Analytical Study for the Mutable Order-Preserving Encoding with Periodically Encryption Key Changing

دراسة تحليلية لطريقة التشفيـــر المتغيرة المُحافِظَة على الترتيب مع تغيير مفتاح التشفير دورياً

Prepared by
Suha Wasof Omar

Supervisor

Prof. Ahmad Al- Kayed

Associate Prof. Oleg Viktorov

A Thesis Submitted in Partial Fulfillment of the Requirements for the Master Degree in Computer Science

Department of Computer Science

Faculty of Information Technology

Middle East University

Amman, Jordan

May, 2016

AUTHORIZATION STATEMENT

I, Suha Wasof Omar, authorize the Middle East University to provide hard copies or soft copies of my thesis to libraries, institutions or individuals upon their request.

Name: Suha Wasof Omar

Signature: 18/5/2016

Data: [illegible]

Examination Committee Decision

This is to certify that the thesis entitled Analytical Study for the Mutable-Order Preserving Encoding with Periodically Encryption Key Changing.

Examination Committee Members Signature

(Supervisor)

Dr. Ahmad Al-Kayed

[illegible]

(Associate)

Dr. Oleg Victorov

[illegible]

(Head of the Committee and Internal Committee Members)

Dr. Ahmad Abu-Shareha

[illegible]

(External Committee Members)

Dr. Hesham Abusaimeh

[illegible]

ACKNOWLEDGMENT

I would like to thank to almighty God for blesses which enabled me to achieve this thesis.

I would like to express my sincere appreciation to Prof. Ahmad AL-Kayed for his guidance, helping and encouraging my efforts during this research and motivation throughout Master’s Thesis.

I would like to thank Prof. Oleg Viktorov for his guidance and supporting during writing this thesis.

I would like to thank my best friends. They were always standing with me through the good times and bad.

Dedications

I dedicate this thesis to my parents. I hope that this achievement complete the dream that you had for me all those many years age when you chose to give me the best education you could.

Also, I dedicate this thesis to my sisters for their support, encouragement, quiet patience and unwavering love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviations	Meaning
AES	Advanced Encryption Standard
DaaS	Database as a Service
DBaaS	DataBase as a Service
DBMS	database management system
DES	Data Encryption Standard
DET	Deterministic Encryption Scheme
DOPE	Digital-based Order Preserving Encryption
DBA	database administrator
EOE	Efficiently Orderable Encryption
GUI	Graphical User Interface
IaaS	Infrastructure as a Service
IND-CPA	Indestinguishability-against Chosen Plaintext Attack
IND-FAOCPA	Indestinguishability- Frequency-Analyzing Ordered Chosen Plaintext Attack
IND-OCPA	Indestinguishability-against Order Chosen Plaintext Attack
MOPE	Most Order-Preserving Encoding
MV-OPES	Multi-Valued - Order-Preserving Encryption Scheme
OE	Oblivious Encryption
OE-DOPE	Oblivious Encryption- Digital based Order Preserving Encryption
OPE	Order-Preserving Encryption

OPES	Order-Preserving Encryption Scheme
PaaS	Platform as a Service
QoS	Quality of Service
RND	Randomized Encryption Scheme
ROPE	Randomized Order Preserving Encryption
ROPF	Random Order Preserving Function
RSA	Rivest Shamir Adleman
SaaS	Software as a Service

Analytical Study for the Mutable Order-Preserving Encoding with Periodically Encryption Key Changing

Prepared by

Suha Wasof Omar

Supervisor

Prof. Ahmad AL-Kayed

Associate: Prof. Oleg ViKtorov

Abstract

The security of cloud database is the most challenge in the cloud paradigm. Encryption is the solution for the data security problem but some encryption techniques do not preserve data order. Order Preserving Encryption (OPE) is an encryption technique that used to preserve the order of data while executing range queries over encrypted data. OPE suffers from leakage of information.

This research has used the order preserving encryption to preserve the order of data. It found a balancing between performance and security level in Mutable Order Preserving Encoding (MOPE). MOPE algorithm keeps its binary search tree balanced on each transaction applied by the system. This research proposed to change the encryption key for a certain time or a certain number of transactions. To enhance the security level without degrading the performance, the idea of this research is to utilize the time consumed to rebalance part of the tree in MOPE algorithm to change the encryption key. Changing the encryption key caused disorder of data thus the whole tree should be restructured.

Changing encryption key from time to time enhances the security of the system while keeping the encryption key for a long time leads to leak of data. Also, changing the encryption key more often decreased the performance. Thus, this research found an optimal point to change the encryption key with an acceptable losing of performance. It used two parameters to monitor the performance, time and transaction window. It found the best time and best number of transactions to change the encryption key.

The proper time window for changing the encryption key that lead to an acceptable losing of performance was equal to 0.4 millisecond to insert 100 data records with 42% losing of performance. The proper time window for changing the encryption key that lead to an acceptable losing of performance was equal to 0.4millisecond to delete 40 data records from 100 data records with 26% losing of performance.

The proper transaction window for changing the encryption key that lead to an acceptable losing of performance was equal to 5 transactions to insert 100 data records with 41% losing of performance. The proper transaction window for changing the encryption key that lead to an acceptable losing of performance was equal to 6 transactions to delete 40 data records from 100 data records with 30% losing of performance.

Keywords: Cloud database, encryption key, security, performance, parameters, time window, transaction window, OPE, MOPE.

دراسة تحليلية لطريقة التشفيـر المتغيرة المُحافِظَة على الترتيب مع تغيير مفتاح التشفير دورياً

إعداد

سها واصف عمر

المشرف الرئيسي

الاستاذ الدكتور احمد الكايد

المشرف المساعد

الاستاذ الدكتور اوليك فيكتروف

الملخص

انعدام الامن في قاعدة البيانات السُحابية واحدة من اهم التحديات في النموذج السُحابي. تشفير البيانات هو الحل لمشكلة انعدام الامن للبيانات لكن بعض تقنيات التشفير لا تحافظ على ترتيب البيانات. تقنيات التشفير المتغيرة المُحافِظة على الترتيب هي احدة خوارزميات التشفير المستخدمة للمحافظة على ترتيب البيانات اثناء اجراء عمليات المدى على البيانات المشفرة. لكن تقنيات التشفير المحُافِظة على الترتيب (OPE) تعاني من مشكلة تسرب معلومات.

هذا البحث استخدم تقنية التشفير المتغيرة المحُافِظة على الترتيب (MOPE)للمحافظة على ترتيب البيانات. هذا البحث وجد توازن بين الاداء ومستوى الامن في خوارزمية التشفير المتغيرة المحُافِظة على الترتيب. خوارزمية التشفير المتغيرة المحُافِظة على الترتيبMOPE تحافظ على الشجرة بشكلها المتوازن مع كل عملية تجري على النظام. هذا البحث اقترح تغيير مفتاح التشفير مع وقت محدد او عدد عمليات محددة. لتحسين مستوى الامن بدون هبوط في الاداء فأن فكرة هذا البحث هي استغلال الوقت المستهلَك لإعادة

التوازن الى جزء الشجرة في خوارزمية التشفير المتغيرة المحُافِظة على الترتيب في تغيير مفتاح التشفير. تغيير مفتاح التشفير تسبَبَ في خلل في ترتيب البيانات لذلك فأن كامل الشجرة تحتاج الى اعادة هيكلة.

تغيير مفتاح التشفير من وقت لأخر حَسَن من مستوى الامن للنظام. لكن الاحتفاظ به فترة طويلة ادى الى تسرب في البيانات وكذلك تغييره مرات كثيرة تَسَبَب في تقليل الاداء. لذلك هذا البحث وجد الانقطة الامثل لتغيير مفتاح التشفير مع خسارة مقبولة في الاداء. حيث استخدم متغيريين لمراقبة الاداء, نافذة الوقت ونافذة العمليات, ووجد افضل وقت وافضل عدد من العمليات لتغيير مفتاح التشفير.

نافذة الوقت المُناسِبة لتغيير مفتاح التشفير والتي ادت الى خسارة مقبولة في الاداء عند اضافة 100 حقل من البيانات هي 0.4 millisecond مع خسارة 42% في الاداء. نافذة الوقت المُناسِبة لتغيير مفتاح التشفير والتي ادت الى خسارة مقبولة في الاداء عند حذف 40 حقل من البيانات من اصل 100 حقل هي 0.4 millisecond مع خسارة 26% في الاداء.

نافذة العمليات المُناسِبة لتغيير مفتاح التشفير والتي ادت الى خسارة مقبولة في الاداء عند اضافة 100 حقل من البيانات هي 5 عمليات مع خسارة 41% في الاداء. نافذة العمليات المُناسِبة لتغيير مفتاح التشفير والتي ادت الى خسارة مقبولة في الاداء عند حذف 40 حقل من البيانات من اصل 100 حقل هي 6 عمليات مع خسارة 30% في الاداء.

كلمات مفتاحية: قاعدة البيانات السحابية, مفتاح التشفير, الحماية, الاداء, متغير, نافذة الوقت, نافذة العمليات, تقنيات التشفير المحُافِظة على الترتيب, تقنية التشفير المتغيرة المحُافِظة على الترتيب.

Chapter One
Introduction

1.1 Introduction

Cloud computing is a model for delivering computing as a service over the Internet. These services may be applications, hardware, or system software. Also, the services are sharable among many clients.

Cloud computing is a model for delivering the services in on-demand and pay-for-use manners through network access to datacenter which provide services such as applications, hardware, and system software. The datacenter is named Cloud and the services are named computing utility **(Armbrust, Fox, Griffith, Joseph, Ketz, Konwinski, Lee, Patterson, Rabkin, Stoica & Zaharia, 2009).**

Cloud computing is a subscription-based service, where the user can pay to obtain some resources such as networks, storage space, operating systems, databases, and other computer resources. The cloud helps us to access our information from anywhere at any time. Therefore, it does not need for the user to be in the same physical location that stores the data. Just the user needs to access it through an Internet connection. For example, if you want to look at a specific document stored in the cloud, first establish an Internet connection either through wired or wireless connection. Then the document can be accessed through any device connected to the Internet such as desktop, laptop, tablet, phone **(Hurth & Cebula, 2011).**

The cloud attracted by most level of users starting from individuals to governments. For example, cloud can be used by small companies because they can use it for storing their data and information to reduce the cost of purchasing of memory devices and other resources that they can not afford to buy them as a bigger company.

Each user can choose the cloud type depend on their needs. Also choosing the cloud provider depends on the user needs with the service type that provided by the cloud provider **(Hurth &Cebula, 2011).**

Cloud computing is a concept for providing computing utility for the client in a very similar way to provide traditional public utilities such as electricity, water, and natural gas. The utility computing that can be provided to the client as a service are networking, storage, processors, software system, applications **(Hassan, 2011)**.

Cloud computing is a concept that creates from the evolution of using technology that delivered through the Internet. It comes as a result of advance in virtualization and distributed computing. It is a solution for most IT industry because it provides businesses what they need starting from hardware infrastructure, system software, reaching to applications and databases **(Atayero & Feyisetan, 2011).**

Benefits of cloud computing

The services provided for clients have many benefits such as following **(Hassan, 2011)**:

1. On-demand model: the client/organization can utilize a service depending on their needs.
2. Autonomous: the clients are independent from the technical complexities of the provided services such as physical locations and number of human resources who manage the service.
3. Quality of Service (QoS): the provider informs the clients about the expected level of the provided service and the client check if it meets their technical needs.

4. Internet-based: the client can reach to the requested service using an internet connection. Hence, the cloud computing using cloud graphically to represent the internet.
5. Easy-to-Use: cloud provider offers easy-to-use interface in order to persuade user to use this service.
6. Scalable: clients can scale up or down their usage for the service according to their needs such that they can dynamically create, upload and install their virtual machine images.
7. Inexpensive: the small and medium companies that they can not afford to own their IT infrastructure can use cloud computing as an IT infrastructure in a lower cost.
8. Subscription-based model: the client can use only the interested services and pay according to their usage.

Cloud Computing Service Models:

In cloud computing everything is provided as a service. There are three types of cloud computing providers **(Atayero & Feyisetan, 2009):**

1- Software as a Service (SaaS): in this service the client can access and use the software application hosted, deployed, and managed by the provider. The client use thin client to access to the service such as (web browser) to input data and get output. In SaaS, the client can access to the application with limited control over it. Examples of SaaS are email services (Gmail) and business applications such as customer relationship management (saleforce).

2- Platform as a Service (PaaS): the client (developer) can build, test and deploy applications on the provider's platform. The client in PaaS does not have direct control over the infrastructure.
3- Infrastructure as a Service (IaaS): instead to buy and build a datacenter from the ground. IaaS provides the client (developer) to access the infrastructure based on demand with no direct control over the resources. This infrastructure includes processing, storage, network, or any other computing resources that can be used to run platforms such as operating systems.

Cloud Computing Deployment Models

These services areprovided through four deployment models **(Atayero & Feyisetan, 2011)**:

1- The public cloud: it is the traditional view of cloud computing. In which the infrastructure available to the general public via multi-tenant model and most cost-effective model with an existence of the security and privacy. Amazon and Microsoft Azure are examples of public clouds.
2- The private cloud: in which the infrastructure is used by single tenant and is may be managed by tenant organization or by a third party. Its cost is more than the public cloud and it provides the user more control over its data and resources.
3- The community cloud: in which the infrastructure is sharable among several businesses within specific community and managed by one of the businesses or by a third party.

4- The hybrid cloud: is a combination of two or more of the public, private and community models**.**

Although of all benefits obtained from using cloud computing services, there are many features that must take into account when deciding to use these services. Security is one of features that eliminate the use of cloud computing services because of sharing and multi-tenancy **(Atayero & Feyisetan, 2011).**

There are a lot of personal information and potentially secure data that housed on the cloud. This information is a valuable to individuals for malicious purposes. Hence, it is necessary for the user of the cloud to identify what information will be stored in the cloud, who will take the privileges to access to that stored information, and what the type of and level of protection is used in that cloud. It is important for the user to understand the security measures (criterion) that the cloud provider depend on them. These criterions vary from provider to provider and from cloud type to another such as what protection methods do they use for the hardware that will be used for storing data and do they have firewalls setup **(Hurth & Cebula, 2011).**

The aspect of outsourced database begins to grow with the time as a data rapidly increases. DataBase as a Service (DaaS) is used to manage big data. The provider of database offers the ability for organizations to create, store and access their data stored in outsourced database. But the users of outsourced database must take into account that if the provider is not trusted, then sensitive data may be leaked (stolen) by any adversary that may be the provider is the adversary. Therefore, the preserving privacy of database becomes very important issues **(Lin, Yang, Lin, and Chen, 2014).**

Store data outside of the organization in outsourced database creates the needs to think how that environment is secure. Thus, security of outsourced database is not easy to provide. To protect the confidentiality of each client's data, several security methods are proposed **(Al Shehri, 2013).**

Many solutions for protect privacy on database has been proposed. One of these approaches is data encryption that includes encrypt entire database on the client side before store to the non-trust provider (DaaS) **(Lin, Yang, Lin, and Chen, 2014).**

In general there are two ways for encryption/decryption of data symmetric and asymmetric encryption schemes. The computing time of symmetric scheme is better than asymmetric scheme but the security provided by asymmetric is better than symmetric since the symmetric uses one key for encryption and decryption while asymmetric uses two keys one for encryption and second for decryption **(Lin, Yang, Lin, and Chen, 2014).**
Encryption is one of the methods to protect each client's data stored in cloud database. Also, there are many encryption techniques used for this purpose. Some of encryption algorithms preserve the order, others are not. Order-Preserving Encryption Scheme (OPES) is one of techniques which provide the ability to execute range queries over encrypted data with preserve the order of data before and after encryption. The disadvantage of OPES is leakage of data, so they provide less security even they preserve the order**(Agrawal, Kiernan, Srikant & Xu, 2004).**

1.2 Problem Statement

Security is a certain challenge in the cloud computing model and there are many encryption algorithms proposed to tackle this challenge.

OPES algorithm is an encryption technique that preserves the order of data, allowing database to execute range queries directly over encrypted data without decrypting them but it leaks some information. Thus, changing of encryption key is one of solutions that trying to prevent leakage of data without performance degradation or with an acceptable overhead in performance.

Changing encryption key may reduce the performance of database because the database needs to reindex for each time when encryption key is changed. It is interesting to find a method for changing the encryption key such that reindexing in database does not cause performance degradation (or leads to minimum database performance degradation).

Problem will be accomplished by answering the following questions:

1. When to change the encryption key? In other words, under which criteria the encryption key is better to be changed, time window or transaction window?
2. What is the optimal mechanism with regard to database performance such that the changing of encryption key leads to minimal indices operations?
3. What is the proper value for time window to change the encryption key such that provides an acceptable system performance?
4. What is the proper number of transactions which applied by the system to change the encryption key such that the system provides an acceptable performance?

1.3 Objectives

The main objective of this research is to preserve the reindexing database time, when applying a time window or transaction window for changing an encryption key in MOPE algorithm, with prevent or reduce leakage of data.

1.4 Motivation and Significance of Study

Cloud computing is one of the fast growing technologies based on the Internet to allow users utilizing services provided by the cloud provider. It provides virtualized resources as a service over the Internet. Services are a large sharable pool of computing resources such as (storage, servers, networks, applications, system software) that can be used in an on-demand and pay-as-you-use manners without installation and management of any software. Adaption to the cloud can bring many benefits to people and organizations such as: cost reduction, continuous availability, performance, quick deployment and ease of integration. Although of all benefits obtained from adaption of cloud computing, there are many problems that make a lot of people and organizations hesitate to adapt it and think how safe an environment of it. Thus, one of these problems is the security of data. This problem creates from sharing of resources and multi-tenancy characteristics which lead the cloud computing to be untrusted place to store data because stored data may be captured by any attacker or even curious DBAs. Thus, data has a certain probability to leak and as a result cloud is not enough secure even it has a well performance.

There is a necessity to find a solution for the security of data problem. Encryption is one of the proposed and strong solutions and exactly computing directly over encrypted data without decrypting them. OPES is one of the encryption techniques that provide this ability of computing with the preserve of order of plaintext to corresponding ciphertext and the high database performance.

1.5 Contribution

This research has been contributed to find the optimal point when to change the encryption key which leads to minimal reindexing the database. This research contribution is summarized in the following points:

1- Discovering when to change the encryption key (time or transaction window).

2- The mechanism that leads to optimal database performance (reindexing time of database) when changing of encryption key using time window or transaction window is discovered.

3- The proper time window for changing the encryption key that provides an acceptable performance is discovered.

4- The proper transaction window for changing the encryption key that provides an acceptable performance is discovered.

1.6 Methodology

This research will be a combination of descriptive and quantitative research methodology. The methodology based on studying and implementing MOPE algorithm and observing the performance of the algorithm with several parameters. The proposed methodology will use a quantitative research by building an experiment and apply different data parameters then analyze and evaluate the performance of the system.

The proposed methodology explained in Figure 1.1and consists of the following steps:

1. Implement the traditional MOPE algorithm with different numbers of data and record the performance (from the period of reindexing time of database).
2. Implement the proposed methodology using time window criteria by applying it over many experiments holds different time window values with different numbers of data and record the performance for each experiment until find out the proper time duration for changing the encryption key such that there is no performance degradation or with an acceptable performance degradation.
3. Implement the proposed methodology using transaction window criteria by applying it over many experiments holds different values for number of transactions with different numbers of data and record the performance for each experiment until find out the proper number of transaction for changing the encryption key such that there is no performance degradation or with an acceptable performance overhead.
4. Analyze and evaluate the performance results for each experiment.
5. Determine the optimal mechanism, proper time window and proper transaction window for changing the encryption key that provides an acceptable database reindexing time.

1.7 Proposed Methodology

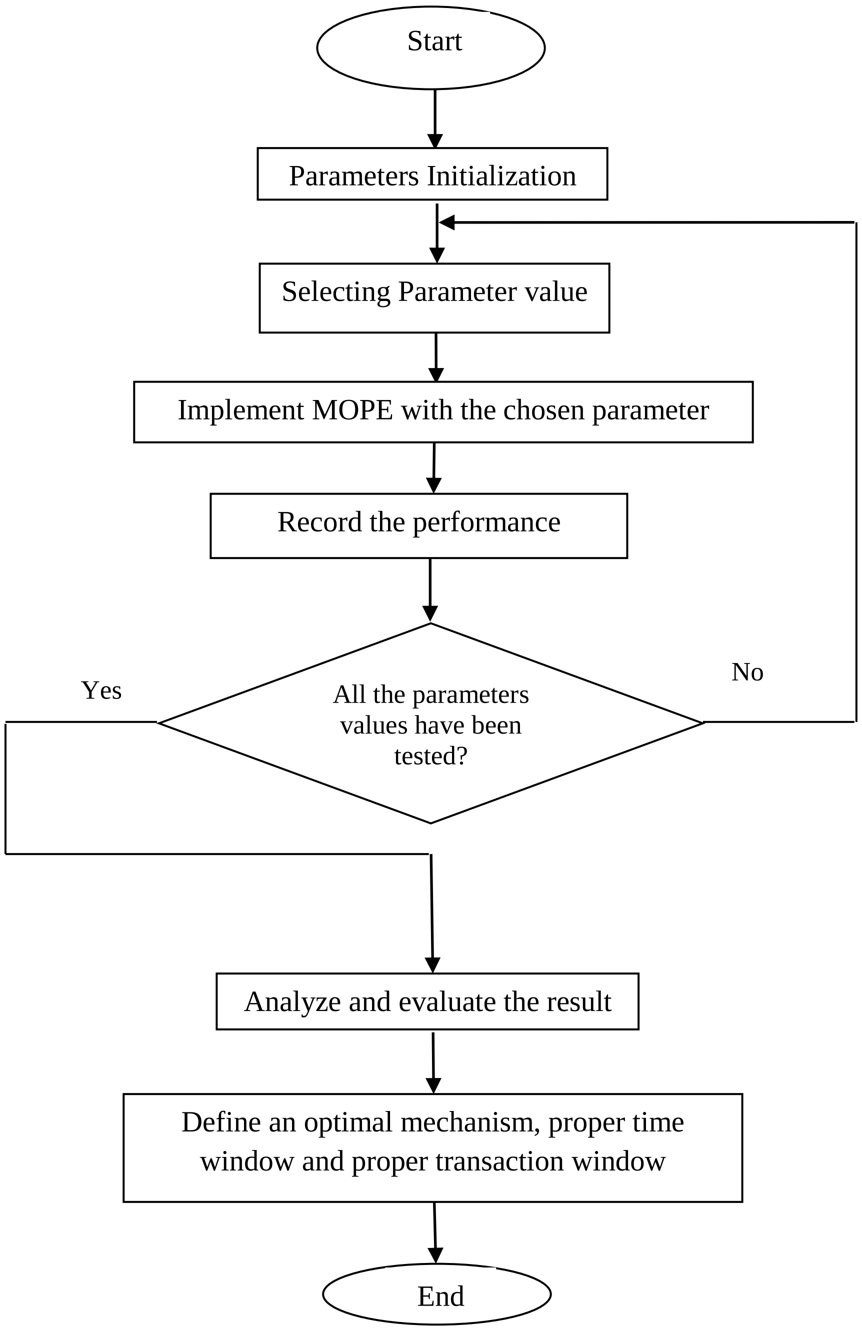

Start
Parameters Initialization
Selecting Parameter value
Implement MOPE with the chosen parameter
Record the performance
All the parameters values have been tested?
Yes
No
Analyze and evaluate the result
Define an optimal mechanism, proper time window and proper transaction window
End

Figure 1.1: General Flowchart of Proposed Solution

Chapter Two Background and Literature Review

Chapter Two

This chapter provides a literature review and background on the main concepts covered by this research. It is divided into three sections. The first section 2.1 defines some concepts about cloud computing security and subjects related to our research. The second section 2.2 explains the enhancement of the MOPE algorithm. Section 3.3 defines many literature reviews and previous studies about OPES.

