sends two numerical values $v_0$ and $v_1$ to $A_1$ which are further forwarded by $A_1$ to $C_1$.

3. $C_1$ sends $E_1(v_{b_1})$ to $A_1$ where $b_1 \in \{0, 1\}$. $A_1$ chooses a random $b_2 \in \{0, 1\}$ and sends $E_1(v_{b_1})$ and $E_2(v_{b_2})$ to $A_3$.

4. $A_3$ sends his guess $b_3$ to $A_1$ who further sends it to $C_1$.

$A_1$ wins the game if $b_3 = b_1$ and advantage of $A_1$ is defined as below.

$$Adv_{A_1} = |Pr[b_3 = b_1] - \frac{1}{2}|$$

**Claim:**

$Pr[b_3 = b_1 | b_1 \neq b_2] > Pr[b_3 = b_1 | b_1 = b_2] - 2\epsilon_2$

**Proof of Claim:** For simplicity let us assume $Pr[b_3 = b_1 | b_1 = b_2] = p$

If claim is not true i.e $Pr[b_3 = b_1 | b_1 \neq b_2] < p - 2\epsilon_2$ then we can construct adversary $A_2$ who can win IND-CPA game in $E_2$ with non negligible advantage as shown below:

$$Adv_{A_2} = |Pr[b_3 = b_2] - 1/2|$$

$$= |Pr[b_3 = b_2 | b_2 = b_1].Pr[b_2 = b_1] + Pr[b_3 = b_2 | b_2 \neq b_1].Pr[b_2 \neq b_1] - 1/2|$$

$$= |p.1/2 + (1 - Pr[b_3 \neq b_2 | b_2 \neq b_1]).1/2 - 1/2|$$

$$> |p.1/2 + (1 - p + 2\epsilon_2).1/2 - 1/2|$$

$$> \epsilon_2$$

which would be against the assumption on $E_2$ and hence claim is true.

$$Adv_{A_1} = |Pr[b_3 = b_1] - \frac{1}{2}|$$

$$= |Pr[b_3 = b_1 | b_2 = b_1].Pr[b_2 = b_1] + Pr[b_3 = b_1 | b_2 \neq b_1].Pr[b_2 \neq b_1] - \frac{1}{2}|$$

$$> |p.1/2 + (p - 2\epsilon_2).1/2 - 1/2|$$

$$> |p - 1/2 - \epsilon_2|$$

$$> |\epsilon_1 + \epsilon_2 - \epsilon_2|$$

$$> \epsilon_1$$
which is against the assumption on $E_1$ and hence with assumption of existence of non negligible adversary against system $S$ we are able to break IND-CPA guarantee of $E_1$ or $E_2$ which would be against the assumption. Hence the theorem 2 is correct.

**Part 2**

From Theorem 2 we can portray our system to a system where encryption of same data encrypted with IND-OCPA secure OPE encryption scheme $E_1$ and IND-CPA secure encryption scheme $E_2$ coexist.

By comparing the definition of IND-OCPA and IND-CPA we can see that in IND-OCPA model definition there are multiple challenge rounds between adversary and challenger where as in IND-CPA definition there is only one challenge round. For our further discussion on security we need both the definitions to be same in terms of challenge rounds. For the same we are going to use concepts mentioned in [6] where they introduced a new security definition model IND-nCPA and related the same to IND-CPA model. Following we define IND-nCPA security model and a security lemma derived from [6].

**IND-nCPA**

IND-nCPA security model for public key encryption scheme $E$ is defined by the game between challenger $C$ and adversary $A$ as below :

1. $C$ generates a secret key $sk$ and public key $pk$ s.t $(sk,pk) = KeyGen(k)$ and shares $pk$ with $A$. $C$ also choses a random $b \in \{0,1\}$

2. $A$ gives $C$ two challenge messages $m_0$ and $m_1$

3. $C$ sends back the encryption $E(m_b)$ of message $m_b$ to $A$.

