Sudan University of Science and Technology
College of Graduate Studies

Improvement of the Search over Encrypted Data in Cloud Computing by Enhancement of the Mutable Order Preserving Encryption Model

تحسين البحث فى البيانات المشفرة بالحوسبة السحابية بتعزيز نموذج تشفير حفظ الترتيب القابل للتغيّر

A Thesis Submitted to the Graduate College of
Sudan University of Science & Technology
In Partial Fulfillment of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY in Computer Science

By

Nasrin Dalil Ali Arabi

Supervisor

Prof. Ahmed Kayed Ahmed

February-2021

Sudan University of Science & Technology

College of Graduate Studies

جامعة السودان للعلوم والتكنولوجيا

كلية الدراسات العليا

كلية الدراسات العليا

Ref: SUST/ CGS/A11

Approval Page

(To be completed after the college council approval)

Name of Candidate: Nasrin | Dalil | Ali | Arabi

Thesis title: Improvment of the search over Encrypted data in cloud computing by Enhancement of the mutable order preserving Encryption Model

Degree Examined for: .....

.....

Approved by:

1. External Examiner

Name: Fakhreldeen Abbas Saeed

Signature: ..... Date: 28-2-2021

2. Internal Examiner

Name: Dr. Yahia Abdalla Mohamed Hamad

Signature: ..... Date: 28-2-2021

3. Supervisor

Name: Ahmed Kayed

Signature: ..... Date: 28-2-2021

In the Name of Allah Most

Gracious Most Merciful

الآيـــــــــة

﴿ بِسْمِ اللهِ الرَّحْمَنِ الرَّحِيمِ ﴾

قال تعالي :

(رَبِّ أَوْزِعْنِي أَنْ أَشْكُرَ نِعْمَتَكَ الَّتِي أَنْعَمْتَ عَلَيَّ وَعَلَىٰ وَالِدَيَّ وَأَنْ أَعْمَلَ صَالِحًا تَرْضَاهُ وَأَدْخِلْنِي بِرَحْمَتِكَ فِي عِبَادِكَ الصَّالِحِينَ)

(النمل : 19)

صدق الله العظيم

Dedications

To my father's soul... Rest in peace

To my beloved Mom

To my family

To all of those who put me on the way and continue encouraging, supporting, and sponsoring...

Acknowledgment

First of all, I would like to thank Allah (Subhanahu Wa Ta'ala) who helps me and supplies all my needs to complete this research.

I would like to express my special deep appreciation and thanks to my advisor Professor Dr. Ahmed Kayed Ahmed for patience, sharing his knowledge, the valuable guidance, and feedback, and for his endless support during the study and research. I appreciate all his contributions, ideas, and time, to make my Ph.D. experience productive and stimulating.

I am grateful to Professor Ezz Eldien Mohammed Osman, the creator of this program, for his kind cooperation God bless him. Also, my sincere thanks extended to the Sudan University of Science and Technology, specifically to the Department of "Computer Science and Information Technology" for giving me the chance to do this thesis.

I would also like to thanks LAP14 family: Dr. Amjad Atta, Dr. Mohamed Adany, Dr. Waleed Ali Mergani, Dr. Elsadig Basheer, Dr. Mohamed Ibrahim, Dr. Modathir Fadul, Dr. Yosif Haroun, and Dr. Mohamed Elfatih for helping me and giving me many valuable comments, guidelines, and suggestions during writing my thesis.

I deeply appreciate the unlimited support of my mother, her prayers, and invocations for me. Also, I would like to thank my family for their continued support of my education, their love and faith in me will always be a guiding force in my life.

Abstract

Cloud computing has been considered as an essential technology to support future pervasive computing. It is becoming increasingly popular as it provides resources as services to its users. One of the beneficial services that the cloud provides to its user is storage. Indeed, outsourcing data to the untrusted domain leads to some vulnerability problems. To protect the outsourced data from such problems the data should be stored encrypted in the cloud. Ideally, to maintain the security of the user's data, queries should be performed over encrypted data. Generally, various approaches support computing over encrypted data. A common approach that enables querying over encrypted data is the Ordered Preserving Encryption (OPE) scheme. It allows the sort operations (such as range query) and the comparison operation (like MIN, MAX) to be directly executed over encrypted data.

Popa's presented the first ideal-security OPE model called mutable Order-Preserving Encryption (mOPE). The ideal-security guarantee for the OPE permits the cipher-texts to leak nothing about the plain-texts besides the order. To achieve this, the mOPE model constructs the OPE traversal tree to involve the client's data. Moreover, to perform operations the server needs the client's help to search over encrypted data on the OPE tree. The dependence of the server on the client to perform the requested operations dramatically produces more requests and responses between the client and the server. Concurrently, slow the system performance.

This thesis presents an enhanced OPE model improved on the mOPE model to reduce the requests and responses between the client and the server to enhance the performance of the search over encrypted data. The proposed enhanced model eliminates the dependence of the server on the client by permitting the server to perform part of the search processes without leaking any information about the original data besides the order. Moreover, to speed up the process of the search operations it uses an indexing mechanism and developed Range_Value to order the client's data in the server. Based on the used indexing mechanism information and the selected Range_Value the client's data will be arranged into two types of indices, one of them follows the Popa's technique to preserve the order of

encrypted data, and the other one applies a different technique. This adds some aspect of security.

The proposed enhanced model was implemented in two case study scenarios and tested on various examples, using simulation programs. Also, the mOPE model was implemented in the same case study scenarios using similar data as the enhanced OPE model. Finally, the findings from the two models were compared to evaluate the enhanced OPE model. The results shows that the enhanced OPE model succeeds in reducing the requests and responses between the client and the server against the original model. The enhanced OPE model behaves better in the small Range_Value. Moreover, the experiments on the used sample of data show the best outcomes when the Range_Value is less than halve of the expected client's data that will be outsourced for storage.

ملخص البحث

تعتبر الحوسبة السحابية تقنية أساسية لدعم الإنتشار الواسع للحوسبة في المستقبل. فقد أصبحت تحظي بشعبية متزايدة لأنها توفر الموارد كخدمات لمستخدميها. التخزين هي واحدة من الخدمات المفيدة التي تقدمها السحابة لمستخدميها. في الحقيقة تصدير البيانات إلي مجال غير موثوق يعرّض البيانات لمشاكل الإختراق. في سبيل حماية البيانات المُصدَرة من مثل هذه المشاكل فإنه يجب تخزينها مشفرة في السحابة. من ناحية مثالية, للحفاظ علي سرية بيانات المستخدم فإنه يجب إجراء الإستعلامات علي البيانات وهي مشفرة. بشكل عام, هناك العديد من الطرق التي تدعم الإستعلام علي البيانات المشفرة. إحدي الطرق الشائعة التي تتيح الإستعلام علي البيانات المشفرة هو تشفير حفظ الترتيب (OPE). فهو يتيح عمليات الفرز (مثل إستعلام النطاق) وعمليات المقارنة (مثل MIN, MAX) ليتم تنفيذها مباشرة علي البيانات وهي مشفرة.

قدم Popa's أول نموذج مثالي الأمان لتشفير حفظ الترتيب يسمي تشفير حفظ الترتيب القابل للتغيّر (mOPE). نموذج الأمان المثالي لتشفير حفظ الترتيب يضمن عدم السماح للرسالة المشفرة بتسريب أي معلومة عن الرسالة الأصلية بجانب الترتيب. للوصول الي ذلك فإن نموذج تشفير حفظ الترتيب القابل للتغيّر يقوم ببناء شجرة بنظام تشفير حفظ الترتيب لتضم بيانات العميل. بجانب ذلك, لإجراء العمليات فإن الخادم يحتاج إلي مساعدة العميل للبحث علي البيانات المشفرة في الشجرة (المبنية بنظام تشفير حفظ الترتيب). في الحقيقة إعتماد الخادم علي العميل لإجراء العمليات المطلوبة ينتج عنه المزيد من الطلبات والإستجابات بين العميل والخادم. في نفس الوقت فإنه يبطئ أداء النظام.

هذا البحث يقدم نموذج مُحسن لتشفير حفظ الترتيب, وهو مُحسن علي نموذج تشفير حفظ الترتيب القابل للتغيّر لتقليل الطلبات والإستجابات بين العميل والخادم لتحسين أداء البحث علي البيانات المشفرة. النموذج المُحسن المُقدم يقلل من إعتماد الخادم علي العميل و ذلك بالسماح للخادم بإجراء جزء من عمليات البحث من غير تسريب أي معلومات عن البيانات الأصلية بجانب الترتيب. علاوة علي ذلك, لزيادة سرعة عملية البحث فإنه يستخدم آلية فهرسة وقيمة مدي مطورة لترتيب بيانات العميل في الخادم. بناءاً علي بيانات آلية الفهرسة المستخدمة وقيمة المدي المختارة فإن بيانات العميل يتم ترتيبها في نوعين من الفهارس, أحدهما يتبع التقنية التي إستخدمها Popa لحفظ ترتيب البيانات المشفرة, والآخر يطبق تقنية ترتيب مختلفة. هذا يضيف بعض جوانب الأمان.

النموذج المُحسن لتشفير حفظ الترتيب المُقدم تم تطبيقه علي سيناريوهين لدراسة الحالة وتم إختباره علي أمثلة مختلفة بإستخدم برامج محاكاة. أيضاً, تم تطبيق نموذج تشفير حفظ الترتيب القابل للتغيّر علي نفس السناريوهين لدراسة الحالة بإستخدام بيانات مشابهة كما في النموذج المُحسن لتشفير حفظ الترتيب. وأخيراً, تمت مقارنة النتائج في النموذجين لتقييم النموذج المُحسن لتشفير حفظ الترتيب. أظهرت النتائج أن النموذج المُحسن لتشفير حفظ الترتيب نجح في تقليل الطلبات والإستجابات بين العميل والخادم مقارنة بالنموذج الأصل. نموذج تشفير حفظ الترتيب المُحسن عمل بشكل أفضل عندما

كانت قيمة المدي صغيرة. بالإضافة لذلك, فقد أظهرت التجارب في عينة البيانات المستخدمة أفضل النتائج عندما كانت قيمة المدي أقل من نصف بيانات المستخدم المتوقع تصديرها للتخزين.

Table of Contents

الآيـــــــة i

Dedications ii

Acknowledgment iii

Abstract iv

ملخص البحث vi

Table of Contents viii

List of Tables xii

List of Figures xiii

LIST OF SYMBOLS / ABBREVIATIONS xiv

CHAPTER I 1

INTRODUCTION 1

1.1 General Introduction: 1

1.2 Research Motivation: 2

1.3 Problem Statement: 2

1.4 Research Questions: 2

1.5 Research Potential Benefits: 3

1.6 Research Objectives: 3

1.7 Research Scope: 3

1.8 Research Methodology: 4

1.9 Research Contribution: 6

1.10 Thesis Organization: 6

CHAPTER II 8

BACKGROUND AND RELATED WORKS 8

2.1 Introduction: 8

2.2 Background: 8

2.3 Cloud Computing: 9

2.3.1 Cloud Computing Characteristics: 9

2.3.2 Cloud Computing Deployment Models: 10

2.3.3 Cloud Computing Layers: 11

2.3.4 Cloud Computing Delivery Models: 11
2.4 Security of Cloud Computing: 12
2.5 Methods for Computing on Encrypted Data: 13
2.5.1 Methods with no Leakage: 14
2.5.1.1 Fully Homomorphic Encryption (FHE): 14
2.5.1.2 Partially Homomorphic Encryption (PHE): 14
2.5.2 Methods with Controlled Leakage: 15
2.5.2.1 Functional Encryption (FE): 15
2.5.2.2 Deterministic Encryption (DE): 15
2.5.2.3 Order-Preserving Encryption (OPE): 16
2.6 OPE Schemes: 16
2.7 Related Works: 18
2.8 Summary: 20
CHAPTER III 21
THE PROPOSED ENHANCED RANGES ORDER PRESERVING ENCRYPTION MODE 21
3.1 Introduction: 21
3.2 The Original mOPE Model: 21
3.3 The Enhanced Ranges Order Preserving Encryption (ROPE) Model: 24
3.3.1 Definitions: 24
3.3.2 The Trusted Party: 25
3.3.3 The Index Table: 25
3.3.4 A Framework for the Enhanced ROPE Model: 28
3.3.4.1 The Client: 30
3.3.4.2 The Trusted Party: 30
3.3.4.3 The Server: 31
3.3.5 Preserving the Order of Data in the Server: 31
3.3.5.1 Dealing with the Id_ranges: 31
3.3.5.2 Organizing the Client's Data into the Ranges: 32
3.3.6 The Access of Data in the Enhanced ROPE Model: 33
3.3.7 A Case Study Example: 38
3.3.7.1 Create the Index Table: 38

3.3.7.2 Data Encryption and Organization: 39
3.3.7.3 The Insert Operation: 40
3.4 Research Methods for the Enhanced Model Evaluation: 41
3.5 Research Tools and Data: 44
3.6 Summary: 44
CHAPTER IV 45
IMPLEMENTATION AND TESTING 45
4.1. Introduction: 45
4.2. The Enhanced ROPE Model Implementation: 45
4.2.1 The Experiments of the Insert Scenario: 48
4.2.2 The Experiments of the Search Scenario: 57
4.3 The mOPE Model Implementation: 59
4.3.1 The Experiment of the Insert Scenario: 61
4.3.2 The Experiment of the Search Scenario: 61
4.4. Summary: 61
CHAPTER V 62
RESULTS AND DISCUSSIONS 62
5.1 Introduction: 62
5.2 Results: 62
5.3. Discussions: 65
5.4. Summary: 67
CHAPTER VI 68
THE ENHANCED ROPE MODEL EVALUATION 68
6.1. Introduction: 68
6.2 Security Evaluation: 68
6.2. Performance Evaluation: 69
6.3. The Enhanced ROPE Model Features: 71
6.4. Comparison between the Enhanced ROPE Model and the mOPE Model: 71
6.5. Summary: 73
CHAPTER VII 74
CONCLUSIONS AND FUTURE WORKS 74
7.1. Conclusion: 74

7.2 Thesis Contributions:..... 75
7.3 The Future Works: 76
References: 77
Appendices 84
Appendix A 84
The enhanced ROPE Model Source Code: 84
Subprogram1: Creates the index table and prepares the database by allocating the Id_ranges..... 84
Subprogram2: The Insertion Program. 87
Subprogram 3: The Search Program. 96
Appendix B..... 102
The mOPE Model Source Code: 102
Subprogram1: The Insertion Program. 102
Subprogram2: The Search Program. 109
List of Publications 114

List of Tables

Table 2.1: Security provided by previous OPE schemes and mOPE scheme (Raluca Ada Popa, 2013). 17

Table 3.1: The index table of the case study 39

Table 3.2: The client's data in the database of the case study 40

Table 4.1: The Index Table for experiment no.1, ECD=100, RV=5 49

Table 4.2: The Index Table for experiment no.2, ECD=100, RV=10 51

Table 4.3: The Index Table for experiment no.3, ECD=100, RV=20 52

Table 4.4: The Index Table for experiment no.4, ECD=100, RV=30 53

Table 4.5: The Index Table for experiment no.5, ECD=100, RV=40 53

Table 4.6: The Index Table for experiment no.6, ECD=100, RV=50 54

Table 4.7: The Index Table for experiment no.7, ECD=100, RV=60 54

Table 4.8: The Index Table for experiment no.8, ECD=100, RV=70 55

Table 4.9: The Index Table for experiment no.9, ECD=100, RV=80 55

Table 4.10: The Index Table for experiment no.10, ECD=100, RV=90 56

Table 4.11: The Index Table for experiment no.11, ECD=100, RV=100 56

Table 4.12: The search operation experiments 58

Table 5.1: The results of the insert operations experiments in the enhanced ROPE model. 63

Table 5.2: The results of the search operations experiments in the enhanced ROPE model 64

Table 5.3: The result of the insert experiment in the mOPE model 65

Table 5.4: The result of the search operations in the mOPE model 65

Table 6.1: Comparison between the mOPE model and the enhanced ROPE model 72

List of Figures

Figure 1-1: The research methodology diagram 5

Figure 2-1: Cloud computing architecture (Qi Zhang, 2010). 12

Figure 3-1: The OPE tree in mOPE model (Raluca Ada Popa, 2013) 22

Figure 3-2: The OPE tree traversal algorithm of the mOPE model (Raluca Ada Popa, 2013). 23

Figure 3-3: The ITC algorithm. 26

Figure 3-4: The ITC algorithm diagram. 27

Figure 3-5: The enhanced ROPE model framework. 29

Figure 3-6: The enhanced ROPE model algorithm. 34

Figure 3-7: Searches for the right Id_range diagram. 36

Figure 3-8: Search for the right location in the target OPE tree diagram. 37

Figure 3-9: The insert of 2 in the database of the case study and the related storage after that. 41

Figure 3-10: The research methods for the evaluation diagram. 43

Figure 4-1: The ROPE model experiments approach. 47

Figure 4-2: The mOPE model experiments approach 60

LIST OF SYMBOLS / ABBREVIATIONS

OPE	Order Preserving Encryption
mOPE	mutable Order Preserving Encoding
CryptDB	Named
BBST	Balanced Binary Search Tree
RRbCS	Requests and Responses between the Client and the Server
RV	Range_Value
DBaaS	Database as a Service
CSP	Cloud Services Provider
NIST	National Institute of Standards and Technology
VMWare	Named for cloud software
vCloud Director	Named for cloud software
Xen	Name for virtualization technology
KVM	Name for virtualization technology
VM	Virtual Machine
API	Application Programming Interfaces
SaaS	Software-as-a-Service
PaaS	Platform-as-a-Service
IaaS	Infrastructure-as-a-Service
ISO	International Standards Organization
CSS	Cloud Service Subscribers
FHE	Fully Homomorphic Encryption
PHE	Partially Homomorphic Encryption
FE	Functional Encryption
pk	Public-key
Sk	Secret key
DE	Deterministic Encryption
IND-OCPA	Indistinguishability under Ordered Chosen-Plaintext Attack

PKE	Public-key Encryption
SE	Searchable Encryption
MV-OPES	Multivalued-Order Preserving Encryption Scheme
DET	Deterministic encryption
ROPE	Real Order Preserving Encryption
IND-OLCPA	Indistinguishability under Ordered Launching Chosen-Plaintext Attack
ROPF	Real Order Preserving Functions
RND	Random encryption
ORE	Order Revealing Encryption
OOPE	Oblivious Order Preserving Encryption
DBMS	Database Management System
ROPE	Ranges Order Preserving Encryption
ECD	Expected Client's Data
ITC	Index Table Creation
AVL	Adelson-Velskii and Landis
BST	Binary Search Tree
TbS	Tests of the Server

CHAPTER I

INTRODUCTION

1.1 General Introduction:

Cloud computing became an appealing technology in the computing era. Today's more and more companies outsourcing their data to the external cloud, outsourcing data to cloud service offers lower cost by sharing hardware and elastic scaling (Sosinsky, 2010). However, a key problem in outsourcing storage is that the sensitive data may be subject to unauthorized access not only from an unknown attacker but also from a curious service provider (Rahman, 2011).

Moreover, one possible approach to protect the confidentiality of outsourced data is encryption (Dan Boneh, 2007) (Manpreet Kaur, 2013). Nevertheless, this solution requires additional decryption operations when there is a need to perform operations on data. To face this limitation the cloud server should be able to perform some computation over encrypted data exactly as unencrypted one. The idea behind this is to use techniques that support processing operations on encrypted data exactly and efficiently as unencrypted ones and without knowing any information about it (Sergei Evdokimov, 2007).

However, one possible approach that permits querying over encrypted data is the Order Preserving Encryption (OPE) scheme. It permits comparisons and sort operations to be directly performed on encrypted data. Also, the OPE scheme can be integrated easily with the existing database systems (Seungmin LEE, 2009). Moreover, (Raluca Ada Popa, 2013) presented the first ideal-security OPE model called the "mutable Order-Preserving Encoding" or "mOPE" model (the word encoding was used instead of encryption). It was constructed for order operations in CryptDB (POPA, 2014). The mOPE model was function by building one OPE tree which is a Balanced Binary Search Tree (BBST) that comprising all the client's data in the server. Also, the server needs the client's help to search over the encrypted data. The server dependences on the client in the mOPE cause more Requests and Responses between the Client and the Server (RRbCS). Sequentially, this leads to slow system performance. To solve this problem this study presents an

enhanced OPE model from the mOPE model. It divides the client's data into parts based on a developed selected Range_Value (RV). Also, it uses an indexing mechanism to speed up search operations.

1.2 Research Motivation:

Recent years show the success of cloud computing as new modern technology. Its popularity comes from the benefits that it offers to the users. To get the benefit of the cloud services, the users outsourced their sensitive data to the cloud server, despite the unauthorized access from the service provider. They need to access their data anytime and from any location. To keep track of this situation, there should be approaches to keep the user's data from leakage while providing flexibility to access them.

1.3 Problem Statement:

In the mOPE model, the server usually needs the client's help to process the client's requests. The dependence of the server on the client to move over the OPE tree generates more RRbCS. Moreover, the involvement of the client's data in one OPE tree increases the tree depth as the storage size is increasing. This leads to generate more and more RRbCS and subsequently the system performance is a slowdown.

1.4 Research Questions:

This work can outline the problem statement in three main questions to map the work direction, and also used as a guide for the study towards fulfilling the main goal. This research aims to answer the following **R**esearch **Q**uestions (**RQ**s):

RQ1: How to reduce the RRbCS in the mOPE model?