2.1 Background

There are many concepts that related with cloud computing. This section will explain cloud computing security and cloud database security are defined in general. Then encryption techniques, OPES and MOPE algorithm are defined. Finally, quick sort algorithm and binary search tree structure are defined.

2.1.1 Cloud Computing Security

Despite of all advantages provided by the cloud computing, there are many challenges that must take into account by IT managers when they decide to depend on it as an IT infrastructure. Security is one of the challenges that faced by cloud computing because of multi-tenancy and resource-sharing. Some of the security issues in cloud computing are:

Availability, in which data being available whenever it required; Data security, this risk comes from loss of control over data because of virtualization vulnerabilities; SaaS vulnerabilities, phishing scams which are lead to data leakage, and interception and loss of encryption key; and Privacy and legal issues, in which the business will store out their data without knowing what future purpose it might use **(Atayero & Feyisetan, 2011).**

In cloud computing, delivery models represent the layer above the deployment model and utilized within a specific deployment model. These delivery models are SaaS, PaaS and IaaS. They represent the brain of the cloud computing and providing many services such as fast deployment, pay-as-you-use, lower cost, scalability, rapid provisioning hypervisor and so on, but despite of those services , still many people hesitate to use it. Thus, security is one of the major challenges which limit the improvement of cloud computing such as accessibility vulnerabilities, virtualization vulnerabilities, web application vulnerabilities and so on. There are different security issues that has launched depend on the nature of the service delivery models of the cloud computing system. The big security challenges in SaaS is that the enterprise do not know how and where data is stored and how are secured. Another point the enterprise data is stored in the same datacenter with the data of other enterprises. So enterprises may uncomfort to use SaaS model. There are many security elements that must be take into account when developing SaaS applications and deployment models such as: Data security, Network security, Data locality, Data integrity, Data segregation. The security issues with IaaS are that developers believe that they have best control over the security such as no problems in the virtualization manager. But in practice there are many security problems in the virtual machines. The other challenge is the reliability of the data stored in the provider's hardware. IaaS provide degrees of security issues based on the cloud deployment model used. Physical security of infrastructure is an important security issues **(Subashini & Kavitha, 2010).**

Ensuring of data security is one of the most important issues that must be executed properly in cloud computing applications in order to get attraction of users to trust on the service provider. One the security issue is that user's data is stored at the SaaS service provider with data of other users. If the provider is public then the data of any SaaS application will be stored with data of other SaaS applications for other users. Also, the provider must replicate in many distributed physical locations in order of high availability. Hence, lack of control over data is one of the security issues. The types of data that need to be secured are: usage data, information that aggregated from computer devices; sensitive data, represents information about health and bank accounts; personally information, information that identify the individuals and human resources; and unique device information, information that might be uniquely traceable such as IP address **(Soofi, Irfan, Talib & Sarwar, 2014).**

2.1.2 Cloud Database Security

As knowing, cloud computing is a new direction in IT industry in terms of cost saving, fast provisioning and releasing and fast application performance. These benefits orient the thinking of companies to start using the cloud application which deliver to users as a service. Cloud database is mostly used as a service. It also called DataBase as a Service (DBaaS). It is used for storing huge data for many companies. There are many service providers for database and there are many parameters that must be considered by users and companies to select the best service provider such as: select DBaaS depending on the company requirement and the services being provided by the company, the capacity of storing data on the cloud database, portability, accessibility, data integrity, data privacy for each user from any attacker and illegitimate users and data security, which is the major threat to the data stored in the database and it depend on the encryption techniques used. Also, data security represents a certain challenge for companies when using cloud database because the companies do not have full control on the server stored data on the cloud database and they can not do anything to make the cloud database security strong. They depend on the service provider only. So, the company must be careful how to select suitable DBaaS provider **(Al Shehri, 2013).**

Cloud storage (DaaS) is classified as a forth service provided by the cloud computing model in addition to SaaS, PaaS, and IaaS. DaaS provide the organization the ability to move their data from their own database to a central database on a cloud. This movement gives many benefits to the organization such as using the latest IT infrastructure in an effective cost because it only pays for use. But this movement also leads to many security challenges like data security, privacy, and integrity **(Sunitha & Prashantha, 2013).**

In data outsourcing, (DBaaS) moves database management system (DBMS) from traditional client-server architecture (the owner is responsible for managing DBMS) to third party architecture (data management is managed by the provider) such as Google, Amazon, and Microsoft. The market of DBMS is growing fast such that large to huge enterprises begins to adapt it. Also, medium to small enterprises adapted it. Security of user's data is a major challenge in IT industry because both data and code stored in the service provider and there is a lack of trust between service provider and users **(Masood, Shibli, Mehak, Ghazi & Khan, 2014).**

Primary Features of DBaaS

DBaaS inherits the same features used to build any cloud computing provider such as **(Masood, Shibli, Mehak, Ghazi & Khan, 2014):**

1. Provide DBMS as on-demand independent service for managing data with no performance overhead.
2. User can access any data from anywhere using any device such as Desktop, Computers, Mobile, and Tablet with just a requirement to internet connection.
3. User can perform desired operation on the abstract resource with no needing to deployment or configuration.
4. Elasticity, which is refer to that DBaaS can adapt to any change in workload with no disruption in the service.
5. The user pays for only data resources used.
6. Agility, which refers to rapid provisioning of resources and the ability to adapt with business changes.

Advantages of DBaaS

The major advantages of DBaaS are following **(Masood, Shibli, Mehak, Ghazi & Khan, 2014)**:

1. Fast and automatic recovery.
2. Data replication at multiple locations.
3. Execute back up regularly.
4. High availability in 24 hours a day, 7 days a week, and 365 days a year.

5. Provides simplified Graphical User Interface (GUI) for data back up and data restore.
6. No hardware utilization by users and no need to hire database expert to manage the database.

(DBaaS) security refers to secure data when it store and use. The security issues includes that the data is mainly secured from any risk (outside or inside adversary), secure data management, confidentiality, integrity, and availability of critical data stored in DBaaS. With confidentiality, it means to secure the execution of operations requested by the user by allow to only authorized users to access and manipulate data. Also, the data must be encrypted before residing in the outsourced database in order to kept confidentiality of user and prevent any attacker from intrude against the user's data. With integrity, it refers to protect data from any unauthorized access. With availability, it means that any user can reach to system resource at any time with no failure **(Masood, Shibli, Mehak, Ghazi & Khan, 2014).**

2.1.3 Encryption of Cloud Computing

Security goal of data in cloud are confidentiality, availability, and integrity. Confidentiality of data in cloud can be achieved by encryption. Encryption algorithms are used to protect data from attackers and to provide the users confidentiality to use the cloud. Encryption algorithm refers to rearrange the data (text, image, audio, video) to convert it to unreadable data, and to decrypt it at the intended recipient. To encrypt and decrypt any data, encryption key must be chosen properly. There are many encryption techniques in cloud computing data security which are classified into symmetric-key encryption (sender and receiver use the same encryption key- private key) and asymmetric-key encryption (sender and receiver use different encryption key- public key) algorithms **(Soofi, Khan & Amin, 2014).**

Encryption is a solution for network security and there are many encryption algorithms that play an important role in information security system and are used to protect the shared data. They are also securing data while transmitting through the network and are classified into symmetric (private) and asymmetric (public) key encryption. In symmetric key encryption, only one key is used to encrypt and decrypt data such as Data Encryption Standard (DES) algorithm which is used to protect sensitive commercial and unclassified data. And Advanced Encryption Standard (AES) algorithm is providing good security and faster than DES algorithm. Asymmetric key encryption is used to solve the problem of key distribution. It uses two keys: public which is used to encryption and private key which is used for decryption such as: Rivest Shamir Adleman (RSA) and Digital Signature algorithms. They based on mathematical functions, computationally intensive. They are slower than symmetric encryption because they require more computational processing power **(Mahajan & Sachdeva, 2013).**

RSA is designed by Ron Rivest, Adi Shamir and Leonard Adleman in 1977. It is an asymmetric (public key) cryptosystem for encryption of blocks of data or digital signatures. This algorithm uses two different keys, the first one is the public key which is used for encryption a message by the sender, and the second one the private key which is used for decryption the (sent) message by the receiver. RSA uses two prime numbers to generate the keys. RSA operations can be executed in three steps: key generation, encryption and decryption. The encryption of data is an interactive algorithm between client and server **(Mahajan & Sachdeva, 2013)**

Deterministic encryption scheme is a cryptosystem such that for any plaintext and key, it produces the same ciphertext. Public key encryption algorithm such as RSA is a deterministic encryption scheme. The advantage for using deterministic encryption scheme is that provides the ability for efficient searching on encrypted database with large sized data. The tree-based data structure is used to store the encrypted field that helps to retrieve the requested data in time logarithmic in the size of the database **(Bellare, Boldyeva & O'Neil, 2007).**

The goal of randomized encryption scheme is enhancing cryptographic security. It means that when encrypting the same data several times, it will produce different ciphertexts. It is used with public key cryptography and is classified as more secure than deterministic encryption scheme **(Stinson, 2006)**.

When the data moved from local database to cloud database, an encryption techniques must be used to secure the operations and the confidential of each customer's data. When data needs to be processed, the cloud provider needs to access to the data. Thus, the encryption methods have the ability to perform operations (calculations, range queries) over encrypted data (ciphertext) without decrypting them and produce the same results, as the operations done directly on the plaintext.

Homomorphic encryption method has the ability to execute calculations of confidential encrypted data without knowing anything about the private key (without decryption). The calculations include (+,$\oplus$) **(Tebaa, El Haji & El Ghazi, 2012).**

Order Preserving Encryption Scheme (OPES) is an encryption technique that allows executing comparison and range queries directly over encrypted data without decrypting them with preserve the order of data. The range queries that can be applied are MAX, MIN, and COUNT queries **(Agrawal, et al. 2004).**

2.1.4 Order-Preserving Encryption Scheme

In DBaaS the data are exported to the cloud database, therefore the data values are encrypted before sent to the cloud database. And the database must have the ability to perform queries over encrypted data without decrypting them. Order-Preserving Encryption (OPE) obtain the ability to perform range queries over encrypted data without knowing the decryption key and give the same result as executed this query on real data **(Kerschbaum, 2015).**

Encryption is the best method to protect the confidentiality of each user's data that stored in untrusted database server. The problem with existing encryption methods is that the integration with the database systems cause to slow the performance since these techniques does not preserve the order such as database indices and lead to unacceptable query execution over encrypted data. Order-Preserving Encryption (OPE) for numeric data allows to perform comparison operations directly over encrypted data without decrypting them with preserving the order of data and gives considerable results **(Agrawal, et al. 2004).**

2.1.5 Mutable Order Preserving Encoding

Many encryption techniques are proposed to protect the confidential data stored in the cloud database. One limitation of these techniques is that when an application needs to process any data, it must be decrypted. This may lead to leak of data for adversaries. OPES is a strong technique that provide the ability to compute directly over encrypted data without decrypting them with the preserving the order of data before and after encryption and to execute comparison operations and range queries. MOPE algorithm uses a balanced search tree to preserve the order of data. The MOPE algorithm preserves its performance through keeping the search tree in its balanced form on each transaction applied by the system. The ideal security protocol that OPES must achieved in (Popa, Li & Zeldovich , 2013)is Indistinguishability against Order Chosen-Plaintext Attack IND-OCPA states that the adversary can not know anything except the order for the plaintexts stored in the system at the same time. Most OPES does not achieve this protocol and leak some information during the order and allow an adversary to destroy the security of data when learning the order relation of old deleted data and new inserted data.

MOPE algorithm as other OPES algorithms suffers from leakage of data especially order relation for the data that are previously stored in the OPE tree and currently stored in the OPE tree. Thus, the MOPE algorithm used RND encryption techniques to accomplish the ideal security protocol IND-OCPA **(Popa, Li & Zeldovich , 2013).**

2.1.6 Quick Sort Algorithm

Quick sort is a high efficient algorithm for sorting large sized data set. It is based on the idea on divide-and-conquer. A large array is divided into two arrays one of them contains values that must be smaller than pivot value and the other one contains values that must be greater than pivot value. The pivot value is a specific value that the division of an array makes depend on it. This division will continue recursively by finding another pivot values for the sub arrays until all sub arrays contains only one element. Then recursively each two sub arrays resulted from division are sorted and combined. The complexity of this algorithm in the best case O(n) and in the worst case O(n log n) **(Pfenning, 2013).**

2.1.7 Binary Search Tree

Binary search tree is a hierarchical data structure with a single pointer at a root node. Each node *n* may represent a root node for sub tree such that all nodes that are at the left side of the *n* hold values less than the value of *n* and all nodes that stay at the right of the *n* hold values greater than the value of *n.*

The root node is in the top of the tree and each child node has exactly one parent node. Each parent has at most two child node left child and right child.

Sibling nodes are nodes that share the same parent node. And a leaf node has no child nodes. Figure 2.1 explains the tree structure in general.

There are many operations can be done on the tree such as create a Binary search tree, find a node in Binary search tree, add node to the Binary search tree, and traverse the Binary search tree **(Stoimen's web Log., 2012).**

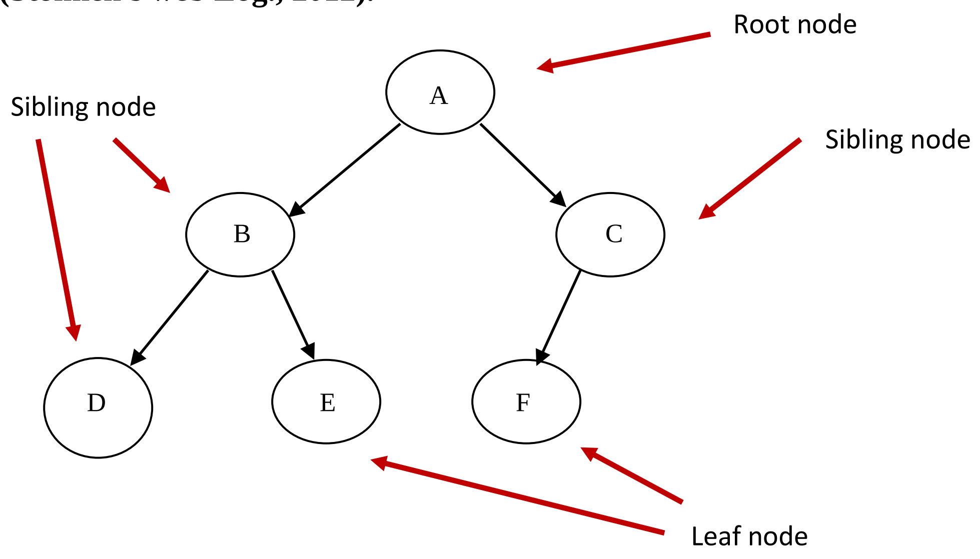


Figure2.1: Explaining the Tree Structure in General

2.2Related Work

Enhancement of MOPE Algorithm

The system consists of two main parts that are working interactively. These two parts are client and server. The client gives commands to server to do actions. The real data (plaintext) is added by the client and is encrypted/decrypted also by the client using enhanced RND encryption scheme.

When the data (plaintext) is added and give the prompt insert by the client to the server, the data (plaintext) is encrypted using specific encryption key and send the produced encrypted data (ciphertext) to the server in order to execute the insertion transaction without knowing anything about the plaintext except the order. Figure 2.2 that adapted from (**Popa, Li, & Zeldovich, 2013)** explaining how the client and server are interacting with each other.

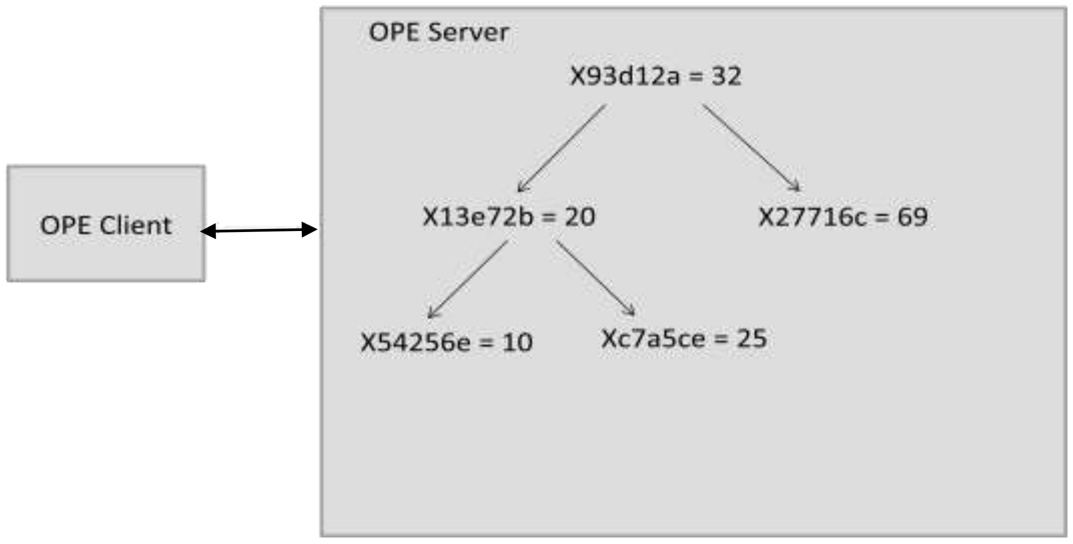


Figure 2.2: Explaining the Interaction between Client and Server as adapted from **(Popa, Li, &Zeldovich, 2013)**

The server does not order the ciphertext. The client directs the server how and when to store the ciphertext and how to retrieve it. Hence, they work interactively and the data is stored in the server in a search tree since the tree has the ability to store the data in an order form. The search tree that used in this methodology is Binary search tree.

The server will be directed by the client for storing (insert) data in the tree in its right order. Such as for insertion transaction, the data (plaintext *P*) is added by the client, then is encrypted also by the client to produce ciphertext *C*, after that the client start to direct the server for insert it through call the tree traversal algorithm that run interactively between them.

In tree traversal algorithm the client begins the action by asking the server for the root node. The server returns the value of the root node (ciphertext *C*) which does not know except it. Then the client decrypts the retrieved ciphertext and compares the produced plaintext *V* with the new value added *P*. If the new added value *P* equals to the retrieved plaintext *V*, then the new value *P* will not inserted. If the new value *P* greater than the retrieved plaintext *V*, then go to get the next node on the right of the retrieved plaintext *V*. If the new value *P* less than the value of the retrieved plaintext *V*, then go to get the next node on the left of the retrieved plaintext *V*. These comparisons go on until reach to the right null position that will new value inserted in it.

Tree traversal algorithm is used to determine the node in the tree in which the new encrypted data (ciphertext *C*) is inserted or deleted. The algorithm is explained in Figure2.3 that adapted from (**Popa, Li, & Zeldovich, 2013)** and the steps of this algorithm are following:

1. The client asks the server about the ciphertext (*C*) at the root node.
2. The client decrypt *C* to *V*
3. If $P<V$ (*P* is the plaintext) then the client tells server to go "left"

   Else if $P = V$ then the client tells server "node found", go to step 5

   Else the client tells server to go "right"
4. The server returns the next C based on the result of step 3. Then go back tostep 2.
5. The algorithm stops when *P* is found, or when the server arrives at an empty spot in the tree. The output of the algorithm is the resulting pointer in the OPE tree, the path in the OPE tree from the root and whether P was found.

<u>Note:</u>

C: ciphertext

V: plaintext produced from decryption of C.

P: plaintext (new added value to insert or delete).

sk: secret key.

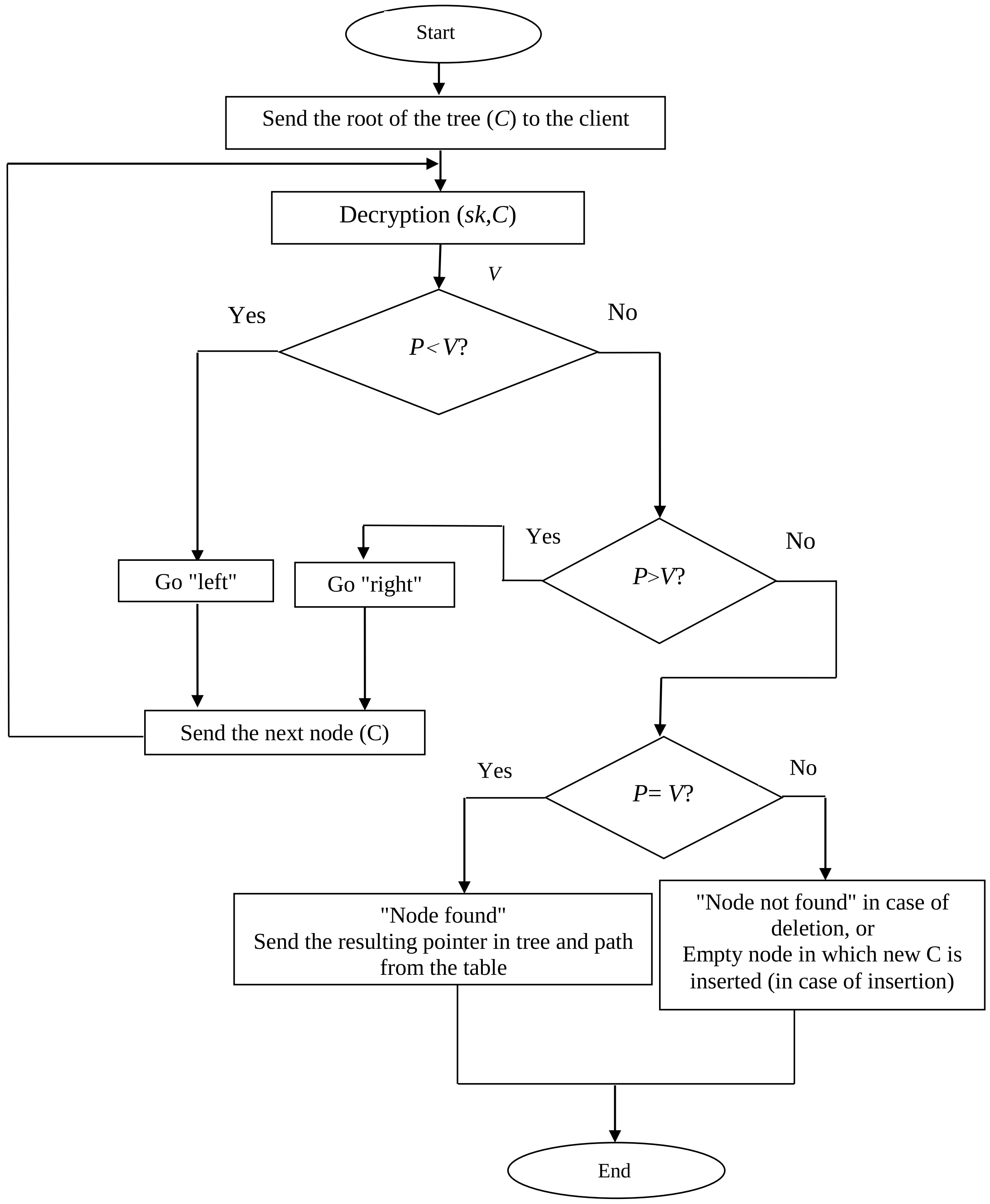

Start
Send the root of the tree (C) to the client
Decryption (sk,C)
V
Yes
No
P< V?
Go "left"
Go "right"
Yes
No
P>V?
Send the next node (C)
Yes
No
P= V?
"Node found"
Send the resulting pointer in tree and path from the table
"Node not found" in case of deletion, or
Empty node in which new C is inserted (in case of insertion)
End

Figure 2.3: Explaining the Tree Traverse Algorithm as adapted from
(**Popa, Li, &Zeldovich, 2013)**

Each ciphertext stored in the Binary search tree is represented by a node *n*, and for each node in the Binary search tree there is an encoding which represent a path to that node starting from the root node such that the left node hold the value Zero, and the right node hold the value One.

