Comparative Analysis for the Performance of Order Preserving Encryption Technique

مقارنة تحليلية لكفاءة طرق التشفير المُحافظة على الترتيب

By

Student Name: Hadeel Bahjet Alkazaz
Student Number: 401320076

Supervisor

Prof. Ahmad K. A. Kayed

A thesis submitted to the Department of Computer Science, Faculty of Information Technology, Middle East University in partial fulfillment of the Requirements for Master Degree in Computer Science.

Department of Computer Science

Faculty of Information Technology

Middle East University

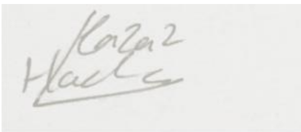
January, 2016

AUTHORIZATION STATEMENT

I, Hadeel Bahjet Alkazaz, authorize the Middle East University to provide hard copies or soft copies of my thesis to libraries, institutions or individuals upon their request.

Name: Hadeel Bahjet Alkazaz

Data: 5/1/2016

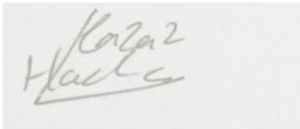
Signature: 

إقرار تفويض

أنا هديل بهجت القزاز، أفوض جامعة الشرق الأوسط للدراسات العليا بتزويد نسخ من رسالتي ورقياً و إلكترونياً للمكتبات أو المنظمات أو الهيئات و المؤسسات المعنية بالأبحاث و الدراسات العليا عند طلبها.

الأسم: هديل بهجت القزاز

التأريخ: 2016/1/5

التوقيع: 

Middle East University

Examination Committee Decision

This is to certify that the thesis entailed "Comparative Analysis for the Performance of Order Preserving Encryption Technique" was successfully defended and approved on, 2016.

Examination Committee Members	**Signature**
Pro. Ahmed A. K. Kayed (Supervisor & Member) Associate professor, Faculty of information technology Middle East University (MEU)	
Dr. Abdelrahman Abuarqoub (Chairman) Associate professor, Faculty of information technology Middle East University (MEU)	
Prof. Jihad Alsady (External member) Department of Computer Science Arab Open University	

ACKNOWLEDGMENT

Thanks to almighty God for hid blesses which enabled me to achieve this work. I would like to thank everyone who encourages me in this thesis. This work would not be accomplished without them.

I would like to thank my supervisor, Prof. Ahmad A. K. Kayed, for the outstanding motivation, guidance, support, and knowledge he provided me throughout my research period.

I am heartily thankful to my family for support and help me until I completed my studies.

I would like also to thank my colleagues for their help and advice in specific topics. My appreciation is extended to my friend Zainab Bayram, for her endless support.

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

"وقل ربي زدني علماً"

Dedication

I would like to exploit this opportunity to dedicate this project to my father, mother, brothers and sisters, without whose invaluable support. I would not have been able to have achieved this in my lifetime.

May God bless them.

Table of Contents

Authorization Statement ……………………………………………………..…………..I
Acknowledgment ……………………………………………………….………....IV
List of Content ………………………………………………………………............VI
List of Tables ……………………………………………………………………...IX
List of Figures ………………………………………………………….…......X
List of Algorithms ……………………………………………….……........…XII
List of Abbreviations…………………..……………………………….…..............XIII
Abstract ………………………..…………………………….………........XIV
الملخص ………………………………………………………………………………XVI
CHAPTER ONE……………………………………………………………………..1
Introduction…………………………………………………………………………...1
1.1 Introduction…………………………………………………..……………...2
1.2 Problem Statement…………………………………………………………...4
1.3 Research Questions…………………………………………………………..4
1.4 Limitations and Scope………………………………………………………….5
1.5 Objective……………………………………………………………………..5
1.6 Contribution…………………………………………………………………….5
1.7 Motivation……………………………………………………………………...6
1.8 Methodology……………………………………………………………………6
1.9 Thesis Outline…………………………………………………………………..7
CHAPTER TWO……………………………………………………………………9
Literature Review……………………………………………………………………..9
2.1Overview…………………………………………………………………………10
2.2 Literature Review………………………………………………………………..10
2.3 Background………………………………………………………………………14
2.3.1 Cloud Computing…………………………………………………………...14
2.3.2 Cloud Deployment……………………………………………………………..16
Public Cloud……………………………………………………………………………16
Private Cloud…………………………………………………………………………...17
Community Cloud……………………………………………………………………...17