RQ2: How to reduce the dependence of the server on the client in the mOPE model?

RQ3: What is the effect of the RV length on the RRbCS?

1.5 Research Potential Benefits:

The hypotheses of this research are formulated as follows:

- Proposes an enhanced OPE model.
- The proposed enhanced OPE model may reduce RRbCS.
- The proposed enhanced OPE model may reduce server dependence on the client.
- The enhanced OPE model may enhance system performance.
- The enhanced OPE model may improve system security.
- The proposed methodology is expected to give promising results.

1.6 Research Objectives:

The general goal is mainly to improve the mOPE model to propose an enhanced OPE model, that reduces the RRbCS and consequently enhances the performance of the search over encrypted data, easily without affecting the efficiency and security of the whole system. Also, the specific objectives are outlined as follows:

1) To reduce the RRbCS in the mOPE model. This objective has been achieved through:
    - Designing an enhanced OPE model architecture based on the mOPE model.
    - Using a new indexing mechanism.
2) To reduce the server relies on the client in the mOPE model.
3) To apply the developed RV that can enhance the performance of search over encrypted data.
4) To evaluate the proposed enhanced model compared to the mOPE model.

1.7 Research Scope:

This research introduces an enhanced OPE model that improved from the mOPE model presented in (Raluca Ada Popa, 2013). The proposed model focuses on the number of RRbCS to enhance the performance of search over encrypted data in cloud computing.

1.8 Research Methodology:

The proposed model uses a mixture of both the qualitative and quantitative research methodologies to investigate the research questions through an effective literature review and also by developing an OPE model to study the RRbCS (performance) of the enhanced model compared to the mOPE model. The literature review forms the qualitative research methodology, and the RRbCS in the enhanced model compared to the mOPE model forms the quantitative part of the research methodology.

At the initial stage, a detailed literature study is made to understand the concept of search over encrypted data and the issues concerning the OPE schemes. Then the study analyzed the issues and challenges that have been emerging in previous works. Besides, based on the results, it abstracted the research objectives, and then set up the enhanced OPE model architecture. After that, a simulation program is designed to observe the behavior of the RRbCS (by means of performance) of the developed model compared to the mOPE model. The execution of the simulation program is to validate the research's initial assumption that the enhanced model can reduce the RRbCS while searching over encrypted data. This could enhance the performance of search over encrypted data. Tests are carried out with the aid of the simulation program for the enhanced model. Having the results, allowing analysis of the findings to be able to evaluate the developed model, and make judgments about the study's usefulness in practice.

In ordered to achieve the above objectives the following steps have been accomplished as shown in:

1. Study literature review and related works on querying over encrypted data and OPE schemes.
2. Define the problem statement.
3. Specify the research objectives.
4. Design and develop an OPE model architecture to enhance the performance of the search over encrypted data (this step will be explained in chapter three).
5. Implement and test the enhanced model using a simulation program to measure the performance enhancement compared to the mOPE model (this step will be discussed in chapter four).
6. Obtain the results (this step will explore in chapter five).

7. Evaluate the developed model compared to the mOPE model (this step will view in chapter six).
8. Publication of the research results.
9. Write the thesis.

Figure 1-1 formulates the steps of the research methodology.

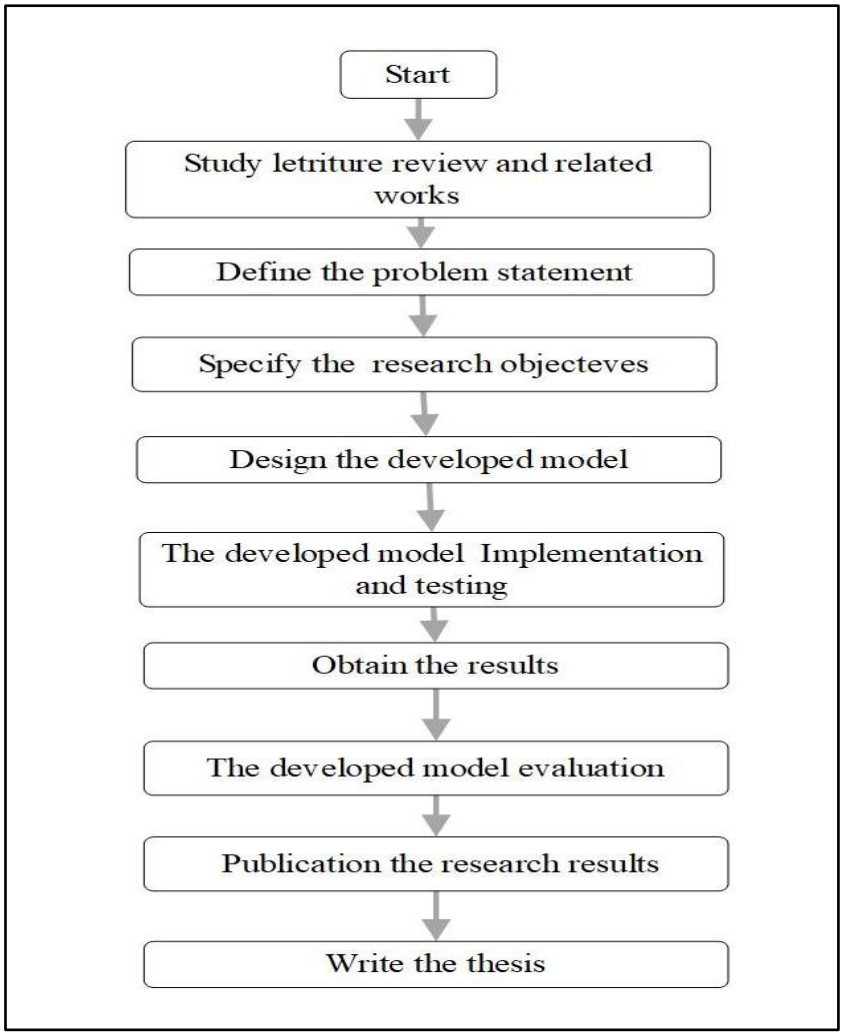

Figure 1-1: The research methodology diagram

1.9 Research Contribution:

The contributions of this dissertation can be summarized as follows:

1. Propose an enhanced OPE model for search over encrypted data.
2. Reduce the RRbCS of search processes over encrypted data.
3. Reduce the dependence of the server on the client while performing the search processes.
4. Permit the server to perform part of the search processes.
5. Provide the IND-OCPA security for the enhanced model.
6. Enhance the performance of search over encrypted data compared to the mOPE model.

1.10 Thesis Organization:

This dissertation is organized into seven chapters as follows:

Chapter One: Introduces the main subject of this study and provides a general outline of the work. It presents an overview of this dissertation; cover the problem statement of the study, the research questions, the potential benefits of the study, the objectives of the study, the research scope, the research methodology, and the contribution of the study. The rest of the thesis is organized as follows:

Chapter two: Presents theoretical background information for studies. Explains the following concepts: general background, the cloud computing characteristics and architecture, the security of cloud computing, methods for computing over encrypted data, the OPE schemes, and the related works.

Chapter three: Introduces the enhanced model, focuses on architecture design. Besides, this chapter describes the proposed enhanced OPE model. It addresses some definitions, illustrates the introduced entities, and presents a framework for the enhanced model. Also, it presents how the enhanced model preserves the order of data in the database server and how it accesses the data. Rather it explains a case study example, the research data and tools, and the research methods for the evaluation of the proposed model.

Chapter four: Covers the case studies that were conducted to test the enhanced model compared to the mOPE model using simulation programs. It presents two scenarios

for testing the enhanced model, one for the insert operations and the other for the search operations. Also, it presents the conducted case studies for the mOPE model.

Chapter five: States the results obtained from the conducted experiments and discusses them compared to the OPE model.

Chapter six: For model evaluation, it evaluates the proposed enhanced model security and performance compared to the mOPE model.

Chapter seven: Presents the conclusion of the study. Also, it mentioned the open issues for new researchers in the same field.

References: Illustrates all the referenced studies that are used in this research.

Appendices: Displays the simulation programs that implement and test the enhanced OPE model and the mOPE model in the insert and search scenarios. Also, it presents the list of publications.

CHAPTER II

BACKGROUND AND RELATED WORKS

2.1 Introduction:

This chapter introduces a background about outsourcing database to the cloud server. It covers cloud computing concepts and architecture. Also, it reviews the security of cloud computing and the methods for computing over encrypted data. Accurately, it focuses on the approaches for computing over encrypted data, especially the Order Preserving Encryption (OPE) schemes. Concretely, this chapter reviews the concept of the OPE schemes and the related works in this area.

2.2 Background:

In cloud computing, resources are provided as a service over the Internet to customers who use them when needed (Safiriyu Eludiora, 2011). One of the most attractive cloud services is the database. In which the cloud database is provided as a service. It is called Database as a Service (DBaaS) (Divyakant Agrawal, 2009) (Shehri, 2013). It allows users to store their data at remote storage and access them anytime and from any location (V. Spoorthy, 2014). To get the benefit of this cloud service, more and more organizations outsourcing their sensitive data (such as e-mails, personal health records, company financial data, and government documents, etc.) to be stored into the cloud database and managed by the Cloud Services Provider (CSP). Unfortunately, the migration of data to the untrusted domain put the outsourced unencrypted data at risk (Treesa Maria Vincent, 2013). A key problem of outsourcing storage is that the sensitive data may be subject to unauthorized access not only from an unknown attacker but also from a curious service provider (Dan Boneh, 2007). However, confidentiality is the most important security mechanism that should be provided to protect the user's data in the cloud (Jakimoski, 2016). Therefore, encryption is a better solution to prevent outsourced data from unauthorized access. So, it is better to encrypt the user's data before storing it in the cloud (R. Velumadhava Rao, 2015).

However, after encryption, it becomes difficult to perform queries over the database. It is not acceptable to decrypt the database before every querying because this leads to security vulnerability again. Another reason is that the decryption for large databases might be very slow. Ideally, queries should be executed directly over the encrypted database (Somayeh Sobati Moghadam, 2016).

2.3 Cloud Computing:

Cloud is an Internet-based technology, where shared resources are provided on-demand and pay-per-use basis. The National Institute of Standards and Technology (NIST) defines cloud computing as ”a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Peter Mell, 2011). Moreover, cloud computing is a computing platform for sharing resources that include infrastructures, software, applications, storage, and services. It provides services to the user in a virtualized manner through the internet. The flexible environment and cheaper cost of using cloud computing attract people to use cloud services related to software, platform, and infrastructure (Krunal Suthar, 2017).

2.3.1 Cloud Computing Characteristics:

(Sosinsky, 2010) And (Gamaleldin, 2013) were summarized the essential characteristics of cloud computing as below:

- **On-demand self-service**: The user can use the resources without the need for human interaction with CSP.
- **Broad network access:** Cloud resources are available over the network and can access through different platforms.
- **Resource pooling:** The CSP creates resources that are pooled together in a system that supports multi-tenant usage to serve multi-consumer, with physical and virtual resources that are assigned or reassigned dynamically according to the user demands. The idea behind pooling is to hide the location of the resource.

- **Rapid elasticity:** Resources can add to the system by either scaling in or scaling out. For the user, resources should be unlimited and can be purchased at any time and in any quantity
- **Measured service:** The resources' usage can be monitored, controlled, and reported. This feature has an advantage of what you are using is that you are paying for.

2.3.2 Cloud Computing Deployment Models:

Further, based on the usage of the Cloud, there are different types of deployment models. The cloud deployment models are private cloud, public cloud, hypride cloud, and community cloud. The first three ones are commonly-used, while the last one is less-commonly used (Gorelik, 2013). The deployment models are described below (Krunal Suthar, 2017) (Gorelik, 2013):

- **Private cloud**: It builds for a single organization and it manages by an organization itself. Organizations use software that enables cloud functionality, such as VMWare, vCloud Director.
- **Public Cloud:** The organization provides a set of computing resources for the general public who can access it from anywhere and pay as per use. The most popular public clouds include Amazon Web Services, Google AppEngine, and Microsoft Azure.
- **Hypride Cloud:** It is a mixture of private cloud and public cloud that provides computing resources of both.
- **Community Cloud:** Computing resources are shared among several organizations that have similar missions and requirements.

2.3.3 Cloud Computing Layers:

The architecture of cloud computing is divided into four layers the hardware layer, the infrastructure layer, the platform layer, and the application layer. The next part describes each of them (Qi Zhang, 2010):

- **The hardware layer:** It concerns managing the underlying physical resources of the cloud such as physical servers, routers, switches, power, and cooling systems.
- **The infrastructure layer:** Also known as the virtualization layer, creates a pool of storage and computing resources by partitioning the physical resources using virtualization technologies such as Xen, KVM, and VMware. Also, it is responsible for dynamic resource assignment and many other features are available in this layer. This layer is known as the virtualization layer.
- **The platform layer:** It consists of operating systems and application frameworks. The purpose of this layer is to facilitate the process of deploying applications into a virtual machine (VM) container. Also, it provides Application Programming Interfaces (API) for implementing web applications.
- **The application layer:** At the top of the hierarchy, it consists of the actual cloud applications.

2.3.4 Cloud Computing Delivery Models:

In cloud computing, three delivery models delivered different types of services to the end-user which are (S. Subashini, 2010) (J.SRINIVAS, 2012):

- **Software-as-a-Service (SaaS):** In a cloud-computing environment SaaS is the software that is owned, delivered, and managed remotely by one or more providers.
- **Platform-as-a-Service (PaaS):** It offers an integrated set of developer environments that can help the consumers to build their applications and deliver them to the users through the internet.
- **Infrastructure-as-a-Service (IaaS):** IaaS completely abstracted the hardware under it and allowed users to consume infrastructure as a service without knowing anything about the underlying complexities.

Figure 2-1 describes the four layers of cloud computing and its three delivery models.

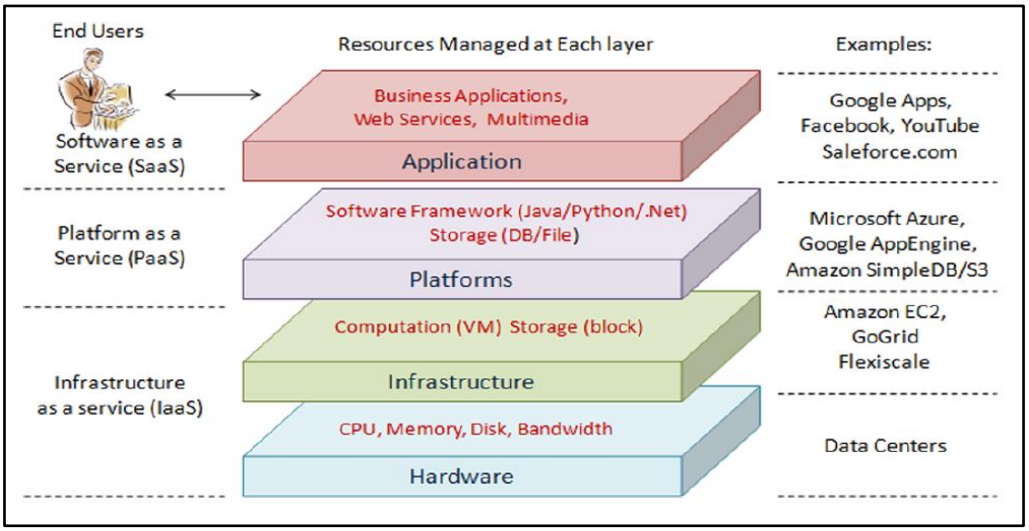

Figure 2-1: Cloud computing architecture (Qi Zhang, 2010).

2.4 Security of Cloud Computing:

Cloud computing has many advantages that attract the user to get benefited. The fact that most of the user's data and software in cloud computing are reside on the internet, makes the system faces some challenges, especially for security. The nature of cloud computing makes it a complicated matter to provide security in cloud computing. For instance, each application may use the resource from multiple servers and the servers are at multiple locations. Also, the cloud may provide services that used different infrastructures across organizations. Therefore, to ensure cloud computing security, various security issues must be taken into account (Liang Yan, 2009).

The International Standards Organization (ISO) recommended some security requirements for Information Security which are:

- Identification & Authentication.
- Authorization.
- Confidentiality.
- Integrity.
- Non-repudiation.
- Availability.

These security requirements should be covered in cloud computing to recognize cloud computing as a secure technology (Ramgovind S, 2010).

Cloud computing is a rich environment of security risk. Hence, there are specific security issues the Cloud Service Subscribers (CSS) should pay more attention to them which are:

- Privileged user access.
- Regulatory compliance.
- Data location.
- Data segregation.
- Recovery.

So, the CSS should agree on the security issues with their providers before selecting them (Safiriyu Eludiora, 2011).

Generally, cloud customers have different motivations to move to the cloud. From the perspective of different cloud clients, the security point of view is different. For instance, academia is concerns about security and performance. Therefore, the cloud can provide approaches to combine security and performance for them, whereas for enterprises, the most important problem is security but performance maybe not as critical as academia (MANDEEP KAUR, 2013).

2.5 Methods for Computing on Encrypted Data:

This section explains some methods that enable computing on encrypted data. These methods were categorized into two types: methods with no leakage about the data and methods that leak a well-defined function of the data. In the second type, the leakage of the data enhances performance (POPA, 2014). In the following subsection, we describe these methods.

2.5.1 Methods with no Leakage:

2.5.1.1 Fully Homomorphic Encryption (FHE):

The FHE permits general computations to be performed on the encrypted data (Khamitkar, 2014). If a user has a function ***f*** and want to obtain ***f***(m_1,…,m_n) for some inputs m_1, …, m_n, it is possible to instead compute on encryptions of these inputs, c_1, …, c_n, obtaining a result which decrypts to ***f***(m_1, …, m_n) (Frederik Armknecht, 2015).

More importantly, FHE allows different types of operations to be performed on encrypted data without decryption (Maha TEBAA, 2013) (Mr. Manish M Poteya, 2016). It permits addition and multiplication to be performed at the same time (SARAH SHIHAB HAMAD, 2018). However, it allows numbers of operations to be performed an unlimited number of times (Abbas Acar, 2018). FHE offers strong security guarantees, called semantic security. It is too slow, because it runs arbitrary types of operations (POPA, 2014).

2.5.1.2 Partially Homomorphic Encryption (PHE):

Partial Homomorphic Encryption (PHE) schemes allow the computation of certain mathematical operations over encrypted data. Unlike the FHE, the PHE permits only one type of operation, either additive or multiplicative (Ayub Hussain Mondal, 2015) (Mr. Manish M Poteya, 2016). For instance, additive homomorphic schemes provide the addition of cipher-texts, such that the result is equal to the addition of the plaintext values (i.e., ENC (m1) + ENC (m2) = ENC (m1 +m2)) (Hossein Shafagh, 2017).

Moreover, PHE allows operations to be performed on encrypted data for an unlimited number of times (Abbas Acar, 2018). So, it provides better performance than the FHE. This method provides the same strong security as FHE (POPA, 2014).

2.5.2 Methods with Controlled Leakage:

2.5.2.1 Functional Encryption (FE):

The FE is public-key (pk) encryption in which a receiver of the cipher-text can learn certain functions of the underlying message based on its secret keys (sk) (O'Neill, 2010). In a FE system, a trusted authority holds a master secret key known only to the authority. When the authority is given the description of some function ***f*** as input, it uses its master secret key to generate a derived secret key sk[***f***] associated with ***f***. In symbols, if E(pk; x) is encryption of x, then decryption accomplishes (Dan Boneh, 2012):

Given E(pk; x) and sk[***f***],

Decryption outputs ***f*** (x) (Dan Boneh, 2012).

This opens possibilities to specify a policy describing what users can learn from the cipher-text (Allison Lewko, 2010).

2.5.2.2 Deterministic Encryption (DE):

A Public-key Encryption (PKE) scheme is said to be deterministic if its encryption algorithm is deterministic. DE allows fast search over encrypted data (Alexandra Boldyreva, 2008). Deterministic leaks encrypted data that corresponds to the same values since it produces the same cipher-text from the same Palin-text. This encryption type allows the server to perform equality checks, which means it can perform selects with equality predicates, equality joins, GROUP BY, COUNT, DISTINCT, etc. (Raluca Ada Popa, 2011). However, deterministic encryption permits logarithmic time search on encrypted data, while randomized encryption only allows linear time search, meaning a search requires scanning the whole database. This difference is crucial for large outsourced databases that cannot afford to slow down search (Bellare, 2008).

2.5.2.3 Order-Preserving Encryption (OPE):

A practical approach that facilitates querying directly over encrypted data (without decrypting) is an Order-Preserving Encryption (OPE) scheme. Assume the plain-text values consist of unique values, which will be represented as $P = p_1, p_2 \ldots p_i < p_{i+1}$. The corresponding encrypted values will be stored as $C = c_1, c_2 \ldots c_i, c_{i+1}$. This means that the storage of c_i and c_{i+1} is preserving the order of p_i and p_{i+1}, such that, the order of the plain-texts is preserved in the cipher-texts (Rakesh Agrawal, 2004). So that, the OPE scheme permits the untrusted server to process SQL queries over encrypted data especially comparison queries and range queries (Hakan Hacıg¨um¨us, 2002) (Hasan KADHEM, 2010).