The path for each node in the Binary search tree will be stored in a table in the server named encoding table (OPE table) as shown below in Table 2.1 that adapted from (**Popa, Li, & Zeldovich, 2013)**. The advantage of encoding table is that helping client to know the order of each ciphertext stored in the tree using the function named Order function that runs at the server, takes as input *ST* (state of the server) and the ciphertext *C* and the output is that the OPE encoding *e* if the *C* is in the tree or signal an error if the *C* is not stored in the tree. The steps are following:

1. The plaintext *P* is encrypted and produced ciphertext *C*.
2. If *C* is in the OPE table then return its encoding *e*.

   Else compute "error signal".

Table 2.1: Encoding Table as adapted from (**Popa, Li, & Zeldovich, 2013)**

Ciphertext	*Encoding[path]10..0*
x93d12a	[ ]100 = decimal 4
x13e72b	[0]10 = decimal 2
x27716e	[1]10 = decimal 6
x54256e	[00]1 = decimal 1
xc7a5ce	[01]1 = decimal 3

The encoding format for each ciphertext is as following equation:

Encoding = [path]10..0 ……..eq1

[path]: mixture of 0,1 which represent the path from the root node to the indicated node.

10..0: represent the pad which contains One and many zeros as necessary to fill the value to a desired ciphertext size *(m).*

The encryption/decryption technique used in MOPE algorithm is RSA (Rivest, Shamir and Adleman) with 128 key sizes.

Initial state *ST* is used in the MOPE algorithm to create the binary tree. The initial state function runs at the server. It uses the security parameter *k* as input and returns an initial state *ST* which indicates the binary tree and OPE table containing only the values ±∞.

As it is known the MOPE algorithm accomplished the ideal security protocol IND-OCPA through using RND encryption techniques. In general, insertion and deletion transactions in search tree need to rebalance them from time to time in order to preserve the order of data.

The rebalancing of tree is done after each transaction (insertion, deletion) applied by the system that lead the OPE tree to be unbalanced.

The MOPE algorithm strategy is that building a balanced search tree which contains all the ciphertexts (represented the encrypted data for each plaintext entered by the client). For each ciphertext there is a specific encoding *e* which represents a path for it starting from the root node reaching it in the tree. The MOPE algorithm states that after each transaction applied by the system (insertion, deletion) the tree is checked if it is still balanced or not (if it is need for rebalancing or not).When the rebalancing condition is satisfied, part of the tree that became unbalanced (as a result of insertion or deletion transaction) will be rebalanced. Rebalancing of a tree includes rearrange the location of some nodes and update their encoding path. When the tree is still balanced, go to apply another transaction.

MOPE's performance is good in both microbenchmarks and in the context of an encrypted database running TPC-C queries **(Popa, Li, & Zeldovich, 2013)**.

According to the security which refers to the probability of leakage of information about the encrypted data, the MOPE algorithm satisfied the ideal security protocol IND-OCPA with probability of leakage of data in the case of deletion transaction when using DET encryption techniques. This deleted data was already has a specific order. This order will be saved by the server even it deleted. When a new data is inserted with the same order relation of the deleted data, the adversary can easily estimates the old deleted data and new inserted data through knowing the order relation. Hence, the MOPE algorithm solves this problem by using RND encryption technique instead of DET encryption technique that used to satisfy the IND-OCPA security goal **(Popa, Li, & Zeldovich, 2013)**.

Thus, the steps of insertion and deletion transaction algorithms are enhanced in MOPE algorithm and implemented using RND encryption techniques to overcome the problem of leakage of data when delete a data in original MOPE algorithm that used DET encryption techniques **(Popa, Li, & Zeldovich, 2013)**.

Insertion transaction with RND encryption techniques

The client will help the server to save the new inserted data in its right position since the server uses the tree structure to store the data. Thus, it needs to know where to go through the tree of left or right until reach to the right position to store the data or the data is already in the tree so does not need to store it again. Figure 2.4 explaining the insertion transaction algorithm and the following steps clarifies this algorithm:

1. Encrypt plaintext *V* and produce ciphertext *C* and send to the server.
2. The client and server run OPE tree traversal algorithm.
3. If *V* for the *C* was already exist in the OPE tree

   Then the server increases ref-count and go to step 6.

   Else (*V* was not already in the tree) the server insert *C* in the tree in the position resulted by the tree traversal algorithm with ref-count of 1, store it in the OPE table and return *C* and encoding *e*.
4. If it still balanced Then return *C*, e and go to step 6.

   Else go to step 5.
5. Rebalance the tree, and as a result update the OPE tree, OPE table and database.
6. End.

Note:

sk: refers to the secret key

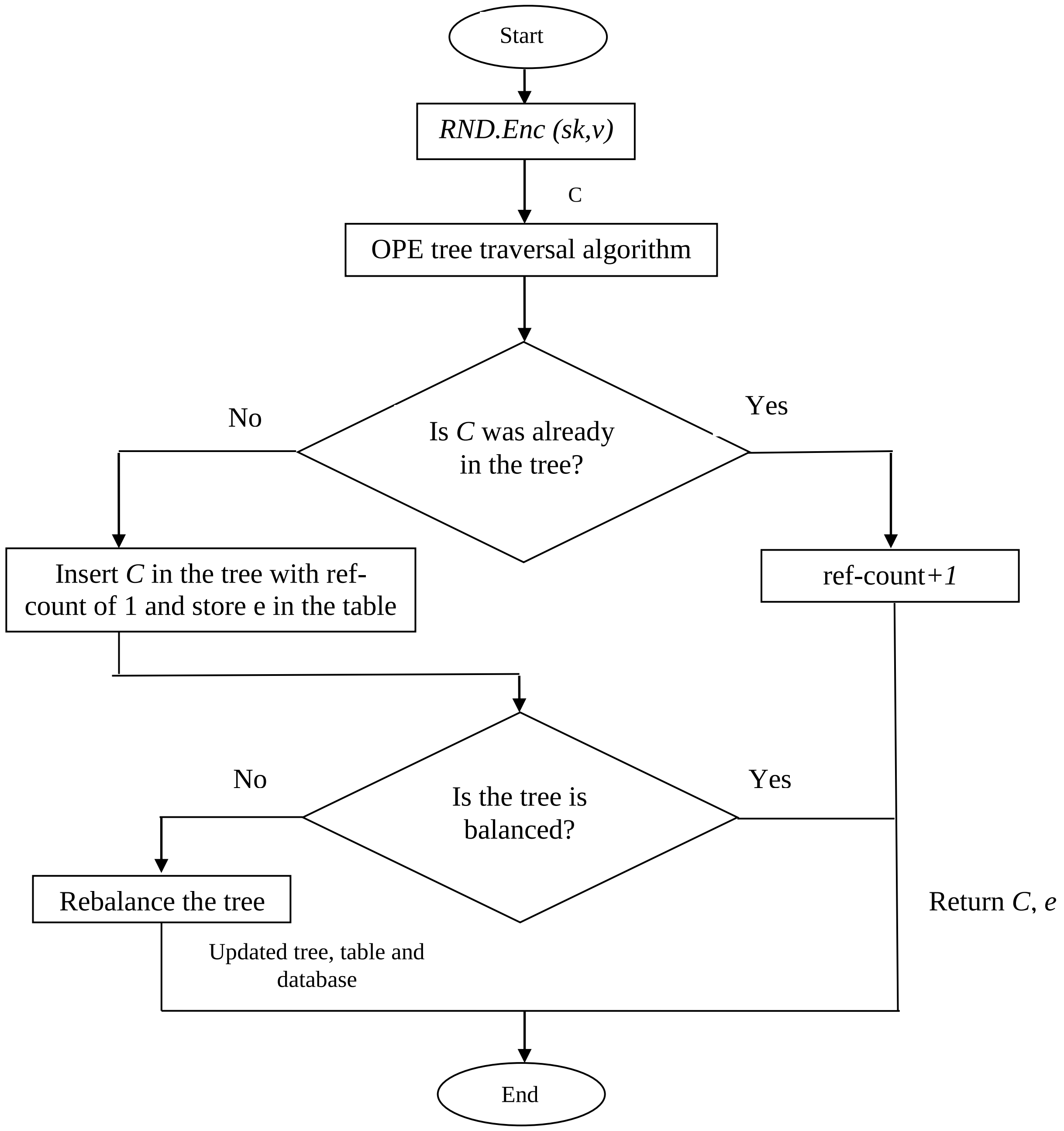

Start
RND.Enc (sk,v)
C
OPE tree traversal algorithm
No
Is *C* was already in the tree?
Yes
Insert *C* in the tree with ref-count of 1 and store e in the table
ref-count+*1*
No
Is the tree is balanced?
Yes
Rebalance the tree
Return *C*, *e*
Updated tree, table and database
End

Figure 2.4: Explaining Insertion Transaction Algorithm
As adapted from **(Popa, Li, &Zeldovich, 2013)**

Deletion transaction with RND encryption techniques

When MOPE algorithm used DET technique faced the problem of leakage of information because the server leaks the order relation of data. MOPE algorithm solved this problem by using RND encryption technique instead of DET encryption techniques.

To delete any data, the client must direct the server to delete this data. The server will call tree traverse algorithm to determine if the requested data exist in the OPE tree or not. If it is existing, then the server return it with its position to the client. Otherwise, the server will send an error message. The steps of deletion transaction are clarified below in Figure 2.5 and explain in the following steps:

1. Encrypt plaintext *V* and produce ciphertext *C* and send to the server.
2. The client and server run the tree traversal algorithm in order to find the node to remove.

   If node not exist then signal an error, go to step 6

   Else the server decreases its ref-count
3. If ref-count = 0 then removes the node from the tree and its encoding *e* from the OPE table.
4. If OPE tree is still balanced after deletion then go to step 6

   Else rebalance the OPE tree and update the OPE table and any storage containing these encoding.
5. The algorithm return *C* (the ciphertext corresponding to V from the OPE tree).
6. End

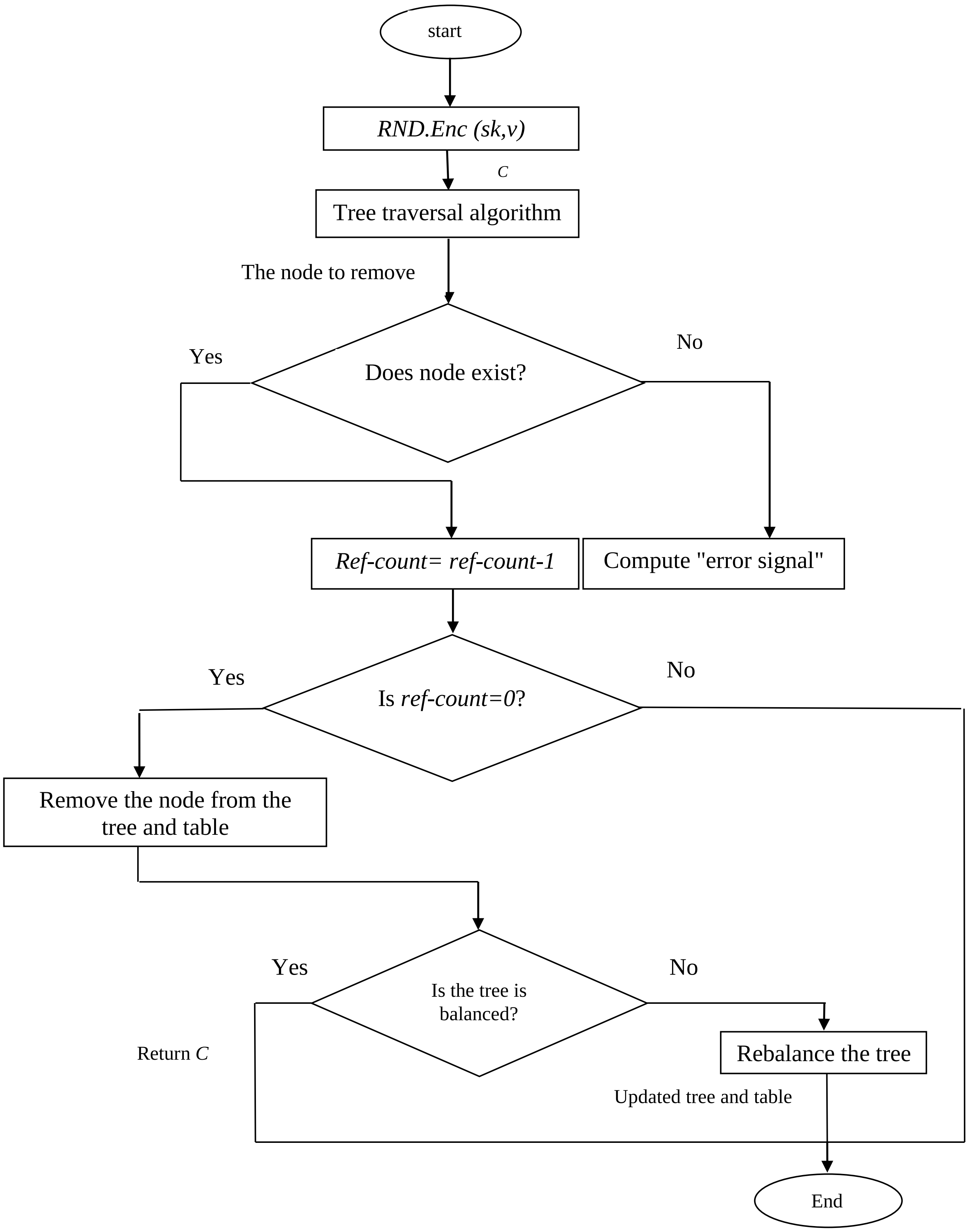


Figure 2.5: Explaining Deletion Transaction Algorithm
As adapted from **(Popa, Li, &Zeldovich, 2013)**.

2.3 Literature Reviews

The Literature review has discussed many concepts and several problems in the cloud computing. Some of these literatures as represented below:

- **Ozsoyoglu, Singer & Chung(2003)**discussed that providing database as a service creates the need to protect the sensitive data from tampering techniques such as disk scan, compromising the data by bypassing the database management system software or database authentication process. So, the way to protect them is to encrypt the data, and to permit queries and transactions over the encrypted data.

A trade-off between the degree of security provided by encryption and the efficient querying of the database has been discussed by (Ozsoyoglu et al., 2003).

The techniques to protect a database and query to encrypted database have been presented by **(Ozsoyoglu et al., 2003).** This technique named anti-tamper database which has many assumptions, such as, the encryption of the original database is transparent, the general encryption mechanism is available to the public (including the adversary) and the commercial database system that host the data does not know that the data in its database is encrypted.

(Ozsoyoglu et al., 2003) has been considered integer values attributes (field)-level encryption for relational database. The authors concentrate the attention on a family of open-form and closed-form homomorphism encryption. In open-form, order-preserving encryption (OPE) is defined for integer values and the goal is to build a family of functions indexed by a key *k* with a little information as possible. To achieve this goal, a pseudorandom sequence using *k* and encryption/decryption functions are generated. The method is reasonable for small values and has some information leakage **(Ozsoyoglu, Singer & Chung, 2003).**

- **Agrawal, Kiernan, Srikant & Xu, (2004)** discovered a method named Order Preserving Encryption Scheme for Numeric Data. This method solved the problem of slow performance for database that produced from integration it with encryption techniques. This method allows performing comparison operations directly over encrypted data without decrypting them and allows range queries over encrypted data such as Equality, MAX, MIN, COUNT, GROUP BY, and ORDER BY **(Agrawal, Kiernan, Srikant&Xu, 2004).**

The OPES gives considerable results when applying those queries over encrypted data, handle updates and easily, handles addition of new value without needing to change the encryption of other values, in addition to the easily integrated with database system such that the indices of database can be structured easily and have a good level of performance such as B-Tree. Also as a result an intruder can access to the encrypted data but cannot encrypt or decrypt any value. OPES is executed via three stages: model stage, flatten stage, and transform stage **(Agrawal, Kiernan, Srikant& Xu, 2004).**

- **Boldyreva, Chenette, Lee & O'Neill, (2009)** have been used Order-Preserving Encryption (OPE) with single code for both encryption and decryption. OPE used in a database community for allowing efficient range queries on encrypted data. It means that the queries over encrypted data must not affect the database performance while it is growing.

There are many standard security notions for encryption that must satisfy the goal of OPE security such as IND-CPA. The security notion is unachievable by an OPE scheme because it can not satisfy all the standard notion of security since it may leak the order-relations among the plaintexts. Another notion named Pseudorandom Function (PRF) which defines the best possible security under order-preserving constraints has been presented in **(Boldyreva et al., 2009).**

This construction is based on a natural relation between a random order-preserving function and the hypergeometric probability distribution **(Boldyreva, Chenette, Lee & O'Neill, 2009).**

-Kahdem, Amagasa & Kitagawa, (2010) state that known encryption schemes provide a privacy range queries over encrypted database at many levels: block, columns, rows, and attributes, but this approach is vulnerable to statistical attacks. A statistical attack against an encrypted database seeks to use some statistical measures to reach to the individual (unencrypted) data. The traditional solution against the statistical attack is to use different encryption key for different field, but this defense lead to performance degradation.

The authors suggest encrypting a value to different multiple values to prevent statistical attacks. Therefore, a new encryption scheme called MV-OPES presented by **(Kahdem et al, 2010)**. In MV-OPES one integer value is encrypted to many values using the same encryption key while preserving the order of the integer values apply comparison operations directly over the encrypted data.

MV-OPES can be easily integrated with existing database systems. It is robust against statistical attacks. The security for MV-OPES can be achieved with reasonable overhead **(Kahdem, Amagasa & Kitagawa, 2010).**

- In **Kahdem, Amagasa & Kitagawa, (2010)** Unfortunately, OPES is not protected against plaintext attacks and statistical attacks. A new database encryption scheme named Multivalued - Partial Order Preserving Encryption Scheme (MV-POPES) is presented in **(Kahdem et al.)**. The security level is improved through face attacks: the plaintext domain is divided into many partitions and assigns a unique random number for each partition. This random number will be the order of partitions in the encrypted domain. Then MV-POPE function is used to generate many encrypted values for each integer value in the partitions using the same encryption key. It is also preserve the order of the integer values within each partition to allow comparison to be directly applied on the encrypted data. The evaluation shows that MV-POPES approach can achieve the security for sensitive data against plaintext and statistical attacks with an acceptable performance overhead **(Kahdem, Amagasa & Kitagawa, 2010).**

- **Popa, Redfield, Zeldovich & Balakrishnan, (2011)**A new mechanism named CryptDB that provides a high level of confidentiality against the threats has been discussed by (Popa et al., 2011). CryptDB allows executing SQL queries over encrypted data. It consists of two parts: a database proxy and unmodified DBMS. CryptDB uses three approaches to solve the problem of threats.

- The first approach is to execute SQL queries over the encrypted data using a SQL-aware encryption strategy.
- The second approach is adjusting the encryption level at run-time to prevent revealing the data to the DBMS using the onion of encryption.

- The third approach is to chain encryption key to user passwords. As a result, CryptDB guarantee security and provide low overhead **(Popa, Redfield, Zeldovich & Balakrishnan, 2011).**

  - The Order-Preserving Encryption (OPE) has been discussed in **(Boldyreva, Chenette& O'Neill, 2011).**.Security requirements for the OPE scheme have been created. An efficient scheme has been proposed and proved that scheme meets their security definition.

In **(Boldyreva et al., 2011)** the Random Order Preserving Function (ROPF) is used as a security notion. ROPF is the "ideal object" in the security definition. To understand the strength and limitation with ROPF must first propose several security notions. One of these notions is one-wayness security. ROPF has been shown that OPE scheme is not a strong secure with this notion because there are many leaks with approximate value of plaintexts and approximate distance between any two plaintexts.

The stronger security notion than ROPF has been achieved in **(Boldyreva et al., 2011)** although are not order-preserving but support range queries. This security notion is named Efficiently Orderable Encryption (EOE).

Finally, the scheme named Modular OPE scheme that improve the security of any OPE scheme, because it does not leak any information about plaintext location, with preserving the efficiency has been presented in Modular OPE**(Boldyreva, Chenette & O'Neill, 2011).**

- Xiao, Yen & Huynh, (2012) state that the problem with OPE algorithms is that they use deterministic single encryption key which not suitable for systems with multi-users access. A solution is to use different encryption key for different data.

Two methods to extend OPE algorithms for multiuser systems have been presented in **(Xiao et al., 2012).** In the first one, the key agent is created in the system and a digit-based OPE (DOPE) protocol has been developed to produce distributed encryption and to ensure that no user in the system has any knowledge of the encryption key. The problem of this protocol is that when key agent is compromised, the data sent through it is compromised too. Therefore, a new oblivious encryption (OE) protocol based on the oblivious transfer concept has been developed. Then OE is integrated with DOPE to get OE-DOPE protocol such that both DOPE and OE-DOPE can be used with an existing OPE algorithms with each user can use different encryption key. Security and correctness of those protocols has been proven but there is a reasonable performance overhead **(Xiao, Yen & Huynh, 2012).**

- Liu, Chen, Yang, Jia & You, (2014) reviewed that the ideal security goal for OPE scheme, IND-CPA, is to reveal no additional information about the plaintext values besides their order.

It seems that only ideal security OPE scheme is mutable OPE (mOPE). **(Popa, Li & Zeldovich , 2013).**

Therefore, the problem of OPE is how to improve security but ensure the function and the efficiency. The security goal is ciphertext- only attacks.

The (Lin et al., 2014)try to build secure OPE scheme which practical on the outsourced database. Also, discuss how to hide data distribution and data frequency for the OPE schemes design. They propose the OPE model which uses message space expansion and nonlinear space split to hide the data distribution and data frequency.