Hybrid Cloud………………………………………………………………………...…17
Virtual Private Cloud…………………………………………………………...…17
Personal Cloud………………………………………………………………………18
2.3.3 Cloud Services…………………………………………………………………18
Infrastructure as a Service (IaaS)…………………………………………………19
Platform as a Service (PaaS)……………………………………………………...19
Software as a Service (SaaS)……………………………………………………...20
2.4 Security of Database Cloud……………………………………………………...20
2.4.1 Order Preserving Encryption…………………………………………………..23
2.4.2 Polynomial Functions………………………………………………………….24
2.5 Summary…………………………………………………………………………26
CHAPTER THREE……………………………………………………………..27
OPE Comparative Analysis…………………………………………………………27
3.1 Introduction……………………………………………………………………...28
3.2 System Description………………………………………………………………31
3.2.1 First Procedure…………………………………………………………………32
3.2.2 Second Procedure……………………………………………………………...33
3.2.3 Third Procedure………………………………………………………………..33
3.2.4 Fourth Procedure………………………………………………………………33
3.3 The Main Algorithm……………………………………………………………..34
Range of Coefficient Algorithm…………………………………………………..35
Degree of Polynomial Algorithm……………………………………………………38
Data Type Algorithm……………………………………………………………...41
Key Size Algorithm…………………………………………………………………44
CHAPTER FOUR…………………………………………………………………47
Results and Analysis…………………………………………………………………47
4.1 Overview………………………………………………………………………...48
4.2 Brief……………………………………………………………………………...48
4.3 Implementation Software………………………………………………………..48
4.4 Evaluation Metrics…………………………………………………………….…52
4.5 Experimental Results and Analysis……………………………………………...54
4.5.1 The effect of using different degree of polynomial function……………………54

4.5.2 The effect of using different range of coefficients……………………………….…62
4.5.3 The effect of using different key size…………………………………………….70
4.5.4 The effect of using different data type……………………………………………77
CHAPTER FIVE………………………………………………………………...86
Conclusion and Future Work………………………………………………………...86
5.1 Conclusion……………………………………………………………………….87
5.2 Future Work……………………………………………………………………...88
References…………………………………………………………………………...90
Appendix…………………………………………………………………………….95

List of Tables

Table 4.1: Results of Different Polynomial Function …………………………….…56

Table 4.2: Compare between Security and Performance in Different Degree…………57

Table 4.3: Results of Different Data Sizes in Different Polynomial Function…………59

Table 4.4: Compare between Security and Performance with Different Data Size……60

Table 4.5: Results of Different Ranges of Coefficients………………………………..63

Table 4.6: Compare between Security and Performance in Different Range…………..65

Table 4.7: Results of Different Data Sizes in Different Range of Coefficients………..66

Table 4.8: Compare between Security and Performance with Different Data Sizes…...68

Table 4.9: Results of Different Key Sizes…………………………………………...72

Table 4.10: Compare between Security and Performance in Different Key Size……...73

Table 4.11: Results of Different Data Sizes in Different Key Sizes…………………..74

Table 4.12: Compare between Security and Performance with Different Data Sizes….75

Table 4.13: Results of Different Data Types…………………………………………..79

Table 4.14: Results of Different Data Sizes with Integer Data Type………………….80

Table 4.15: Compare between Security and Performance with Different Data Sizes….81