2.6 OPE Schemes:

The notion of the OPE scheme was proposed earlier to allow the comparison operation directly performed on encrypted data without decryption. The OPE scheme mapped the ***i th*** value in the plain-text values to the ***i th*** value in the cipher-text values (Rakesh Agrawal, 2004). In such a scheme the order of the plain-texts is preserved in the cipher-texts. Thus, equality queries, range queries, MAX, MIN, and COUNT queries can be directly processed over encrypted data. Also, GROUP BY and ORDER BY operations can be applied. Only there is a need for value decryption when applying SUM or AVG (Rakesh Agrawal, 2004).

However, the first formal study of the concept and its security was performed by (Alexandra Boldyreva, 2009). Also, the study proposed the strongest security definition for order-preserving schemes which is the IND-OCPA (INDistinguishability under Ordered Chosen-Plaintext Attack). The IND-OCPA for order-preserving encryption allows the adversary to learn nothing about the plaintext values except for their order (Alexandra Boldyreva, 2009). Despite the large body of works on OPE schemes (Seungmin LEE, 2009) (Hasan KADHEM, 2010) (Alexandra Boldyreva, 2011), none of them achieves the IND-OCPA. The prior schemes reveal more information about the plain-text besides the order. Table 2.1 explains the security provided by the previous OPE scheme and the mOPE scheme. Furthermore, it shows that the first ideal order-preserving encoding scheme that achieves IND-OCPA security is the mOPE (Raluca Ada Popa, 2013). It reviews the

security provided by previous OPE schemes and the mOPE scheme, including the cryptographic security guarantees provided by each scheme, and the information revealed by each scheme in addition to the order of the plaintext values (Raluca Ada Popa, 2013).

Table 2.1: Security provided by previous OPE schemes and mOPE scheme (Raluca Ada Popa, 2013).

Order-preserving scheme	Security Guarantees	Leakage besides order
(Gultekin Ozsoyoglu, 2004)	None	Yes
(Rakesh Agrawal, 2004)	None	Yes
(Alexandra Boldyreva, 2009) (Alexandra Boldyreva, 2011)	ROPF (Alexandra Boldyreva, 2009)	Half of plaintext bits
(Divyakant Agrawal, 2009)	None	Yes
(Seungmin LEE, 2009)	None	Yes
(Hasan KADHEM, 2010)	None	Yes
(Kadhem, 2010)	None	Yes
(Liangliang Xiao, 2012)	None	Yes
(Liangliang Xiao, 2012)	IND-OLCPA (Liangliang Xiao, 2012)	Yes
(Dae Hyun Yum, 2011)	ROPF (Alexandra Boldyreva, 2009)	Half of plaintext bits
(Dongxi Liu, 2012)	None	Most of the plaintext
(George Weilum Ang, 2014)	None	Yes
(Dongxi Liu, 2013)	None	Most of the plaintext
(Raluca Ada Popa, 2013)	Ideal: IND-OCPA	None

As explains in Table 2.1, the OPE schemes before the mOPE scheme leak more than the order of data and the mOPE scheme is the first one that achieves the IND-OCPA. This study is based on the mOPE model. The study improves the mOPE structure and combines it with an indexing technique to reduce the RRbCS.

Also, in the OPE schemes, the cipher-texts reveal the ordering information of the plain-texts. On the other hand, when it is desirable to have better performance for range queries while providing a reasonable degree of security, the OPE is a good option (Liangliang Xiao, 2012). However, the OPE schemes leak order information for plaintext values as the minimum needs for the order-preserving property, the authors in (Vladimir Kolesnikov, 2012) discusses approaches to limit the impact of this leakage.

2.7 Related Works:

There has been a significant amount of work on OPE schemes that allow querying over encrypted database without any change on the database engine (Gultekin Ozsoyoglu, 2004) (Seungmin LEE, 2009) (Kadhem, 2010) (Liangliang Xiao, 2012) (Liangliang Xiao, 2012).

The authors in (Hasan KADHEM, 2010) proposed MV-OPES (Multivalued-Order Preserving Encryption Scheme). The model allows queries over encrypted databases with an improved security level. Their idea is to change the approach of encrypting a value to another fixed value to prevent statistical attacks. MV-OPES allows an integer value to be encrypted to many values using the same encryption key while preserving the order of the integer values. By this, the attackers cannot infer information from the encrypted database even if they have statistical knowledge about the plaintext database.

(Dongxi Liu, 2012) Proposed an order-preserving indexing scheme based on a linear expression. The indexing scheme maps an input value v to $a * v + b$ + noise, where noise is a random value, where a and b are coefficients. The noise is selected in a way that preserves the order of input values. The scheme is combined with the encryption algorithms to deal with querying over the encrypted database. It has the proxy to communicate between the database applications and the encrypted database. The proxy uses the indexing mechanism to generate the index for the input value and also encrypts the

value with some encryption algorithm. The index and the encrypted value are then stored into corresponding fields in the same record of the database. When a range query is made, the proxy calculates the index of the value in the query condition, which is then used by the database service to search indexes stored in the databases.

(Raluca Ada Popa, 2013) Proposed the mutable Order Preserving Encryption (mOPE) model, the first OPE scheme that achieves the IND-OCPA. The mOPE scheme uses the Deterministic (DET) encryption to encrypt the values. Also, it orders the values in a BBST. Since it achieves the IND-OCPA, it reveals no information for the data except for the order of the plain-texts.

However, some works were built on the mOPE model. (K. Srinivasa Reddy, 2014) Introduced a novel scheme called "Randomized Order Preserving Encryption" (ROPE). This scheme follows the mOPE scheme proposed in (Raluca Ada Popa, 2013). It replaces the DET encryption with RND (random encryption) for the cipher-texts which prevents the leakage of information about the distribution of plaintext values. Although the cipher-texts are randomized they still maintain the plaintext order. Also, the scheme shows a query retrieval time overhead. Nevertheless, it achieves the perfect security objective IND-OCPA, in which the adversary can know nothing except for the order of the plain-texts. Also, the study by (Florian Kerschbaum, 2014) was improved on Popa's mOPE results and provided IND-OCPA.

The OPE scheme is also related to cryptographic schemes for performing queries over encrypted data. (POPA, 2014) Provided CryptDB, a system to protect data confidentiality against threats by executing SQL queries over the encrypted database. The architecture of CryptDB consists of a database proxy that intercepts the client's requests, transforms them into an understandable form to the Database Management System (DBMS) server, and returns the expected results to the client. It is also responsible to perform all the encryption and decryption processes. Interestingly, CryptDB uses the mOPE scheme (Raluca Ada Popa, 2013) to perform order operations efficiently on an encrypted database. Actually, they replace the BBST in the mOPE model with the Adelson-Velskii and Landis (AVL) binary search tree which reduced the cost of the OPE encryption (Raluca Ada Popa, 2011).

(Do Hoang Giang, 2017) Investigated certain types of complex queries namely multi-dimensional range queries over encrypted data in cloud platforms (where the plaintexts are multi-numerical attributes). Their solution combines a packetization based scheme with order-preserving encryption-based techniques.

However, (Kim, 2019) Presented a construction to improve the round and client-side storage on the ideally-secure mutable order-preserving encryption protocol. The construction of the OPE is based on the Order Revealing Encryption (ORE: a special type of symmetric encryptions). To present a non-interactive, ideally-secure OPE with a constant-size client storage construction; they used the public comparison functionality of the ORE. Their resulting constructions achieve better enhancement both on the number of rounds and the client-side storage, but cannot guarantee the ideal security.

However, the study in (Anselme Tueno, 2020) presented the Oblivious Order Preserving Encryption (OOPE) scheme based on the mOPE scheme to introduce an interactive protocol. It's a secure multiparty protocol that enables secure range queries for multiple users (such as data owners and data analyst) in the OPE scheme. It allows the data analyst to execute queries on the OPE database without revealing his private queries or revealing the encryption key of the data owner.

2.8 Summary:

This chapter provides background about outsourcing database to the cloud server and the related issues. It explains cloud computing concepts and architecture. Also, it reviews the security of cloud computing and explores the methods for computing over encrypted data especially for the OPE schemes. And, it covers the related works in the area of the OPE schemes and querying over encrypted data. Especially, it reviews some studies that enhanced the mOPE model.

CHAPTER III

THE PROPOSED ENHANCED RANGES ORDER PRESERVING ENCRYPTION MODEL

3.1 **Introduction**:

This chapter describes the enhanced OPE model which is proposed in this research. It works out how to reduce the Requests and Responses between the Client and the Server (RRbCS) regarding the existing mOPE model, to improve the mOPE model, and to develop an enhanced OPE model that reduces the RRbCS and sequentially enhanced the performance of search over encrypted data.

3.2 The Original mOPE Model:

One of the most efficient approaches that allow the server to compute over encrypted data is the OPE scheme. (Raluca Ada Popa, 2013) Presented the first IND-OCPA order-preserving encryption scheme called the "mutable Order-Preserving Encoding" or "mOPE" model (the word encoding was used instead of encryption). Moreover, the proposed enhanced OPE model in this research add improvement to the mOPE model.

However, in the beginning, the mOPE model will be explained to understand the behavior of the original model. The mOPE model functions by building a Balanced Binary Search Tree (BBST) comprising all the encrypted data on the server. In such a tree for each node v, all the nodes in the left subtree are smaller than v and all the nodes in the right subtree are greater than v. Figure 3-1 shows the OPE tree which adapted from Popa's mOPE model. In the figure each node contains the cipher-text, the filled blocks show the corresponding plain-text value (Raluca Ada Popa, 2013).

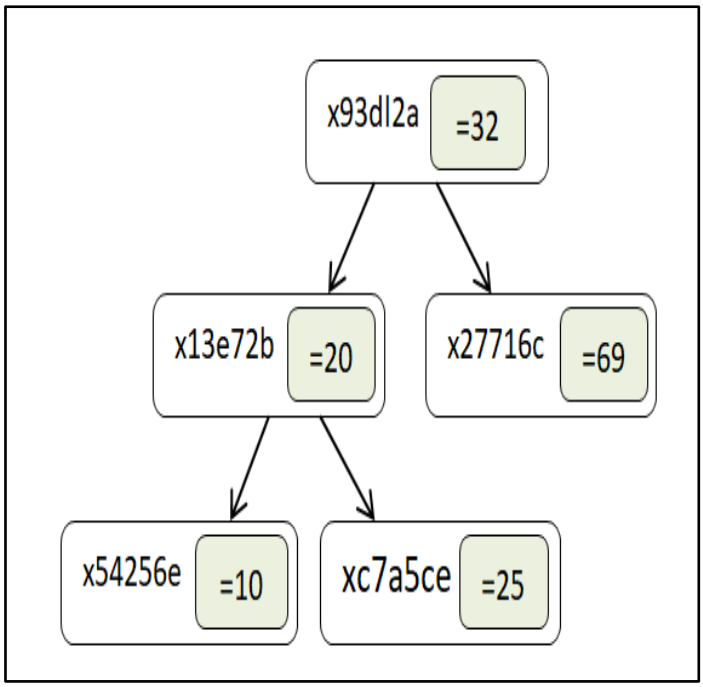

Figure 3-1: The OPE tree in mOPE model (Raluca Ada Popa, 2013)

Figure 3-2 describes the general algorithm that functions the OPE tree in Figure 3-1. The algorithm inputs a value v and outputs the resulting pointer in the OPE tree, the path in the OPE tree from the root, and whether v was found. It uses the variables: c′ : the cipher-text at the tree node, v′ : the decryption of the cipher-text, c": the next cipher-text. Also, the algorithm uses the Cl for Client and Ser for Server to indicate at which party each piece of computation happens (Raluca Ada Popa, 2013).

Algorithm 1 (OPE Tree traversal for a value v).

1: Cl $\leftrightarrow$ Ser: The client asks the server for the ciphertext c' at the root of the OPE tree.
2: Cl: The client decrypts c' and obtains v'.
3: Cl $\rightarrow$ Ser: If $v < v'$, the client tells the server "left"; if $v = v'$ the client tells the server "found"; if $v > v'$, the client tells the server "right".
4: Ser: The server returns the next ciphertext c'' based on the client's information, and goes back to step 2.
5: Ser, Cl: The algorithm stops when v is found, or when the server arrives at an empty spot in the tree. The outcome of the algorithm is the resulting pointer in the OPE Tree, the path in the OPE Tree from the root, and whether v was found.

Figure 3-2: The OPE tree traversal algorithm of the mOPE model (Raluca Ada Popa, 2013).

For instance, consider the setting in which the client inserts 55 into the server using the OPE tree shown in Figure 3-1 and the algorithm explained in Figure 3-2. In the beginning, the client demands the server for the root node of the OPE tree, and then the server returns the encrypted value x93d12a. Which the client decrypts to 32 and compares 55 with 32 since 55 greater than 32 the client demands the server for the right child of the root node. The server responds with the encrypted value x27716c. The client decrypts x27716c to 69 and compares 69 with 55 since 55 lesser than 69 then the client demands the server for the left child of the new node. The server responds that there is no such child. This means that the server can insert 55 in this position (Raluca Ada Popa, 2013).

As we see in Popa's OPE tree, the server always needs to ask the client to move over the OPE tree. It returns to the client to know if it goes left or right in the next move. Unfortunately, this scenario generates many RRbCS. Also, the increasing of the database size increases the tree depth and the mOPE model generates more and more RRbCS.

3.3 The Enhanced Ranges Order Preserving Encryption (ROPE) Model:

This study aims to enhance the existing mOPE model by reducing the RRbCS in the mOPE model. To achieve this goal, the study presents an enhanced OPE model named "Ranges Order Preserving Encryption" (ROPE). The name Ranges comes from the fact that the outsourced client's data will be organized into ranges of data at the server-side according to the indexing mechanism. Also, it reduces the server relies on the client by allowing the server to perform some of the tasks without revealing any information except for the order of data. The next subsections will illustrate the proposed enhanced ROPE model in detail.

3.3.1 Definitions:

The following definitions clarify the terms and expressions used in the enhanced ROPE model.

- **The Client (or an enterprise):** The owner of data to be outsourced, thus the client is trusted.
- **The Database Server:** The storage where the outsourced client's data is stored, it's untrusted.
- **The Trusted Party:** An introduced party allocated in the client. It acts as a communication channel between the client and the server, it is trusted.
- **The Index Table:** A developed structure which stored in the trusted party. It plays the role of the indexing mechanism.
- **The Expected Client's Data (ECD):** The expected total number of records or tuples that the client will outsource to be stored in the database server.
- **The Range_Value (RV):** It's a suggested number that is used to divide the ECD by it to obtain groups of equal elements (1<RV<=ECD).
- **The Range:** It's a group of sequenced elements that contains some elements equal to the RV.
- **The Index Table Creation (ITC) algorithm:** The algorithm explains how to create the index table.

- **The Ranges Order Preserving Encryption (ROPE) algorithm:** The algorithm describes how the ROPE model works and how does it access the data stored in the database server.

3.3.2 The Trusted Party:

It's an introduced party in the enhanced ROPE model. It authenticates the client and acts as a communication channel between the client and the database server. It builds at the client-side allocated inside the client boundaries. It concerns about the encryption keys generation, encryption operations, and decryption operations. Moreover, it receives all the client's requests and intercepts all the RRbCS. The trusted party is responsible for encrypts the client's requests, decrypts the returned encrypted results from the database server, and gives the client the exact results. Besides, the trusted party is used to make some preparations in the initial stage of the enhanced ROPE model. Also, it will be explained that it applies two different techniques to preserve the order of the client's data.

3.3.3 The Index Table:

It is an essential part of the ROPE model. It should be created at the initialization stage of the database system and stored in the trusted party. To create the index table, firstly, it's required to know the ECD that will be outsourced to the database server, for example:

- If the client will outsource 10 values then the ECD = 10.
- If the client will outsource 100 values then the ECD = 100.
- If the client will outsource 1000 values then the ECD = 1000 …etc.

However, the second requirement to create the index table is the value of the suggested RV. Then, the ECD is divided by the suggested RV. As a result, we obtain some groups of elements; each group of elements is became a Range. Of course, the total number of Ranges depends on the value of the selected RV. Thus, the smaller RV has a shorter Range and produces many Ranges, whereas the greatest RV has a longer Range and produces a few Ranges. Note that, we construct the ROPE model for numeric data. For

simplicity, we implement the ECD and the RV in the forms of integer values with no repetition.

Therefore, the client determines the value of the ECD and sends it to the trusted party. Then, the trusted party uses the ECD and the selected RV to create the index table. However, the steps of the index table creation are formulated in an algorithm named "Index Table Creation" (ITC) algorithm. The following is the ITC algorithm:

Algorithm1: Index Table Creation (ITC)

Input: ECD, RV

Output: The index table

1. Input the values of the ECD and the RV.
2. Divide the ECD by the RV to obtain the Ranges.
3. Specify the start value and the end value for each Range.
4. Give each Range a unique Id_range.
5. Create the index table with three columns: the Id_range (key-value), the Start_value, and the End_value.
6. Insert the values of the Id_range, the Start_value, and the End_value for all ranges in the index table.
7. Commit the index table.

Figure 3-3: The ITC algorithm.

As we see in Figure 3-3, the ITC algorithm inputs the ECD and the RV; and outcomes the index table. Also, the algorithm uses the variables: Id_range, Start_value, and End_value to name the table columns; and the Range: to indicate a part of the client's data. The steps of the ITC algorithm are described in Figure 3-4.

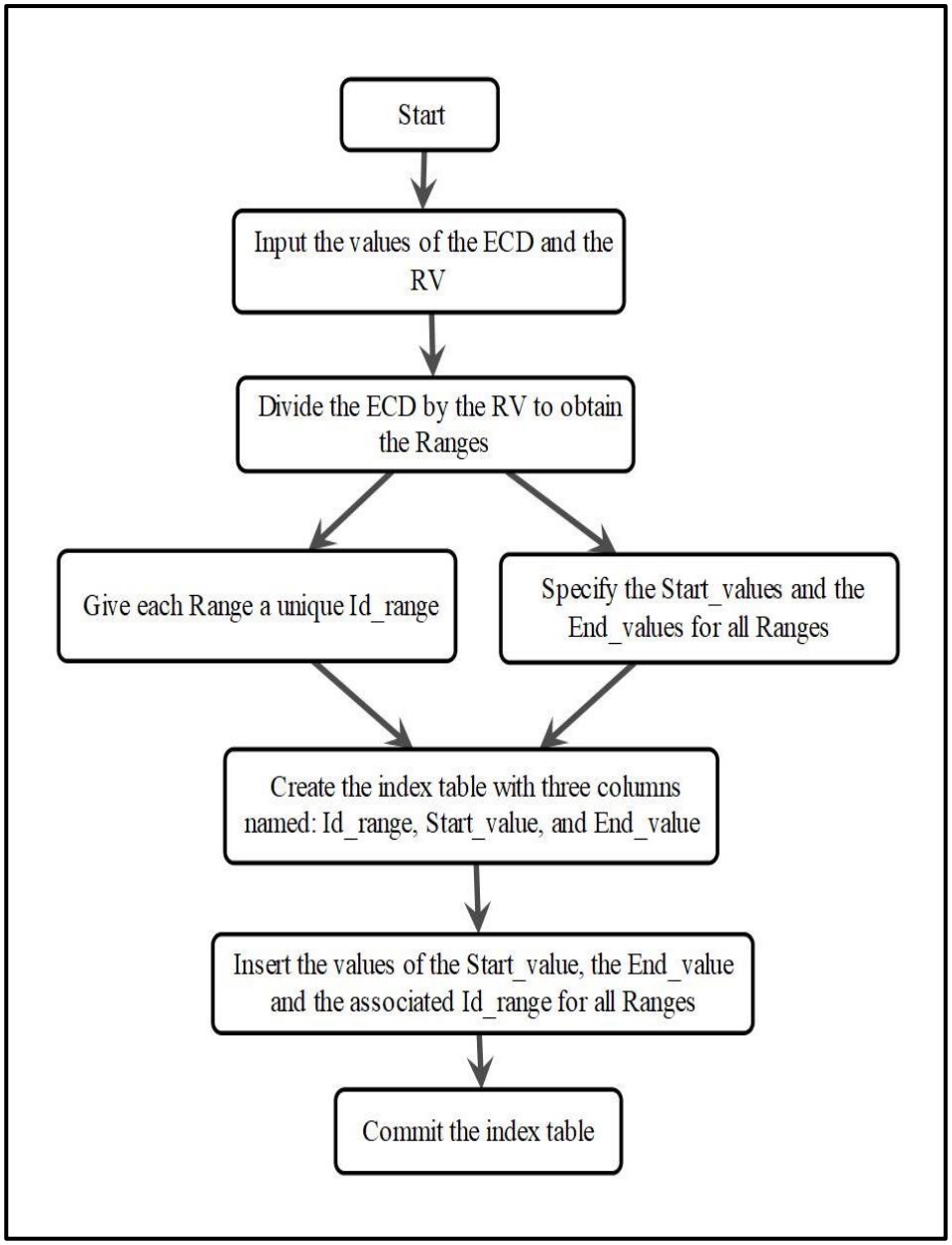

Figure 3-4: The ITC algorithm diagram.

As a result, we have the index table in the trusted party with several rows equal to the total number of Ranges. And each row contains a unique Id_range, Start_value, and End_value restricted to a specific Range. The two factors that affect the index table information are the value of the ECD and the length of the suggested RV. In the coming subsection, we will see how the index table information plays a basic role in storing ordered encrypted data in the server and how the results can be retrieved from the server.

3.3.4 A Framework for the Enhanced ROPE Model:

In this section, we design a Framework to be a formative tool for the ROPE model. In this Framework, the inputs are defined as the Data that should be outsourced to the cloud server or retrieved from. The framework is the integration of three main parts which are: the Client, the Trusted party, and the Server. These parts are communicating together in terms of exchanging data. Figure 3-5 shows the ROPE model framework.