4. Step 2, 3 are repeated multiple polynomial number of times.

5. $A$ outputs his guess $b'$

$A$ wins the game if $b = b'$. And a scheme is $(t, \epsilon)$ IND-nCPA secure if there doesn’t exist an adversary $A$ with running time at most $t$ and advantage atleast $\epsilon$ who wins the above game. Advantage of $A$ is defined as below:

$$Adv_A = |Pr[b = b'] - \frac{1}{2}|$$

**Lemma 1:** If public key encryption $E$ is $(t,\epsilon)$ IND-CPA secure then the same scheme is $(t',n\epsilon)$ IND-nCPA secure in multiuser encryption setting.
i.e \( \text{Adv}_{\text{IND−nCPA}} < n\text{Adv}_{\text{IND−CPA}} \)

**Theorem 3:** If \( \mathcal{E}_1 \) is \((t_1, \epsilon_1)\) IND-CPA secure OPE encryption scheme and \( \mathcal{E}_2 \) is \((t_2, \epsilon_2)\) IND-nCPA secure encryption scheme then the underlying system \( S \) which uses these encryption schemes side by side in multiple encryption setting is \((t_3, \epsilon_3)\) IND-OCpairA secure where 
\[
\epsilon_3 = \epsilon_1 + \epsilon_2 \quad \text{and} \quad t_3 \leq \min(t_1, t_2).
\]

**Proof:** The flow of proof is similar to proof of Theorem 1. We prove the theorem by contradiction with the assumption of some adversary say \( A_3 \) against challenger \( C_3 \) of system \( S \) who breaks the IND-OCpairA claim of \( S \) with some non negligible advantage \( > \epsilon_1 + \epsilon_2 \). With this adversary at hand we will break the IND-OCpairA claim of either \( \mathcal{E}_1 \) or IND-nCPA claim of \( \mathcal{E}_2 \) which would be against the assumption of \( \mathcal{E}_1 \) or \( \mathcal{E}_2 \).

**IND-OCpairA game between \( A_3 \) and \( C_3 \)**

The game proceed in similar fashion as described in Section 3.3 with only difference that \( A_3 \) will get two encryption values \( \mathcal{E}_1(v_{i}^{b_1}), \mathcal{E}_2(v_{i}^{b_1}) \) in each challenge round of challenge phase from \( C_3 \) where \( b_1 \epsilon_r \{0, 1\} \). Assuming the guess made by \( A_3 \) be \( b_3 \) then we say \( A_3 \) wins the game if 
\[
b_1 = b_3 \quad \text{and according to assumption.}
\]

\[
\text{Adv}_{A_3} = |Pr[b_1 = b_3] - \frac{1}{2}| > \epsilon_1 + \epsilon_2 \quad (4.6)
\]

Now given this \( A_3 \) we construct new adversary \( A_1 \) against challenger \( C_1 \) who tries to break IND-OCpairA assumption on \( \mathcal{E}_1 \).

**Construction of \( A_1 \)**

1. \( A_1 \) chooses the instance of \( \mathcal{E}_2 \) and runs its keyGen to get its sk, pk of \( \mathcal{E}_1 \). \( A_1 \) then sends pk of \( \mathcal{E}_2 \) to \( A_3 \). \( C_1 \) choses \( b_1 \epsilon_r \{0, 1\} \). \( A_1 \) chooses a \( b_2 \epsilon \{0, 1\} \).

2. \( A_3 \) sends two numerical values \( v_{i}^{0} \) and \( v_{i}^{1} \) to \( A_1 \) which are further forwarded by \( A_1 \) to \( C_1 \).

3. \( C_1 \) sends \( \mathcal{E}_1(v_{i}^{b_1}) \) to \( A_1 \). Further \( A_1 \) sends \( \mathcal{E}_1(v_{i}^{b_1}) \) and \( \mathcal{E}_2(v_{i}^{b_2}) \) to \( A_3 \).

4. step 2, 3 are repeated polynomial number of times where value in each round is represented as \( v_{i}^{0}, v_{i}^{1} \) with condition for any \( i, j \) iff \( v_{i}^{0} < v_{j}^{0} \) then \( v_{i}^{1} < v_{j}^{1} \).

5. \( A_3 \) sends his guess \( b_3 \) to \( A_1 \) who further sends it \( C_1 \).
$A_1$ wins the game if $b_3 = b_1$ and advantage of $A_1$ is defined as below.

$$Adv_{A_1} = |Pr[b_3 = b_1] - \frac{1}{2}|$$

**Claim:**

$Pr[b_3 = b_1 | b_1 \neq b_2] > Pr[b_3 = b_1 | b_1 = b_2] - 2\epsilon_2$

**Proof of Claim:** For simplicity let us assume $Pr[b_3 = b_1 | b_1 = b_2] = p$

If claim is not true i.e $Pr[b_3 = b_1 | b_1 \neq b_2] < p - 2\epsilon_2$ then we can construct adversary $A_2$ who can win IND-nCPA game in $E_2$ in same setting with non negligible advantage as shown below:

$$Adv_{A_2} = |Pr[b_3 = b_2] - 1/2|$$

$$= |Pr[b_3 = b_2 | b_2 = b_1].Pr[b_2 = b_1] + Pr[b_3 = b_2 | b_2 \neq b_1].Pr[b_2 \neq b_1] - 1/2|$$

$$= |p.1/2 + (1 - Pr[b_3 \neq b_2 | b_2 \neq b_1]).1/2 - 1/2|$$

$$>|p.1/2 + (1 - p + 2\epsilon_2).1/2 - 1/2|$$

$$>|\epsilon_2|$$

which would be against the assumption on $E_2$ and hence claim is true.