Table 4.16: Results of Different Data Sizes in String Data Type……………………...83

Table 4.17: Results of Different Data Sizes in Both Data Type……………………….84

List of Figures

Chapter Two

Figure 2.1: Deployment Model……………………………………………..…….16

Figure 2.2: Services of Cloud Computing…………………………………..……….19

Chapter Three

Figure 3.1: The Flowchart of the Methodology………………………….………….30

Figure 3.2: System Description………………………………………………….....32

Figure 3.3: Flowchart of Range of Coefficient Algorithm……………………….…36

Figure 3.4: Flowchart of Degree of Polynomial Algorithm……………………………39

Figure 3.5: Flowchart of Data Type Algorithm…………………………………….....42

Figure 3.6: Flowchart of Key Size Algorithm……………………….………………45

Chapter Four

Figure 4.1: Main Interface of Our Proposed Software…………………………………49

Figure 4.2: The Interface of Using Different Degree of Polynomial Function………...50

Figure 4.3: The Interface of Using Different Range of Coefficients…………………...51

Figure 4.4: The Interface of Using Different Key Sizes………………………………..51

Figure 4.5: The Interface of Using Different Data Types/ Sizes……………………….52

Figure 4.6: Different Polynomial Function…………………………………………….56

Figure 4.7: Different Data Sizes with Different Polynomial Function…………………59

Figure 4.8: Different Ranges of Coefficient……………………………………………64

Figure 4.9: Different Data Sizes in Different Range of Coefficients…………………..67

Figure 4.10: Results of Different Key Sizes……………………………………………72

Figure 4.11: Different Data Sizes in Different key sizes……………………….……75

Figure 4.12: Results of Different Data Types…………………………………………..79

Figure 4.13: Results of Different Data Sizes with Integer Data Type………………….80

Figure 4.14: Results of different Data Sizes in String Data Type……………………...84

Figure 4.15: Results of Different Data Sizes in Both………………………………….85

List of Algorithms

Algorithm 1: Main Algorithm……………………………………………………………35

Algorithm 2: Range of Coefficient Algorithm…………………………………………37

Algorithm 3: Degree of Polynomial Algorithm……………………………………….40

Algorithm 4:   Data Type Algorithm…………………………………………………...43

Algorithm 5: Key Size Algorithm……………………………………………………...46

List of Abbreviations

ASCII	:	American Standard Code for Information Interchange
DaaS	:	Database as a Services
DOPE	:	Digit Order Preserving Encryption
FHE	:	Fully Homomorphic Encryption
GOPE	:	Generalized Order Preserving Encryption
IaaS	:	Infrastructure as a Services
IT	:	Information Technology
MOPE	:	Modular Order Preserving Encryption
OPE	:	Order Preserving Encryption
PaaS	:	Platform as a Services
PHE	:	Partially Homomorphic Encryption
ROPF	:	Random Order Preserving Function
SaaS	:	Software as a Services
VPC	:	Virtual Private Cloud

Comparative Analysis for the Performance of Order Preserving Encryption Technique

Prepared by: Hadeel Bahjet Alkazaz

Supervisor: Prof. Ahmad K. A. Kayed

Abstract

Cloud Computing has been defined as a new business model. It is an emerging paradigm because of that it has been given high attention from many researchers. The main advantage of cloud computing is to reduce the cost of computing while, at the same time, the main disadvantage of using cloud computing is the lack of security. The security of the database in the cloud computing is more critical. One solution for the security problem is encryption, using encryption technique to secure database in the cloud will cause other problem such as not preserve data order.

Order Preserving Encryption (OPE) is an encryption technique that used to preserve the operations of data. OPE scheme solves partially the problem of searching over encrypted data, but it leaks some information.

This thesis was used the order preserving encryption technique to preserve the order of data. It has been investigated encryption techniques that preserve operations of data such as OPE. This thesis studied the OPE function to identify the impact of several parameters on the performance and security level of OPE. It is used the polynomial function with the following parameters: the degree of polynomial function, the range of coefficients, key sizes and data types/ data sizes. Accordingly, the parameters have been studied to find which parameter achieves a high performance with an optimal security level.

This thesis was designed and implemented the software to compare and analyze the performance of OPE function with the previous parameters. It has been imported data from Northwind database, run the software using all the parameters, and analyzed the results for each parameter.