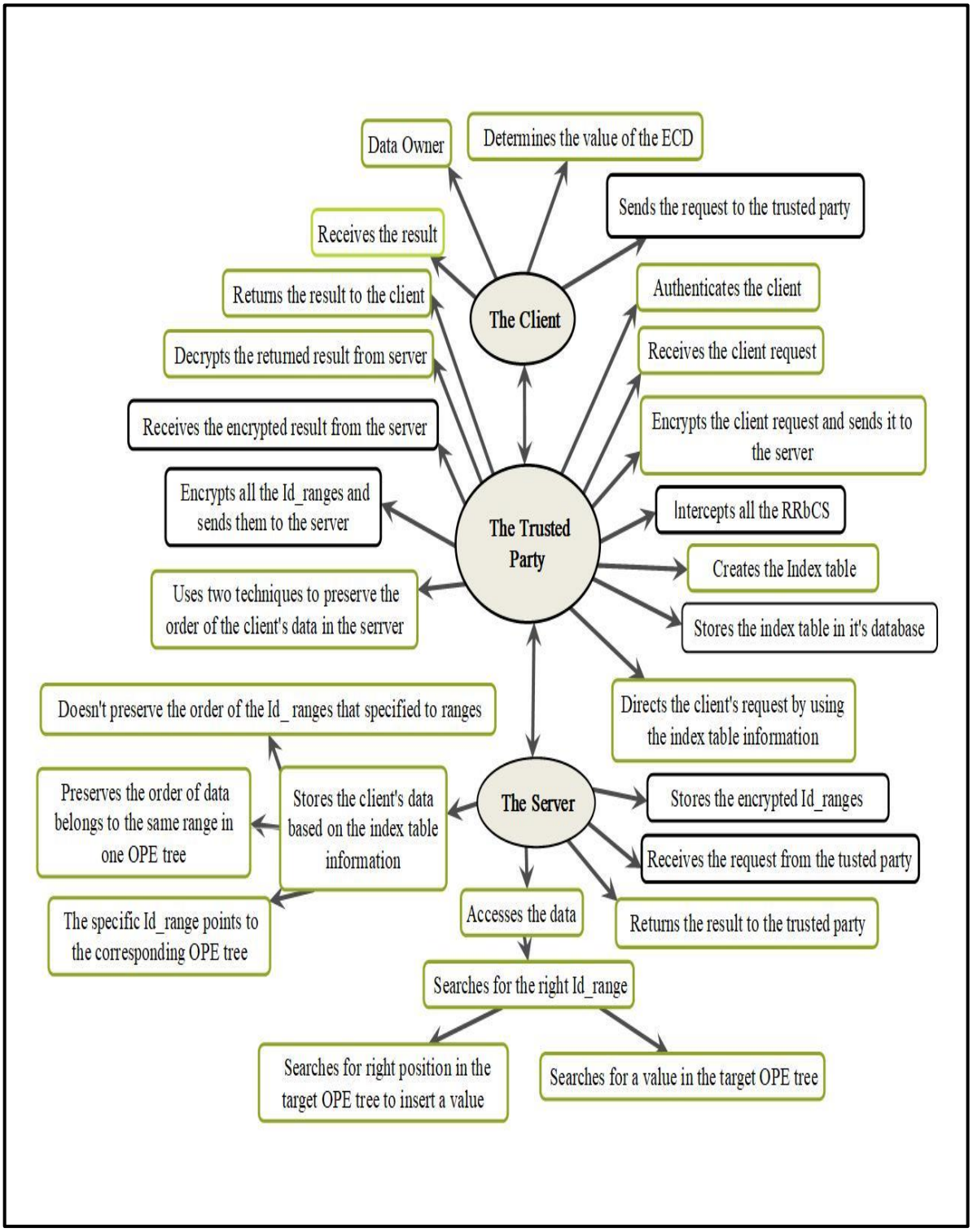

Figure 3-5: The enhanced ROPE model framework.

The following describes the three parts of the ROPE model framework and their responsibilities, as shown in Figure 3-5:

3.3.4.1 The Client:

The client is the owner of the data and it is authorized to determine the value of the ECD. It has a communication link with the trusted party to send its requests and receives the result.

3.3.4.2 The Trusted Party:

Most of the important duties are accomplished by a trusted party. It performs the following tasks:

- Authenticates the client.
- Receives the client's request.
- Encrypts the client request and sends it to the server.
- Creates the Index table.
- Stores the index table in it’s database.
- Encrypts all the Id_ranges of the index table and sends them to the server.
- Directs the client’s request by using the index table information.
- Intercepts all the RRbCS.
- Receives the encrypted result from the server.
- Decrypts the returned result from the server.
- Returns the result to the client.
- Uses the technique that doesn’t preserve the order of the Id_ranges of the index table in the server, and the other one that preserves the order of data in OPE trees.

3.3.4.3 The Server:

The server is the data storage and it is accomplished the following tasks:

- Stores the encrypted Id_ranges.
- Receives the trusted party request.
- Stores the client's data based on the index table information. It doesn't preserve the order of Id_ranges that specified to Ranges, preserves the order of data that belongs to the same Range in one OPE tree, and lets the specific Id_range points to the corresponding OPE tree.
- Searches for the right Id_range to reach the right value in the OPE tree (in case of the search operation) or to achieve the right position in the OPE tree (in case of the insert operation).
- Returns the result to the trusted party.

3.3.5 Preserving the Order of Data in the Server:

However, the power of the index table is appearing clearly when we need to access the database server. There is a main relationship between the index table information and the arrangement of data in the database server. It is important to deal with the client's values concerning the index table information (the Id_ranges and their related Ranges from the Start_values up to the End_values). That is, the client's data must reside in their corresponding Ranges as declared in the index table. Therefore, we need to link the client's data with the index table information to preserve the order of the client's data in the database server.

3.3.5.1 Dealing with the Id_ranges:

However, after the creation of the index table, all the Id_ranges of the index table must be encrypted. Then, the encrypted Id_ranges will be sent to store in the server by using a technique that might not preserve the order of data. As a result, the encrypted Id_ranges are presented in the server as unordered. More importantly, the trusted party must encrypt all the Id_ranges and store them in the database server before start

outsourcing the client's data to the database server. The secret behind this is to make the stored Id_ranges act as indices for the client's data.

3.3.5.2 Organizing the Client's Data into the Ranges:

In general, based on the index table information, the client's data is dividing into parts of data in the server. Each part contains values that belong to a specific Range as defined earlier in the index table (values from the Start_value up to the End_value that is associated with the exact Id_range). This means each part of the client's data corresponds to the Range in the index table. Moreover, the trusted party uses the technique that preserves the order of data to order the data that belongs to these Ranges. The trusted party follows Popa's mOPE to preserve the order of data in different Ranges. This means every Range of the client's data is organized into an OPE tree at the server. Ideally, the root node of each OPE tree is pointed by the appropriate encrypted Id_range that stored initially (as defined in the index table).

As a result, we have some OPE trees in the server equal to the number of Ranges in the index table. Every OPE tree contains nodes of values related to a specific Range in the index table (values from the Start_value to the End_value as defined in the index table). And every OPE tree is pointed by the encrypted value of the associated Id_range in the index table. In general, it can be said that the ROPE model preserves the order of data that belongs to the same Range, while it doesn't preserve the sequence of Ranges. This provides some aspect of security to user's data in the database server.

However, practically we implement the OPE trees instead of the AVL tree, as Popa does. Of course, the AVL tree has advantages of the BST, in the AVL tree for each node v, all the left subtree nodes of v are smaller than v and all the right subtree nodes of v are greater than v. Also it is a self-balance tree, so that after balancing the OPE tree some nodes may change their positions (Princiraj, 2019).

Further, we implement the nodes of the tree in the server as pre-ordered traversal (root node, left subtree nodes, and right subtree nodes respectively). This allows us to restore the original tree construction from the database to perform operations. Because of the tree balancing and the need for maintaining the pre-order traversal implementation, the

server must update any storage related to the balanced OPE tree. Note that, such an update affects only the target OPE tree not all the other OPE trees corresponding to the other Ranges.

3.3.6 The Access of Data in the Enhanced ROPE Model:

As it mentioned before, there are basic requirements to function the enhanced ROPE model, which are:

- Build the index table (by using the ECD, the RV, and run the ITC algorithm).
- Encrypt all the Id_ranges of the index table and store them in the server initially.

After provided these requirements it can be able to run the ROPE algorithm and build the OPE trees. The enhanced ROPE algorithm runs by the client, the trusted party, and the server. The algorithm obtains v as an input from the client and outcome the resulting pointer in the target OPE tree (to insert v) and whether v was found. Also, the algorithm functions either to insert a value v in the server or to search for v. The algorithm variables are v: the client value, Enc(Id_range): the encrypted Id_range, c^{-}**:** the cipher-text at the root node of the target OPE tree, $c^{\approx}$**:** the cipher-text at OPE tree node**,** and v^{-}**:** the decryption of cipher-text. Figure 3-6 describes the enhanced ROPE model algorithm.

Algorithm 2: The enhanced ROPE scheme

Input: The value v

Output: The resulting pointer in the target OPE tree (to insert v) and whether v was found.

1. The client sends a value v to the trusted party (either to insert or search for).
2. The trusted party receives v and finds a suitable Id_range for v from the index table.
3. The trusted party encrypts the id rang into Enc(Id_range) and sends it to the server.
4. The server obtains the Enc(Id_range), then:
    a) It starts a sequential search for a matching value to Enc(Id_range), by performing equality checks.
    b) When a matching value is found it tells the trusted party.
5. The trusted party asks the server for the cipher-text c^{-} at the root node of the target OPE tree that the Enc(Id_range) points to.
6. The trusted party receives c^{-}, decrypts it, and obtains v^{-}.
7. The trusted party compares v^{-} with the client value v:
    a) If $v < v^{-}$ the trusted party demands the server for the left child of the current node.
    b) If $v > v^{-}$ the trusted party demands the server for the right child of the current node.
    c) If $v = v^{-}$ the trusted party tells the server it is found.
8. The server returns the cipher-text $c^{\approx}$ at the current tree node to the trusted party and goes back to step 6.
9. The algorithm stops when v is found, or when the server achieves an empty node in the tree.

Figure 3-6: The enhanced ROPE model algorithm.

However, the ROPE algorithm shows that we have to go through two steps to get the expected outcome. The first step is to reach the right Id_range in the server that points to the intended OPE tree and the second step is to search for the appropriate location in the target OPE tree, as follows:

- **Step 1:** Searches for the right Id_range in the server (step 1 to 4 in the ROPE algorithm). As we explain in Figure 3-7, after the trusted party receives v, it looks in the index table to find the corresponding Id_range. Then it encrypts the found value by using the same encryption algorithm that is used before to encrypt the Id_ranges and obtains Enc(Id_range). Next, the trusted party sends the Enc(Id_range) to the server. Then, the server receives the Enc(Id_range), starts searching for a matching value, and finds the matching value. By this, we reach the exact pointer to the target OPE tree.

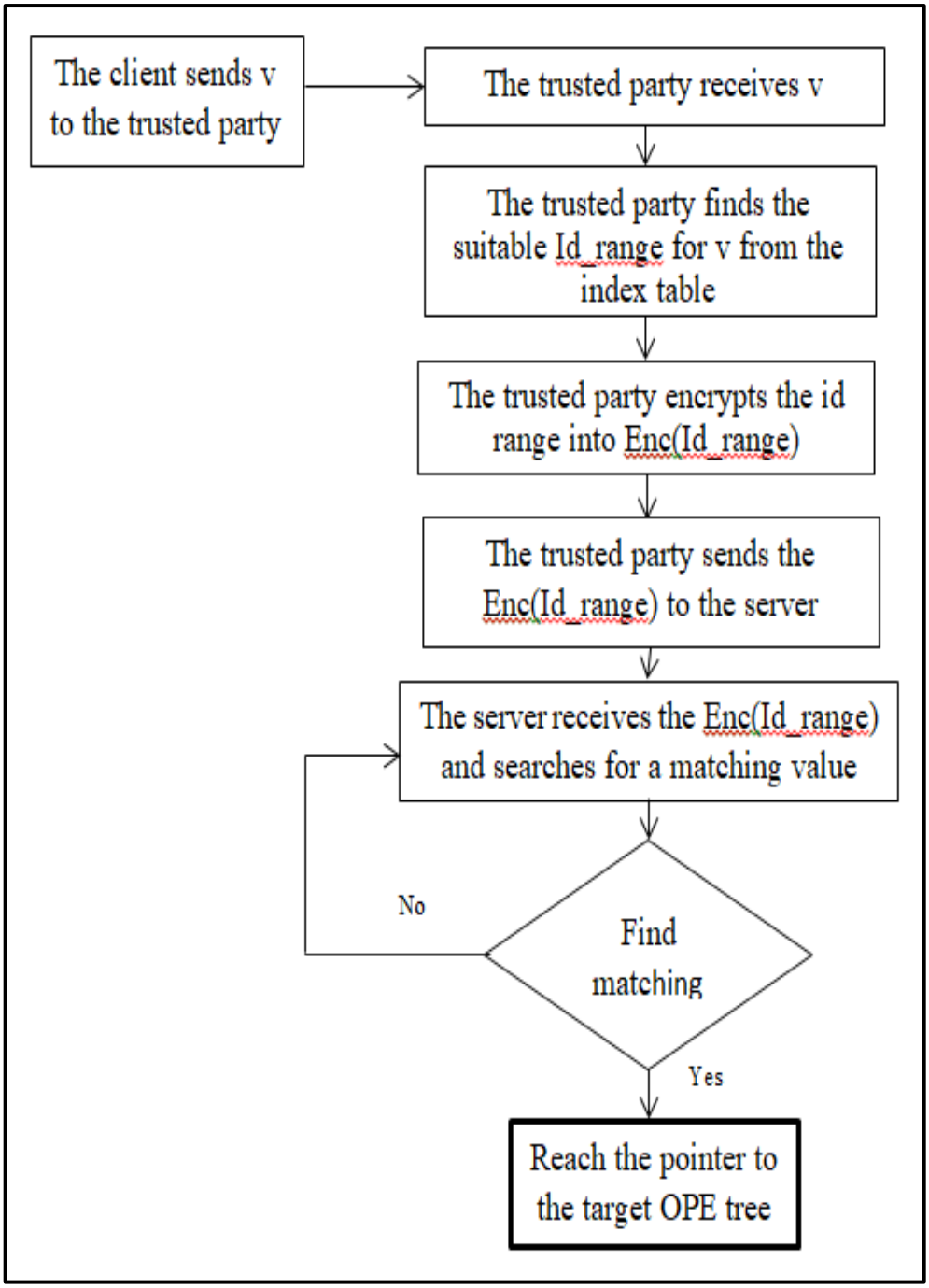

Figure 3-7: Searches for the right Id_range diagram.

- **Step 2:** After reaching the pointer of the target OPE tree, the next step is to search for the right location in the target OPE tree (steps 5 to 9 in the ROPE algorithm). As Figure 3-8 illustrates, the trusted party asks the server for the pointed cipher-text c^- at the root node of the target OPE tree. The trusted party receives c^-, decrypts it to v^-, and compares v with v^-. The next move in the server is based on the result of this comparison. If $v < v^-$ the trusted party demands the server for the left child of the current node; if $v > v^-$ the trusted party demands the server for the right child of the current node; and if $v = v^-$ the trusted party tells the server it is found. Moreover, unless $v= v^-$, the server returns the next cipher-text $c^{\approx}$ at the current tree node to the trusted party. Then, the same cycle is repeated until $v = v^-$ or the server achieves an empty node in the OPE tree.

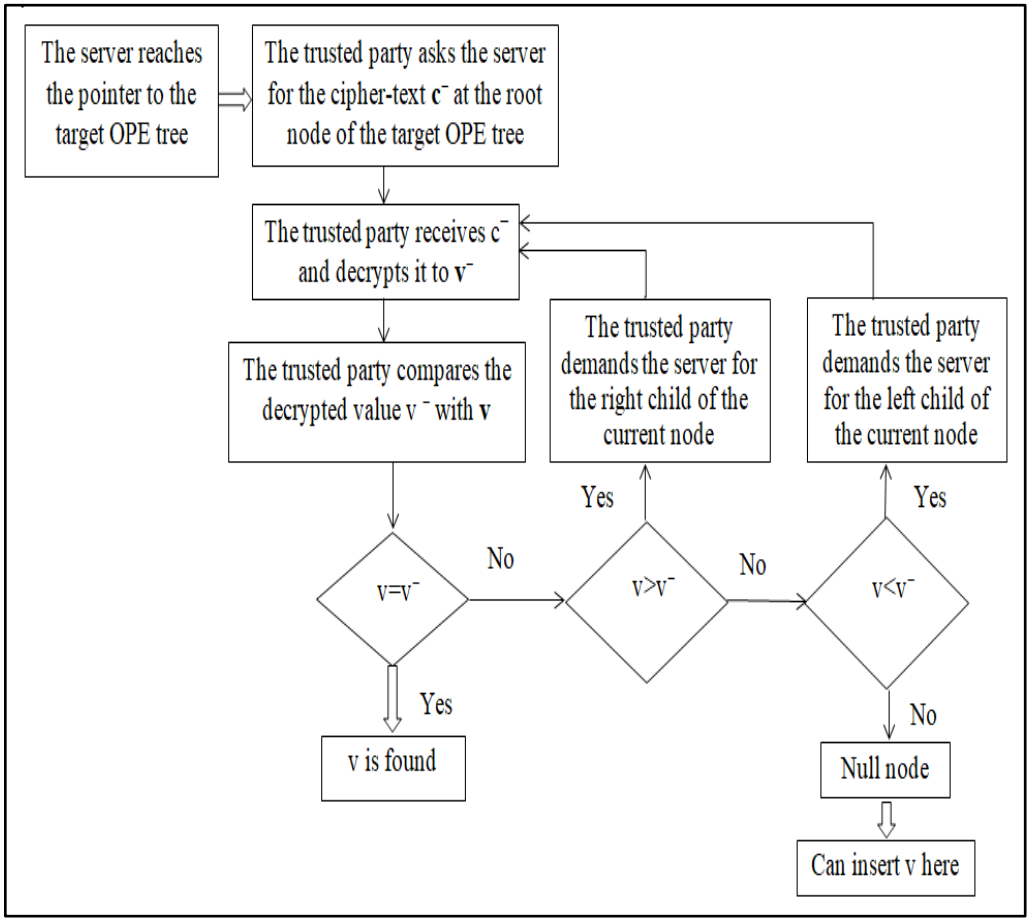

Figure 3-8: Search for the right location in the target OPE tree diagram.

The following points can be observed from the ROPE algorithm:

- The server can search for the right Id_range independently to reach the intended OPE tree.
- After reaching the intended OPE tree the server needs the client's help to move over the OPE tree (actually the trusted party at the client-side will help the server).
- The search for the right Id_range is performed locally on the server.
- The search over the OPE tree is performed globally across the network (generates RRbCS).
- So, instead of return to the trusted party in all cases we just return when the data preserves the order.

It can be summarize, the enhanced ROPE model replaces the one OPE tree in the mOPE model with sub-OPE trees combined with the indexing mechanism. Further, despite maintains the order of data in the sub-OPE trees it doesn't preserve the sequence of them.

3.3.7 A Case Study Example:

This section explains an example to illustrate how the enhanced ROPE model works. The example uses ECD=20 from 1 to 20 and RV=5. The next part will explain how to function the enhanced ROPE model regarding this case study.

3.3.7.1 Create the Index Table:

To create the index table we apply the ITC algorithm using the ECD=20 and the RV=5. The obtained index table can be as shown in Table 3.1.

Table 3.1: The index table of the case study

Id_range	Start_value	End_value
1	1	5
2	6	10
3	11	15
4	16	20

3.3.7.2 Data Encryption and Organization:

Moreover, considering the index table in Table 3.1, the data can be ordered in the server like the following:

- The Id_ranges: 1, 2, 3, and 4, are encrypted and then they stored in the server by using the technique that doesn't preserve the order of them.
- The incoming client's data will be encrypted and then stored in the server by using the technique that preserves the order of data like the Popa's one.

Therefore, the client's data will be organized into four OPE trees. The nodes of the OPE trees will contain values as described below:

- An OPE tree that has nodes values from 1 to 5.
- An OPE tree that has nodes values from 6 to 10.
- An OPE tree that has nodes values from 11 to 15.
- An OPE tree that has nodes values from 16 to 20.

For instance, suppose the actual client's values are: 1, 3, 4, 13, 14, 15, 16, and 19 using the index table shown in Table 3.1. The organization of this data in the database server can be illustrated as shown in Table 3.2. The grey blocks show the encrypted Id_ranges, while the white one shows the encrypted client's data. Also, the label ($\tilde{}$) is used to denote the part of data that applies the technique does not preserve the order of data. And the label ($\bar{}$) is used to denote the part of data that applies the technique preserves the order of data.

Table 3.2: The client's data in the database of the case study

$2^{\approx}$					
$4^{\approx}$	16^{-}	19^{-}			
$1^{\approx}$	3^{-}	1^{-}	4^{-}		
$3^{\approx}$	14^{-}	13^{-}	15^{-}		

As it is illustrated in Table 3.2, the encrypted client's values are arranged in their suitable location as declared earlier in the index table. Every encrypted Id_range references an OPE tree that contains nodes of values related to a specific Range as specified in the index table. And all of the OPE trees are implemented in the form of the pre-ordered traversal tree (the root node, the nodes of the left subtree, and the nodes of the right subtree).