The security analysis and performance evaluation show that our OPE scheme is secure enough and more efficient **(Liu, Chen, Yang, Jia & You, 2014, July).**

- **Reddy & Ramachandram, (2014)** state OPE is a well scheme for encrypting database as it allows to execute comparison operations and range queries in sufficient way over encrypted data directly. OPE preserve the order of values and reveal nothing else.

One of the security goal for OPE schemes is IND-OCPA (Indestinguishability-against Order Chosen Plaintext Attack), and the first OPE scheme achieved this security is mutable OPE (mOPE) which uses DET (deterministic encryption) to encrypt plaintext values with leakage of distribution of plaintext values **(Popa, Li & Zeldovich , 2013).**

(Reddy & Ramachandram, 2014) present a new scheme named Randomized Order Preserving Encryption (ROPE) that follows the mOPE scheme by contributing randomness to it to tackle IND-CPA security.

ROPE is based on RND for the ciphertext, which implies that two equal values are mapped to different ciphertexts. It allows executing insertion, deletion and querying functions instantly on an encrypted data.

The evaluation of this scheme is that POPE permit to instantly secure and efficient SQL queries (range queries, aggregate functions, equality queries, and join queries) on encrypted data. The performance evaluation for this scheme is that there is a time overhead for query retrieval **(Reddy & Ramachandram, 2014).**

- The security of OPE is still un provable. This paper presents new security protocol named Indestinguishability under Frequency- Analyzing Ordered Chosen Plaintext Attack (IND-FAOCPA) and claim that this new protocol is stronger than (IND-OCPA). This security states that the security of order-preserving encryption is increased through randomizing the ciphertext such that no frequency information from repeated ciphertexts leaks and as a result hide the frequency of plaintext. This paper introduces a new tradeoff between security and storage cost. The implementation of this idea proved its security protocol (IND-FAOCPA) **(Kerschbaum, 2015).**

Chapter Three
Methodology

Chapter Three

Several research studies have been built to improve the security level in cloud computing. OPES is one of the schemes that used in this research but the main problem of OPES is leakage of information. There is a need to increase the security level to decrease losing of information to improve OPES. Whenever the security level is improved, the performance will be reduced. This research builds a model to improve the security level and study its effect on the performance using MOPE algorithm.

The main topics of this chapter include the introduction about the balancing between performance and security level in section 3.1. section 3.2 illustrates proposed methodology and the MOPE algorithm performance then it explains the proposed transaction window (which refers to a proper number of transactions that applied by the system to change the encryption key) and proposed time window (which refers to an absolute time to change the encryption key), and a comprehensive exploratory of the experiments are provided in section 3.3.

3.1 Introduction

One of the proposed solutions to improve the security level of OPES is to change encryption key. In general, it is preferred for the encryption techniques to change the encryption key from time to time in order to get more secure system. Keeping the encryption keys for long time without changing leads to security problems. From other side, changing the encryption key more often will decrease the performance of the system. Thus, it is important to find an optimal point to change the encryption key.

This research investigates how to find the optimal point where we can to balance between security level and performance. The main goal is to change the encryption key with acceptable performance degradation. MOPE will be explained in detail, to be able to reach to this optimal point.

This research will enhance the security level of MOPE algorithm by changing the encryption key. At the same time, it will monitor the performance by finding the best number of transactions or best time to change the encryption key.

This chapter will explore the main security for MOPE algorithm. These properties include: the leak of information using MOPE algorithm, the effect of encryption over database indices, the structure of these indices, and its operations such as balancing algorithm, re-ordering algorithm, etc.

The security protocol that should be achieved by OPE schemes is named Indesitinguishability- against Chosen Plaintext Attack (IND-CPA). This security protocol states that the ciphertext can not reveal anything about the plaintext value except their order. Most OPES cannot satisfy all standards notions of this security protocol IND-CPA since most OPES used deterministic encryption techniques and leak the order-relations among the plaintexts. Therefore, it is important for each application to choose the best security protocol that it requires **(Boldyreva, Chenette, Lee & O'Neill, 2009).**

MOPE algorithm used the security protocol named as Indesitinguishability- against Order Chosen Plaintext Attack (IND-OCPA) that states the adversary cannot know from the ciphertexts anything about the plaintexts values except the order for the data that are stored at the same time in the database **(Popa, Li, & Zeldovich, 2013).**

The original MOPE algorithm suffers from leakage of order-relation for the previously stored data and new inserted data. That leak helps an adversary to estimate the real data (plaintext) if these two values have the same order-relation. **(Popa, Li, & Zeldovich, 2013)** proposed a solution to achieve IND-OCPA through replacing DET encryption techniques with RND encryption techniques.

In this research, we will continue to use DET encryption techniques with applying some conditions to achieve IND-OCPA security protocol and as a result improve the security level of the system.

Searching over database needs to use binary search tree or B-tree. These trees need to be rebalanced from time to time to keep a good performance. Thus, the idea of this research is to change the encryption key utilizing the rebalancing time and replace the rebalancing of tree to do restructuring which includes rebuilding of the tree.

This research will be a combination of descriptive and quantitative research methodology through using the MOPE algorithm to develop a new technique as following:

1. Implement MOPE algorithm. This section will measure the performance of the MOPE algorithm that has been proposed by **(Popa, Li & Zeldovich, 2013).**
2. Modify the MOPE algorithm by adding transaction window and time window to record the time in order to compute the execution time which related to the performance.
3. The a quantitative methodology will be applied by building an experiment and apply different data parameters then analyze and evaluate the performance of the system.

3.2 Proposed Methodology

3.2.1 Performance of the MOPE Algorithm

As it is previously clarified this research is a combination of descriptive and quantitative research methodology. The descriptive part includes studying and implementing the enhanced MOPE encryption algorithm and observing the performance of the algorithm by recording the running time. In this research, the running time is added to the insertion and deletion transaction algorithms of the enhanced MOPE algorithm as a separate step (step1, step 7) in order to study the running time for the MOPE algorithm to be able to propose the proper time and transaction windows to enhance the security level.

Step 1 includes record the start time T_0 and step 7 includes compute running time such that $T_r = T_f - T_0$, where T_f refers to the finish time.

According to the condition of rebalancing OPE tree and steps of rebalancing, the enhanced MOPE algorithm **(Popa, Li, & Zeldovich, 2013)** did not clarify how to check if the OPE tree is balanced or not after insertion or deletion transactions and how to rebalance the OPE tree.

In this research we apply the condition that taken from the general definition of balanced search tree in **(Stoimen's web log, 2012)** which states a balanced Binary search tree is a data structure when the height of the left and the right sub tree can vary by one level at most. The following condition and balancing steps is the ideal steps that applied in our proposed solution.

The condition of rebalancing is that for each node inserted in or deleted from the OPE tree, the tree must be checked if it is still balanced or not. This checking is executed by measure the number of levels in the left side and number of levels in the right side for each node starting from the new inserted node (in case of insertion) or the replaced node (in case of deletion) up of the tree until reach to the root node **(Pfenning, 2011)**.

The following steps are the rebalancing algorithm **(Pfenning, 2011):**

Let us suppose the inserted or replaced node is represented by (*S*)

```
1. $X$ = number of levels in the left side of $S$
2. $Y$ = number of levels in the right side of $S$
3. If $X= Y$ then the tree is balanced
        Else if $X>Y$ then $Z=X-Y$
        Else $Z=Y-X$
4. If $Z>1$ then go to step 5 (rebalance the tree from that node).
      Else go to the parent node, put its value in $S$, and go to step 1.
```

5. Linked list is built out of a binary search tree by walk through the tree from left-root-right.

6. The middle item in the linked list must be found. Finding the middle item is the same way to find the middle index of an array that its length is knownby using the following equation: *X= (number of elements/2) +1*

And make *X* as a root node, go recursively in this step.

7. Balanced search tree is built again **(Stoimen's web log, 2012).**

In case of deletion when an existing node is deleted, the decision must be taken to replace the deleted node with another node. Choosing the replaced node *H* depending on the location of the deleted node in the OPE tree. The following steps clarify the rebalancing conditions in case of deletion **(Pfaff, 2004):**

1- If the deleted node *D* is a leaf node, then there is no node stay instead of it but must be checked if its deletion effect on the balancing of the search tree or not by calling the rebalancing algorithm.

2- If the deleted node *D* has only one child, then binding its parent with its child which represents the replaced node *H*, put the value of *H* in *S* and call the rebalancing algorithm

3- If the deleted node *D* has two children, then splice out its successor, and replace the data in the node to be deleted by the data from the spliced out successor which represents the replaced node *H*. Or it is possible to splice out the predecessor instead of successor. Then put the value of *H* in *S* and call the rebalancing algorithm.

3.2.2 Proposed Transaction Window:

MOPE algorithm satisfied the ideal security protocol IND-OCPA using RND encryption techniques instead of DET encryption techniques since MOPE algorithm leak of data when used it.

This research tries to treat the problem of security level obtained by the MOPE algorithm when depending on DET encryption techniques through supporting to change the encryption key with proper time window or transaction window.

In general, changing encryption key on each transaction applied by the system leads the system with high security level but at the same time effect on the system performance. Therefore, this research improves security level provided by MOPE algorithm with preserving performance level or with an acceptable degradation of system performance provided by MOPE algorithm.

This research finds a balancing between security level and performance such that improving the security with an acceptable overhead in system performance.

As it is known preserving the performance of systems that used MOPE algorithm is done through rebalancing the binary search tree on each transaction applied by the system that leads the tree to be unbalanced.

Here, the rebalancing time consumed in MOPE algorithm is utilized to change the encryption key. Changing the encryption key leads to disorder the tree. Thus, the whole tree needs to be restructured instead of rebalancing part of the tree that became unbalanced as a result of insertion or deletion transaction.

The performance is measured each time the encryption key is changed and tree is restructured. Two parameters will be used to measure the performance such as (time window, transaction window) until reach to a situation which represents the proper transaction window or proper time window to change the encryption key (that represent improved security level) with suitable performance level of the system.

The performance of system represents the running time *Tr* starting from the command of insert or delete. The total time taken is measured in order to find the effect of changing the encryption key on the system performance.

The encryption/decryption techniques used by the client is DET encryption techniques which always produces the same ciphertext for a given plaintext and key. RSA (Rivest, Shamir and Adleman) with 128 bit key size is used in this research.

The first parameter that will be tested in our proposed solution is transaction window that refers to the proper number of transactions that applied by the system to change the encryption key.

Insertion transaction using transaction window

The steps of inserting new plaintext to the OPE tree is the same as in the MOPE algorithm with DET encryption techniques. Our proposed transaction window will use DET encryption techniques to achieve the security protocol IND-OCPA instead of RND encryption techniques which used by MOPE algorithm. Our proposed transaction window will add some conditions to satisfy this security protocol using DET encryption techniques to treat the problem the of leakage of data that MOPE algorithm suffered from it when using DET encryption techniques and leads to replace it with RND encryption techniques.

After inserting new data and checking the balancing of tree there are two branches. The first branch supports that the tree after insertion is not balanced. Therefore, the decision is to change the encryption key, restructuring the tree, record the time duration and the time in which the restructuring operation is applied.

The second branch is applied when the tree remains balancing after insertion transaction. The number of transactions applied by the system since the last restructuring is measured and store in *J*. After that the value of *J* is compared with the transaction window value that stores in *W*. If *J* equals to or exceeds *W*, then the encryption key must be changed since there is a big probability that leakage of data happened, tree must be structured, time duration is recorded , time of restructuring is record and running time is computed.

J: Parameter refers to the number of transactions (insertion or deletion) since last restructuring time.

W: Proper number of transactions that can be applied by the system to change the encryption key. The values of *W* are concluded from the execution of the original MOPE algorithm.

The steps of this algorithm is clarified in Figure 3.1 and explained as following:

1. Records time T_0.
2. Plaintext P is encrypted and produces its ciphertext C.
3. The client starts to direct the server for insert the ciphertext C but first asks the server to check if C is already in the tree by searching its OPE encoding in the OPE table.

   If the server find C then return e and C, go to step 13

   Else go to step 4
4. Run the OPE tree traversal algorithm in order to determine where to insert the C in the tree and obtain path of C.
5. Insert C in the OPE tree and compute its encoding e based on eq1 and insert it in the OPE table.
6. Check if tree is still balanced then go to step 12

   Else go to step 7
7. Change the encryption key.
8. Restructure the tree and produce new balancing OPE tree and new OPE table.
9. Records the time duration.
10. Record the time of restructuring, go to step 13.
11. Check the number of transactions since last restructuring and store in J.
12. If $J \geq W$ then go to step 7

    Else go to step 13
13. Compute running time $Tr = T_f - T_0$.
14. End.

Note: Eq1 :*Encoding = [path]10..0*

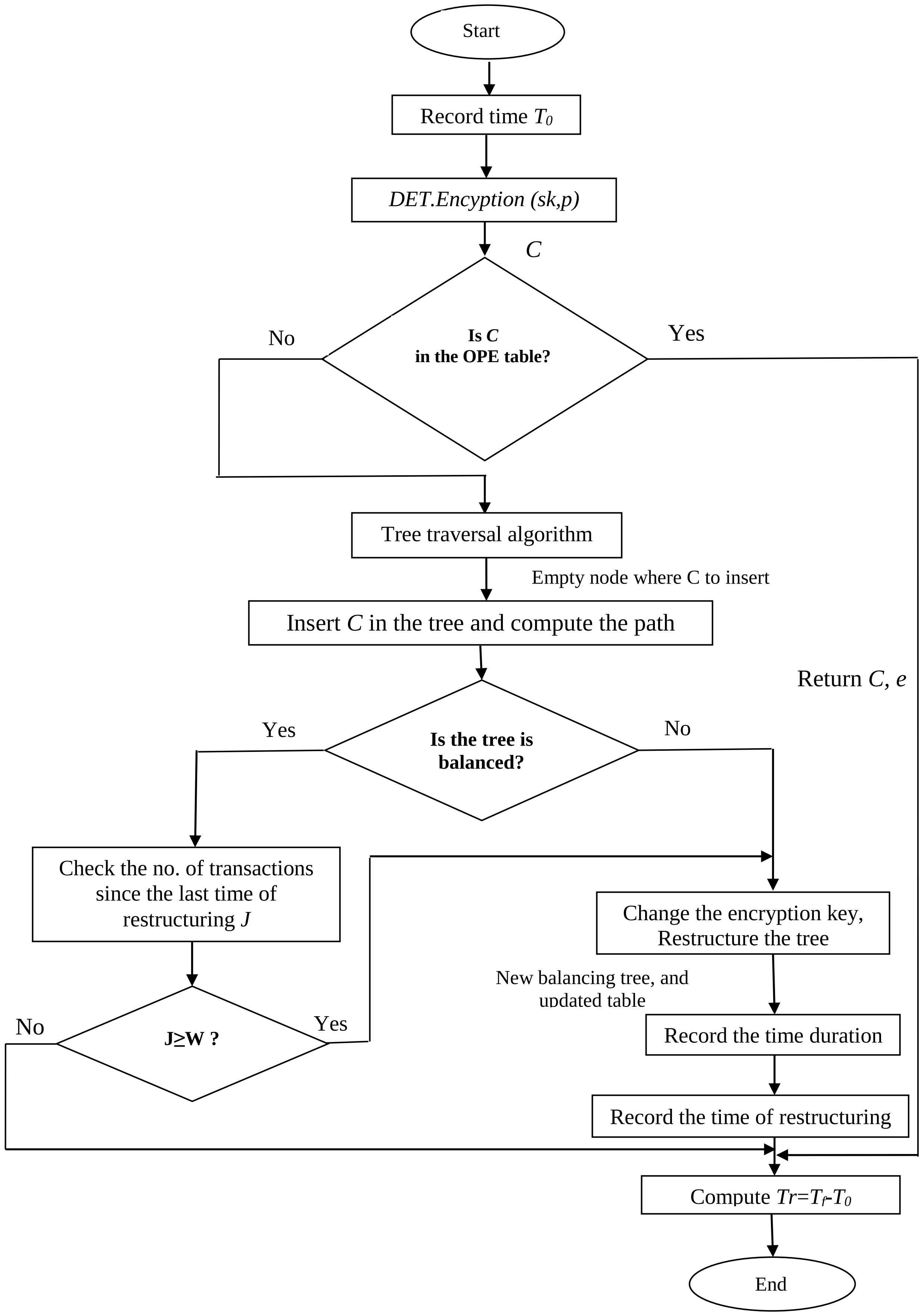

Start
Record time T_0
DET.Encyption (sk,p)
C
Is *C* in the OPE table?
No
Yes
Tree traversal algorithm
Empty node where C to insert
Insert *C* in the tree and compute the path
Return *C*, *e*
Is the tree is balanced?
Yes
No
Check the no. of transactions since the last time of restructuring *J*
Change the encryption key, Restructure the tree
New balancing tree, and updated table
J≥W ?
No
Yes
Record the time duration
Record the time of restructuring
Compute $Tr=T_f-T_0$
End

Figure 3.1: Explaining Insertion Transaction Algorithm using Proposed Transaction Window

Deletion transaction using transaction window:

The steps of deleting plaintext from the OPE tree here is the same as in the MOPE algorithm with DET encryption techniques. When the requested data is determined to remove (through runs the tree traversal algorithm), then after removing, the tree is checked if it is still balanced or not.

If it is not balanced after deletion, the decision is taken to change the encryption key and restructure the tree. Also, the time duration is measured, the time of restructuring is recorded and the running time is computed.

If tree is still balanced after deletion, then the decision is taken to check the number of transaction (insertion, deletion) that applied by the system since last time of restructuring and store in a parameter named *Q*.

If *Q* reaches to or exceeds a value of *R* (*R* represents the transaction window value) then there is ability for data to be leaked. Therefore, when $Q \geq R$ then the key must be changed, tree must be structured, time duration is recorded, time of restructuring is record and running time is computed.

Otherwise, there is another condition that must be checked which is the only number of deletion transactions that applied by the system since last restructuring time. (This is in order to reduce the probability of leakage of data that may happen with deletion as a result for discovering of the order relation for the previously deleted data). The status of this condition is stored in a parameter named *S*. Then *S* is compared with a parameter named *Z* that represents the number of deletion transactions such that when *S* reaches to or exceeds *Z*, then the encryption key must be changed and the tree must be restructured, time duration is measured, time of restructuring is record and running time is computed. Otherwise go to compute the performance through measure the running time *Tr*.

R: is the proper number of transactions (insertion or deletion) that applied by the system to change the encryption key. The values of *R* are concluded from the execution of the original MOPE algorithm

Z: is the proper number of deletion transactions to change the encryption key. The values of *Z* are concluded from the execution of the original MOPE algorithm

Q: refers to the number of transactions since last restructuring time.

S: refers to the number of deletion transactions since last restructuring time.

The steps of deletion using transaction window is clarified in figure 3.2 and explained as following:

1. Records time T_0.
2. Plaintext *P* is encrypted and produces its ciphertext *C*.
3. The client and server run the tree traversal algorithm in order to find the node to remove.

   If node not exist then compute "error signal", go to step 13

   Else go to step 4
4. Delete the node from the tree and its *e* from the OPE table.
5. If the tree is balanced then go to step 10

   Else go to step 6
6. The encryption key is changed.
7. The tree is restructured and produces new balanced OPE tree and new updated OPE table.

```
8.  The time duration is measured.
9.  The time of restructuring is recorded, go to step 13.
10. The number of transactions since last restructuring is measured and stores in $Q$
        If $Q \geq R$ then go to step 6
            Else go to step 11
11. Measure the number of deletion transactions that applied by the system since last
    restructuring and stores in $S$.
12. If $S \geq Z$ then go to step 6
        Else go to step 13
13. Compute running time $Tr = T_f - T_0$.
14. End.
```

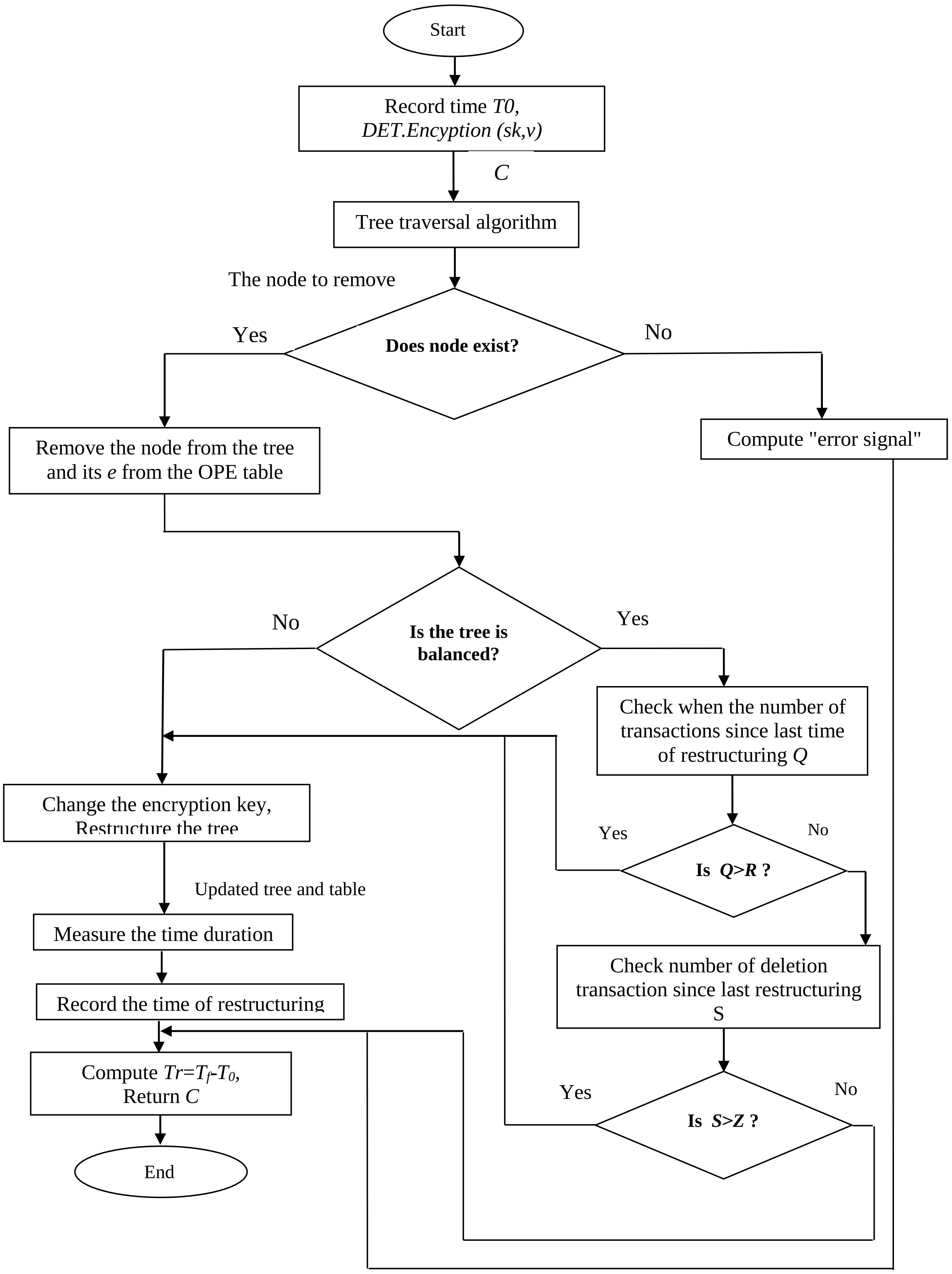

Start
Record time T0, DET.Encyption (sk,v)
C
Tree traversal algorithm
The node to remove
Does node exist?
Yes
No
Remove the node from the tree and its e from the OPE table
Compute "error signal"
Is the tree is balanced?
No
Yes
Check when the number of transactions since last time of restructuring Q
Change the encryption key, Restructure the tree
Yes
No
Is Q>R ?
Updated tree and table
Measure the time duration
Check number of deletion transaction since last restructuring S
Record the time of restructuring
Compute Tr=Tf-T0, Return C
Yes
No
Is S>Z ?
End

Figure 3.2: The Deletion Transaction Algorithm using Proposed Transaction Window

3.2.3 Proposed Time Window:

This research checks another parameter to monitor the performance. This parameter is named time window which refers to the proper time duration to change the encryption key.