Experiments were conducted to study the effect of several parameters on the performance of OPE. This thesis computed the efficiency as trade- off between performance and security level. The optimal efficiency level would be in the situation of minimum loss in the performance with a high gain of security. This thesis found that when increased the degree from 1 to 4 of the polynomial function, we will gain 75% security level with loss 7% performance. The result showed that degree 4 of the polynomial function is the optimal solution for that situation. This thesis found that increasing the range of coefficients from range 1000 to range 10000 will gaining 90% security levels with losing 3% performance. The result showed that range 10000 of the coefficients is the optimal solution for that situation. This thesis found that increasing the size of the key from size equal 16 bit to size equal 32 bit will gaining 50 % security level with losing 9% performance. The result showed that key size equal 32 bit is the optimal solution for that situation. However, this thesis found that the data type affect the performance of OPE, but it is not significant. The result showed that the data size (6 bytes) achieved a high performance compared with the other sizes.

Keywords: Cloud Computing, Encrypted Data, Order Preserving Encryption Technique, Polynomial function.

مقارنة تحليلية لكفاءة طرق التشفير المُحافظة على الترتيب

إعداد: هديل بهجت القزاز

أشراف: أ.د. أحمد الكايد

المُلخص

تعتبر الحوسبة السحابية أحدى أهم المواضيع التي حصلت على أهتماما كبيراً من قبل العديد من الباحثين. تتميز الحوسبة السحابية بميزة أساسية وهي أنخفاض تكاليف الحوسبة وفي الوقت نفسه أحدى عيوب الحوسبة السحابية هو أنعدام الامن. يعتبر أمن قاعدة البيانات من أهم المحاور حيث يتم معالجة مشكلة أمن قاعدة البيانات من خلال تشفير البيانات. تشفير البيانات سوف يسبب مشكلة اخرى وهو عدم المُحافظة على البيانات ولكن يوجد تقنيات تشفير تحافظ على ترتيب البيانات و أحدى هذه التقنيات هي تقنية التشفير المُحافظة على الترتيب (OPE). هذه التقنية عملت على حل جزء من مشكلة البحث على البيانات المشفرة وذلك بسبب تسرب لبعض المعلومات. أستخدمت هذه التقنية لغرض المحافظة على ترتيب البيانات.

درست هذه الاطروحة عملية البحث على البيانات المشفرة و تقنيات التشفير التى تحافظ على عمليات البيانات و أستخدمت تقنية التشفير المُحافظة على الترتيب لغرض المحافظة على ترتيب البيانات. دراسة تقنية التشفير المُحافظة على الترتيب (OPE) مع أستخدام الدالة كثيرة الحدود (Polynomial function) مع المعاملات التالية : درجة وظيفة الدالة كثيرة الحدود، مجموعة مختلفة من المعاملات، عدة أحجام من المفاتيح، أنواع بيانات مختلفة مع أحجام مختلفة. دراسة تأثير العديد من المعاملات على مستوى أداء و أمن تقنية التشفير المُحافظة على الترتيب. بناءاً على ذلك، فقد تم دراسة المعاملات لايجاد اي منها يحقق مستوى أداء جيد مع مستوى أمن مرتفع في نفس الوقت.

صممت و نفذت هذه الأطروحة برنامج لاجراء مقارنة تحليلية لقياس أداء تقنية التشفير المُحافظة على الترتيب. هذه الأطروحة جمعت البيانات من خلال أستخدام قاعدة بيانات (Northwind) وتم تشغيل البرنامج باستخدام كل المعاملات للحصول على النتائج و امكانية المقارنه فيما بينها.

أجريت التجارب من أجل دراسة تأثير المعاملات على أداء تقنية التشفير المُحافظة على الترتيب. تم حساب الكفاءة من خلال أجراء ميزانية بين مستوى الاداء و مستوى الامان. يعتبر مستوى الكفاءة الامثل عندما تكون الخسارة في مستوى الاداء قليلة مع زيادة في مستوى الأمان.