3.3.7.3 The Insert Operation:

Moreover, assume the client wants to insert 2 using the database shown in Table 3.2 and the related index table expressed in Table 3.1. Whenever the trusted party receives the value, it looks in the index table for a suitable Id_range. It finds that the appropriate Id_range that 2 can reside in is 1. Next, it encrypts 1 by the same encryption technique that is used before to encrypt the Id_ranges and obtains $\mathbf{1}^{\approx}$. Then, it asks the server to search for $\mathbf{1}^{\approx}$. The server makes its first try: $2^{\approx}$ doesn't match $\mathbf{1}^{\approx}$, the second try: $4^{\approx}$ doesn't match $\mathbf{1}^{\approx}$, and the third try: $\mathbf{1}^{\approx}$ matches $\mathbf{1}^{\approx}$. At this point, the server reaches the right Id_range that points to the target OPE tree.

After that, the server needs the help of the trusted party to move over the target OPE tree to find an empty node. Figure 3-9 illustrates the insert of 2 in the database of the case study and the related database after that. Figure 3-9 (a) shows the target OPE tree. Firstly, the trusted party requests the root node of the OPE tree, and the server returns 3^{-}, which the trusted party decrypts to 3. Since $2 < 3$, the trusted party requests the left child of the root node, and the server responds with 1^{-}, which the trusted party decrypts to 1. $2 > 1$, so the trusted party requests the right child of the last node requested, and the server responds that there is no such child. This means that the trusted party can insert 2^{-}, the encrypted value of 2, in this position. Remember that, the trusted party encrypts 2 by the same encryption

technique that is used before to encrypt the client's data. Figure 3-9 (b) displays the OPE tree after inserting 2⁻. Lastly, there is a need to update the part of the database storage where the OPE tree is storing. This is because we have to maintain the pre-order traversal tree in the server. Figure 3-9 (c) shows the related storage part after updates.

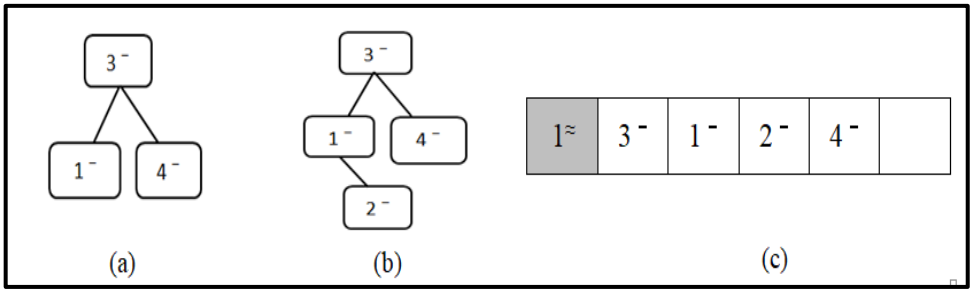

Figure 3-9: The insert of 2 in the database of the case study and the related storage after that.

From the previous example, it is observed that the server can perform the search for the right Id_range without any trusted party involvement, while it needs the help of the trusted party to find a suitable location in the OPE tree where to insert 2⁻. Also, we see that the server makes three tries until it finds the intended Id_range, after that, there are 6 requests and 6 responses between the trusted party and the server before reach an empty node to insert 2⁻. Thus, it can be said that the server is responsible to perform all the tasks that are related to the data that does not preserve the order independently (the encrypted Id_ranges), while it needs the help of the trusted party to accomplish the works over the OPE trees. More importantly, the trusted party told the server order information only for the exact OPE tree and the interaction between them doesn't reveal anything else besides the order of the intended OPE tree.

3.4 Research Methods for the Enhanced Model Evaluation:

However, to evaluate the performance of the ROPE model the study go through several steps, as it addresses in Figure 3-10. The figure describes that there are two different simulation programs were designed, one to simulate the enhanced ROPE model and the other one to simulate the mOPE model. The two simulation programs process the same data with the same characteristics. To measure the performance of the enhanced

ROPE model, the study recognizes two parameters. The first parameter monitors the works performed by the server. It computes how many times the server tests the stored encrypted Id_ranges to find the right Id_range that points to the target OPE tree. This parameter is named “Tests by the Server” (TbS). The second parameter monitors the works that are performed across the network by the client and the server. And it counts the RRbCS to reach the right position in the target OPE tree. Note that, the trusted party is the actual entity that sends the requests and receives the responses at the client-side, and it builds in the client. Because of this, the client is used instead of the trusted party.

Furthermore, unlike the enhanced ROPE model, the mOPE model involves only one parameter to compute the RRbCS. This is because the server completely depends on the client to perform the tasks. Of course, the findings of RRbCS from the two models are used for the comparison purpose, whereas the findings of the TbS in the ROPE model should be acceptable. Bring in your mind, in the two models we don’t take into account the RRbCS of making the tree balancing.

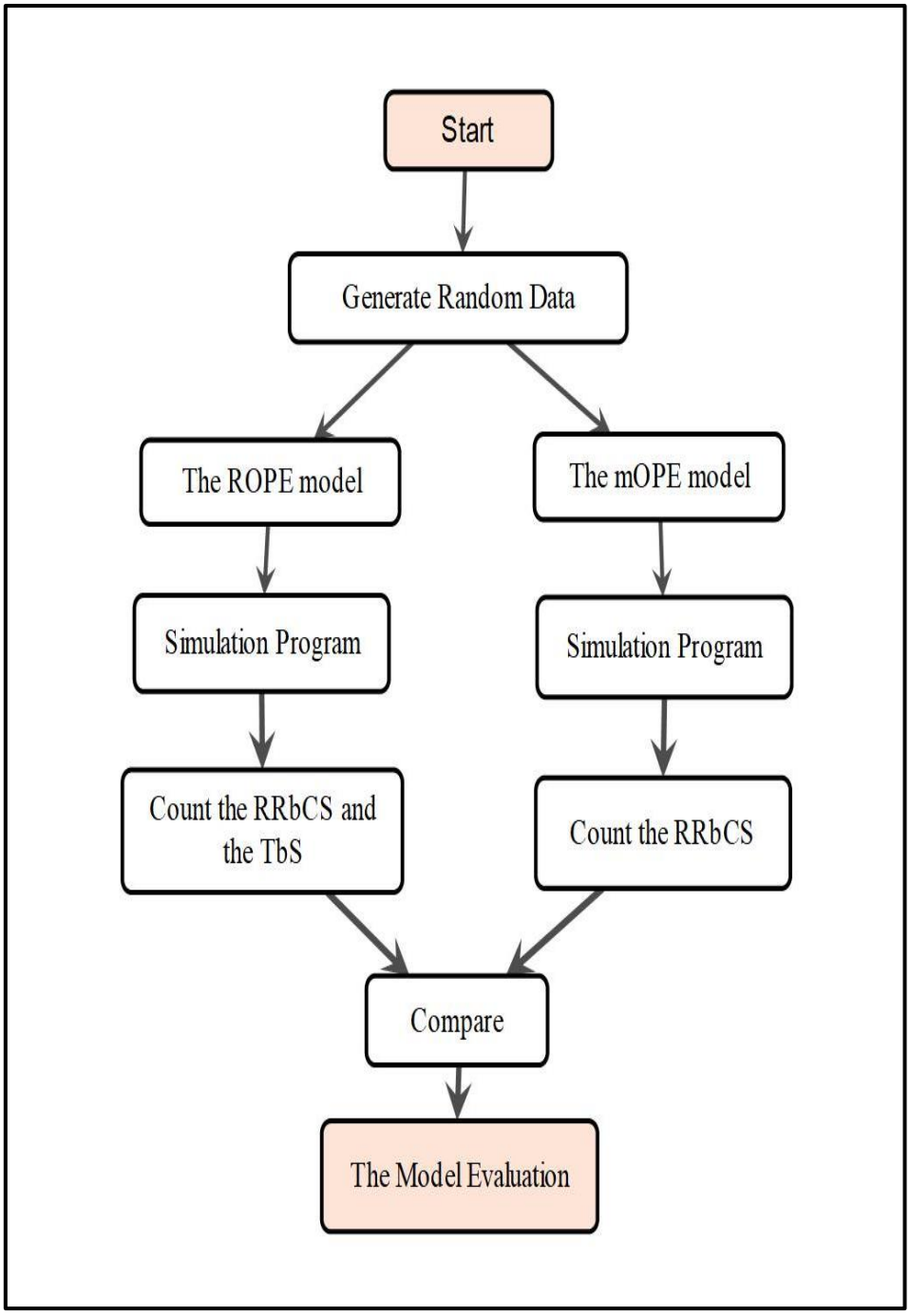

Figure 3-10: The research methods for the evaluation diagram.

3.5 Research Tools and Data:

The validation of the proposed enhanced model is done by using a simulation program in C code. User-defined functions were used to describe all the scheme activities, and to conduct the experimental studies. Moreover, random distinct integer numbers were generated for conducting experimental studies and testing the validity of the proposed model. Borland C++ 5.02 was used to implement the mOPE model, while C Free 2015 was used to execute the structure of the enhanced ROPE model. And, Mind Maple Lite was used to draw figures and diagrams.

3.6 Summary:

This chapter discusses the proposed enhanced ROPE model. It explains the ROPE model architecture and designs the ITC algorithm to create the index table. Also, it presents a framework to be a formative tool for the enhanced model. Especially, it presents how the ROPE preserves the order of data as an approach to achieve the research objectives. Further, it illustrates the used techniques to preserve the order of encrypted data, the ROPE algorithm to access the database, and explains a case study as an example to practice the ROPE model. Also, it explains the research methods for the model evaluation and the used tools and data. Besides, an enhanced OPE model is proposed for search over encrypted data.

CHAPTER IV

IMPLEMENTATION AND TESTING

4.1. Introduction:

This thesis proposes the enhanced Ranges Order Preserving Encryption (ROPE) model for reducing the Requests and Responses between the Client and the Server (RRbCS) and enhancing the performance of search over encrypted data in cloud computing. This chapter shows the implementation of the enhanced ROPE model facing the mutable Order Preserving Encryption (mOPE) model. To do that, it presents two case study scenarios for both of the models. In the first scenario, the client sends his data for storing in the database server. And in the second scenario, the client retrieves his data from the database server. Also, some experiments are conducted to test the enhanced model in the two scenarios.

4.2. The Enhanced ROPE Model Implementation:

Moreover, to investigate the enhanced ROPE model and evaluate its performance, we design a simulation program partitioning into three subprograms that are:

- **Subprogram 1:** The system initialization. Creates the index table by applying the ITC algorithm (page -26). Also, it prepares the database by allocating the Id_ranges.
- **Subprogram 2:** Inputs the data and builds the OPE trees of the ROPE model. It is the insertion program that uses the index table created by subprogram 1. This program applies the ROPE algorithm (page 34) to insert the data.
- **Subprogram 3:** Searches for target values. It uses the resulted database from the insertion program and the corresponding index table. And it also applies the ROPE algorithm (page 34) to search for the target values.

However, the investigation of the enhanced ROPE model is formulated into two scenarios: one to implement the insert operations, and the other one to implement the search operations. Further, the model is implementing in different Range_Values (RV) to extract the effect of the RV on the outcomes of the tests made by the server to find the target OPE trees and the requests and responses between the client and the server to reach the right positions in the target OPE trees (TbS and RRbCS parameters) on both of the scenarios. The study uses a fixed value for the Expected Client's Data (ECD) equals 100 to conduct the insert operations experiments. Actually, it generates random distinct integer numbers from 1 to 100 that represent the client's data. On the other side, the same values with a fixed sequence are used to conduct the search operations experiments.

As Figure 4-1 displays, the implementation of the ROPE model for a specific RV is accomplished in three steps:

- **In the first step:** Run subprogram 1 using the ECD and the selected RV, to create the index table and to allocate the Id_ranges in the database.
- **In the second step:** Run subprogram 2 to conduct the insert experiment using the same index table created in the first step. Before this, random distinct integer numbers from 1 to 100 were generated to represent the inputs for subprogram 2. The outcomes of this step are the results of the RRbCS and the TbS of the insert experiment in a specific RV plus its ordered database.
- **In the third step:** Run subprogram 3 to conduct the search experiment using the ordered database that resulted from the second step. Random twenty numbers were selected to be the searched values in this step. The outcomes of this step are the results of the RRbCS and the TbS of the search experiment in a specific RV.

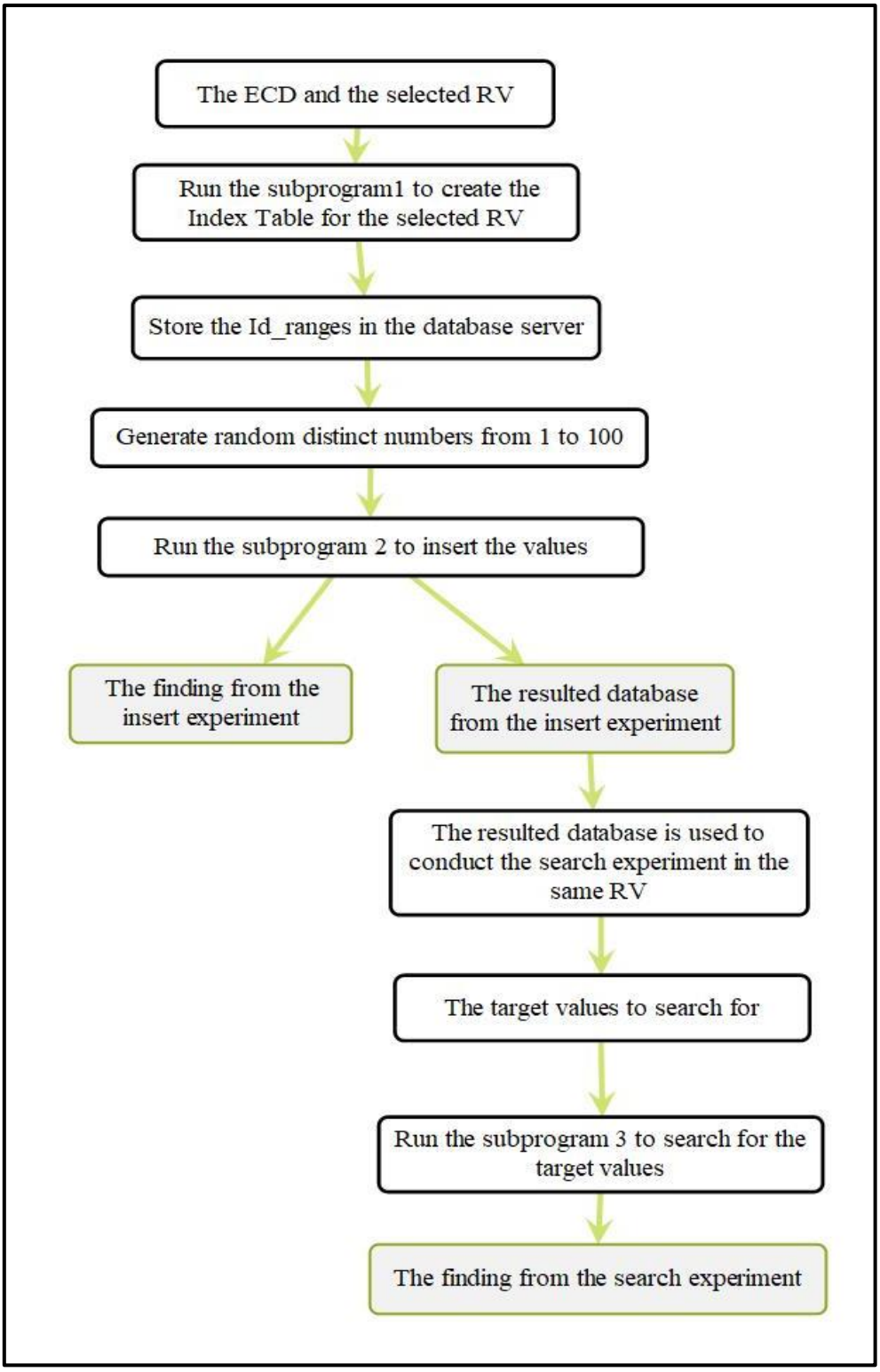

Figure 4-1: The ROPE model experiments approach.

4.2.1 The Experiments of the Insert Scenario:

In the first scenario of the experiments, the study examines the efficiency of the enhanced ROPE model in the case of inserting data into the database server. It performs eleven experiments to test the insert operations in the ROPE model using fixed ECD and different RV values that vary from small to great. The eleventh experiments aimed to compute TbS and RRbCS parameters after inserting the client's data. The experiments strategy is to start with a small RV and increasing it for every experiment. In the beginning, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 are selected to be the RVs for the eleven experiments. Then, each RV is used with the ECD to create the index table for a separate experiment. Further, the generated random distinct integer numbers are saved in a file as a fixed sample of data. This sample of data is used to form the inputs for all of the insert operations experiments (same data in fixed sequence). Of course, all of the experiments have ECD equal to 100. Actually, in each experiment the study did the following:

1. The index table for the specific RV was created by running subprogram 1.
2. The Id_ranges of the index table were stored in the database server (actually they were stored randomly to implement them as unordered).
3. The data was inserted into the ROPE database by running subprogram 2.

The next part will explain the selected RV and the index table details for every experiment.

- **Experiment No.1:** The ECD=100, the RV=5. Table 4.1 shows the index table that was created according to the selected RV. The index table had twenty Ranges and there were twenty Id_ranges specified to them. The Id_ranges, the Start_values, and the End_values used in this experiment are explained below:
    - Id_range = 1, the Start_value = 1, and the End_value = 5.
    - Id_range = 2, the Start_value = 6, and the End_value = 10.
    - Id_range = 3, the Start_value = 11, and the End_value = 15.
    - Id_range = 4, the Start_value = 16, and the End_value = 20.
    - Id_range = 5, the Start_value = 21, and the End_value = 25.
    - Id_range = 6, the Start_value = 26, and the End_value = 30.
    - Id_range = 7, the Start_value = 31, and the End_value = 35.

- Id_range = 8, the Start_value = 36, and the End_value = 40.
- Id_range = 9, the Start_value = 41, and the End_value = 45.
- Id_range = 10, the Start_value = 46, and the End_value = 50.
- Id_range = 11, the Start_value = 51, and the End_value = 55.
- Id_range = 12, the Start_value = 56, and the End_value = 60.
- Id_range = 13, the Start_value = 61, and the End_value = 65.
- Id_range = 14, the Start_value = 66, and the End_value = 70.
- Id_range = 15, the Start_value = 71, and the End_value = 75.
- Id_range = 16, the Start_value = 76, and the End_value = 80.
- Id_range = 17, the Start_value = 81, and the End_value = 85.
- Id_range = 18, the Start_value = 86, and the End_value = 90.
- Id_range =19, the Start_value = 91, and the End_value = 95.
- Id_range = 20, the Start_value = 96, and the End_value = 100.

Table 4.1: The Index Table for experiment no.1, ECD=100, RV=5

Id_range	Start_value	End_value
1	1	5
2	6	10
3	11	15
4	16	20
5	21	25
6	26	30
7	31	35
8	36	40

9	41	45
10	46	50
11	51	55
12	56	60
13	61	65
14	66	70
15	71	75
16	76	80
17	81	85
18	86	90
19	91	95
20	96	100

- **Experiment No.2:** The ECD=100, the RV=10. Table 4.2 explains the index table that was used for this experiment. The index table had ten Ranges and ten Id_ranges from 1 to 10 corresponding to those Ranges. The Id_ranges and their corresponding Start_values and End_values are explained below:
    - Id_range = 1, the Start_value = 1, and the End_value = 10.
    - Id_range = 2, the Start_value = 11, and the End_value = 20.
    - Id_range = 3, the Start_value = 21, and the End_value = 30
    - Id_range = 4, the Start_value = 31, and the End_value = 40.
    - Id_range = 5, the Start_value = 41, and the End_value = 50.
    - Id_range = 6, the Start_value = 51, and the End_value = 60.

- Id_range = 7, the Start_value = 61, and the End_value = 70.
- Id_range = 8, the Start_value = 71, and the End_value = 80.
- Id_range = 9, the Start_value = 81, and the End_value = 90.
- Id_range = 10, the Start_value = 91, and the End_value = 100.

Table 4.2: The Index Table for experiment no.2, ECD=100, RV=10

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

- **Experiment No.3:** The ECD=100, the RV=20. Table 4.3 shows the index table that was created for this experiment. The index table had five Ranges and five Id_ranges from 1 to 5 were specified to Ranges. The Start_values and the End_values for the five Id_ranges are:

  - Id_range = 1, the Start_value = 1, and the End_value = 20.
  - Id_range = 2, the Start_value = 21, and the End_value = 40.
  - Id_range = 3, the Start_value = 41, and the End_value = 60.
  - Id_range = 4, the Start_value = 61, and the End_value = 80.
  - Id_range = 5, the Start_value = 81, and the End_value = 100.

Table 4.3: The Index Table for experiment no.3, ECD=100, RV=20

Id_range	Start_value	End_value
1	1	20
2	21	40
3	41	60
4	61	80
5	81	100

- **Experiment No.4:** The ECD=100, the RV=30. Table 4.4 illustrates the index table for that experiment. The index table had four Ranges and four Id_ranges from 1 to 4 for those Ranges. The Start_values and the End_values specified to the four Id_ranges are:

  - Id_range = 1, the Start_value = 1, and the End_value = 30.
  - Id_range = 2, the Start_value = 31, and the End_value = 60.
  - Id_range = 3, the Start_value = 61, and the End_value = 90.
  - Id_range = 4, the Start_value = 91, and the End_value = 100.