Insertion transaction using time window

Insertion with time window has the same steps of insertion that has been applied with transaction window except here the concentration is on the time duration instead of number of transactions such that when the condition of tree balancing is checked and the answer is yes. Then the last time the restructuring operation is checked and stores in *T*. This checking is done in order to prevent or reduce leakage of data in case that many insertion transactions are applied and the tree is still balanced.

N_0 is the proper time duration that the system needs to change the encryption key. It will compare with the last time of restructuring *T*. The values of N_0 that will be checked are concluded from execution of the original MOPE algorithm .If $T \geq N_0$ then the encryption key is changed and tree is restructuring.

Deletion transaction using time window:

The steps of deletion with time window are the same that have been applied by deletion with transaction window except when checking the condition of tree balancing and the answer is yes, then the last time of restructuring is checked and stores in parameter named *T*. This parameter is compared with another parameter *X*. *X* is the time duration to change the encryption key. The values of *X* are concluded from execution of the original MOPE algorithm.

If *T* is equal to or greater than *X* then the decision taken to change the encryption key, restructure the tree, measure the time duration and record the last time of restructuring *T*.

If the last time of restructuring *T* is not reached to a value of *X* then check the number of deletion transactions done since last restructuring time *S*. Then *S* is compared with a parameter named *Z* that represents the number of deletion transactions since last restructuring. The values of *Z* are concluded from the execution of the original MOPE algorithm. When *S* reaches to or exceeds *Z*, then the key must be changed and the tree must be restructured. Otherwise go to compute the performance through measure the running time *Tr*.

How to Restructure the OPE Tree?

As it is known the idea of this research is to utilize the time consumed to rebalance the tree, in changing the encryption key to enhance the security level of MOPE algorithm. Changing encryption key leads to change in the structure of database indices. Thus, changing the encryption key needs to restructure the whole tree to preserve the order of data. As a result the time consumed to do rebalancing of the OPE tree (in MOPE algorithm) will be utilized to change the encryption key and restructure the OPE tree (in our proposed methodology).

In our proposed methodology, when the OPE tree needs to rebalance as a result of insertion or deletion, the decision is taken to change the encryption key and restructure the whole tree. Restructuring the tree is done by deleting all the nodes of the tree and rebuild it as a new tree with the new inserted data (in case of insertion transaction) or without the deleted data (in case of deletion transaction) using MOPE algorithm.

We found that this way took long running time since the time of rebalancing will be very long. Thus, we decided to do the restructuring process in different way. All data will be sorted using quick sort and send to the server to store in the OPE tree.

3.3 Follow Up

Many experiments are created using different data digits and sizes to implement the MOPE algorithm in order to measure its performance. Also experiments are created to implement our proposed methodology using two parameters transaction window and time window and study the effect of these parameters on the database reindexing time until reach to the proper transaction window and time window which provide an acceptable database reindexing time.

The main evaluation process will be based on comparing the execution time for the original MOPE algorithm and the two approaches (transaction and time window).

Also the evaluation process will determine which approach is better to improve the security level, transaction or time window.

Chapter Four
Results and Analysis

Chapter Four

This chapter includes five sections; section 4.1 introduces the parameters used. The characteristics of the software tools used are explained in section 4.2. Section 4.3 explains implementation software. Execution of the MOPE algorithm is defined in section 4.4. Execution of the proposed methodology using transaction window is explained in section 4.5. Section 4.6 explains the execution of the proposed methodology using time window. And section 4.7 analyzes the results.

4.1 Introduction

In general, the main goal of this research is to enhance the security level and study the effect of this enhancement on the performance in OPES. Many experiments will be built and many parameters will be tested in order to achieve to this goal. The parameters includes using transaction window which refers to the proper number of transactions that applied by the system for changing the encryption key and restructuring the OPE tree. Different values for transaction window will be checked and measure the effect of each value used on the execution time of each operation and on the total operations in order to test the system's performance. Many experiments will be applied until reach to a proper number of transactions for changing the encryption key and restructuring the OPE tree that provide an acceptable system performance.

Another parameter will be experimented that is time window. Through it the proper time needed to change the encryption key and restructuring the OPE tree is estimated. Many experiments will be applied with different values of time window and measure the effect of each value used on the performance of the system through compute the running time for each operation until reach to a proper value of the time window to change the encryption key and restructure the OPE tree.

4.2 Tools

This research has been designed and implemented a software to study and analyze the performance of OPES using Visual Basic project file 2010 as a programming language. The properties of the device used to execute this software are Intel(R) Core (TM) i3-4000 CPU 2.40 GHz, 4 GB of RAM and 64-bit operating system. According to the network connection and cloud computing environment, we did not concentrate on them since all we want to check is the database indices. Thus, we have been built a database using Structured Query Language SQL and interfaces through them carry out the insertion and deletion transactions.

4.3 Implementation Software

This research consists of two main parts, the first part is implementing the MOPE algorithm **(Popa, Li, & Zeldovich, 2013)** and the second part is implementing our proposed methodology. Figure A.1 in Appendix A shows the main interface.

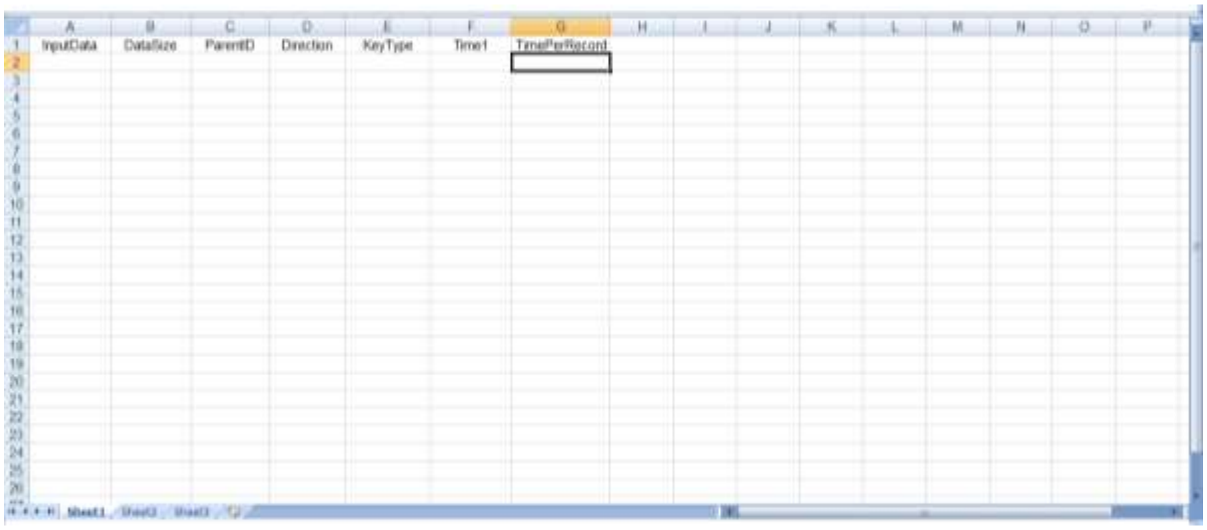
4.3.1 Implementation of MOPE Algorithm

Figure A.2 in Appendix A displays the interface of the MOPE algorithm program.

The output includes exporting two tables: the first table is the tree table that shown in Figure 4.1 and contains the following fields:

- InputData: represents the inserted data.
- DataSize: represents how many digits of InputData

- ParentID: represents the parent node (data) for the inserted data.
- Direction: represents the position of the inserted data according to its parent, if it is on the left or right of the ParentID or it is a root node if the data is the first insertion or became a root as a result of rebalancing operation.
- KeyType: represents the size of the encryption key used.
- TimePerRecord: represents the requested time for each transaction to stay in its right position in the OPE tree.
- Time1: represents the accumulated time for all insertion transactions such that the last record contains the total time.



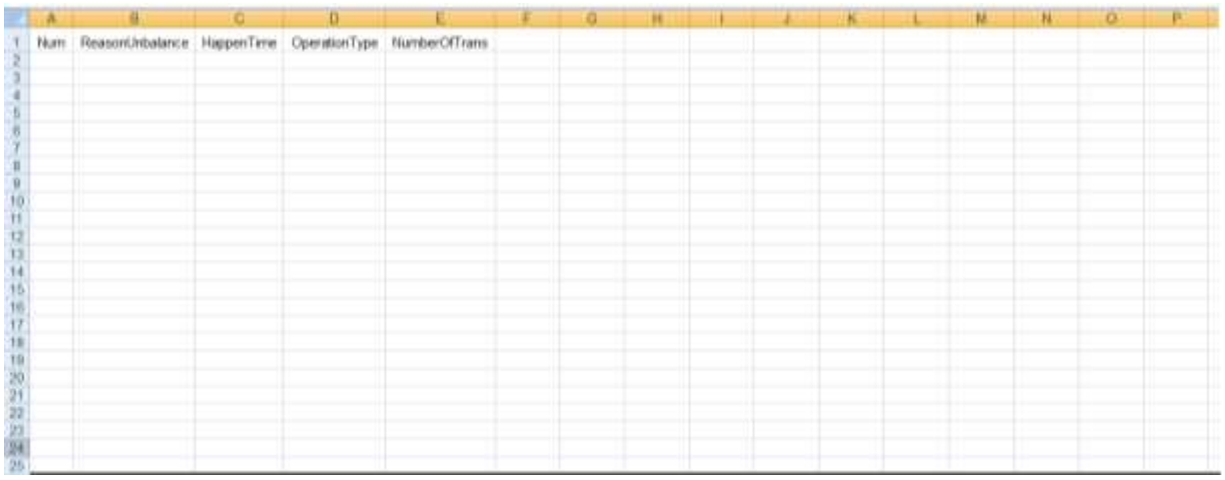
InputData	DataSize	ParentID	Direction	KeyType	Time1	TimePerRecord

Figure 4.1: Tree Table Exporting File

The second table contains the data that their insertion leads to the OPE tree to be unbalanced. Figure 4.2 shows the Unbalance table and its fields are described below:

- ResonUnbalance: represents the data that lead the OPE tree to be unbalanced.

- Num: represents the data node in the OPE tree that from it the rebalancing operation started to apply.
- HappenTime: represents the time of rebalancing (when the rebalancing of tree is executed).
- NumberOfTrans: represents the number of transactions that applied by the system.
- OperationType: represents the type of operation applied if it insertion or deletion.



Num	ReasonUnbalance	HappenTime	OperationType	NumberOfTrans

Figure 4.2: Unbalance Table Exporting File

4.3.2 Implementation of the Proposed Methodology

Our proposed methodology used two parameters (transaction and time windows) to change the encryption key. Many values will be tested and study the effect of these values on the performance of MOPES. Our goal is reaching to a proper value for changing the encryption key with an acceptable system performance. The program can be executed when click on the (Time-Transaction Operation) button in Figure A.1 in Appendix A. The interface of this program is shown in Figure B.1 in Appendix B.

4.4 Execution of MOPE Algorithm

In this section, the MOPE algorithm is executed in order to compute the running time, the number of rebalancing operations that will be executed by the system to insert or delete data, record the exact time of when the rebalancing operation was executed and with which data. This recorded information can be used in our proposed solution to estimate the proper time window and transaction window to change the encryption key.

Many experiments were tested and record the above parameters. The first experiment inserted 25 records of data with different digits sizes. The total time to insert 25 data records was approximately equal to 2 seconds; the tree was rebalanced 13 times to insert 25 data records. Second experiment inserted 50 data records and the total time increased to 4.6 seconds with 29 rebalancing operations. The following Table 4.1 shows the output where the field Exp No. represents the experiment Number, No. of Inserted Data represents the number of data inserted using experiment specified in the Exp No., No. of Rebalancing field contains how many times the rebalancing operation are called to rebalance the tree, the Total time field represents the output which contains the time needed to execute this experiment.

Table 4.1: Execution of Experiments for Insertion Transaction in MOPE Program

Exp No.	No. of Inserted Data	No. of Rebalancing	Total Time (Sec)
Exp1	25	13	2.1
Exp2	50	29	4.6
Exp3	100	51	10.8
Exp4	200	97	21.1
Exp5	400	208	44
Exp6	500	251	82.5
Exp7	700	348	165.3
Exp8	1000	515	322.6

As we see from the table, the number of times rebalancing operation were called are increased with the number of inserted data. Figure 4.3 clarified how the number of rebalancing operations is increased with number of data inserted. The X-axis represents the number of inserted data and the Y-axis represents the number of rebalancing operations.

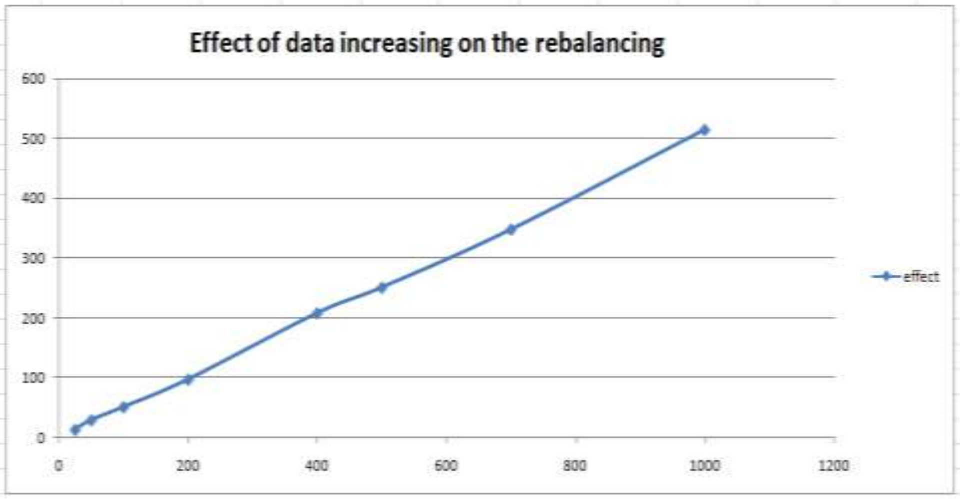


Figure 4.3: Effect of Data Increasing on the Rebalancing

Also from the Table 4.1 the increasing of inserted data leads to increase in the total time. Figure 4.4 shows that such that the X-axis represents the number of inserted data and the Y-axis represents the number of running time.

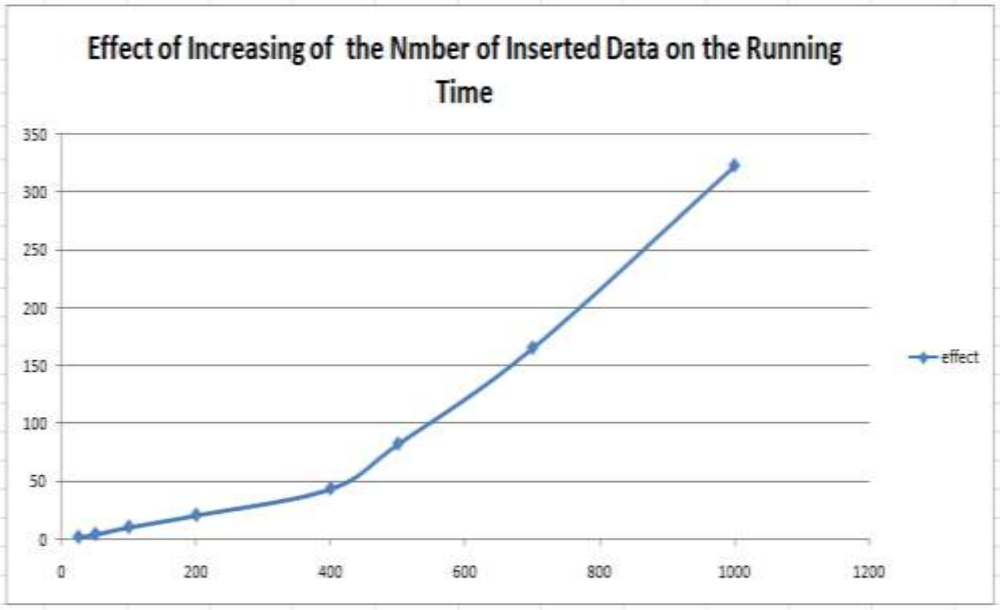


Figure 4.4: Effect of Data Increasing on the Total Time

Many experiments were used to test the deletion transaction for the MOPE algorithm. These experiments took the Exp3 in the Table 4.1 as a sample and apply its test on the data of this experiment. Exp3 from Table 4.1 includes insert 100 data records to the OPE tree with 51 rebalancing operations and 10.8 seconds running time. We chose Exp3 for many reasons: we executed our program on the device with limited options and we did not use real server or real database. Also, when we executed insertion transaction with different numbers of data and measured the running time, we found that there is a linear relationship between the number of inserted data and the produced running time such that when 100 data records are inserted, the running time was equal to 10.8 seconds and when 200 data records are inserted, the running time was equal to 21.1 seconds and so on. Finally, our goal is proof of concept. Thus, we carried out our experiments with 100 data records.

To delete data from this tree we need to import the data that requested to delete from an excel sheet. We stepped to delete the data from 5 data records to 80 data records from 100 data records and study the effect of deletion transaction on the running time as shown in Table 4.2. The first column Exp No. represents the experiment number, the second column named No. of Deleted Data, column three represents how many rebalancing operation needed for that deletion. This column named No. of Rebalancing, and the last column named Total Time measures the running time.

Table 4.2: Execution of Experiments for Deletion Transaction in MOPE Program

Exp No.	No. of Deleted Data	No. of Rebalancing	Total Time
Exp 1	5	2	0.151
Exp2	10	3	0.410
Exp3	20	5	0.707
Exp4	40	10	1.258
Exp5	60	19	2.310
Exp6	80	25	3.152

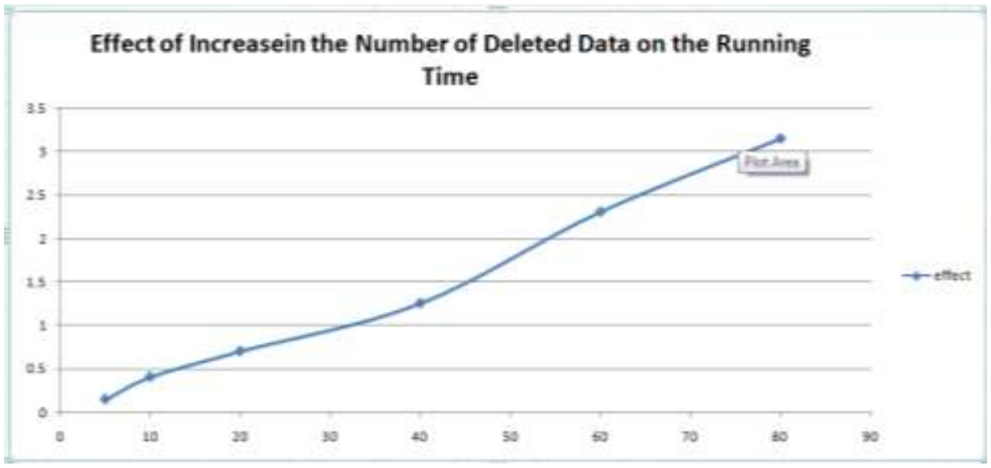


Figure 4.5: Effect of Deleted Data Increasing on the Total Time

In Case of Deletion

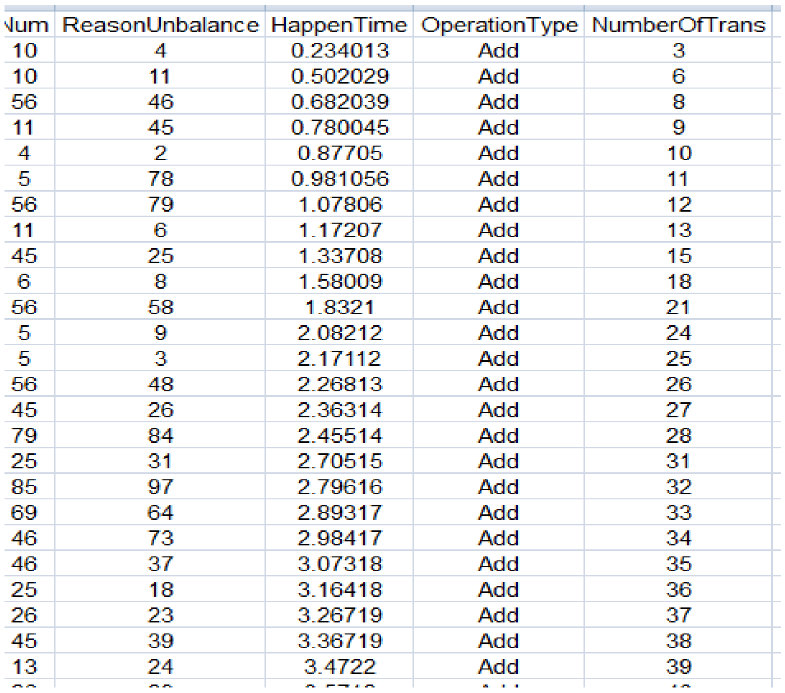
This Figure 4.5 clarify how the number of deleted data effect on the running time such that X-axis represents the number of deleted data and Y-axis represents the running time.

4.5 Execution of the Proposed Methodology using Transaction Window

Our idea in this research is to change the encryption key with each time the rebalancing operation is executed. The time that already consumed to rebalance the OPE tree will be consumed to change the key (to enhance the security level). As it is known changing the encryption key leads to change the structure of the OPE tree. Thus, the time consumed to rebalance the OPE tree that needs from time to time to preserve the order of the data will be utilized to enhance the security level of OPES through changing the encryption key and restructuring the OPE tree. To achieve this idea, two parameters used time window and transaction window. In this section transaction window parameter will be discussed.

Transaction window refers to the number of transactions (data entry) that applied by the system. The proper number of transactions to change the encryption key will be estimated depending on analysis one of the experiments in Table 4.1. Exp3 will be discussed as an example.

In this experiment, 100 data records were inserted. The total time was needed is approximately equal to 10.8 second with 51 rebalancing operations as shown in Table 4.1.