أظهرت نتائج البحث، عند أستخدام الدرجة الرابعة للدالة كثيرة الحدود بدل من الدرجة الأولى سوف نحصل على مستوى أمان بنسبة 75% مع فقدان في الأداء بنسبة 7 %. وأظهرت النتيجة بأن الدرجة الرابعة للدالة كثيرة الحدود هي الحل الأمثل لهذا الوضع. كما أظهرت النتائج عند أستخدام معدل المعاملات (10000) بدل من معدل المعاملات (1000) سوف نحصل على مستوى أمان 90% مع فقدان في الأداء بنسبة 3 %. تبين بأن معدل المعاملات (10000) هو الحل الأمثل لهذا الوضع. كما أظهرت النتيجة عند أستخدام حجم مفتاح التشفير 32 بت بدل من حجم مفتاح التشفير 16 بت، سوف نحصل على مستوى أمان بنسبة 50% مع فقدان في الأداء بنسبة 9%. أوضحت النتيجة بأن حجم مفتاح التشفير 32 بت هو الحل الأمثل لهذا الوضع. وجدت هذه الأطروحة بإن نوع البيانات يؤثر على أداء تقنية التشفير المُحافظة على الترتيب بنسبة ليست كبيرة.

كلمات البحث: سحابة الحوسبة، بيانات مُشفرة، تقنية التشفير المُحافظة على الترتيب، الدالة كثيرة الحدود.

Chapter One

Introduction

1.1 Introduction

Nowadays, there are many developments in the Information Technology (IT) field. The cloud computing technology is one of these developments. Cloud computing can be considered as a new business model. It is an emerging paradigm for that reason it has been given high attention from many researchers.

Cloud computing appears as an important enabler for the IT industry. Cloud enables the user to use it like "pay – as- you- use". Users moved their data and application to remote outsource storage and can access it at any time. For this reason, it was one of the IT interested terminologies. Cloud computing is considered as a group of services, providing scalable, quality of service guaranteed, and inexpensive computing platform on demand. Users have to manage various software installation, configuration and update their data and applications (Dillon et al., 2010).

Cloud computing is defined as a pool of many concepts from virtualization, distributed application design, grid computing, utility computing, and clustering. It helps companies to access, manipulate, and configure the application over the network. Also, it enables users to use cloud without the need to install software. Therefore; it reduces the cost of computing, application hosting, and content storage (Mell & Grance, 2011).

Security of cloud computing has appeared as the significant issue because the data in the cloud is typically in a shared environment nearby data from other users. Ramgovind et al. stated that security issue used to prevent data loss, running software from an unauthorized user, and sharing the resource. They stated that the data must be encrypted to prevent an unauthorized user from accessing it. Encryption is an effective method that makes data unusable and safe (Ramgovind et al., 2010).

The main advantage of cloud computing is to reduce the cost of computing while; at the same time the disadvantage of the cloud computing is the security leakage. One solution for securing database problem is the encryption; encryption in the cloud will cause other problem. Many encryption techniques do not preserve operations on data. There are two types of encryption technique that preserve operations on data: Homomorphic encryption technique and Order Preserving Encryption technique (OPE) (Agrawal et al., 2004).

Homomorphic encryption techniques preserve the arithmetic operation (+,-,*, /).There are two types of homomorphic: Fully Homomorphic Encryption (FHE) and Partially Homomorphic Encryption (PHE). FHE is used to preserve all arithmetic operations. PHE is used to preserve some arithmetic operations (Mohanty, 2013).

The main important points in the encryption database are searching operation and indexing. This thesis investigated on searching over encrypted data and the encryption techniques that preserve operations on data such as OPE. This thesis studied the OPE function and used the polynomial function with the following parameters: the degree of polynomial function, the range of coefficients, key sizes and data types/ data sizes. The purpose of these parameters is: the degree and the range of coefficients decide the security level of the polynomial function. The key sizes have been used in the encryption technique to study the effect of these key sizes on the performance of OPE. This thesis has been suggested three data types of studying the effect of the data type on the performance of OPE.

It has been studied the impact of several parameters on the performance and security level of OPE. The aim of this thesis is to find the optimal point as a trade-off between security level and performance.