Table 4.4: The Index Table for experiment no.4, ECD=100, RV=30

Id_range	Start_value	End_value
1	1	30
2	31	60
3	61	90
4	91	100

- **Experiment No.5:** The ECD=100, the RV=40. Table 4.5 shows the index table that was used for this experiment. The index table had three Ranges and there were three Id_ranges from 1 to 3 associated with them. The Start_values and the End_values related to the three Id_ranges are:
  - Id_range = 1, the Start_value = 1, and the End_value = 40.
  - Id_range = 2, the Start_value = 41, and the End_value = 80.
  - Id_range = 3, the Start_value = 81, and the End_value = 100.

Table 4.5: The Index Table for experiment no.5, ECD=100, RV=40

Id_range	Start_value	End_value
1	1	40
2	41	80
3	81	100

- **Experiment No.6:** The ECD=100, the RV=50. Table 4.6 explains the index table that was used for this experiment. The index table had two Ranges and we specified 1 and 2 as the Id_ranges for those two Ranges. The Start_values and the End_values that specified the two Id_ranges are:
    - Id_range = 1, the Start_value = 1, and the End_value = 50.
    - Id_range = 2, the Start_value = 51, and the End_value = 100.

Table 4.6: The Index Table for experiment no.6, ECD=100, RV=50

Id_range	Start_value	End_value
1	1	50
2	51	100

- **Experiment No.7:** The ECD=100, the RV=60. The index table that was created for this experiment is shown in Table 4.7. The index table had two Ranges and we determined 1 and 2 as the Id_ranges for those Ranges. The Start_values and the End_values related to those Id_ranges are:
    - Id_range = 1, the Start_value = 1, and the End_value = 60.
    - Id_range = 2, the Start_value = 61, and the End_value = 100.

Table 4.7: The Index Table for experiment no.7, ECD=100, RV=60

Id_range	Start_value	End_value
1	1	60
2	61	100

- **Experiment No.8:** The ECD=100, the RV=70. The index table for this experiment is shown in Table 4.8. The index table had two Ranges and we specified 1 and 2 as the Id_ranges for the Ranges. The Start_values and the End_values for the two Id_ranges are:
  - Id_range = 1, the Start_value = 1, and the End_value = 70.
  - Id_range = 2, the Start_value = 71, and the End_value = 100.

Table 4.8: The Index Table for experiment no.8, ECD=100, RV=70

Id_range	Start_value	End_value
1	1	70
2	71	100

- **Experiment No.9:** The ECD=100, the RV=80. The index table for this experiment is explained in Table 4.9. The index table had two Ranges and we determined 1 and 2 as the Id_ranges for the Ranges. The Start_values and the End_values specified for the two Id_ranges are:
  - Id_range = 1, the Start_value = 1, and the End_value = 80.
  - Id_range = 2, the Start_value = 81, and the End_value = 100.

Table 4.9: The Index Table for experiment no.9, ECD=100, RV=80

Id_range	Start_value	End_value
1	1	80
2	81	100

- **Experiment No.10:** The ECD=100, the RV=90. The index table for this experiment is shown in Table 4.10. The index table had two Ranges and we specified 1 and 2 as the Id_ranges for the Ranges. The Start_values and the End_values for the two Id_ranges are:
    - Id_range = 1, the Start_value = 1, and the End_value = 90.
    - Id_range = 2, the Start_value = 91, and the End_value = 100.

Table 4.10: The Index Table for experiment no.10, ECD=100, RV=90

Id_range	Start_value	End_value
1	1	90
2	91	100

- **Experiment No.11:** The ECD=100, the RV=100. The index table for this experiment is illustrated in Table 4.11. In this experiment, the index table had one Range and we specified 1 as an Id_range for this Range. The determined Start_value and End_value for this Id_range are 1 and 100 respectively.

Table 4.11: The Index Table for experiment no.11, ECD=100, RV=100

Id_range	Start_value	End_value
1	1	100

4.2.2 The Experiments of the Search Scenario:

In the second scenario of the study experiments, the enhanced ROPE model is examined in the case of retrieving data from the database server. It performs eleven experiments, corresponded to those of the insert operation, to test the search operation in the enhanced ROPE model. These experiments aimed to compute the TbS and the RRbCS parameters while searching for similar values with a fixed sequence. Hence, random twenty numbers from 1 to 100 are selected to search for them in all of the search experiments. The twenty numbers are: 95, 93, 40, 64, 26, 66, 19, 46, 3, 47, 91, 38, 12, 16, 8, 53, 56, 52, 99, and 24 sequentially. The search experiments are performed on the databases that resulted from the insert experiments. That is, the resulting ordered database from each insert experiment in a specific RV is used to search for the target values in the same RV. Practically, the resulted database from experiment no.1 of the insert scenario is used to conduct experiment no.1 of the search scenario. And the resulted database from experiment no.2 of the insert scenario is used to conduct experiment no.2 of the search scenario. Table 4.12 shows the details of the eleven experiments of the search operation. It explains the database, the index table, and the RV that are used for every experiment.

Table 4.12: The search operation experiments

The search operation experiment no.	The used database	The Index Table	The RV
Experiment no.1	From experiment no.1 of the insert operation	Table 4.1	5
Experiment no.2	From experiment no.2 of the insert operation	Table 4.2	10
Experiment no.3	From experiment no.3 of the insert operation	Table 4.3	20
Experiment no.4	From experiment no.4 of the insert operation	Table 4.4	30
Experiment no.5	From experiment no.5 of the insert operation	Table 4.5	40
Experiment no.6	From experiment no.6 of the insert operation	Table 4.6	50
Experiment no.7	From experiment no.7 of the insert operation	Table 4.7	60
Experiment no.8	From experiment no.8 of the insert operation	Table 4.8	70
Experiment no.9	From experiment no.9 of the insert operation	Table 4.9	80
Experiment no.10	From experiment no.10 of the insert operation	Table 4.10	90
Experiment no.11	From experiment no.11 of the insert operation	Table 4.11	100

4.3 The mOPE Model Implementation:

On the other hand, the study simulates the mOPE model in two subprograms to conduct the experiments for the two scenarios as well as the enhanced ROPE model. The first subprogram, subprogram 1, is to conduct the insert operations experiment. And the other one, subprogram 2, is to conduct the search operations experiment. The data used for the experiments is the same as the enhanced ROPE model. The experiments of this model are aimed to compute only the requests and responses between the client and the server during performing the operations. This is because the mOPE model organizes all the encrypted data in one OPE tree, and the server depends on the client to move over the OPE tree.

The approach of mOPE model experiments is explained in Figure 4-2. At the start, the study conducts the insert operations experiment by running subprogram 1. This program inputs the same sample of data that is used in the insert experiments of the enhanced ROPE model. Then it runs subprogram 2 to conduct the search operations experiment. It uses the resulted database from the insert experiment and the target values to search for them.

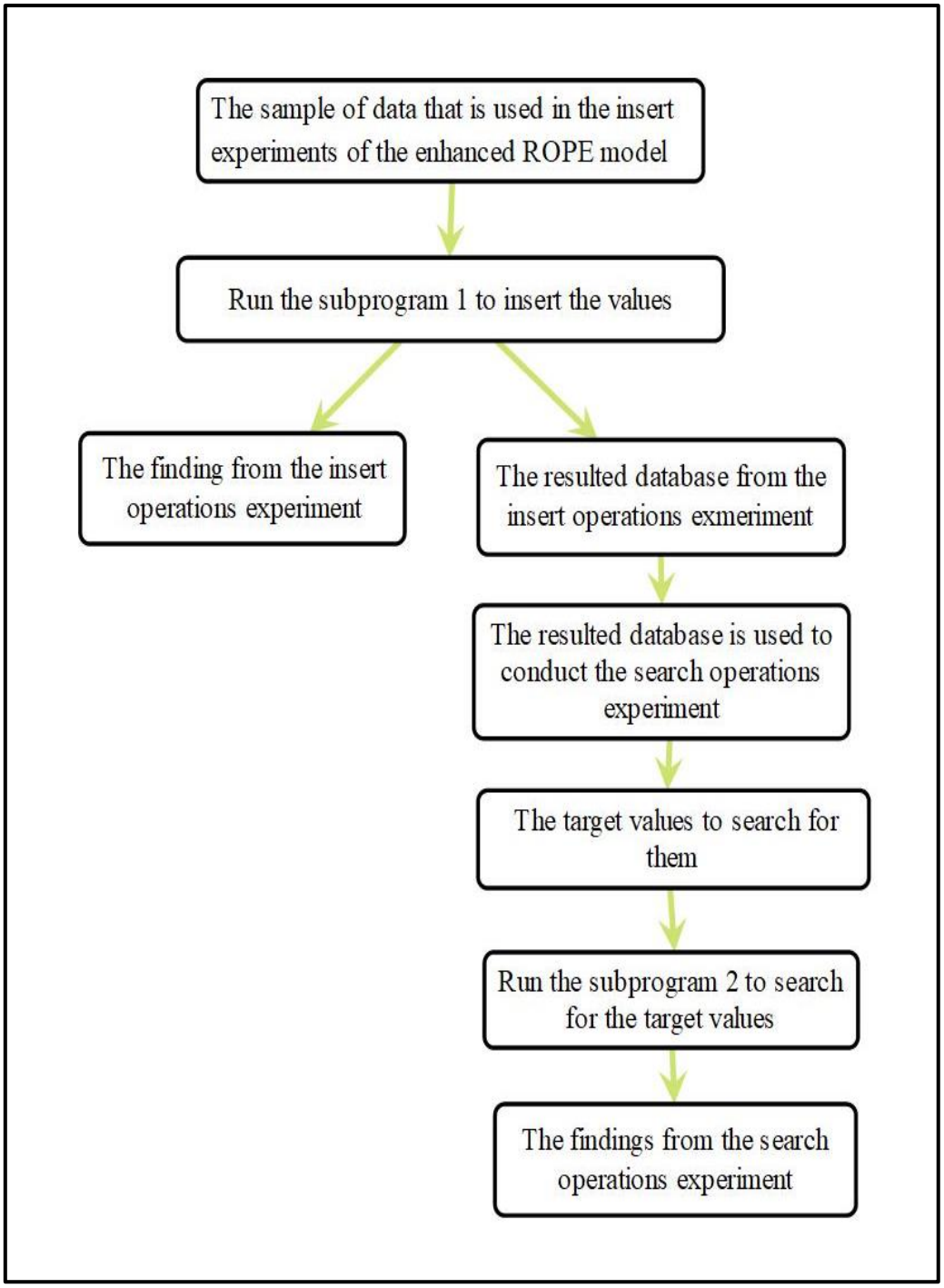

Figure 4-2: The mOPE model experiments approach

4.3.1 The Experiment of the Insert Scenario:

On the other hand, the study conducts the insert experiment in the mOPE model. To be more precise, the experiment used the sample of data that was used in the insert experiments of the enhanced ROPE model. And run subprogram 1 to simulate the model structure.

Note that, this model organizes all the encrypted data in one OPE tree, this is why it conducts one experiment for the insert operations. Also, the server relies on the client to move over the OPE tree. Therefore, the experiment aims to compute the results of the RRbCS after inserting the 100 numbers.

4.3.2 The Experiment of the Search Scenario:

In contrast, the study implements search operations in the mOPE model. It conducts the search experiment by running subprogram 2 to search for the target values. To be more precise, it searches for the same target values in the same sequence as the enhanced ROPE search experiments. It uses the database that resulted from the insert experiment. Also, this experiment aims to find RRbCS after completing the search operations.

4.4. Summary:

This chapter presents the implementation of the ROPE model facing the mOPE model. To validate the research methodology, eleven experiments of the insert operations were performed using eleven different RVs for the ROPE model. Also, the same sample of data in a fixed sequence was used for all experiments. The resulted databases from the insert experiments were used to conduct another eleven experiments of the search operations scenario in the ROPE model. Besides, two experiments were conducted for the mOPE model. The first one for the insert operations scenario, which used the similar sample of data as the insert experiments of the enhanced ROPE model. The resulted database from the insert experiment was used for the search operations experiment. The search experiment was searched for the same target values in the search experiments of the enhanced ROPE model.

CHAPTER V

RESULTS AND DISCUSSIONS

5.1 Introduction:

This chapter presents the results of the enhanced Ranges Order Preserving Encryption (ROPE) model experiments and the mutable Order Preserving Encryption (mOPE) model experiments and discusses the achieved results.

5.2 Results:

Table 5.1 shows the findings after inserting the data in the enhanced ROPE model which are the final results of the Tests by the Server (TbS) and the Requests and Responses between the Client and the Server (RRbCS). The table explains the outcomes from the eleven experiments, each row for a particular experiment with a specific number. For each one of them, the RV column addresses the Range_Value (RV) that used to create the index table, the TbS column describes how many times the server tests the stored Id_ranges to find the right Id_ranges that points to the target OPE trees and the RRbCS column shows the requests and responses between the client and the server to reach the right positions in the target OPE trees.

Table 5.1: The results of the insert operations experiments in the enhanced ROPE model

Experiment no.	RV	TbS	RRbCS
1	5	1050	482
2	10	550	652
3	20	300	816
4	30	240	918
5	40	200	970
6	50	150	1066
7	60	160	1058
8	70	170	1084
9	80	180	1108
10	90	190	1174
11	100	100	1272

Moreover, Table 5.2 shows the final results of the eleven experiments of the search operations in the enhanced ROPE model. For the eleven experiments, the table illustrates the RV that used to create the index table before, the TbS to find the right Id_ranges that points to the target OPE trees, and the RRbCS to reach the right positions in the target OPE trees respectively.

Table 5.2: The results of the search operations experiments in the enhanced ROPE model

Experiment no.	RV	TbS	RRbCS
1	5	198	88
2	10	124	106
3	20	56	132
4	30	51	146
5	40	42	180
6	50	31	180
7	60	34	180
8	70	36	216
9	80	36	214
10	90	36	212
11	100	20	224

Furthermore, Table 5.3 shows the final result of the requests and responses between the client and the server after inserting the data in the mOPE model. As we mentioned before, the mOPE model performs all tasks over the network. Because of this, The TbS column records nothing.

Table 5.3: The result of the insert experiment in the mOPE model

RRbCS	TbS
1272	–

However, Table 5.4 shows the outcomes after searching the target values in the mOPE model.

Table 5.4: The result of the search operations in the mOPE model

RRbCS	TbS
224	–

5.3. Discussions:

This section discusses the achieved results and makes some comparisons. It focuses on the results of the TbS and the RRbCS in the enhanced ROPE model, and the results of the RRbCS in the mOPE model. Moreover, the RRbCS of the two models is employed to make comparisons between the two models, while monitoring the TbS behavior of the enhanced model.

Back to the results of the insert operations experiments in the enhanced ROPE model (Table 5.1 – page-63). It is observed that the enhanced ROPE scheme behaves better for the small RVs than the greater ones. The increase of the RV produces a significant increase in the RRbCS besides decreasing in the TbS. This is not surprising

since the great RV generates a few Ranges in the index table, hence a few OPE trees in the server. Consequently, the size of the OPE trees is growing and hence their height. For the same reasons, those experiments of the RVs are greater than 50 have a lower change in the TbS and the RRbCS than those of the RVs are less than or equal to 50. The RRbCS records the greatest value when the RV is equivalent to the ECD (experiment no 11). In this case, all the data organized in one OPE tree, the TbS records the lowest result, and the RRbCS records the highest result. This case is the worst case of the enhanced ROPE model since it performs most of the tasks across the network.

On the other side, return to the results of the insert operations experiments in the mOPE model (Table 5.3 – page-65). It is observed that the insertion of the data costs the system 1272 RRbCS. Remember that, the insert operations experiment in the mOPE model behaves like the eleventh experiment of the insert operations in the enhanced ROPE model (Table 5.1 – page -63) since both of them are inserting the same data and organized the data in one OPE tree. Moreover, the mOPE model and the last experiment of the insert operations in the enhanced ROPE model are executing 1272 RRbCS to perform the insert operations. Recall from the previous mentioned that, the order of encrypted data in one OPE tree represents the worst case of the ROPE model. This means, the enhanced ROPE model gets the same result as the mOPE model in it is worst-case otherwise it behaves better than the mOPE model.

Moreover, in the results of the search operations experiments in the enhanced ROPE model (Table 5.2 – page-64), it is observed that the greater RV produces few TbS and more RRbCS than the smaller one to retrieve data. The more RV decreases, the higher the TbS gets and the lower RRbCS obtains. This attributes to the fact that the small RV produces more OPE trees and hence lower tree depth in the server than the greater RV. This is appearing clearly in experiments no: 1, 2, 3, and 4 they record a significant decrease in the TbS and significant increases in the RRbCS due to the RV increasing. Whereas in experiments no: 7, 8, 9, 10, and 11 they record a few changes in the TbS and the RRbCS despite the increase of the RV. This is because all of these experiments are order the same data in two OPE trees. Another reason that affects the findings of these experiments is the distribution of the target values in the OPE trees, especially in experiment 5, 6, and 7.

Besides, experiment no 11 registers the lowest value in the TbS and the highest value in the RRbCS.

On the other hand, the study registers the results of the search operations in the mOPE model (Table 5.4 – page-65). It is observed that the search operations for the target numbers cost the system 224 RRbCS, the same result as the eleventh experiment of the search operations in the enhanced ROPE model. Remember that, the search operations experiment in the mOPE model behaves as same as the eleventh experiment of the search operations in the enhanced ROPE model. (Table 5.2 – page-64) since both of them are searching for the same values in the same database structure (one OPE tree).

Generally, the enhanced ROPE approach eliminates the server's relies on the client and allows the server to perform part of tasks. The use of the small RVs reduces the RRbCS rather produces higher overload on the server (TbS), of course, the greater RV results in the opposite.

5.4. Summary:

This chapter presents the results of the conducted experiments. It shows the results for the eleven experiments of the insert operations and the eleven experiments of the search operations in the enhanced ROPE model. Also, it illustrates the results of the insert operation experiment and the search operations in the mOPE model. Moreover, it compares the obtained results from the two models and discusses the achieved results.

CHAPTER VI

THE ENHANCED ROPE MODEL EVALUATION

6.1. Introduction:

This chapter evaluates the enhanced Ranges Order Preserving Encryption (ROPE) model facing the mutable Order Preserving Encryption (mOPE) model. The evaluation is presenting in terms of two aspects: security, and performance. The security evaluation is proved from the mOPE model security. The performance evaluation is judged by comparing the achieved results of the Requests and Responses between the Client and The Server (RRbCS), and the Tests by the Sever (TbS) in the two scenarios in both of the models.

6.2 Security Evaluation:

Since the enhanced ROPE model is order-preserving encryption, it reveals nothing in addition to order. The justification of why the enhanced ROPE model is secure can be proved from the mOPE model. In Popa's mOPE, they prove that their OPE model is ideal-security OPE model and it achieves Indistinguishability under Ordered Chosen-Plaintext Attack (IND-OCPA) security where the data is ordered in one OPE tree. Further, the server knows nothing besides the order of data in the whole OPE tree consistently. Indifference to theirs, the enhanced ROPE scheme orders the data in sub-OPE trees based on the length of the Range_Value (RV). The server knows nothing besides the order of the target OPE tree. Therefore, the enhanced ROPE model is ideal-security OPE and achieves IND-OCPA security. More interesting, the sub-OPE trees aren't sequentially presented on the server. This feature adds some aspects of security to the ROPE model.

An important fact of the enhanced ROPE model is that when the RV is equivalent to the Expected Client's Data (ECD) then the data is ordered in one OPE tree. This case has the same structure as the mOPE model. Therefore, it provides the same level of security as the mOPE one.

Moreover, If we decrease the RV value then the depth of the OPE trees becomes smaller and the amount of the ordering information known by the server becomes lower. The security level provided in such a case needs to be justified. Hence, it can be said that the security of the enhanced ROPE model is at least as good as the security of the mOPE model.

6.2. Performance Evaluation:

The performance evaluation of the enhanced ROPE model depends on the making efforts in both of the client and the server. The client efforts and the server efforts that affect the enhanced ROPE model performance are:

- **The client efforts:**
  - Searches the appropriate Id_range in the index table for the searched value.
  - Encrypts the Id_range and asks the server to find it.
  - Performs the encryption operation for the inserted value.
  - Helps the server to move over the OPE tree to reach the right position or to find the searched value, by doing the following:
    - ✓ Performs the decryption operations for the returned results from the server.
    - ✓ Performs the comparison operations between the searched value and the returned results from the server.
- **The server efforts:**
  - Searches the encrypted Id_range, which produced the TbS.
  - Searches the right position in the OPE tree, which produced the RRbCS.

However, to be more precise the search operation in the enhanced ROPE model needs a search for a suitable Id_range in the index table and an encryption operation for the searched Id_range. Recall that, these two operations are performed locally in the trusted party (at the client-side) and only one time before requesting the server. It's suggested that to serve these operations as part of the services. Also, it needs to study the impact of the search for the right Id_range in the index table and its encryption operation on the overall system performance. Hence, the part of the client efforts is added to future works. This research focuses on the RRbCS as mentioned earlier.

Moreover, this research evaluates the performance of the ROPE model employing the RRbCS and the TbS (the server efforts). To do that, the research aims to answer the following questions:

- What is the impact of the RV length on the outcomes of the RRbCS and the TbS?
- Does the enhanced ROPE model reduce the RRbCS more than Popa's mOPE model?