Num	ReasonUnbalance	HappenTime	OperationType	NumberOfTrans
10	4	0.234013	Add	3
10	11	0.502029	Add	6
56	46	0.682039	Add	8
11	45	0.780045	Add	9
4	2	0.87705	Add	10
5	78	0.981056	Add	11
56	79	1.07806	Add	12
11	6	1.17207	Add	13
45	25	1.33708	Add	15
6	8	1.58009	Add	18
56	58	1.8321	Add	21
5	9	2.08212	Add	24
5	3	2.17112	Add	25
56	48	2.26813	Add	26
45	26	2.36314	Add	27
79	84	2.45514	Add	28
25	31	2.70515	Add	31
85	97	2.79616	Add	32
69	64	2.89317	Add	33
46	73	2.98417	Add	34
46	37	3.07318	Add	35
25	18	3.16418	Add	36
26	23	3.26719	Add	37
45	39	3.36719	Add	38
13	24	3.4722	Add	39

Figure 4.6: Sample of Unbalance Table for Exp3

Figure 4.6 includes sample of Unbalance table when Exp3 was executed in Table 4.1. The NumberOfTrans field contains the number of transactions (number of data entry) when the rebalancing was executed. In our proposed methodology we need to count the number of transactions since last restructuring time and store in *J*. Then the value of *J* compares with transaction window value *W* in order to take the decision if the encryption key is changed or not. Now, the values of transaction window *W* were concluded from Table 4.10 by subtract each value in NumberOfTrans field from the next value of the same field. By applying this equation, we concluded the transaction window values as following: 1, 2, 3, 5, 6, 7, 8. Then the system will be executed with each value. The total time will be computed with each transaction window value in order to reach to a proper value for transaction window for changing the encryption key with an acceptable performance. Table 4.3 shows how insertion algorithm using different transaction window values applied by the system.

Table 4.3: Execution of Insertion Algorithm using Different Transaction Windows

Exp No.	No. of Inserted Data	Transaction Window	No. of Restructuring	Total Time
Exp1	100	1	65	543.374
Exp2	100	2	59	518.805
Exp3	100	3	58	280.441
Exp4	100	5	55	276.835
Exp5	100	6	55	243.503
Exp6	100	7	53	234.306
Exp7	100	8	52	222.071

From the Table 4.3 we can see that when the transaction window is decreased, the running time is increased for the same number of inserted data. When the transaction window value increased the total time is decreased as shown in Figure 4.7. In this figure the X-axis represents the transaction window values and Y-axis represents the running time.

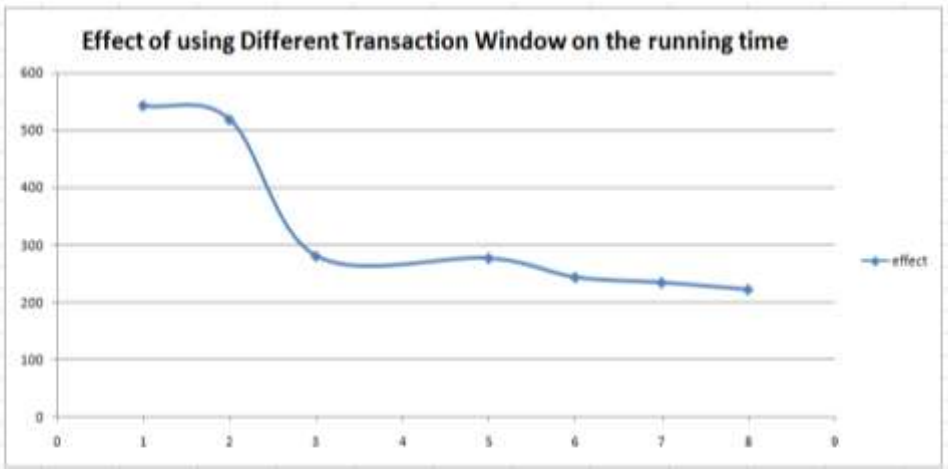


Figure 4.7: Effect of using Different Transaction Windows on the Running Time in Case Of Insertion

Also increasing transaction window value leads to decreasing the number of restructuring. Figure 4.8 clarifies this relation such that the X-axis represents the transaction window values and Y-axis represents the number of restructuring operations.

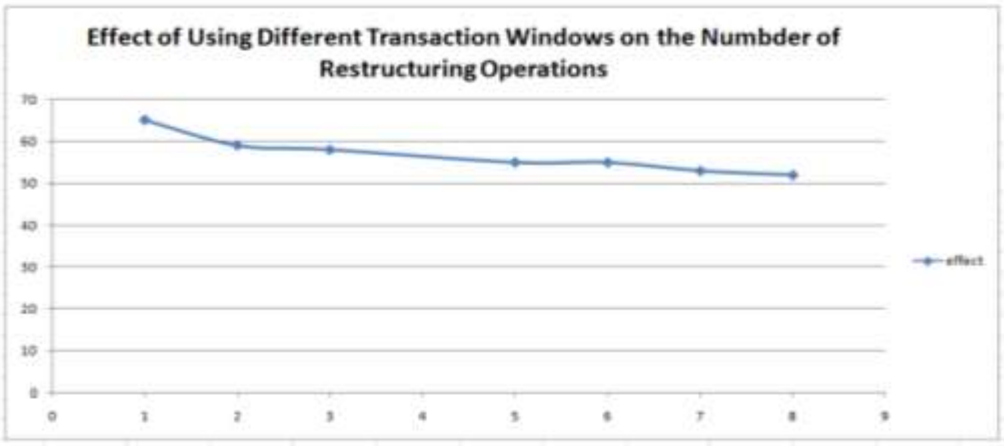
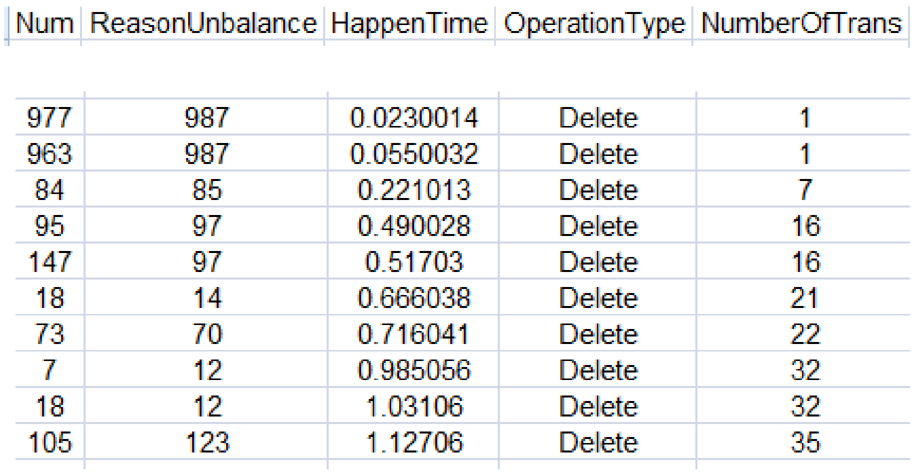


Figure 4.8: Effect of using Different Transaction Windows on the Number of Restructuring in Case of Insertion

To apply deletion transaction using transaction window, Exp4 from Table 4.2 will be tested as a sample. This experiment includes delete 40 data records from 100 data records with result 10 rebalancing operations and 1.258 second running time. Figure 4.9 contains unbalance table for this experiment.



Num	ReasonUnbalance	HappenTime	OperationType	NumberOfTrans
977	987	0.0230014	Delete	1
963	987	0.0550032	Delete	1
84	85	0.221013	Delete	7
95	97	0.490028	Delete	16
147	97	0.51703	Delete	16
18	14	0.666038	Delete	21
73	70	0.716041	Delete	22
7	12	0.985056	Delete	32
18	12	1.03106	Delete	32
105	123	1.12706	Delete	35

Figure 4.9: Sample of Unbalance Table for Exp4 in Table 4.10

In the Figure 4.9 the NumberOfTrans field refers to the number of data entry deleted before current deletion transaction. In deletion with transaction window, two conditions must be checked if the OPE tree still balanced after delete data. First condition checks the number of transactions (insertion or deletion) since last restructuring time. Then compare its value with transaction window value to take decision if the encryption key will be changed or not. If this condition does not satisfy then go to the second condition that checks the number of deletion transaction since last time of restructuring and store its value in a variable *S*. Then, the value of *S* checked with the value of transaction window *Z* in order to take the decision if the encryption key will be changed or not. Z refers to the transaction window that holds the number of deletion transactions since last time of restructuring. The values of Z were concluded by subtract each value in NumberOfTrans field from the next value of the same field in Figure 4.9. The transaction window generated from applying this equation takes the following values: 1, 3, 5, 6, 9, and 10. In our experiment we concentrated on the second condition and remain the first condition constant since we test deletion transaction.

Our experiment now is to delete 40 data records from 100 data records using different deletion transaction windows (1, 3, 5, 6, 9, and 10) and compute the running time for each value that will be applied. Table 4.4 shows the result.

Table 4.4: Execution of Deletion Algorithm using Different Transaction Windows

Exp No.	No. of Deleted Data	Transaction Window	No. of Restructuring	Total Time
Exp1	40	1	35	316.616
Exp2	40	3	20	299.932
Exp3	40	5	12	274.873
Exp4	40	6	12	271.661
Exp5	40	9	11	260.552
Exp6	40	10	11	228.616

From Table 4.4 we can note how the values of transaction window effect on the running time such that when the transaction window holds small value, the running time holds high value (running time becomes worst) and each time the value of transaction window becomes higher the running time becomes better. Show Figure 4.10 shows the relation between transaction window values and running time such that the X-axis represents the transaction window values and Y-axis represents the running time.

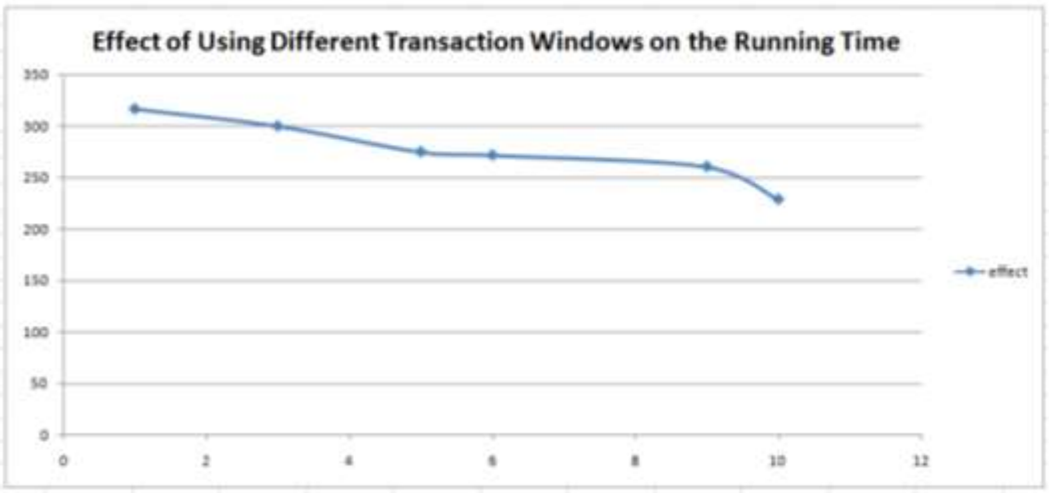


Figure 4.10: Effect of using Different Transaction Windows on the Running Time in Case Of Deletion

Also we can see from Table 4.4 the effect of increasing number of transactions on the number of restructuring operations such that when the number of transactions is increased , the number of restructuring operations is decreased and vice versa. Figure 4.11 clarifies this relation such that the X-axis represents the transaction window values and Y-axis represents the number of restructuring operations.

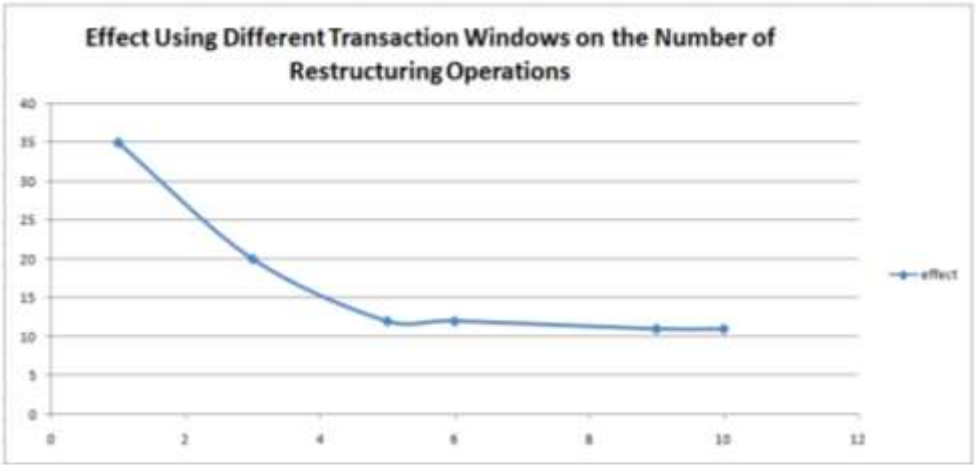


Figure 4.11: Effect of using Different Transaction Windows on the Number of Restructuring Operations in Case Of Deletion

As we note the running time from Table 4.3 which includes insert 100 data records using different transaction windows. The running time is very huge in comparison with the running time of inserting the same number of data records using MOPE algorithm. The running time using MOPE algorithm was 10.8 seconds while using our proposed solution with different transaction window values (1, 2, 3, 5, 6, 7, and 8) the running time is ranged from 543.374 to 222.071 seconds. It is also noted that the time difference for each experiment in Table 4.3 is not logical. For example see EXP2 with 59 restructuring takes 518.805 seconds while EXP3 with 58 restructuring takes 280.441 seconds.

After analyzing the problem, we noted that the restructuring problem was to delete all data and to insert them one by one using the same MOPE algorithm. That means each time the tree needs to restructure, the rebalancing operations will be executed for each insertion. Thus, the alternative solution is when the tree needs to restructure, we do not need to use MOPE algorithm to rebuild the tree since all the inserted data records now is known to the client. We can encrypt all data at the same time, rebuild the tree in its balanced form and send them to the server. Building tree in its balanced form will be done by using quick sort algorithm. In this way we can avoid the huge running time that resulted in Table 4.3 Tables 4.5a, 4.5b explain the result of this enhancement on the restructuring operation.

Table 4.5 a: Execution of Insertion Algorithm using Different Transaction Windows With Enhancement of Restructuring Operation

Exp No.	No. of Inserted Data	Transaction Window	No. of Restructuring	Total Time
Exp1	100	1	65	21.52
Exp2	100	2	59	20.4
Exp3	100	3	58	19.5
Exp4	100	5	55	18.2
Exp5	100	6	55	17.5
Exp6	100	7	53	16.4
Exp7	100	8	52	15.9

Table 4.5 b: Execution of Deletion Algorithm using Different Transaction Windows With Enhancement of Restructuring Operation

Exp No.	No. of Deleted Data	Transaction Window	No. of Restructuring	Total Time
Exp1	40	1	35	3.5
Exp2	40	3	20	2.7
Exp3	40	5	13	1.9
Exp4	40	6	12	1.8
Exp5	40	9	11	1.6
Exp6	40	10	11	1.5

4.6 Execution of the Proposed Solution using Time Window

As we explained in section 4.5, to achieve the idea of enhanced security level of OPES and study the effect of this enhanced on the performance, two parameters are used transaction and time windows. Transaction window parameter was discussed in section 4.5. In this section time window will be discussed. Time window refers to the time duration to change the encryption key. The proper time window to change the encryption key will be estimated depending on the analysis the experiments in the Table 4.1. Exp3 will be analyzed.

Exp3 is an experiment to insert 100 data records with total time equal to 10.8 seconds and 51 rebalancing operations as shown in Table 4.1.

When we note Figure 4.6 which contains the exported sheet of execution Exp3, this table includes the data records that lead to execute rebalancing operation in ReasonUnbalance field, and the time when the rebalancing operations is executed stored in HappenTime field.

Table 4.6: Execution of Insertion Algorithm using Different Time Windows

Exp No.	No. of Inserted Data	Time Window	No. of Restructuring	Total Time
Exp1	100	0.1	70	441.806
Exp2	100	0.2	67	433.747
Exp3	100	0.3	59	406.084
Exp4	100	0.4	57	332.530
Exp5	100	0.6	53	269.036
Exp6	100	0.7	52	223.142
Exp7	100	0.8	52	221.009

In insertion with time window we need to check one condition that states when the last time of restructuring and store its value in a variable *T*. *T* then is compared with the value of time window N_0 to decide if the encryption key is changed or not. The values of time window N_0 were concluded by measure the time difference between each two sequential rebalancing operations in Figure 4.6. To compute the values of N_0, subtract each value in HappenTime field from the next value of the same field. When we applied this equation on the HappenTime column in Figure 4.6, the values of N_0 were equal to 0.1, 0.2, 0.3, 0.6, 0.7 and 0.8.

The time window will take these values sequentially to execute inserting 100 data records. Running time will be computed with each time window used. Table 4.6 shows the result.

From Table4.6 we can see how the value of time window effect on the running time such that when the value of time window increases the running time decreases and vice versa. Figure 4.12 and Figure 4.13 show the effect of time window values on the number of restructuring operations and running time.

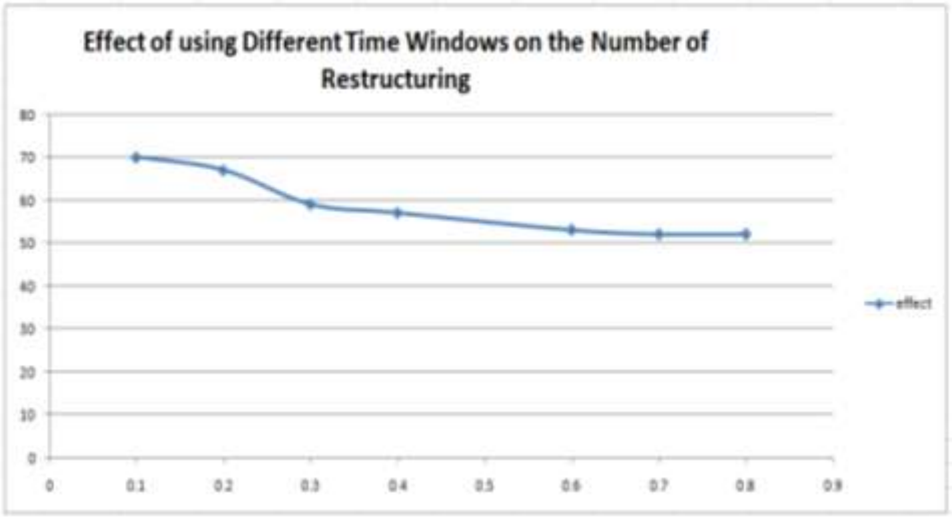


Figure 4.12: Effect of using Different Time Windows on the Number of Restructuring in Case of Insertion

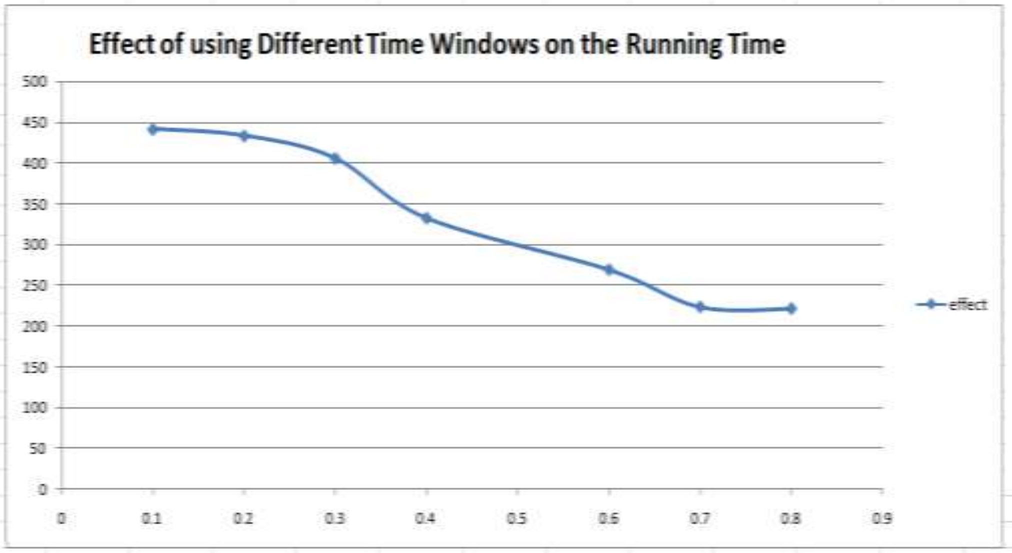


Figure 4.13: Effect of using Different Time Windows on the Running Time in Case of Insertion

As we know deletion with time window needs to check two conditions if the OPE tree still balanced after delete data. The first condition checks when the last time of restructuring and stores the value in a variable *T*. Then *T* compares with specific time window value that store in *X* in order to decide if the encryption key need to change or not. The values of *X* were concluded from HappenTime field in Figure 4.9 by subtract each value in this field from the next value of the same field. The second condition checks the number of deletion transaction since last restructuring time and store in a variable *S*. Then *S* compares with a specific transaction window value *Z* that holds the number of deletion transactions between each two sequential rebalancing operations. In our experiment we focused on the first condition since we are testing the time window and remain the value of *Z* constant.

The values of *X* are (0.1, 0.2, 0.3, 0.4, 0.6, 07, and 0.8) will be tested to delete 40 data records from 100 data records. The running time will be recorded with each time window value that will be used. Table 4.7 shows the result.