$$Adv_{A_1} = |Pr[b_3 = b_1] - \frac{1}{2}|$$

$$= |Pr[b_3 = b_1 | b_2 = b_1].Pr[b_2 = b_1] + Pr[b_3 = b_1 | b_2 \neq b_1].Pr[b_2 \neq b_1] - \frac{1}{2}|$$

$$>|p.1/2 + (p - 2\epsilon_2).1/2 - 1/2|$$

$$>|p - 1/2 - \epsilon_2|$$

$$>|\epsilon_1 + \epsilon_2 - \epsilon_2|$$

$$>|\epsilon_1|$$

which is against the assumption on $E_1$ and hence with assumption of existence of non negligible adversary against system $S$ we are able to break IND-OCPA guarantee of $E_1$ or IND-nCPA guarantee $E_2$ which would be against the assumption. Hence the theorem 3 is correct.

Using Lemma 1 and Theorem 3 we can say that in multi-encryption setting where one encryption scheme is IND-CPA secure and the other scheme is IND-OCPA secure then overall
system is IND-OCPA secure.

Using Part 1 and Part 2 we can say that numerical data encryption in eDB is IND-OCPA secure.

4.3 eDB Tools

Few extensions have been made in eDB w.r.t. PhantomDB model as below:

4.3.1 Floating Point Support

Till now no model has considered on handling query processing for floating point numbers. In current work we have developed a scheme to handle non negative floating point numbers. Floating point numbers can be represented in the form of mantissa and exponent as below:

\[ M \times 10^E \]

Here M and E are integers representing mantissa and exponent parts of the floating point number. In our scheme before encrypting M and E we change the floating point number to the representation where the number of digits in M are same throughout all numbers. For example numbers 1.234, 10, 15.2 , 0.05 are brought to representation as \((M=1234 \ E=-3), (M=1000 \ E=-2), (M=1520 \ E=-1), (M=5000 \ E=-5)\) respectively. However 0 would not go by this representation and therefore it can be represented after giving it a small calibration like \((M=1000 \ E=-23)\).

Thereafter we encrypt the mantissa part on server using OPE, MULT and ADD schemes and exponent part under OPE and ADD scheme. Queries related to floating points can be processed in the following ways:

1. **Order predicate:** Two number N1 and N2 can be compared as following in eDB:

   \[ \text{CompareF} \ (N1,N2) \]

   \[
   \begin{align*}
   &\text{if} \ (N1.E > N2.E) \ \text{then N1 is larger} \\
   &\text{else if} \ (N1.E < N2.E) \ \text{then N2 is larger} \\
   &\text{else compare N1.M and N2.M}
   \end{align*}
   \]

   The queries involving order predicates can be re-written for the execution on the server like below:

   \[
   \text{select name from account where balance} > 1.067 \times 10^{14} \\
   \]

   \[
   \Downarrow
   \]

   Select name from account.E where balance.E.OPE > OPE(11)
UNION
Select name from account.E where balance.E_OPE = OPE(11) and balance.M_OPE > OPE(1067)

2. Multiplication: Two number N1, N2 can be multiplied as below:

MULTF (N1,N2)
Return Value.E = ADD(N1.E,N2.E)

These queries can be re-written as below:

\[
\text{select balance} \times 0.99 \text{ from account} \\
\downarrow \\
\text{select } MULT(\text{balance}_M\_MULT,\text{E}_{MULT}(99)), \\
ADD(\text{balance}_E\_ADD,\text{E}_{ADD}(-2)) \text{ from account}_E
\]

3. Aggregation: Aggregation can be performed in two stages as below:

SUM (Salary)
Stage1: Group Salary according to salary.E_OPE
For each group calculate Sum_E_OPE
Return all Sum_E_OPE and E_OPE
Stage2: Client decrypts all Sum_E_OPE and E_OPE and perform the final summation at his end.