1.2 Problem Statement

Cloud computing is one of the recent technologies and provides many services to users. Search over encrypted data is a challenging problem in the cloud security field. Security is a fundamental issue in the cloud computing paradigm. The traditional encryption techniques are preventing unauthorized access to sensitive data while at the same time do not preserve the order of data. OPE scheme solves the problem of searching over encrypted data partially, but it leaks some information. Enhancing the security of OPE will reduce the performance (execution in time). This thesis studies the effect of using OPEs function (polynomial function) with regards to performance and security level. This thesis investigated the performance and security level of encryption protocols by using several parameters. These parameters are the degree of polynomial function, the range of coefficients, key sizes and data types/sizes.

1.3 Research Questions

Goals of this thesis have been accomplished by answering the following questions:

1. Which OPEs' function (Polynomial degree) can achieve a "high" performance with "an optimal" security level over OPE?
2. Which range of coefficients can achieve a high performance with an optimal security level?
3. What is the effect of using several key sizes on the performance of OPE and security level?

4. What is the effect of using several data types and sizes on the performance of OPE?

1.4 Limitations and Scope

This thesis studies the performance and security level of encryption protocol without enhancing the OPE technique itself. To apply our experiments, we run several polynomial degrees with all the parameters without using an actual database over actual cloud computing. The time of each experiment is computed by using a computer with properties (Intel (R) Core (TM) i3, CPU 2.50GHz, RAM 4GB, 64-bit Operating System, Windows 8). The focus of this thesis will be on the execution time of the parameters not on database re-indexing time.

1.5 Objective

The objective of this thesis is to find the effect of the parameters on the OPE performance. The objective of this thesis is to find the efficiency of the combination of the parameters as a trade-off between security level and performance.

1.6 Contribution

The contribution of this research can be summarized in the following:

- Identified how much the effect of following parameters degree of the polynomial, the range of coefficients, key sizes, data types/ sizes on the performance and security of OPEs.
- Identified the efficiency for many cases for several parameters as a trade-off between security level and performance.
- Identified the optimal efficiency as the minimum increment in the performance with a high gain of security.

1.7 Motivation

During the recent years, cloud computing was obtaining more interest from IT companies and people. Cloud computing shares data and resources on the cloud, therefore; cloud must be protected data from attackers. Securing of cloud database is more critical because of this reason the people reduced using of the cloud. The encryption technique is a suitable way to protect data.

Understanding how one can searching over encrypted data by using order preserving encryption technique (OPE). Searching over encrypted data is important for any database developers and IT students to encrypt the data with preserve the order of the original data. Cloud computing is an important model. Finding the solution is a big motivation because there are several organization interests with the security. It has motivated us to work on securing cloud database with preserve the order of data.

1.8 Methodology

The methodology of this thesis is a combination of descriptive and quantitative research. The methodology mainly based on building several experiments by using OPEs technique to study the performance and security level. Data will be collected and implemented using OPE technique. Performance and security level will be recorded then analyzed to answer the main questions of this thesis.

This thesis adopted polynomial function because the security level of it was approved by (Ozsoyoglu et al., 2003). They stated that the security level for each function will be monitored by counting their coefficients and also stated that the high number of coefficients means high-security level.

This thesis ran many experiments by using the polynomial function with several parameters (degree of the polynomial function, the range of coefficients, key sizes, data types/sizes). The performance (execution in time) will be measured by using tools such as (set begin - time and set end- time). This thesis compared and evaluated the results of each experiment based on the trade-off between performance and security level. The optimal efficiency has been defined as the minimum loss in the performance with a high gain of security. The methodology has been divided into the following phases:-

- **Studying Phase**

  This phase described and studied the OPE technique and the type of each OPE function.

- **Design and Implementation Phase**

  This phase has been designed and implemented OPE technique and used polynomial function with several parameters (degree of the polynomial, range of coefficients, key sizes, data types/ sizes).

- **Evaluation Phase**

  This phase compared and analyzed the results for each experiment to find which one of parameters (degree of the polynomial, range of coefficients, key sizes, data types/ sizes) achieved a good performance with high security level.

1.9 Thesis Outline

The list of this thesis is structured as follow:

- Chapter 2 provides the summary of the literature and reviews the related works. We explain the main concept of cloud computing, types of cloud deployment, and also several service models. We describe different types of encryption techniques and specifically concern on order preserving encryption technique in searching over encrypted data.