Table 4.7: Execution of Deletion Algorithm using Different Time Windows

Exp No.	No. of Inserted Data	Time Window	No. of Restructuring	Total Time
Exp1	40	0.1	30	252.334
Exp2	40	0.2	32	255.605
Exp3	40	0.3	32	256.122
Exp4	40	0.4	28	253.507
Exp5	40	0.6	33	255.369
Exp6	40	0.7	34	257.231
Exp7	40	0.8	32	254.396

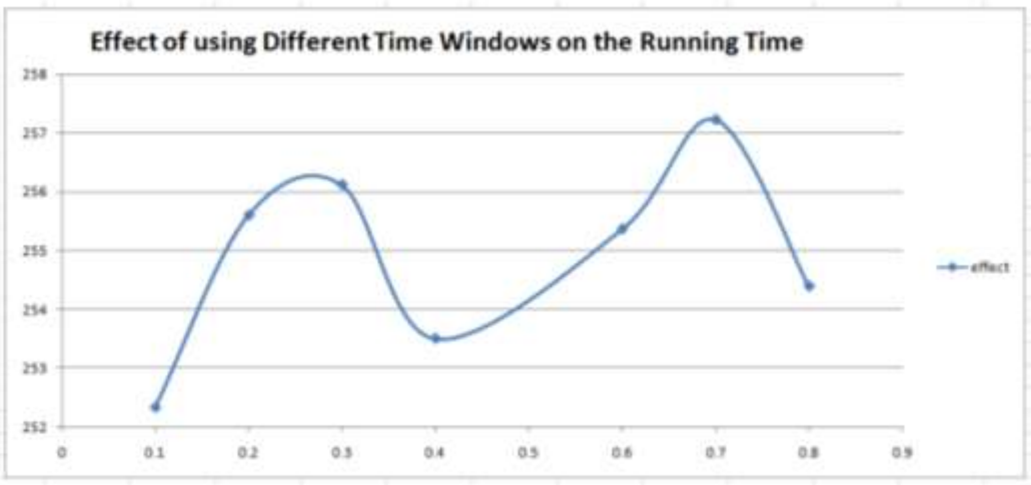


Figure 4.14: Effect of using Different Time Windows on the Running Time
in Case of Deletion

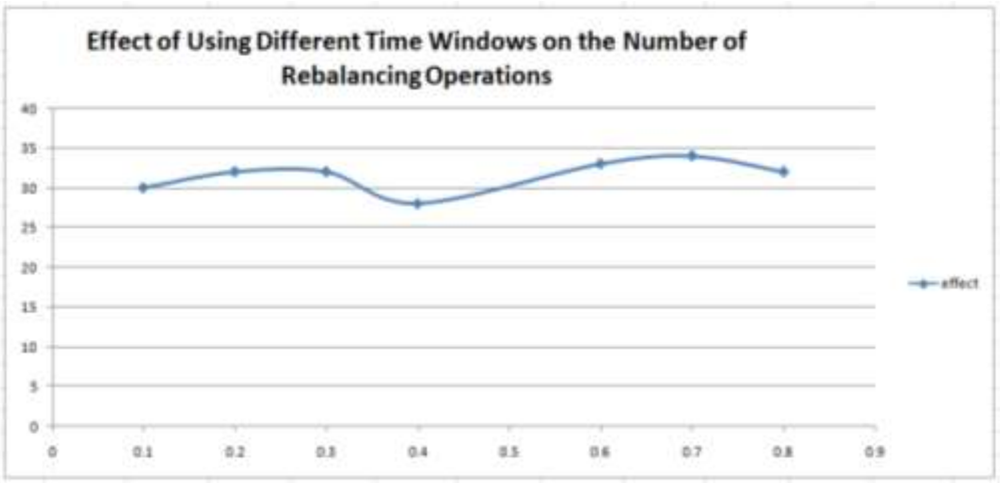


Figure 4.15: Effect of using Different Time Windows on the Number of Restructuring
operations in Case of Deletion

As we note the total time column in Table 4.7 represents the running time to delete 40 data records from 100 data records using different time windows. The running time was increased with the increasing of time duration in time window. This increasing is done since in our research we used DET encryption techniques which suffer from leakage of data especially with deletion transaction. Thus, we added two conditions to prevent or reduce this leakage. First condition checks when the last time of restructuring and compare with the time window value and according to this comparison the decision is taken if the encryption key is changed or not. The second condition check the number of deletion transactions since last restructuring and compare to specific number that we concluded according to our experiments. According to the result of comparison the decision is taken to encryption key is changed or not.

Thus, there is a big probability to do the restructuring operation many time and leads to increasing of running time since there is a mixed between time and transaction criteria such that if the first condition not satisfied then there is a big probability for second condition to be satisfied..

As we note form Table 4.6 which includes insert 100 data records using different time window. We note that the running time is very huge in comparison with the running time when inserting the same number of data using MOPE algorithm. The running time using MOPE algorithm was 10.8 seconds while using our proposed methodology with different time window values 0.1, 0.2, 0.3, 0.6, 0.7, and 0.8 the running time is ranged from 441.806 to 221.009 seconds.

After analyzing the problem, we noted that the restructuring problem was to delete all data and to insert them one by one using the same MOPE algorithm. That means each time the tree needs to restructure, the rebalancing operations will be executed for each insertion. The solution is when the tree needs to restructure, we do not need to use the MOPE algorithm to rebuild the tree since all the inserted data records became known to the client. The tree can be built using quick sort algorithm and store it in its balanced form in the server. In this way we can avoid the huge running time that produced in Table 4.6 Following Tables 4.8 show how the running will be reduced using quick sort algorithm in restructuring process.

Table 4.8: Execution of Insertion Algorithm using Different Time Windows With Enhancement of Restructuring Operation

Exp No	No. of Inserted Data	Time Window	No. of Restructuring	Total Time
Exp1	100	0.1	70	20.97
Exp2	100	0.2	67	19.85
Exp3	100	0.3	59	19.03
Exp4	100	0.4	57	18.70
Exp5	100	0.6	53	17.66
Exp6	100	0.7	52	17.08
Exp7	100	0.8	52	16.32

Also the running time with deletion transaction in Table 4.7 produce a huge time to delete 40 data records form 100 data records. The solution is the same in the insertion that includes restructure the tree using quick sort algorithm since the values of the tree became known to the client. Thus, the client can directly encrypt all data, sorting using quick sort algorithm and store them in their balanced form in the server. Table 4.9 show how the running time will be reduced when enhanced the process of restructuring.

Table 4.9: Execution of Deletion Algorithm using Different Time Windows With Enhancement of Restructuring Operation

Exp No.	No. of Deleted Data	Time Window	No. of Restructuring	Total Time
Exp1	40	0.1	30	1.66
Exp2	40	0.2	32	1.79
Exp3	40	0.3	32	1.90
Exp4	40	0.4	28	1.71
Exp5	40	0.6	33	1.80
Exp6	40	0.7	34	2.22
Exp7	40	0.8	32	2.04

4.7 Discussion of the Results

Many experiments are built and tested in previous sections using different values for time and transaction windows. We found that transaction window very closed to time window such that the time window provided 20.97 second running time and the transaction window provided 21.52 second running time. Thus, the difference is very small. These approximated results belong to that we did not apply real data in our experiments; we assumed these data were imported from real database. Also, in real life (real database), the insertion or deletion (transaction)will not happened in the same time while in our experiments we force to enter 100 data records at the same time.

As we clarified above the time window and transaction window are closed to each other. In real life, transactions happened with different time depends on the application domain. Some application transactions happened in very fast manner (for example, stock market system) while other applications have long time between transactions (for example, changing basic information for students). Thus, this research reaches to that the time window should be interested with regards to the application domain to be useful. It is not logical to give the same optimal time window for all applications. This research will extend and investigate this idea later in the future work.

In this section, the obtained results will be analyzed to determine how much security level was gained and how much performance was loosed to define the optimal mechanism to change the encryption key that provides minimal database reindices time. Two comparisons will be done between the MOPE algorithm running time and the transaction window parameter running time. And between the MOPE algorithm running time and the time window parameter running time.

- To compare between running time in MOPE algorithm and proposed transaction window, the following Table 4.10 clarify the running time for inserting100 data record in MOPE algorithm and proposed transaction window when transaction window holds the minimum value equal to one which refer to high security and maximum value equal to eight which refer to low security.

Table 4.10: Explaining the Running Time of MOPE Algorithm and Proposed Transaction Window in Insertion Transaction

MOPE algorithm Running Time	Proposed Transaction Window Running Time	Transaction Window Value	Security Level
10.8	21.52	1	High
10.8	15.9	8	Low

We can note in transaction window how the running time reached to maximum value when using transaction window equal to one that provide high security. And the running time reached to minimum value when using transaction window equal to eight that provide low security.

To measure the loss of performance the following rule will be applied:

Loss of Performance= $\frac{\text{running time of transaction window} - \text{running time of MOPEalgorithm}}{\text{running time of transaction window}}$

1. When transaction window =1

The loss of Performance = $\frac{21.52 - 10.8}{21.52}$ = 0.498 = 50%

2. When transaction window = 8

The loss of Performance =$\frac{15.9 - 10.8}{15.9}$ = 0.320 = 33%

Table 4.11 clarified the result.

Table 4.11: Explaining the percentage of Losing of Performance to Get Optimal Mechanism with Insertion in Proposed Transaction Window

Num	Proposed Transaction Window Running Time	Transaction Window Value	Security Level	Losing of Performance
1	21.52	1	High	50%
2	15.9	8	Low	33%

To compare between running time in MOPE algorithm and proposed transaction window when we delete 40 data records from 100 data records. Table 4.12 clarify the running time required to delete 40 data records from 100 data records in MOPE algorithm and proposed transaction window when holds the minimum value equal to one which refers to high security level and maximum value equal to ten which refers to low security.

Table 4.12: Explaining the Running Time of MOPE Algorithm and Proposed Transaction Window in Deletion Transaction

MOPE algorithm Running Time	Proposed Transaction Window Running Time	Transaction Window Value	Security Level
1.258	3.5	1	High
1.258	1.5	10	Low

To measure the loss of performance the following rule must be applied:

Loss of Performance=

$$\frac{\text{running time of transaction window} - \text{running time of MOPE algorithm}}{\text{running time of transaction window}}$$

1. When transaction window = 1

The loss of Performance = $\frac{3.5 - 1.258}{3.5}$= 2.2424/3.5 = 0.640 = 64 %

2. When transaction window = 10

The loss of Performance = $\frac{1.5 - 1.258}{1.5}$= 0.242/1.5 = 0.161 = 16 %

Table 4.13clarified the result. Table 4.13: Explaining the percentage of Losing of Performance to Get Optimal Mechanism with Deletion in Proposed Transaction Window

Num	Proposed Transaction Window Running Time	Transaction Window Value	Security Level	Losing of Performance
1	3.5	1	High	64 %
2	1.5	10	Low	16 %

- To compare between running time in MOPE algorithm and proposed time window, the following Table 4.14 clarify the running time for inserting 100 data record in MOPE algorithm and proposed time window when time window holds the minimum value equal to 0.1 millisecond which refer to high security and maximum value equal to 0.8 millisecond which refer to low security.

Table 4.14: Explaining the Running Time of MOPE Algorithm and Proposed Time Window in Insertion Transaction

OPE algorithm Running Time	Proposed Time Window Running Time	Time Window Value	Security Level
10.8	20.97	0.1	High
10.8	16.32	0.8	Low

We can note in time window how the running time reached to maximum value when using time window equal to 0.1millisecond that provided high security. And the running time reached to minimum value when using time window equal to0.8 millisecond that provided low security.

To measure the loss of performance the following rule will be applied:

Lose of Performance= $\frac{\text{running time of time window} - \text{running time of MOPE algorithm}}{\text{running time of time window}}$

1. When time window = 0.1

The loss of Performance = $\frac{20.97 - 10.8}{20.97}$= 0.484 = 49%

2. When time window = 0.8

The loss of Performance = $\frac{16.32- 10.8}{16.32}$= 0.3380 = 34%. Table 4.15 shows the results.

Table 4.15: Explaining the percentage of Losing of Performance to Get Optimal Mechanism in Proposed Time Window with Insertion

Num	Proposed Time Window Running Time	Time Window Value	Security Level	Losing of Performance
1	20.97	0.1	High	49%
2	16.32	0.8	Low	34%

Now to compare between running time in MOPE algorithm and proposed time window to delete 40 data records form 100 data records. Table 4.16 explains the running time required to delete 40 data records form 100 data records in MOPE algorithm and proposed time window when holds minimum value equal to 0.1 millisecond which refers to high security level and maximum value equal to 0.8 millisecond which refers to low security.

Table 4.16: Explaining the Running Time of MOPE Algorithm and Proposed Time Window in Deletion Transaction

MOPE algorithm Running Time	Proposed Time Window Running Time	Time Window Value	Security Level
1.258	1.66	0.1	High
1.258	2.04	0.8	Low

To measure the loss of performance the following rule will be applied:

Loss of Performance=$\frac{\text{running time of time window} - \text{running time of MOPE algorithm}}{\text{running time of time window}}$

1. When time window = 0.1

The loss of Performance =$\frac{1.66 - 1.258}{1.66}$ = 0.402/1.66 = 24 %

2. When time window = 0.8

The loss of Performance = $\frac{2.04 - 1.258}{2.04}$= 0.782/2.04 = 38 %

Table 4.17 shows the results.

Table 4.17: Explaining the percentage of Losing of Performance to Get Optimal Mechanism in Proposed Time Window with Deletion

Num	Proposed Time Window Running Time	Time Window Value	Security Level	Losing of Performance
1	1.66	0.1	High	24 %
2	2.04	0.8	Low	38 %

As we see from Table 4.17 losing of performance for time window 0.1millisecond is very small. The reason is that when concluded the values of time window from Figure 4.13, we found that most of time difference between each rebalancing operations were 0.1 millisecond. Thus, when we used time window that holds that value, then the running time resulted 1.66 seconds very closed to the running time using MOPE algorithm 1.25 seconds.

Chapter Five
Conclusion and Future Work

Chapter Five

5.1 Conclusion

This research is based on the idea of improving the security level with an acceptable losing of performance for systems that used OPES. OPES is used to preserve the order of data while executing range queries over encrypted data but it leaks of data. MOPE replaced the DET encryption techniques to RND encryption techniques to satisfy the IND-OCPA security protocol in order to prevent leakage of data. In this research we supposed to change the encryption key instead of replacing DET encryption techniques to RND encryption techniques. We found that the best situation to change the encryption key was through utilizing the time consumed in rebalancing operation in MOPE algorithm to change the encryption key and restructure the whole tree in our methodology. Two parameters was used, time and transaction windows. The running time was studied with each parameter. The contribution of this research was the following:

1. The first contribution was about which window should be considered, time or transaction window? The answer for this question is that we found that transaction window very closed to time window such that the time window provided 20.97 second running time and the transaction window provided 21.52 second running time. Thus, the difference is very small.
2. The second contribution was about defining a mechanism for changing the encryption key using time or transaction window such that the reindexing time of database is optimal.

To find the suitable mechanism for changing the encryption key that provide an optimal database reindices time, two comparison were done between the MOPE algorithm running time from one side and the transaction and time window running time from other side.

We concluded the mechanism to change the encryption key that provides optimal database indices time when using transaction window were following:

- To insert 100 data records, when increase the security to its high level, losing of performance reach to 50%.
- To insert 100 data records, when decrease the security level to its low level, losing of performance reach to 33%.
- To delete 40 data records from 100 data records, when increase the security to its high level, losing of performance reach to 64 %
- To delete 40 data records from 100 data records, when decrease the security to its low level, losing of performance reach to 16 %

We concluded the mechanism to change the encryption key that provides optimal database indices time when using time window were following:

- To insert 100 data records, when increase the security to its high level, losing of performance reach to 49%.
- To insert 100 data records, when decrease the security level to its low level, losing of performance reach to 34%.
- To delete 40 data records from 100 data records, when increase the security to its high level, losing of performance reach to 24 %
- To delete 40 data records from 100 data records, when decrease the security to its low level, losing of performance reach to 38 %

3. To determine the proper time window to change the key that provide an acceptable performance for OPES, we concluded that the time window that holds the value equal to 0.4 millisecond provides an acceptable running time equal to 18.7 seconds to insert 100 data records in comparison with maximum running time equal to 20.97 seconds (when using minimum time window equal to 0.1) and minimum running time equal to 16.32 seconds (when using maximum transaction window equal to 0.8).

According to the proper time window to change the encryption key that provide an acceptable performance for OPES when delete 40 data records from 100 data records, we concluded that the time window that holds the value equal to 0.4 millisecond provides an acceptable running time equal to 1.71 second in comparison with maximum running time equal to 2.04 seconds (when using maximum time window equal to 0.8) and minimum running time equal to 1.66 second (when using minimum time window equal to 0.1).

4. To determine the proper transaction window to change the encryption key that provide an acceptable performance for OPES, we concluded that the transaction window that holds the value five transactions provides an acceptable running time equal to 18.2 seconds to insert 100 data records in comparison with maximum running time equal to 21.52 seconds (when using minimum transaction window equal to one) and minimum running time equal to 15.9 seconds (when using maximum transaction window equal to eight).

   According to the proper transaction window to change the encryption key that provide an acceptable performance for OPES when delete 40 data records from 100 data records, we concluded that the transaction window that holds the value equal to six transactions provides an acceptable running time equal to 1.8 second in comparison with maximum running time equal to 3.5 seconds (when using minimum transaction window equal to one) and minimum running time equal to 1.5 second (when using maximum time window equal to ten).

5.2 Future Work

Received results can be tested on real server and real database which give a better insight of which window (time or transaction) is worth to change the encryption key. Hopefully, to eliminate the leakage of data can be done by changing encryption key with other OPES algorithms. Also, changing of encryption key can be applied with RND encryption techniques.

Utilizing new formulas for changing the encryption key can enhance the security level with preserving the performance or acceptable degradation in performance.

References

Agrawal, R., Kiernan, J., Srikant, R., &Xu, Y. (2004).Order preserving encryption for numeric data.***In Proceedings of the ACM SIGMOD International Conference on Management of Data***, 563-574.

Al Shehri, W. (2013). Cloud Database as a Service. ***International Journal of Database Management Systems (IJDMS),*** *5*(2).

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Ketz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, L. &Zaharia, M. Above the cloud: A Berkeley view of cloud computing. ***University of California at Berkeley***.Technical report UCB/ EECS-2009-28.

Atayero, A.A. &Feyisetan, O. (2009). Security issues in cloud computing: The potentials of homomorphic encryption. ***Journal of Emerging Trends in Computing and Information Sciences.****2*(10),546-552.

Bellare, M., Boldyreva, A., & O'Neill, A. (2007).Deterministic and efficiency searchable encryption.***CRYPTO 07 Proceeding.***Lecture notes in Computer Science, Springer, 4622, 535-552.

Boldyreva, A., Chenette, N., & O'Neill, A. (2011). Order-preserving encryption revisted: Improved security analysis and alternative solutions. ***CRYPTO 2011, 31st Annual International Cryptology Conference, P.Rogawayed.***,Springer Heidelberg, 578-595.

Boldyreva, A., Chenette, N., Lee, Y., & O'Neill, A. (2009).Order-preserving symmetric encryption.***A.Joux (Ed): EUROCRYPT 2009, International Association for Cryptologic Research,***SpringerHeidelberg, 5479, 224-241.

Hamdaqa, M., &Tahvildari, L. (2012).Cloud computing uncovered: A research landscape.***Software Technologies Applied Research (RTAR) Group.***University of waterloo, 85,0065-2458, Waterloo, Ontario Canada.

Hassan, Q.F. (2011). Demystifying cloud computing. ***Faculty of Computers and Information,***Mansoura University, Egypt.

Hurth, A. &Cebula, J. (2011). The basics of cloud computing. ***United States Computer Emergency Readiness Team (UC-CERT),*** Carnegie Mellon University, 88-145.

Kahdem, H., A, Amagasa, T., & Kitagawa, H. (2010). MV-OPES: Multivalued-Order Preserving Encryption Scheme: A novel scheme for encryption integer value to many different values. ***IEICE Transactions on Information and Systems***, E93-D(9), 2520-2533, Tokyo, Japan

Kahdem, H., Amagasa, T., & Kitagawa, H. (2010).A secure and efficient order preserving encryption scheme for relational databases.***In International Conference on Knowledge Management and Information Sharing (KMIS),*** 25-35, Valencia, Spain.

Kerschbaum, F. (2015).Frequency-hiding order-preserving encryption.***Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communication Security***,656-667, New York, USA.

Lin, Y., Yang, L., Lin, L., and Chen, Y. (September 2014). Preserving privacy in outsourced database.***International Journal of Computer and Communication Engineering,*** 3(5), DOI: 10.7763/IJCCE.2014.V3.350.

Liu, Z., Chen, X., Yang, J., Jia, C., & You, I. (2014 July).New order preserving encryption model for outsourced databases in cloud environment.***Journal of Network and Computer Applications*.**

Mahajan, P. &Sachdeva, A. (2013).A study of encryption algorithms AES, DES and RSA for security.***Global Journal of Computer Science and Technology Network, Web & Security,***13(15), Version 1.0, Type: Double Blind Peer Reviewed International Research Journal, Publisher: Global Journals Inc. (USA).

Masood, R., Shibli, A, Mehak, F., Ghazi, Y., and Khan, S. (January 2014). Security aspects of DataBase-as-a-Service (DBaaS) in cloud computing. ***Springer International Publishing***, 297-324, Pakistan.

Ozsoyoglu, G., Singer, D.A., & Chung, S.S. (2003). Anti-tamper databases: querying encrypted database. ***In 17th Annual IFIP WG 11.3 Working Conference on Database and Applications Security*.**

Pfaff, B. (2004). An introduction to binary search trees and balanced trees.***Free Software Foundation,*** 1, version 2.0.2.

Pfenning, F. (2011). Lecture notes on AVL trees. ***Principle of Imperative Computation,*** Lecture 8, 15-122.

Pfenning, F. (February, 2013). Lecture notes on quicksort. ***Principles of Imperative Computation,***Lecture 8, 15-122.

Popa, R.A., Li, F.H. &Zeldovich, N. (2013).An Ideal-Security protocol for order-preserving encoding.***In IEEE Symposium on S&P,*** 463-477, San Francisco, CA.

Popa, R.A., Redfield, C.M., S., Zeldovich, N. &Balakrishnan, H. (2011). CryptoDB: Protecting confidentiality with encrypted query processing. ***In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles,***85-100.

Reddy, K.S., &Ramachandram, S. (2014). A new randomized order preserving encryption scheme.***International Journal of Computer Application,*** 108, (12).

Soofi, A.A., Irfan Khan, M., Talib, R., and Sarwar, U. (March 2014). Security issues in SaaS delivery model of cloud computing. ***International Journal of Computer Science and Mobile Computing,*** *3*(3), 15-21.

Soofi, A.A., Khan, M.I. & Amin, F. (2014).Encryption techniques for cloud data confidentiality.***International Journal of Grid Distribution Computing,*** *7*(4),11-20.

Stinson, D. R. (2006).*Cryptography theory and practice (3rd ed).*Chapman & Hall/CRC.

Stoimen's web Log. (2012). Computer algorithms: a binary search tree.http://www.stoimen.com/blog/2012/06/22/computer-algorithms-binary-search-tree-data-structure/

Stoimen's web Log. (2012). Computer algorithms: balancing a binary search tree. http://www.stoimen.com/blog/2012/07/03/computer-algorithms-balancing-a-binary-search-tree/

Subashini, S. &Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing**. *Journal of Network and Computer Applications***, *34*(1), 1-11.

Sunitha, K. &Prashantha, S., K. (2013). Enhancing privacy in cloud service provider using cryptographic algorithm.***IOSR Journal of Computer Engineering,*** *12*(5), 62-64.

Tebaa, M., El Haji, S. & El Ghazi, A. (2012, 14 July). Homomorphic encryption applied to the cloud computing security. ***Proceedings of the World Congress on Engineering,****1*, 536-539, London, UK.

Xiao, L., Yen, I. & Huynh, D., T. (2012).Extending order preserving encryption for multi-user systems.***IACR cryptology ePrint Archine,***192.

Appendix

Appendix A

Implementation of MOPE Algorithm

The main interface of the program clarify the implementation of the MOPE algorithm in the first button (Add-Delete Tree Table) **(Popa, Li, & Zeldovich, 2013)** and our proposed methodology in the second button (Time-Transaction Operation), as shown in Figure A.1.