To answer the above questions, the study observes the behavior of the RRbCS and TbS of the enhanced ROPE model in different RVs compared to the mOPE model. It considers the results of the insert operations experiments and the search operations experiments that were performed to implement and test the enhanced ROPE model compared to the mOPE model (chapter V). It is been noticed that from the achieved results (Table 5.1 page-63, Table 5.2 page-64, Table 5.3 page-65, Table 5.4 page-65):

- The more RV decreases in the enhanced ROPE model, the more RRbCS reduces and more TbS produces.
- Hence, the smallest RVs have a lower RRbCS than the mOPE model, further; they have higher TbS.

Generally, we can summarize the following:

- The small RV enhances the RRbCS over the network while produces more overload on the server.
- The great RV eliminates the overload on the server while increasing the RRbCS over the network.

However, to reduce the RRbCS of the mOPE model the enhanced ROPE model adds some overload in the server (TbS). Recall that, one of the research objectives is to reduce the server relies on the client in the mOPE model by making the server doing parts of the search operations and without knowing any things besides the order. Moreover, this research concerns about the RRbCS factor because it performs the tasks globally through the network. As we know, transmitting over the network may face more problems than performing tasks locally in the server (TbS). Moreover, the overload on the server can be solved by upgrading the equipment at the server-side to provide high processing. Another

solution is to take advantage of the provided services to serve the TbS as a part of the service. To be more precise the study needs to unify the measurement of the performance in the enhanced ROPE model (the client efforts and the server efforts). This point will be added to future works.

From all the above mentioned, it can be said that the enhanced ROPE model succeeds in reducing the RRbCS of the mOPE model. Moreover, the enhanced ROPE model behaves better than the mOPE model, especially with the small RV. The experiments on the sample data (where the Expected Client's Data =100) shows the best outcomes when the RV is less than halve of the ECD, especially when the RV equals 20 and 30. Finally, it is suggested to use the small RV concerning the overhead produced at the server. By means, if the system provides the highest processors it can be allowed to implement the smallest RV. Consequentially, it enhances system performance.

6.3. The Enhanced ROPE Model Features:

Moreover, the enhanced ROPE model has some key features that distinguish it from the mOPE model. The following points describe these features:

1. The data is organized in sub-OPE trees in the server.
2. The search operation is performed in the target sub-OPE tree.
3. The update of the database storage is accomplished only on the related sub-OPE tree.
4. The server knows only the order of the target sub-OPE tree.

6.4. Comparison between the Enhanced ROPE Model and the mOPE Model:

Table 6.1 makes a comparison between the mOPE model and the enhanced ROPE under some criteria.

Table 6.1: Comparison between the mOPE model and the enhanced ROPE model

Criteria	The mOPE Model	The Enhanced ROPE Model
The data organization in the server	In one OPE tree	In sub-OPE trees
Using an indexing mechanism	No	Yes
The server relies on the client to process the search operations	Totally	Partially
The update of the database storage	In the whole OPE tree	In the target sub-OPE tree
The client efforts in the search process	✓ Performs encryption operations. ✓ Performs the decryption operations. ✓ Makes the comparison operations.	✓ Searches the Id_range in the index table. ✓ Encrypts the Id_range. ✓ Performs encryption operations. ✓ Performs the decryption operations. ✓ Makes the comparison operations.
The server efforts in the search process	✓ Searches the right position in the whole OPE tree with the client's help.	✓ Searches the encrypted Id_range. ✓ Searches the right position in the target OPE tree with the client's help.
The server knows	The order of the whole OPE tree	The order of the target sub-OPE tree
Security	IND-OCPA	IND-OCPA

6.5. Summary:

This chapter evaluates the enhanced ROPE model in terms of security and performance. It proves the security of the enhanced ROPE model from the security of the mOPE model. It observes the achieved results to show the impact of the RV on the RRbCS and the TbS of the enhanced ROPE. Moreover, it is seen that it is better to implement the small RV for the enhanced ROPE model to reduce the RRbCS.

CHAPTER VII

CONCLUSIONS AND FUTURE WORKS

7.1. Conclusion:

This study proposes the enhanced Ranges Order Preserving Encryption (ROPE) model improved from the mOPE model, to enhance the performance of the search over encrypted data in cloud computing, by reducing the Requests and Responses between the Client and the Server (RRbCS). The ROPE model employs the index table to speed up the search processes and to facilitate the search processes for the appropriate positions in the server. Based on the index table knowledge and the applied technique for preserving the order of encrypted data, the client's data is organized in sub-OPE trees in the server. Moreover, the OPE trees are not sequenced in the server. Besides, the proposed ROPE model enables the server to perform part of the search processes independently.

From the evaluation, it was found that the proposed enhanced ROPE model is a very efficient OPE scheme. Because it was demonstrated that, it provides some good features in the model design that were not found in the mOPE model. Rather, it reduces the RRbCS more than the mOPE model.

At the beginning of this study, the research problem was outlined in three Research Questions (RQ) which were mapped to lead the direction of the study as well as guide the model towards the research's main goal. Here it shall examine these questions one more time to show how the achieved results help to answer them.

RQ1: How to reduce the RRbCS in the mOPE model?

One problem of the mOPE model is the high RRbCS it performed. This because it comprises the client's data in one OPE tree. This is solved in the proposed enhanced ROPE model by splitting the one OPE tree to be sub-OPE trees regarding the index table. Moreover, the search process is performed in the intended sub-OPE tree. And the server needs to search only over the intended OPE tree. Therefore, instead of returning to the client in all cases, the server just return to the client after it reaches the intended OPE tree.

RQ2: How to reduce the dependence of the server on the client in the mOPE model?

In the mOPE model, the use of one OPE tree makes the server usually needs the client's help to perform the search processes. Indifference to theirs, the enhanced ROPE model performs the search processes specifically in the target OPE tree. Also, to reach the target OPE tree, the enhanced ROPE model enables the server to search for the suitable Id_range that points to the intended OPE tree independently without returning to the client.

RQ3: What is the effect of the RV length on the RRbCS?

The proposed enhanced ROPE model results show that the length of the RV is mainly affected the achieved findings of the RRbCS. The proposed enhanced model evaluation demonstrates that the small RV reduces the RRbCS more than the mOPE model and subsequently affects the performance of the search processes.

7.2 Thesis Contributions:

This thesis proposes the enhanced ROPE model to enhance the performance of search over encrypted data. The proposed model can be adapted to the encrypted databases and could use to compute over encrypted data especially for the sort operations. The main contributions of this thesis are:

1. Propose an enhanced OPE model for search over encrypted data.
2. Reduce the RRbCS of search operations over encrypted data compared to the mOPE model.
3. Reduce the dependence of the server on the client to process the search operations.
4. Permit the server to perform part of the search processes.
5. Provide the IND-OCPA for the enhanced OPE model.

7.3 The Future Works:

The thesis achieved all the objectives of the study by designing the sub-OPE trees combined with the introduced index table. An enhanced OPE model is proposed to improve the mOPE model. However, several research opportunities still exist and further research can be conducted into them.

The future work will be achieved as the following objectives:

- ✓ To study the client efforts in the enhanced ROPE model and its impact on the whole system performance.
- ✓ To unify the performance measurement in the enhanced ROPE model, by unifying the measurement of the client efforts and the server efforts.
- ✓ To study the security of the enhanced ROPE model by considering an accurate measurement.
- ✓ To study how to deal with the extra storage for the client without affecting the system performance.
- ✓ To study the ROPE model in reality by enabling integration with the cloud Database system.

References:

Abbas Acar, H. A. A. S. U. M. C., **2018**. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Computing Surveys,* 51(4), pp. 1-35.

Alexandra Boldyreva, N. C. A. O., **2011**. *Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions.* Springe, Berlin, Heidelberg, In Annual Cryptology Conference, pp. 578-595.

Alexandra Boldyreva, N. C. Y. L. a. A. O., **2009**. *Order-Preserving Symmetric Encryption.* Springer,Berlin, Heidelberg, Annual International Conference on the Theory and Application of Cryptographic Techniques, pp. 224-241.

Alexandra Boldyreva, S. F. A. O., **2008**. *On Notions of Security for Deterministic Encryption, and Efficient Constructions without Random Oracles.* Berlin, International Association for Cryptologic Research, pp. 335-359.

Allison Lewko, T. O. A. S. K. T. B. W., **2010**. *Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption.* Springer, Berlin, Heidelberg, In Annual International Confernce on the Theory and Applicatation of Cryptologic Techniques, pp. 62-91.

Anselme Tueno, F. K., **2020**. *Efficient Secure Computation of Order-Preserving Encryption.* In Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, pp. 193-207.

Ayub Hussain Mondal, M. R. M. S., **2015.** A Brief Overview of Homomorphic Cryptosystem and Their Applications. *International Journal of Computer Applications.*

Bellare, M. a. F. M. a. O. A. a. R. T., **2008**. Deterministic encryption: Definitional equivalences and constructions without random oracles. *International Association for Cryptologic Research*, pp. 360-378.

Dae Hyun Yum, D. S. K. J. S. K. P. J. L. S. J. H., **2011**. *Order-Preserving Encryption for Non-Uniformly Distributed Plaintexts.* In Intl. Workshop on Information Security Applications.

Dan Boneh, A. S. B. W., **2012**. Functional Encryption: A New Vision for Public Key Cryptography. *Communication of the ACM,* 55(11), pp. 56-64.

Dan Boneh, B. W., **2007**. *Conjunctive, Subset, and Range Queries on Encrypted Data.* Springer, Berlin, Heidelberg, In Theory of Cryptography Conference, pp. 535-554.

Dan Boneh, G. D. C. R. O. G. P., **2004**. *Public Key Encryption with Keyword Search.* Springer, Berlin, Heidelberg, In International conference on the theory and applications of cryptographic techniques, pp. 506-522.

Divyakant Agrawal, A. E. A. F. E. A. M., **2009**. *Database Management as a Service: Challenges and Opportunities.* IEEE, pp. 1709-1716.

Do Hoang Giang, N. W. K., **2017**. Multi-dimensional Range Query on Outsourced Database with Strong Privacy Guarantee. *I. J. Computer Network and Information Security,* Issue 10, pp. 13-23.

Dongxi Liu, S. W., **2012**. *Programmable Order-Preserving Secure Index for Encrypted Database Query.* In IEEE Fifth International Conference on Cloud Computing.

Dongxi Liu, S. W., **2013**. Nonlinear order preserving index for encrypted database query in service cloud environments. *Concurrency and Computation: Practice and Experience,* 25(13), pp. 1967-1984.

Florian Kerschbaum, A. S., **2014**. *Optimal Average-Complexity Ideal-Security Order-Preserving Encryption.* In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 275-286.

Frederik Armknecht, C. B. C. C., **2015**. A Guide to Fully Homomorphic Encryption. *IACR Cryptol. ePrint Arch,* p. 1192.

Gamaleldin, A. M., **2013**. *An Introduction to Cloud Computing.* Software Engineering Competence Center.

George Weilum Ang, J. H. W. T. P. W., **2014**. *System and Method of Sort_Order Preserving Tokenization,* U.S. Patent 8,739,265.

Gorelik, E., **2013**. *Cloud Computing Models.* Massachusetts Institute of Technology.

Gultekin Ozsoyoglu, D. A. S. S. S. C., **2004**. *Anti-Tamper Databases: Querying Encrypted Databases.* Springer, Boston, MA, In Data and Applications security XVII, pp. 133-146.

Hakan Hacıg¨um¨us, B. I. C. L. S. M., **2002**. *Executing SQL over Encrypted Data in the Database Service Provider.* In Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pp. 216-227.

Hasan KADHEM, T. A. H. K., **2010**. MV-OPES: Multivalued-Order Preserving Encryption Scheme: A Novel Scheme for Encrypting Integer Value to Many Different Values. *IEICE TRANS. INF. & SYST,* E93–D(9).

Hasan KADHEM, T. A. H. K., **2010**. *Optimization Techniques for Range Queries in the Multivalued-Partial Order Preserving Encryption Scheme.* Springer, Berlin, Heidelberg, In International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management, pp. 338-353.

Hossein Shafagh, A. H. L. B. P. F. S. D., **2017**. *Secure Sharing of Partially Homomorphic Encrypted IoT Data.* Delft, Netherlands, 15th ACM Conference on Embedded Networked Sensor Systems.

J.SRINIVAS, K. S. R. A. Q., **2012**. CLOUD COMPUTING BASICS. *International Journal of Advanced Research in Computer and Communication Engineering,* 1(5).

Jakimoski, K., **2016**. Security Techniques for Data Protection in Cloud Computing. *International Journal of Grid and Distributed Computing,* 9(1), pp. 49-56.

K. Srinivasa Reddy, S. R., **2014**. A New Randomized Order Preserving Encryption Scheme. *International Journal of Computer Applications,* 108(12), p. 0975 – 8887.

Kadhem, H., **2010**. *A SECURE AND EFFICIENT ORDER PRESERVING ENCRYPTION SCHEME FOR RELATIONAL DATABASES.* In International Conference on Knowledge Management and Information Sharing, pp. 25-35.

Khamitkar, S., **2014**. A survey on Fully Homomorphic Encryption. *IOSR Journal of Computer Engineering,* 17(6), p. .

Kim, K. S., **2019**. New Construction of Order-Preserving Encryption Based on Order-Revealing Encryption. *Journal of Information Processing System,* October, 15(5), pp. 1211-1217.

Krunal Suthar, J. M. P., **2017**. *Data Security in Cloud Computing using Encryption and Obfuscation Techniques.* ResearchGate.

Liang Yan, C. R. a. G. Z., **2009**. *Strengthen Cloud Computing Security with Federal Identity Management Using Hierarchical Identity-Based Cryptography.* Springer-Verlag Berlin Heidelberg, In IEEE International Conference on Cloud Computing (pp. 167-177).

Liangliang Xiao, I.-L. Y., **2012**. *A Note for the Ideal Order-Preserving Encryption Object and Generalized Order-Preserving Encryption,* IACR Cryptology ePrint Archive, 2012, p.350.

Liangliang Xiao, I.-L. Y. D. T. H., **2012**. *Extending Order Preserving Encryption for Multi-User Systems,* IACR Cryptology ePrint Archive, 2012, p.192.

Maha TEBAA, S. E. H., **2013**. Secure Cloud Computing through Homomorphic Encryption. *International Journal of Advancements in Computing Technology(IJACT),* 5(16).

MANDEEP KAUR, M. M., **2013**. Using encryption Algorithms to enhance the Data Security in Cloud Computing. *International Journal of Communication and Computer Technologies,* 1(3).

Manpreet Kaur, R. S., **2013**. Implementing Encryption Algorithms to Enhance Data Security of Cloud in Cloud Computing. *International Journal of Computer Applications,* 70(18), p. 0975 – 8887.

Mr. Manish M Poteya, D. C. A. D. M. D. H. S., **2016**. Homomorphic Encryption for Security of Cloud Data. *Procedia Computer Science,* Volume 79, p. 175 – 181.

O'Neill, A., **2010**. Definitional Issues in Functional Encryption. *IACR Cryptology ePrint Archive,* Volume 2010, p. 556.

Peter Mell, T. G., **2011**. *The NIST Definition of Cloud Computing,* National Institute of Standards and Technology Special Publication 800-145.

POPA, R. A., **2014**. *BUILDING PRACTICAL SYSTEMS THAT COMPUTE ON ENCRYPTED DATA.* s.l.:Massachusetts Institute of Technology.

Princiraj, **2019**. *GreeksforGreeks.* [Online] Available at: www.GreeksforGreeks.org

Qi Zhang, L. C. ,. R. B., **2010**. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications.*

R. Velumadhava Rao, K. S., **2015**. *Data Security Challenges and Its Solutions in Cloud Computing.* Elsevier B.V..

Rahman, A. B. a. S. (. M., **2011**. AN OVERVIEW OF THE SECURITY CONCERNS IN ENTERPRISE CLOUD COMPUTING. Volume 3.

Rakesh Agrawal, J. K. R. S. Y. X., **2004**. *Order Preserving Encryption for Numeric Data.* Paris, France, In Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp. 563-574.

Raluca Ada Popa, C. M. S. R. N. Z. H. B., **2011**. *CryptDB: Protecting Confidentiality with Encrypted Query Processing.* In Proceeding of the Twenty-Third ACM Symposium on Operating System Principles, pp. 85-100.

Raluca Ada Popa, F. H. L. N. Z., **2013**. *An Ideal-Security Protocol for Order-Preserving Encoding.* In 2013 IEEE Symposium on Security and Privacy, pp. 463-477.

Ramgovind S, E. M. S. E., **2010**. *The Management of Security in Cloud Computing.* In 2010 Information Security for South Africa (pp. 1-7). IEEE.

S. Subashini, V., **2010**. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications.*

Safiriyu Eludiora, O. A. A. O. A. O. C. O. L. K., **2011**. A User Identity Management Protocol for Cloud Computing Paradigm. *Int. J. Communications, Network and System Sciences,* 4(3), pp. 152-163.

Sandhya Kohli, K. S. D. ,. R. K., **2015**. A Review on Functional Encryption Schemes and their usage in VANET. *International Journal of Advanced Research in Computer and Communication Engineering,* 4(11).

SARAH SHIHAB HAMAD, A. M. S., **2018**. PUBLIC KEY FULLY HOMOMORPHIC ENCRYPTION. *Journal of Theoretical and Applied Information Technology,* 96(7).

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

Seungmin LEE, T.-J. P. D. L. T. N. N. S. K., **2009**. Chaotic order preserving encryption for efficient and secure queries on databases. *IEICE Transactions on Information and Systems,* 92(11), pp. 2207-2217.

Shehri, W. A., **2013**. CLOUD DATABASE DATABASE AS A SERVICE. *International Journal of Database Management Systems,* 5(2).

Somayeh Sobati Moghadam, G. G. J. D., **2016**. *Secure Order-Preserving Indexing Schemes for Outsourced Data.* In 2016 IEEE International Carnahan Conference on Security Technology (ICCST), pp. 1-7.

Sosinsky, B., **2010**. *Cloud Computing Bible.* John Wiley & Sons.

Treesa Maria Vincent, J., **2013**. Data Storage in Cloud Environment Enhance Privacy. *International Journal of Computer Trends and Technology,* 4(3).

V. Spoorthy, M. M. B. S. K., **2014**. A Survey on Data Storage and Security in Cloud Computing. *International Journal of Computer Science and Mobile Computing,* 3(6).

Vladimir Kolesnikov, A. S., **2012**. On The Limits of Privacy Provided by Order- Preserving Encryption. *Bell Labs Technical Journal 17(3).*

Yan-Cheng Chang, M. M., **2005**. *Privacy Preserving Keyword Searches on Remote Encrypted Data.* Springer, Berlin, Heidelberg, In International Conference on Applied Cryptography and Network Security, pp. 442-455.

Appendices

Appendix A

The enhanced ROPE Model Source Code:

Subprogram1: Creates the index table and prepares the database by allocating the Id_ranges.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>

int range_val=100;
int data_size=100;
int num[500];

/*The trusted party Generates ranges and store them randomly in a file.
 This function run only one time..
 Build the file with range id in the first column and zeros in rows.*/
void insert_id(int range_val)
{  FILE *f;  FILE *f2;
   int temp,i,j,r,k;
int count_range=(data_size/range_val);
   if((data_size%range_val)!=0)
   count_range++;
   printf("\n The index table has %d ranges\n",count_range);
   for(temp=0, i=1; temp<count_range; i++,temp++)
 num[temp]=i;
   srand(time(NULL));
   for(i=count_range-1; i>0; i--)
```

```
  {r=rand()%i;
   temp=num[i];
   num[i]=num[r];
  num[r]=temp;
   }
  f = fopen("range100.txt", "w");
  f2 = fopen("r100.txt", "w");
  if((f == NULL)||(f == NULL))
  printf("\n Error opening file");
 else
     {
  for(i=0; i<count_range; i++)
   {
         k=num[i];
  fprintf(f,"%d ",k);
             fprintf(f2,"%d ",k);
         for(j=1; j<=(range_val*2); j++)
         {
             fprintf(f,"%d ",0);
             fprintf(f2,"%d ",0);
            }
        fprintf(f,"\n"); fprintf(f2,"\n");
           }
   }//else
  printf("\n The two files are ready\n");
  fclose(f);  fclose(f2);
}

 int main()
{
  printf("\n Generate ranges and store them in file");
```

```
   insert_id(range_val);
return 0;
 }
```

Subprogram2: The Insertion Program.