- Chapter 3 explains the methodology in details. This chapter includes the comparative analysis that is used to solve the research problem.
- Chapter 4 presents the experimental results. The experiments were carried out based on the performance and security level. This chapter displays the screen of the implementation software and analyzes the result of this thesis.
- Finally, Chapter 5 describes the conclusion of this thesis and suggests future work to improve the performance.

Chapter Two

Literature Review

2.1 Overview

Search over encrypted data in the cloud computing became important and had attention by researchers. It is one of the challenge processes because it is difficult to let the data storage conduct the search and response the query without loss of data confidentiality.

This thesis sheds a light on the previous related about the field of search over encrypted data in the cloud computing that the data is outsourcing in the cloud. This thesis focuses on the encryption technique that preserve the order of data such as OPE and focus on how to use different encryption function with OPE that preserve the order.

This chapter provides a literature review and background on the main concepts covered by this research. It is divided into four sections. Section 2.2 displays the most related studies in the field of security of database cloud, homomorphic encryption techniques, and order preserving encryption techniques. Section 2.3 discusses the main concept of cloud computing, types of cloud deployment, and cloud services that are needed to understand topics embedded in this thesis. Section 2.4 discusses the different types of encryption techniques that are used to protect the data in the cloud then, discuss the Order Preserving Encryption technique and the polynomial functions that are used to preserve the order of data. Section 2.5 presents a summary for the chapter.

2.2 Literature Review

Many works of literature discussed several concepts and problems in the cloud computing as follow.

Jadeja & Modi discussed the idea of cloud computing, and they defined it as an emerging field of computer science. They said cloud computing as a computing environment because one party can be outsourced to another party and need a connection network to access the computing power or resource like a database. They present the main advantage of cloud computing such as users do not have to pay for infrastructure. They also described the architecture of cloud computing such as deployment and services of cloud computing (Jadeja & Modi, 2012).

Alam & Shakil reviewed the concept of cloud computing. The authors presented the cloud computing concept and its characteristics, and then identified the several types of cloud deployment and services models as public, private, hybrid, and community. They also defined several cloud service as Software as a Service, Platform as a service, database as a service and Infrastructure as a Service. They presented different examples of cloud computing platform, security the cloud, reference architectures and data storage in cloud computing (Alam & Shakil, 2015).

Sunitha & Prashanth discussed the data storage models and data security in cloud computing system. They suggested the efficient algorithm for cloud security. This method has offered important security services such as key generation, encryption and decryption are provided in cloud computing system. They also presented the main goal of manage the data and securely stored (Sunitha & Prashanth, 2010).

Liu et al. discussed an efficient technique for search over encrypted keyword in cloud computing. They presented how the user can store and retrieve their personal data in an encrypted form in the cloud, and they presented how the user can send queries in the form of encrypted keywords. The authors explained that the encryption scheme may not work well when a user wants to retrieve only files that contain certain keywords. They presented the difficulties when the service provider could not determine which files contain keywords. They proposed an adequate privacy preserving keyword search schema in cloud computing to allow search over encrypted data without leaking any information (Liu et al., 2009).

Gentry described a fully homomorphic encryption scheme to solve the problem in cryptography that allowed to compute arbitrary function over encrypted data without decrypt it. They used a fully homomorphic scheme to support all arithmetic operations. They also presented what homomorphic encryption scheme and how to use it to improve the efficiency of secure multiparty computation (Gentry, 2009).

Mohanty explained the model of a framework that can be used to protect and manage their data stored in the untrusted server. The author used the homomorphic to achieve confidentiality; homomorphic is considered as one of the encryption techniques used to protect the data from the users who want to store the data in the untrusted server. Mohanty used a homomorphic encryption scheme for protecting the data, and update encrypted files, instead of transmitting entire encrypted versions each time after an update was explored (Mohanty, 2013).

Boldyreva et al. studied order preserving symmetric encryption for allowing efficient range queries on encrypted data. They presented standard security notions for encryption such as indistinguishability against chosen- plaintext attack is unachievable by a practical OPE scheme. They presented a security of pseudorandom function and related primitives asking that OPE scheme look as- random- as- possible subject to order preserving constraints. They design an efficient OPE scheme and prove the security based on pseudorandom of an underlying block cipher (Boldyreva et al., 2009).