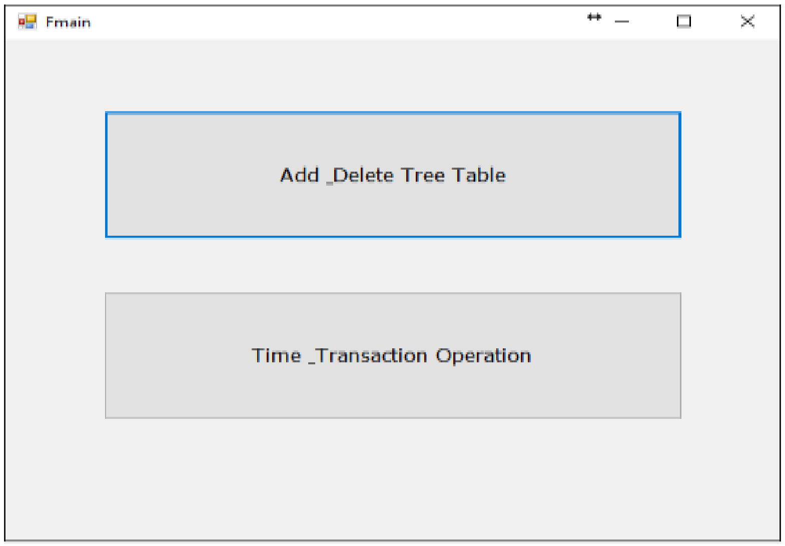


Figure A.1: Main Interface of the Program

The button (Add-Delete Tree Table) in Figure A.1 used to implement the MOPE algorithm to test specific parameters (running time, difference time between each two sequential rebalancing operations and number of transactions between each two sequential rebalancing operations) that will be used in our proposed methodology to determine when to change the encryption key with an acceptable performance.

The following Figure A.2 displays the interface of the MOPE algorithm program that has the following items:

- Add Import button which is used to import data from an excel sheet in order to insert them.
- Deleted Import button which is used to import data that will be deleted.
- Delete Data button that is used to delete the OPE tree.
- Add button used to begin insertion operation for the imported data.
- Delete button used to delete the imported data.
- Append Data check box used to insert another data.
- Two export buttons which are used to export the output to an excel sheet.

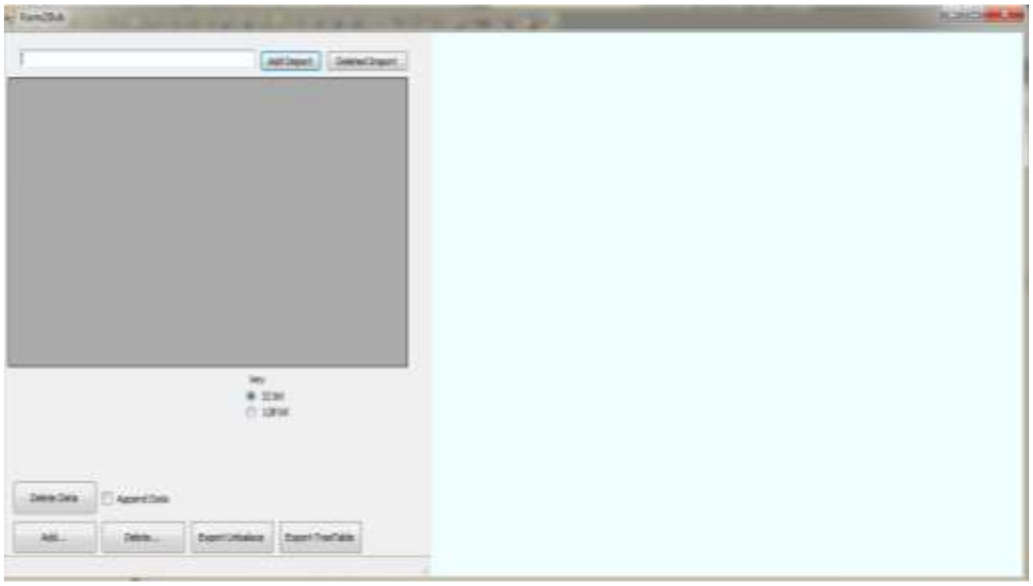


Figure A.2: The Interface of the MOPE Program

The OPE tree resulted and exported to tree table is also drawn as a graph as following Figure A.3

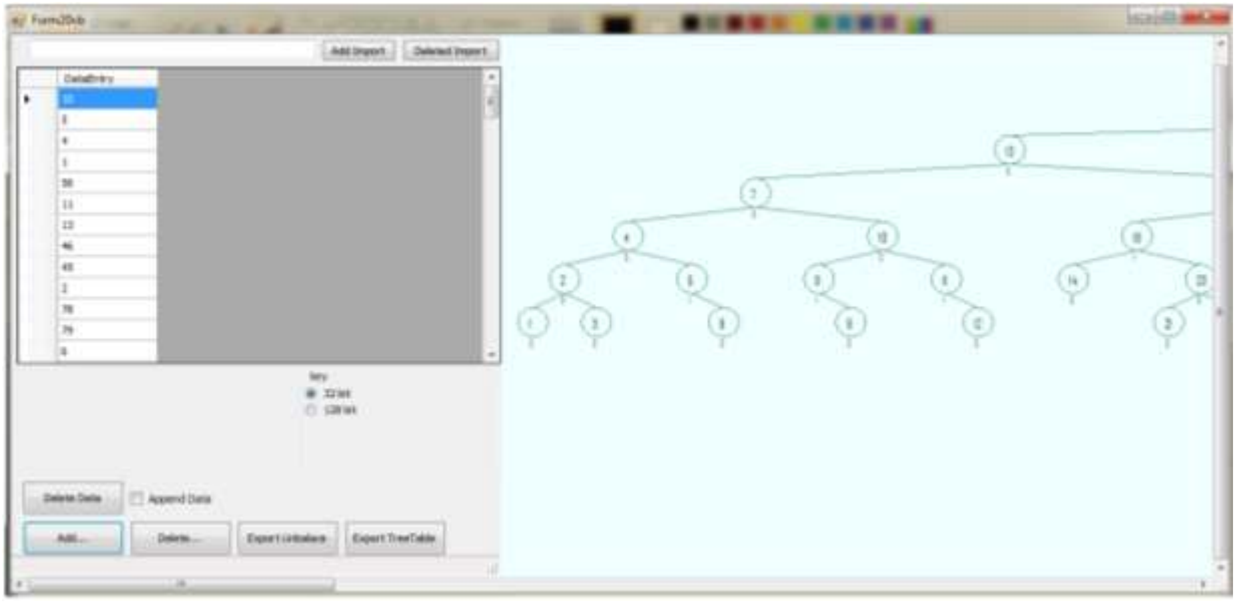


Figure A.3: The Graph of the Tree Table Exporting File

In this figure the imported data are stayed on the left side of the interface. When Add or Delete button is clicked the insertion or deletion transaction of these data is beginning and the result (OPE tree) is drawn in the right side of the interface and exported as a table to tree table that explained in Figure 4.1 in chapter Four.

Appendix B

Implementation of the Proposed Solution

The following Figure B.1 shows the interface of this program

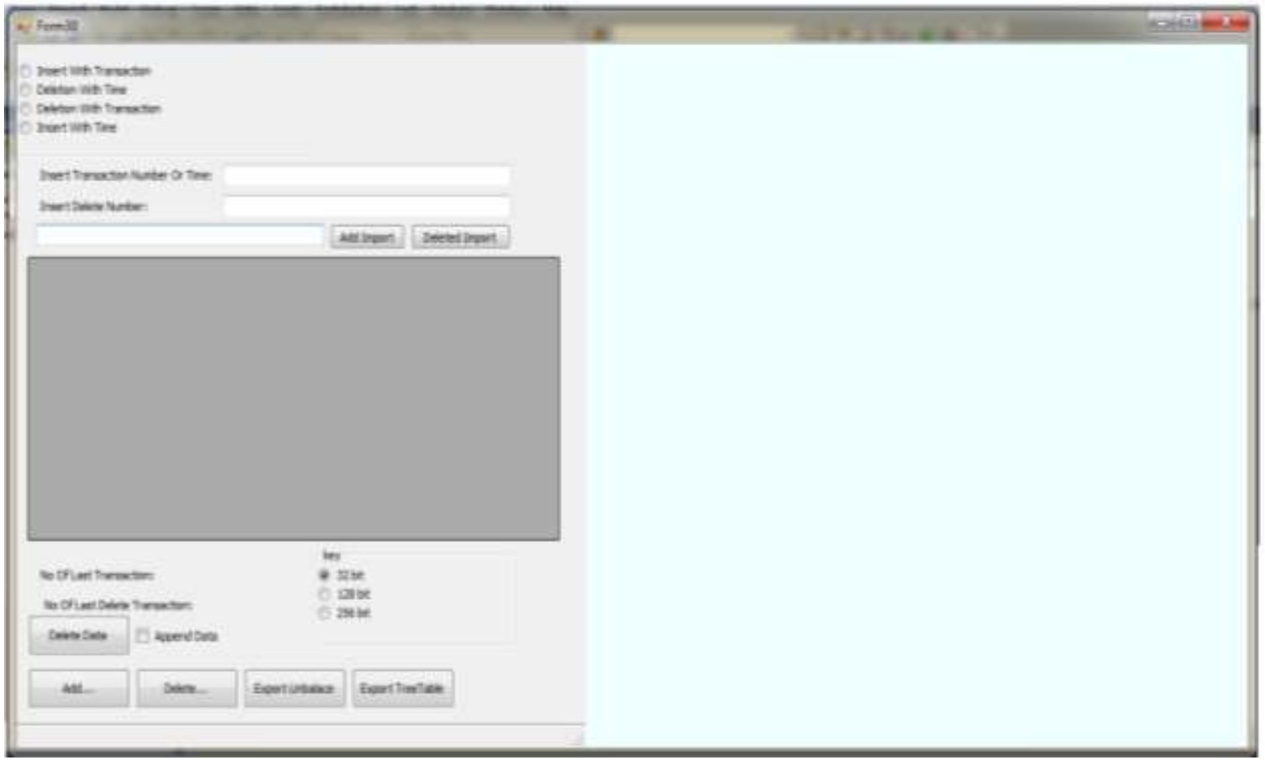


Figure B.1: The Interface of Our Proposed Solution

The Figure B.1 includes the following items:

- Add Import button which is used to import data from an excel sheet in order to insert them.
- Deleted Import button which is used to import data that will be deleted.
- Delete Data button that is used to delete the OPE tree.
- Add button used to begin insertion operation for the imported data.
- Delete button used to delete the imported data.

- Append Data check box used to insert another data.
- Insert with Transaction: this check box is chosen when we want to build our tree using the criteria of transaction window to study the effect of this chosen value on the performance. The number of transaction must be determined in the field Insert Number
- Delete with Transaction: this check box is chosen when we want to use the criteria of transaction window to delete values from the tree and the number of transactions must be determined in the fields Insert Transaction Number and Insert Delete Number to study the effect of this chosen values on the performance.
- Insert with Time: this check box is chosen when we want to use the criteria of time window to insert data in the OPE tree using specific time duration to change the encryption key that must be determined in the field Insert Time and study the effect of this value on the performance.
- Delete with Time: this check box is chosen when we want to use the criteria of time window to delete data from the tree using two parameters. First one contains specific time duration determined in the field Insert Time and second one contains number of deletion transaction sine last restructuring which store in the Insert Delete Number field and study the effect of these values on the performance.
- Two export buttons which are used to export the output to an excel sheet. These exported sheets are explained in Figures 4.1 and 4.2 in chapter Four.

Appendix C

The Following tables clarified the execution of MOPE algorithm to insert 100 records of data:

Table C.1: Execution of MOPE Algorithm to Insert 100 Data Records (Exp3)

InputData	DataSize	ParentID	Direction	KeyType	Time1	TimePerRecord
78	2	0	root	32	0.463	0.446
36	2	78	Left	32	0.556	0.093
13	2	36	Left	32	0.649	0.093
7	1	13	Left	32	0.766	0.088
4	1	7	Left	32	0.854	0.088
2	1	4	Left	32	0.953	0.099
1	1	2	Left	32	0.078	0.101
3	1	2	Right	32	0.169	0.09
5	1	4	Right	32	0.281	0.091
6	1	5	Right	32	0.402	0.096
10	2	7	Right	32	0.521	0.095
8	1	10	Left	32	0.635	0.092
9	1	8	Right	32	0.733	0.075
11	2	10	Right	32	0.831	0.074
12	2	11	Right	32	0.906	0.075
25	2	13	Right	32	0.007	0.08
18	2	25	Left	32	0.09	0.083
14	2	18	Left	32	0.172	0.082
23	2	18	Right	32	0.278	0.083
21	2	23	Left	32	0.362	0.084
24	2	23	Right	32	0.44	0.078
29	2	25	Right	32	0.545	0.081
26	2	29	Left	32	0.628	0.083

33	2	29	Right	32	0.707	0.079
31	2	33	Left	32	0.801	0.075
34	2	33	Right	32	0.896	0.075
58	2	36	Right	32	0.998	0.078
46	2	58	Left	32	0.1	0.079
43	2	46	Left	32	0.199	0.079
39	2	43	Left	32	0.287	0.088
37	2	39	Left	32	0.366	0.079
42	2	39	Right	32	0.47	0.08
44	2	43	Right	32	0.57	0.076
45	2	44	Right	32	0.671	0.081
48	2	46	Right	32	0.777	0.084
47	2	48	Left	32	0.88	0.08
56	2	48	Right	32	0.98	0.08
55	2	56	Left	32	0.086	0.083
57	2	56	Right	32	0.188	0.082
69	2	58	Right	32	0.288	0.075
64	2	69	Left	32	0.384	0.074
62	2	64	Left	32	0.462	0.078
60	2	62	Left	32	0.552	0.09
66	2	64	Right	32	0.628	0.076
65	2	66	Left	32	0.702	0.074
67	2	66	Right	32	0.805	0.084
73	2	69	Right	32	0.906	0.08
70	2	73	Left	32	0.088	0.084
77	2	73	Right	32	0.166	0.078
76	2	77	Left	32	0.242	0.076
147	3	78	Right	32	0.319	0.077
100	3	147	Left	32	0.393	0.074
87	2	100	Left	32	0.471	0.078
84	2	87	Left	32	0.605	0.11
79	2	84	Left	32	0.713	0.086

83	2	79	Right	32	0.79	0.077
85	2	84	Right	32	0.896	0.085
95	2	87	Right	32	0.996	0.079
91	2	95	Left	32	0.092	0.077
88	2	91	Left	32	0.196	0.079
92	2	91	Right	32	0.294	0.077
97	2	95	Right	32	0.397	0.082
99	2	97	Right	32	0.498	0.076
114	3	100	Right	32	0.602	0.082
105	3	114	Left	32	0.71	0.083
145	3	114	Right	32	0.811	0.083
123	3	145	Left	32	0.886	0.075
741	3	147	Right	32	0.969	0.083
369	3	741	Left	32	0.071	0.082
258	3	369	Left	32	0.232	0.08
156	3	258	Left	32	0.382	0.078
159	3	156	Right	32	0.547	0.08
321	3	258	Right	32	0.63	0.083
513	3	369	Right	32	0.709	0.079
456	3	513	Left	32	0.87	0.078
426	3	456	Left	32	0.978	0.088
654	3	513	Right	32	0.079	0.079
531	3	654	Left	32	0.181	0.081
684	3	654	Right	32	0.287	0.082
963	3	741	Right	32	0.396	0.085
852	3	963	Left	32	0.478	0.082
789	3	852	Left	32	0.581	0.079
753	3	789	Left	32	0.679	0.076
759	3	753	Right	32	0.757	0.078
793	3	789	Right	32	0.831	0.074
846	3	793	Right	32	0.905	0.074
954	3	852	Right	32	0.985	0.08

921	3	954	Left	32	0.085	0.08
950	3	921	Right	32	0.165	0.08
957	3	954	Right	32	0.249	0.084
977	3	963	Right	32	0.335	0.086
965	3	977	Left	32	0.416	0.081
971	3	965	Right	32	0.49	0.074
987	3	977	Right	32	0.563	0.073
1000	4	987	Right	32	0.642	0.079

Table C.2: Unbalance Table for the Execution of MOPE Algorithm
to Insert 100 Data Records (Exp3)

Num	ReasonUnbalance	HappenTime	OperationType	NumberOfTrans
10	4	0.650037	Add	3
10	11	0.953055	Add	6
56	46	1.16907	Add	8
11	45	1.28107	Add	9
4	2	1.40208	Add	10
5	78	1.52109	Add	11
56	79	1.63609	Add	12
11	6	1.7331	Add	13
45	25	1.90611	Add	15
6	8	2.17212	Add	18
56	58	2.44014	Add	21
5	9	2.70715	Add	24
5	3	2.80116	Add	25
56	48	2.89617	Add	26
45	26	2.99817	Add	27
79	84	3.10118	Add	28
25	31	3.36619	Add	31
85	97	3.4702	Add	32
69	64	3.5702	Add	33
46	73	3.67121	Add	34
46	37	3.77722	Add	35
25	18	3.88122	Add	36
26	23	3.98123	Add	37
45	39	4.08723	Add	38
13	24	4.18824	Add	39
26	29	4.28825	Add	40
64	77	4.70227	Add	45
84	100	4.80527	Add	46
97	145	4.90628	Add	47
145	114	5.47131	Add	53
58	156	5.60532	Add	54

31	34	5.79033	Add	56
45	43	5.89634	Add	57
145	147	5.99634	Add	58
114	258	6.09235	Add	59
156	369	6.19635	Add	60
100	963	6.29436	Add	61
369	852	6.39737	Add	62
258	741	6.49837	Add	63
95	123	6.60238	Add	64
36	456	6.71038	Add	65
369	654	6.9694	Add	68
85	87	7.87145	Add	75
95	92	7.97846	Add	76
88	91	8.07946	Add	77
963	977	8.18147	Add	78
852	965	8.28747	Add	79
147	921	8.47848	Add	81
100	159	8.58149	Add	82
456	513	8.98551	Add	87
39	42	9.64255	Add	95

The Following tables clarified the execution of MOPE algorithm to delete 40 data records from 100 data records:

Table C.3: Execution of MOPE Algorithm to Delete 40 Data Records from 100 Data Records (Exp4)

InputData	DataSize	ParentID	Direction	KeyType	Time1	TimePerRecord
78	2	0	root	32	0.164	0.164
36	2	78	Left	32	0.341	0.177
25	2	36	Left	32	0.508	0.167
18	2	25	Left	32	0.719	0.171
5	1	18	Left	32	0.834	0.115
4	1	5	Left	32	0.924	0.09
12	2	5	Right	32	0.034	0.086
21	2	18	Right	32	0.125	0.091
23	2	21	Right	32	0.243	0.096
29	2	25	Right	32	0.462	0.167
26	2	29	Left	32	0.632	0.132
33	2	29	Right	32	0.743	0.09
31	2	33	Left	32	0.848	0.085
58	2	36	Right	32	0.96	0.087
46	2	58	Left	32	0.046	0.086
43	2	46	Left	32	0.15	0.083
39	2	43	Left	32	0.236	0.086
48	2	46	Right	32	0.331	0.095
47	2	48	Left	32	0.646	0.251
56	2	48	Right	32	0.87	0.224
55	2	56	Left	32	0.019	0.149
57	2	56	Right	32	0.209	0.152
70	2	58	Right	32	0.366	0.157
65	2	70	Left	32	0.525	0.159
60	2	65	Left	32	0.71	0.146

76	2	70	Right	32	0.906	0.157
73	2	76	Left	32	0.1	0.153
77	2	76	Right	32	0.297	0.152
741	3	78	Right	32	0.488	0.151
156	3	741	Left	32	0.642	0.154
87	2	156	Left	32	0.796	0.154
83	2	87	Left	32	0.942	0.108
84	2	83	Right	32	0.049	0.088
114	3	87	Right	32	0.156	0.084
91	2	114	Left	32	0.264	0.087
88	2	91	Left	32	0.363	0.079
97	2	91	Right	32	0.465	0.081
123	3	114	Right	32	0.561	0.077
369	3	156	Right	32	0.661	0.079
258	3	369	Left	32	0.763	0.082
321	3	258	Right	32	0.868	0.081
531	3	369	Right	32	0.951	0.083
456	3	531	Left	32	0.035	0.084
426	3	456	Left	32	0.119	0.084
684	3	531	Right	32	0.205	0.086
852	3	741	Right	32	0.308	0.08
789	3	852	Left	32	0.423	0.09
759	3	789	Left	32	0.608	0.083
793	3	789	Right	32	0.692	0.084
846	3	793	Right	32	0.768	0.076
965	3	852	Right	32	0.85	0.082
954	3	965	Left	32	0.938	0.088
950	3	954	Left	32	0.014	0.076
957	3	954	Right	32	0.116	0.078
971	3	965	Right	32	0.227	0.091

Table C.4: Unbalance Table for the Execution of MOPE Algorithm to Delete 40 Data Records from 100 Data Records (Exp4)

Num	ReasonUnbalance	HappenTime	OperationType	NumberOfTrans
10	4	0.508029	Add	3
10	11	0.924053	Add	6
56	46	1.12506	Add	8
11	45	1.24307	Add	9
4	2	1.46208	Add	10
5	78	1.63209	Add	11
56	79	1.7431	Add	12
11	6	1.84811	Add	13
45	25	2.04612	Add	15
6	8	2.33213	Add	18
56	58	3.01917	Add	21
5	9	3.5252	Add	24
5	3	3.71021	Add	25
56	48	3.90622	Add	26
45	26	4.10123	Add	27
79	84	4.29825	Add	28
25	31	4.79627	Add	31
85	97	4.94228	Add	32
69	64	5.04929	Add	33
46	73	5.1573	Add	34
46	37	5.2643	Add	35
25	18	5.36331	Add	36
26	23	5.46531	Add	37
45	39	5.56132	Add	38
13	24	5.66132	Add	39
26	29	5.76333	Add	40
64	77	6.20636	Add	45
84	100	6.30836	Add	46
97	145	6.42437	Add	47
145	114	7.0144	Add	53

58	156	7.11641	Add	54
31	34	7.30742	Add	56
45	43	7.41042	Add	57
145	147	7.51643	Add	58
114	258	7.61344	Add	59
156	369	7.72344	Add	60
100	963	7.82945	Add	61
369	852	8.14247	Add	62
258	741	8.40548	Add	63
95	123	8.6645	Add	64
36	456	8.92651	Add	65
369	654	9.29153	Add	68
85	87	10.3706	Add	75
95	92	10.4776	Add	76
88	91	10.5766	Add	77
963	977	10.6896	Add	78
852	965	10.7936	Add	79
147	921	10.9766	Add	81
100	159	11.1836	Add	82
456	513	11.6247	Add	87
39	42	12.3127	Add	95
977	987	0.0680039	Delete	1
963	987	0.114007	Delete	1
84	85	0.35102	Delete	7
95	97	0.672038	Delete	16
147	97	0.713041	Delete	16
18	14	0.912052	Delete	21
73	70	0.980056	Delete	22
7	12	1.33808	Delete	32
18	12	1.37808	Delete	32
105	123	1.53409	Delete	35