These queries are re-written as below:

\[
\text{select sum(balance) from account} \\
\downarrow \\
\text{select } ADD(\text{balance}_M) \text{ from account}_E \text{ group by balance}_E
\]

Once the result is received by the client, he will further add the numbers having different exponent values.
4.3.2 Encryption Dictionary model

In eDB for numerical data we are using OPE (deterministic), MULT (nondeterministic) and ADD (nondeterministic) encryption schemes. And if multiple instances of same value exist in Plain Database then according to current storage models we have to encrypt that same value multiple times under OPE, ADD and MULT which is storage inefficient and also by doing this we are giving the randomness factor of MULT and ADD to server who can get the information by comparing OPE values. To make it storage efficient we create a separate table (core table) which will store ADD, MULT encryption values w.r.t. different OPE values with a primary index made on OPE value column. And all other tables will store only the OPE encryption values.

It has been seen that this storage model gave a good results for the queries involving only order predicates. The possible reason for the same is that table size of target table is reduced significantly and so the scan on that table takes less time. At the same time queries involving an aggregation function like the one given below takes little more time because of an extra join operation between target table and core table.

\[
\begin{align*}
\text{select } & \text{sum(price) from part} \\
\downarrow \\
\text{select } & \text{ADD(core_ADD) from part_enc,Core where price_OPE = core_OPE}
\end{align*}
\]

The query where there are multiple aggregation function will require even more time as that many number of join operations would be required. For example if the same query wants to find \(\text{sum(price)}, \text{sum(cost)}\) that is two aggregations it need to perform two join operations. The alternate technique that can be performed in these queries is as below:
**Input:** col1,..coln,Enc_target_T,core_T  
**Output:** sum(col1),...,sum(coln)  

**Step 1:**  
select col1_ope,...,coln_ope from Enc_target_T  

**Step 2:**  
Make tables Tab1,..., Tabn on  
col1_ope,..coln_ope respectively.  
Sort the tables Tab1,..., Tabn according  
to values of col1_ope,...,coln_ope  
Make master MINHEAP masterH by  
accessing first row values from Tab1,..., Tabn  

**Step 3:**  
start scanning core_T for Core_OPE values  
while Core_OPE == masterH.min do  
    add Core.ADD to that specific sum(coli)  
    delete masterH.min;  
    masterH.insert(Tabi.next);  
end while  

**Step 4:**  
return sum(col1),...,sum(coln)  

**Algorithm 1:** Aggregation Technique

In step 2, we are making multiple tables on different columns and then sort each table according to coli_ope value. This sorting can be done in parallel threads as operation is performed on separate tables. And as Step 3 involves only one disk scan on the core table core_T the overall execution time would expected to be much lesser comparing to the case where we will require multiple joins which leads to multiple scans.
Chapter 5

Experimental Evaluation

5.1 Set-up

The experimental setup consists of two machines with machine 1 having Intel Core2 Quad Core 3.0 GHz processor, 8 GB memory, 4*300 GB 15000 RPM SAS running Ubuntu Linux 12.04. Both the client and server processes were running on the machine 1. Machine 2 have Dual Quad-core Intel Xeon 2.0GHz , 16 GB memory, 5 * 2TB SATA 7200 RPM hard disk, running with Ubuntu Linux 12.04. Machine 2 was primarily used for storing mOPE encrypted tree which is used to calculate the order preserving encoding for value predicates used in query.

The database system used was “MySQL”. Data generation tool provided by TPCH was used to generate the data base for experiments. For the experiments 1 GB TPCH database was considered as baseline. For a specific TPCH query experiment only those columns of plain TPCH database were considered for encryption which were required in that query execution.

Encrypted database was created using multi threaded programs which were run on 128 core queue of Tyrone Cluster [3].

In mOPE scheme 128 bit AES and 128 bits were used to represent the order preserving encoding. The final state of mOPE tree was created beforehand by accessing all the values from all the tables of plaintext database. Implementation of interactive mOPE for database system is left as future work. For ADD(paillier encryption) and MULT(elgamal encryption) we used 256 bit size group for encryption. Operating System buffer cache as well as database query and table cache was flushed before running each query.

We use primary key definition specified by TPCH. For the encrypted database Order Preserving Encryptions of columns specified as primary key in TPCH were made the primary key. No secondary indexes were created. All the calls for operations on ADD and MULT are replaced by custom UDF’s (User Defined Functions).
5.2 Performance results

Performance evaluation for few of the TPCH queries over eDB is as below:

1. TPCH Q4

As mentioned earlier for testing a specific TPCH query over eDB only those columns were encrypted from plain database which were required for query execution. Here we encrypted the required columns under all encryption schemes as defined earlier.