```
 #include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>

int range_val=5;
int data_size=100;
int num[500];
int count1;
int count2;
//int val;
FILE *f;  FILE *f2;

struct Node
       {
         int key;
         struct Node *left,*right;
         int height;
       };

//Get maximum of two integers.
int max(int a,int b)
{
return(a>b)? a:b;
}
//get the height of the tree.
int height(struct Node *N)
{
if(N==NULL)
  return 0;
  return N->height;
}
//Allocte a new node with the give null left and right pointer.
struct Node*newNode(int key)
{
struct Node* node= (struct Node*) malloc(sizeof(struct Node));
  node->key=key;
  node->left=NULL;
  node->right=NULL;
```

```
  node->height=1;
  return(node);
}

struct Node *rightRotate(struct Node *y)
{
  struct Node *x=y->left;
  struct Node *t2=x->right;

  //perform rotation
  x->right = y;
  y->left = t2;

  //update heights
  y->height = max(height(y->left), height(y->right))+1;
  x->height = max(height(x->left), height(x->right))+1;

  //Return new root
  return x;
}

 struct Node *leftRotate(struct Node *x)
{
  struct Node *y = x->right;
  struct Node *t2 = y->left;

  //perform rotation
  y->left = x;
  x->right = t2;

  //update heights
  x->height = max(height(x->left), height(x->right))+1;
  y->height = max(height(y->left), height(y->right))+1;

  //Return new root
  return y;
}

//Get balance factor of node N.
int getbalance(struct Node *N)
{
```

```
if (N == NULL)
  return 0;
    else
    return (height(N->left)- height(N->right));
}

//Insert a value to AVL tree in a row with counter.
struct Node* insert(struct Node* node, int key)
{
// 1. Perform the normal BST insertion.
if(node == NULL)
  {
     count2=count2+2;
     return(newNode(key));

    }
 if(key < node->key)
  {
     count2=count2+2;
     node->left = insert(node->left,key);

    }
  else if(key > node->key )
      {
       count2=count2+2;
       node->right = insert(node->right,key);

      }
  else //equal keys are not allowed in BST
      {count2=count2+2;
       return node;

      }

  // 2. Update height of this ancestor node.
  node->height = 1 + max(height(node->left), height(node->right));

  //Get balance factor of this ancestor node.
  int balance = getbalance(node);

  //If this node becomes unbalanced, then there are 4 cases:
```

```
  //Left Left case:
  if((balance > 1) && (key < node->left->key))
  return rightRotate(node);

  //Right Right case:
  if((balance < -1) && (key > node->right->key))
  return leftRotate(node);

  //Left Right case:
  if((balance > 1) && (key > node->left->key))
  {node->left = leftRotate(node->left);
  return rightRotate(node);
  }
  //Right Left case:
  if((balance < -1) && (key < node->right->key))
  {node->right = rightRotate(node->right);
  return leftRotate(node);
  }
  //return the unchanged node pointer.
  return node;
}
//Reconsrtuct & Build AVL tree from row without counter.
struct Node* insert2(struct Node* node, int key)
{
// 1. Perform the normal BST insertion.
if(node == NULL)
  {
     return(newNode(key));
    }
 if(key < node->key)
  {
     node->left = insert2(node->left,key);
    }
  else if(key > node->key )
       {
        node->right = insert2(node->right,key);
       }
  else //equal keys are not allowed in BST
       {
        return node;
```

```
        }

    // 2. Update height of this ancestor node.
    node->height = 1 + max(height(node->left), height(node->right));

    //Get balance factor of this ancestor node.
    int balance = getbalance(node);

    //If this node becomes unbalanced, then there are 4 cases:

    //Left Left case:
    if((balance > 1) && (key < node->left->key))
    return rightRotate(node);

    //Right Right case:
    if((balance < -1) && (key > node->right->key))
    return leftRotate(node);

    //Left Right case:
    if((balance > 1) && (key > node->left->key))
    {node->left = leftRotate(node->left);
    return rightRotate(node);
    }
    //Right Left case:
    if((balance < -1) && (key < node->right->key))
    {node->right = rightRotate(node->right);
    return leftRotate(node);
    }
    //return the unchanged node pointer.
    return node;
}

// Writes the tree values in a file in preorder traversal.
 void store(FILE* fp,struct Node  *root)
 {
    if(root!= NULL)
        {
         fprintf(fp,"%d ",root->key);
         store(fp,root->left);
         store(fp,root->right);
        }
```

```
  }
//function to write a row contents in a correspondind row in an helper file
 void copy(int id)
 {
 int k,k2;
  //int m=0;
  f  = fopen("range5.txt", "r+");
  f2 = fopen("r5.txt", "r");
  if((f == NULL)||(f2 == NULL))
  printf("\n Error opening file");
 else
 {
     fscanf(f2,"%d",&k2);// first id in file
     while(!feof(f2))
     {
       if(k2==id)
       {
          fscanf(f,"%d",&k);
          while(!feof(f))
          { if(k==id)
              {
               fseek(f,1,SEEK_CUR);
               fseek(f2,1,SEEK_CUR);
               fscanf(f2,"%d",&k2);
               while(k2>0)
               {
                  fprintf(f,"%d ",k2);
                  fscanf(f2,"%d",&k2);
                 }
               if(k2==0)
                break;
              }
          else
            fseek(f,(range_val*4),SEEK_CUR);
            fscanf(f,"%d",&k);
          }//while
          if(k2==0)
          break;
       }
       else
       fseek(f2,(range_val*4),SEEK_CUR);
```

```
      fscanf(f2,"%d",&k2);
     }//while
  }//else
  fclose(f); fclose(f2);
 }
 //The server Checks a suitable range_id inside the file to insert a value .
void check_id(int id,int val)
{
struct Node *root=NULL;
  int x,x2,m;
  f = fopen("range5.txt", "r");
  f2 = fopen("r5.txt", "r+");
  if((f == NULL)||(f2 == NULL))
  printf("\n Error opening file");
 else
  {  fscanf(f,"%d",&x);// search for id
  while(!feof(f))
  {
       if(x==id)//find range_id for a value
        {
          count1++;
            fseek(f,1,SEEK_CUR);
            fscanf(f,"%d",&m);
            while(m>0)//restore a tree from a row
               {
                root = insert2(root,m);
                fscanf(f,"%d",&m);
                }
            root= insert(root,val);
            if(m==0)
              break;
        }

        else//k!= id
       fseek(f,(range_val*4),SEEK_CUR);//4 digits for each number
        count1++;
        fscanf(f,"%d",&x);
    }//while
    // copy a row in an assistant file
    fscanf(f2,"%d",&x2);
    while(!feof(f2))
```

```
    {
if(x2==id)
     {
      fseek(f2,1,SEEK_CUR);
      store(f2,root);
      break;
       }
     else
    fseek(f2,(range_val*4),SEEK_CUR);//4 digits for each number
    fscanf(f2,"%d",&x2);
    }//while
  }
  fclose(f); fclose(f2);
}
//Proxy Calcolate suitable range_id for a value to insert it.
int get_range(int val)
 {
int id=(val/range_val);
   if((val%range_val)!=0)
   id++;
   return id;
 }

//Generate random numbers ftom min to max and insert them in model2
void rand_in( )
{
 FILE *f;
 f=fopen("in100.txt","r");
 count1=0; count2=0;
 int k,id;
 while(!feof(f))
    {
      fscanf(f,"%d ",&k);
      id=get_range(k);
      check_id(id,k);
      copy(id);
 }
 fclose(f);
}

int main()
```

```
{

  FILE *f;

  printf( "\n Uses the generated data ");

  rand_in();
  printf("\n Insert is complete\n");
  f=fopen("counter.txt","a+");
  fprintf(f,"\n Range = 5  TbS=%d  RRbCS=%d \n",count1,count2);
  fclose(f);
 return 0;
}
```

Subprogram 3: The Search Program.

```
#include <stdio.h>
 #include <conio.h>
 #include <stdlib.h>
 #include <malloc.h>

int range_val=100;
int count1=0;
int count2=0;
  struct Node
       {
         int key;
         struct Node *left,*right;
         int height;
       };
struct Node *root;

//Get maximum of two integers.
int max(int a,int b)
{
return(a>b)? a:b;
}
//get the height of the tree.
int height(struct Node *N)
{
if(N==NULL)
   return 0;
   return N->height;
}
//Allocte a new node with the give null left and right pointer.
struct Node*newNode(int key)
{
struct Node* node= (struct Node*) malloc(sizeof(struct Node));
   node->key=key;
   node->left=NULL;
   node->right=NULL;
   node->height=1;
   return(node);
}
```

```
struct Node *rightRotate(struct Node *y)
{
  struct Node *x=y->left;
  struct Node *t2=x->right;

  //perform rotation
  x->right = y;
  y->left = t2;

  //update heights
  y->height = max(height(y->left), height(y->right))+1;
  x->height = max(height(x->left), height(x->right))+1;

  //Return new root
  return x;
}

 struct Node *leftRotate(struct Node *x)
{
  struct Node *y = x->right;
  struct Node *t2 = y->left;

  //perform rotation
  y->left = x;
  x->right = t2;

  //update heights
  x->height = max(height(x->left), height(x->right))+1;
  y->height = max(height(y->left), height(y->right))+1;

  //Return new root
  return y;
}

//Get balance factor of node N.
int getbalance(struct Node *N)
{
if (N == NULL)
  return 0;
    else
    return (height(N->left)- height(N->right));
```

```
}

//Insert a value to AVL tree in a row with counter.
struct Node* insert(struct Node* node, int key)
{
// 1. Perform the normal BST insertion.
if(node == NULL)
   {
      return(newNode(key));
     }
  if(key < node->key)
   {
      node->left = insert(node->left,key);
     }
   else if(key > node->key )
        {
         node->right = insert(node->right,key);
        }
   else //equal keys are not allowed in BST
        {
         return node;
        }

   // 2. Update height of this ancestor node.
   node->height = 1 + max(height(node->left), height(node->right));

   //Get balance factor of this ancestor node.
   int balance = getbalance(node);

   //If this node becomes unbalanced, then there are 4 cases:

   //Left Left case:
   if((balance > 1) && (key < node->left->key))
   return rightRotate(node);

   //Right Right case:
   if((balance < -1) && (key > node->right->key))
   return leftRotate(node);

   //Left Right case:
   if((balance > 1) && (key > node->left->key))
```

```
  {node->left = leftRotate(node->left);
  return rightRotate(node);
  }
  //Right Left case:
  if((balance < -1) && (key < node->right->key))
  {node->right = rightRotate(node->right);
  return leftRotate(node);
  }
  //return the unchanged node pointer.
  return node;
}
//Proxy Calcolate suitable range_id for a value to insert it.
int get_range(int val)
 {
int id=(val/range_val);
  if((val%range_val)!=0)
  id++;
  return id;
 }

void search(struct Node* node, int key)
{

if(node !=NULL)
{
 if(key < node->key)
  {
  count2=count2+2;
      search(node->left,key);
     }else
  if(key > node->key )
       {
        count2=count2+2;
        search(node->right,key);
       }else //equal keys are not allowed in BST
       if(key==node->key)
       {
         count2=count2+2;
       // printf("\n Element is found");
       }else
       printf("\n element is not found.");
```

```
}
}
void restore(FILE* f,int id, int val)
{
struct Node *root=NULL;
int m,x;
//count1=0;
fscanf(f,"%d",&x);
 while(!feof(f))
  {
   if(x==id)//find range_id for a value
    { //read all elements in this row val and build tree
          count1++;
           fseek(f,1,SEEK_CUR);
           fscanf(f,"%d",&m);
           while(m>0)//&&(c<=range_val))
           {
              root = insert(root,m);
              fscanf(f,"%d",&m);
           }
  search(root,val);
           if(m==0)
             break;
     }
     else//x!= id
      fseek(f,(range_val*4),SEEK_CUR);//10numbers + 10space
        count1++;
        fscanf(f,"%d",&x);
    }//while
 }

 int  main()
  {
   FILE* f;  FILE* f2;  FILE* f3;
   int val,id;
   f2 =fopen("20rand.txt", "r");
   f3 =fopen("Search2_count.txt", "a+");
   if((f==NULL)||(f2==NULL)||(f3==NULL))
   printf("\n Error to open file");
else
{fscanf(f2,"%d",&val);
```

```
    while(!feof(f2))
   {

   id=get_range(val);
   f =fopen("range100.txt", "r");
      restore(f,id,val);
      fscanf(f2,"%d",&val);
      fclose(f);
}
fprintf(f3,"\nRange :100 count1=%d  count2=%d\n",count1,count2);
printf("\n Search is Complete..\n");
fclose(f2); fclose(f3);
   return 0;
```

Appendix B

The mOPE Model Source Code:

Subprogram1: The Insertion Program.

```
 #include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>

int num[500];
int count=0 ;

struct Node
       {
         int key;
         struct Node *left,*right;
         int height;
       };

struct Node *root;

//Get maximum of two integers.
int max(int a,int b)
{
return(a>b)? a:b;
}
//get the height of the tree.
int height(struct Node *N)
{
if(N==NULL)
   return 0;
   return N->height;
}
//Allocte a new node with the give null left and right pointer.
struct Node*newNode(int key)
{
```

```
struct Node* node= (struct Node*) malloc(sizeof(struct Node));
  node->key=key;
  node->left=NULL;
  node->right=NULL;
  node->height=1;
  return(node);
}

struct Node *rightRotate(struct Node *y)
{
  struct Node *x=y->left;
  struct Node *t2=x->right;

  //perform rotation
  x->right = y;
  y->left = t2;

  //update heights
  y->height = max(height(y->left), height(y->right))+1;
  x->height = max(height(x->left), height(x->right))+1;

  //Return new root
  return x;
}

 struct Node *leftRotate(struct Node *x)
{
  struct Node *y = x->right;
  struct Node *t2 = y->left;

  //perform rotation
  y->left = x;
  x->right = t2;

  //update heights
  x->height = max(height(x->left), height(x->right))+1;
  y->height = max(height(y->left), height(y->right))+1;

  //Return new root
  return y;
}
```

```
//Get balance factor of node N.
int getbalance(struct Node *N)
{
if (N == NULL)
  return 0;
    else
    return (height(N->left)- height(N->right));
}

// To reconstruct a tree from a file without counter..
struct Node* insert2(struct Node* node, int key)
{

// 1. Perform the normal BST insertion.
if(node == NULL)
  {
     return(newNode(key));
    }
 if(key < node->key)
  {
     node->left = insert2(node->left,key);
    }
  else if(key > node->key )
      {
       node->right = insert2(node->right,key);
      }
  else //equal keys are not allowed in BST
      {
       return node;

      }

  // 2. Update height of this ancestor node.
  node->height = 1 + max(height(node->left), height(node->right));

  //Get balance factor of this ancestor node.
  int balance = getbalance(node);

  //If this node becomes unbalanced, then there are 4 cases:
```

```
  //Left Left case:
  if((balance > 1) && (key < node->left->key))
  return rightRotate(node);

  //Right Right case:
  if((balance < -1) && (key > node->right->key))
  return leftRotate(node);

  //Left Right case:
  if((balance > 1) && (key > node->left->key))
  {node->left = leftRotate(node->left);
  return rightRotate(node);
  }
  //Right Left case:
  if((balance < -1) && (key < node->right->key))
  {node->right = rightRotate(node->right);
  return leftRotate(node);
  }
  //return the unchanged node pointer.
  return node;
}
// Build AVL tree.
struct Node* insert(struct Node* node, int key)
{
// 1. Perform the normal BST insertion.
if(node == NULL)
  {
     count=count+2;
     return(newNode(key));

    }
 if(key < node->key)
  {
      count=count+2;
     node->left = insert(node->left,key);

    }
  else if(key > node->key )
      {
        count=count+2;
       node->right = insert(node->right,key);
```

```
    }
  else //equal keys are not allowed in BST
    { count=count+2;
     return node;

    }

   // 2. Update height of this ancestor node.
  node->height = 1 + max(height(node->left), height(node->right));

  //Get balance factor of this ancestor node.
  int balance = getbalance(node);

  //If this node becomes unbalanced, then there are 4 cases:

  //Left Left case:
  if((balance > 1) && (key < node->left->key))
  return rightRotate(node);

  //Right Right case:
  if((balance < -1) && (key > node->right->key))
  return leftRotate(node);

  //Left Right case:
  if((balance > 1) && (key > node->left->key))
  {node->left = leftRotate(node->left);
  return rightRotate(node);
  }
  //Right Left case:
  if((balance < -1) && (key < node->right->key))
  {node->right = rightRotate(node->right);
  return leftRotate(node);
  }
  //return the unchanged node pointer.
  return node;
}
 // Writes the tree values in a file in preorder traversal.
 void store(FILE* fp,struct Node  *root)
 {
  if(root!= NULL)
```

```
        {
         fprintf(fp,"%d ",root->key);
         store(fp,root->left);
         store(fp,root->right);
        }
   }

// Recostruct the tree from file.
void restore(FILE* fp)
 {
  struct Node *root=NULL;
      int val;
     while(!feof(fp))
        {
        fscanf(fp,"%d",&val);
        root = insert2(root,val);
       }
   }


//read fixwd data from a file to insert in mOPE model
void read100()
{
FILE* f; FILE* f1; int k, size;
f=fopen("in100.txt","r");
f1=fopen("data.txt","w");
while(!feof(f))
 {
  fseek(f1,0,SEEK_END);
 size=ftell(f1);
   if(size==0)
   {fscanf(f,"%d ",&k);
    fprintf(f1,"%d ",k);
    fclose(f1);
   }
   else
   {
   f1=fopen("data.txt","r");
        restore(f1);
        fscanf(f,"%d ",&k);
        root=insert(root,k);
```

```
        fclose(f1);
        f1=fopen("data.txt","w");
        store(f1,root);
        fclose(f1);
    }

 }
 fclose(f);
}
int main()
{
  FILE *f;

  read100();
  f=fopen("counter.txt","a+");
  fprintf(f," RRbCS=%d\n",count);
  fclose(f);
  printf("insert is completed");
  return 0;
}
```

Subprogram2: The Search Program.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>

int count=0;
 struct Node
      {
       int key;
       struct Node *left,*right;
       int height;
      };
struct Node *root=NULL;

//Get maximum of two integers.
int max(int a,int b)
{
return(a>b)? a:b;
}
//get the height of the tree.
int height(struct Node *N)
{
if(N==NULL)
  return 0;
  return N->height;
}
//Allocte a new node with the give null left and right pointer.
struct Node*newNode(int key)
{
struct Node* node= (struct Node*) malloc(sizeof(struct Node));
  node->key=key;
  node->left=NULL;
  node->right=NULL;
  node->height=1;
  return(node);
}

struct Node *rightRotate(struct Node *y)
```

```
{
  struct Node *x=y->left;
  struct Node *t2=x->right;

  //perform rotation
  x->right = y;
  y->left = t2;

  //update heights
  y->height = max(height(y->left), height(y->right))+1;
  x->height = max(height(x->left), height(x->right))+1;

  //Return new root
  return x;
}

 struct Node *leftRotate(struct Node *x)
{
  struct Node *y = x->right;
  struct Node *t2 = y->left;

  //perform rotation
  y->left = x;
  x->right = t2;

  //update heights
  x->height = max(height(x->left), height(x->right))+1;
  y->height = max(height(y->left), height(y->right))+1;

  //Return new root
  return y;
}

//Get balance factor of node N.
int getbalance(struct Node *N)
{
if (N == NULL)
  return 0;
    else
    return (height(N->left)- height(N->right));
}
```

```
//Insert a value to AVL tree in a row with counter.
struct Node* insert(struct Node* node, int key)
{
// 1. Perform the normal BST insertion.
if(node == NULL)
   {
      return(newNode(key));
     }
  if(key < node->key)
   {
      node->left = insert(node->left,key);
     }
   else if(key > node->key )
        {
         node->right = insert(node->right,key);
        }
   else //equal keys are not allowed in BST
        {
         return node;
        }

   // 2. Update height of this ancestor node.
   node->height = 1 + max(height(node->left), height(node->right));

   //Get balance factor of this ancestor node.
   int balance = getbalance(node);

   //If this node becomes unbalanced, then there are 4 cases:

   //Left Left case:
   if((balance > 1) && (key < node->left->key))
   return rightRotate(node);

   //Right Right case:
   if((balance < -1) && (key > node->right->key))
   return leftRotate(node);

   //Left Right case:
   if((balance > 1) && (key > node->left->key))
   {node->left = leftRotate(node->left);
```

```
  return rightRotate(node);
  }
  //Right Left case:
  if((balance < -1) && (key < node->right->key))
  {node->right = rightRotate(node->right);
  return leftRotate(node);
  }
  //return the unchanged node pointer.
  return node;
}
 //Search for an element in tree
void search(struct Node* node, int key)
{
if(node !=NULL)
{
  if(key < node->key)
   {
   count=count+2;
       search(node->left,key);
     }else
   if(key > node->key )
       {
        count=count+2;
        search(node->right,key);
       }else //equal keys are not allowed in BST
       if(key==node->key)
       {
         count=count+2;
        //printf("\n Element is found");
       }else
       printf("\n element is not found.");
}

}
void restore(FILE* f)
{
int m;
 while(!feof(f))
   {
    fscanf(f,"%d",&m);
 root = insert(root,m);
```

```
    }

 }

 int main()
  {
    FILE* f;  FILE* f2;  FILE* f3;
    int val;
    f =fopen("data.txt", "r");
    f2 =fopen("20rand.txt", "r");
    f3 =fopen("Search1_count.txt", "a+");
    if((f==NULL)||(f2==NULL)||(f3==NULL))
    printf("\n Error to open file");
else
{

    restore(f);
    //fscanf(f2,"%d",&val);
   while(!feof(f2))
   {
   fscanf(f2,"%d",&val);
 search(root,val);
    }

    fprintf(f3,"\n count=%d",count);
}
printf("\n Search is completed.");
    return 0;
  }
```

List of Publications

1. Nasrin Dalil1 and Ahmed Kayed (2015) “Preserving Data in Cloud Computing”, IJCSI International Journal of Computer Science Issues, Volume 12, Issue 2, March 2015, ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784. www.IJCSI.org

2. Nasrin Dalil1 and Ahmed Kayed (2019) “Enhancing Performance of Search over Encrypted Data in Cloud Computing”, International Journal of Computer Science Trends and Technology (IJCST) – Volume 7 Issue 6, Nov - Dec 2019, ISSN: 2347-8578. www.ijcstjournal.org