Table 5.1 shows space usage for database made for Q4. Table 5.2 shows the execution time of Q4 at varying selectivity.

<table>
<thead>
<tr>
<th>Disk Space</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Plaintext Database</td>
<td>190 MB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eDB</td>
<td>6.4 GB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Database Size for Q4

<table>
<thead>
<tr>
<th></th>
<th>0%</th>
<th>25%</th>
<th>50%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain Text DB</td>
<td>8</td>
<td>16</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>eDB</td>
<td>26</td>
<td>225</td>
<td>216</td>
<td>220</td>
</tr>
</tbody>
</table>

Table 5.2: Time taken (in seconds) to execute Q4 at varying selectivity

2. TPCH Q21

Following table shows the database size and execution time of Q21 with standard predicate values.

<table>
<thead>
<tr>
<th>Disk Space</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Plaintext Database</td>
<td>180 MB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eDB</td>
<td>6.2 GB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Database Size for Q21

<table>
<thead>
<tr>
<th>Execution Time</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain Text DB</td>
<td>32</td>
</tr>
<tr>
<td>eDB</td>
<td>133</td>
</tr>
</tbody>
</table>

Table 5.4: Time taken (in seconds) to execute Q21
3. TPCH Q1

This query was used to test the aggregation operation in eDB. The query involves calculating complex expression like \((\text{price} \times (1 - \text{discount}))\) for each row which involved aggregation and multiplication. For the testing purposes we precomputed these kind of expressions and stored the value in separate new column of encrypted database.

Below tables shows the stats for Q1.

<table>
<thead>
<tr>
<th></th>
<th>Disk Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plaintext Database</td>
<td>200 MB</td>
</tr>
<tr>
<td>eDB</td>
<td>8.3 GB</td>
</tr>
</tbody>
</table>

Table 5.5: Database Size for Q1

<table>
<thead>
<tr>
<th></th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain Text DB</td>
<td>18</td>
</tr>
<tr>
<td>eDB</td>
<td>510</td>
</tr>
</tbody>
</table>

Table 5.6: Time taken (in seconds) to execute Q1

The main reason for increase in query execution time in encrypted database is data explosion which can clearly be seen from the results of TPCH Q1. The data base size for Q1 increased 33 times in eDB model and so does the query execution time by 30 times. For the other queries, same can not be said due to the fact that mysql computed intermediate temporary table while executing the query.

5.2.1 Encryption Dictionary evaluation

For TPCH Q4 encryption dictionary model reduced the space from 6.4 GB to 3.2 GB. And the performance result for the same are shown in table below: Aggregation query containing

<table>
<thead>
<tr>
<th></th>
<th>10%</th>
<th>25%</th>
<th>50%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal Storage Model</td>
<td>240</td>
<td>225</td>
<td>216</td>
<td>220</td>
</tr>
<tr>
<td>Encryption Dictionary</td>
<td>42</td>
<td>43</td>
<td>55</td>
<td>77</td>
</tr>
</tbody>
</table>

Table 5.7: Time taken (in seconds) for eDB to execute Q4

aggregation over one column as given below was also executed over eDB.

\[
\text{select sum(orderkey) from orders where orderdate > sel1 and orderdate < sel2}
\]
Table 5.8: Time comparison (in seconds) for aggregation query in eDB

<table>
<thead>
<tr>
<th></th>
<th>10%</th>
<th>25%</th>
<th>50%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal Storage Model</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>Encryption Dictionary</td>
<td>30</td>
<td>32</td>
<td>33</td>
<td>38</td>
</tr>
</tbody>
</table>

The performance table for the same is given below: We can see that there is a trade off for storing the data in encryption dictionary model, where query execution time for order predicate queries improves significantly and for queries involving aggregation it takes slightly more time.
Chapter 6

Conclusions

In this thesis, we presented a DaaS model eDB, which is the extension of PhantomDB where multiple encryption schemes were used side by side. With the encryption schemes chosen in eDB we were able to prove the IND-OCPA security of eDB. We developed a basic idea for handling query processing for floating point numbers in eDB. eDB also presented a new efficient storage model called *Encryption Dictionary* which helps in significant improvement of the query execution time for queries having order predicate clauses with trade off on aggregate queries where it performs little badly.
Chapter 7

Future Work

Various directions for the future work are:

• Extend eDB to handle analytic queries and data mining algorithms efficiently.

• To provide protocols to handle various other functions existing in database like under root, logarithm etc.
Bibliography