Boldyreva et al. studied and revisited security of order-preserving symmetric encryption (OPE). They discussed the problem of characterizing what encryption via random order preserving function (ROPF) leaks the information. They presented ROPF encryption leaks approximate value of any plaintext as well as the approximate distance between any two plaintexts. They studied a different type of order preserving encryption technique such as random order preserving function (ROPF), modular order preserving encryption (MOPE), Generalized Order preserving encrypting (GOPE), and they improved the security of each OPE scheme to be a pseudorandom order preserving function. They displayed the result that help the researchers to estimate the risk and security guarantees of using a secure OPE in their application. They proposed an efficient transformation that can be applied to any OPE scheme (Boldyreva et al., 2011).

Popa et al. discussed order preserving encryption technique that is used to sort the order of ciphertext that matches with the sort of the order of plaintext. Order preserving encryption enabled database and other application to process queries involving order over encrypted data. Authors displayed several types of OPE function and explained mOPE function, the first order-preserving encryption technique that achieves excellent security. The problem of mOPE function, an adversary learns nothing but the order of elements based on the ciphertexts. They proposed a stronger notion of same-time OPE security that allows an adversary to learn only the order of elements present in an encrypted database. They used OPE technique by adding the random noise to increase the ambiguity (Popa et al., 2013).

Özsoyoglu et al. discussed possibilities of encrypting the database and how to allow querying of encrypted database efficiently. The authors believed that the best way to protect data was by encrypt the database and allow querying over the encrypted data.

They focused on finding the best technique to database encryption and finding the best technique on querying over encrypted data. They classified encryption functions into different functions such as open-Form encryption, closed-Form encryption, and decryption function. They reviewed advantages and disadvantages of both types. They encrypted an integer valued attributes. They used polynomial functions as the encryption function and used the coefficients of the polynomial function as the key with a specific range of coefficients (Özsoyoglu et al., 2003).

Ren, et al. discussed privacy preserving ranked multi keyword search leveraging polynomial function in cloud computing. They discussed how can protect data privacy before outsourcing to the cloud that makes the data utilization a challenging task so, it is enabled to search over encrypted data for supporting effective and efficient data retrieval over a large number of documents in the cloud. The authors presented how can search over encrypted data by using encryption function and Hash function to prevent the adversary from learning the index keywords.

The used polynomial function to hide the encryption input keywords and search query can describe as the coefficient of the polynomial function to prevent the adversary from learning input keyword (Ren et al., 2014).

2.3 Background

2.3.1 Cloud Computing

Cloud computing is a model of computing which is dynamically access the applications, resources as a service from anywhere at any time. It provides data storage, infrastructure, and applications over the network (Zhang et al., 2010).

Cloud computing providers both hardware and software necessary to run the applications, it provides data storage as service so; the user can use it without the need to be in the same location as the hardware that stores the data (Hut & Cebula, 2011).

In the recent years, the cloud computing became very popular and interested by many users because it have several advantages as listed below (Marston et al., 2011).

- The main advantage of cloud computing is cost saving because its reduce management cost through its offer online development and deployment through the platform as a service model.
- It eliminates requires of organizations to run their platforms and maintain hardware/ software infrastructure.
- It offers On-demand self-service to permit the user to access cloud services through the online control panel.
- It offers accessibility, configuration, and manipulate the application over the network at any time.
- It doesn't require installing software to access the application.
- It considered an efficiency model because cloud services have standard APIs (application program interface), which provides easy communication with two application or data sources, also easy backup and restore data.
- It has a feature of flexibility because it allows the user to pay per use; it means to allow them to use the resource as much as they need.

2.3.2 Cloud Deployment

Cloud computing is very interested from many organizations and individuals because it does not need any development and maintenance. Cloud is an efficient model because it provides pay per use or nothing pays for unused. Figure 2.1 shows the cloud deployment models that are classified into six different types based on characteristics and purpose of it.

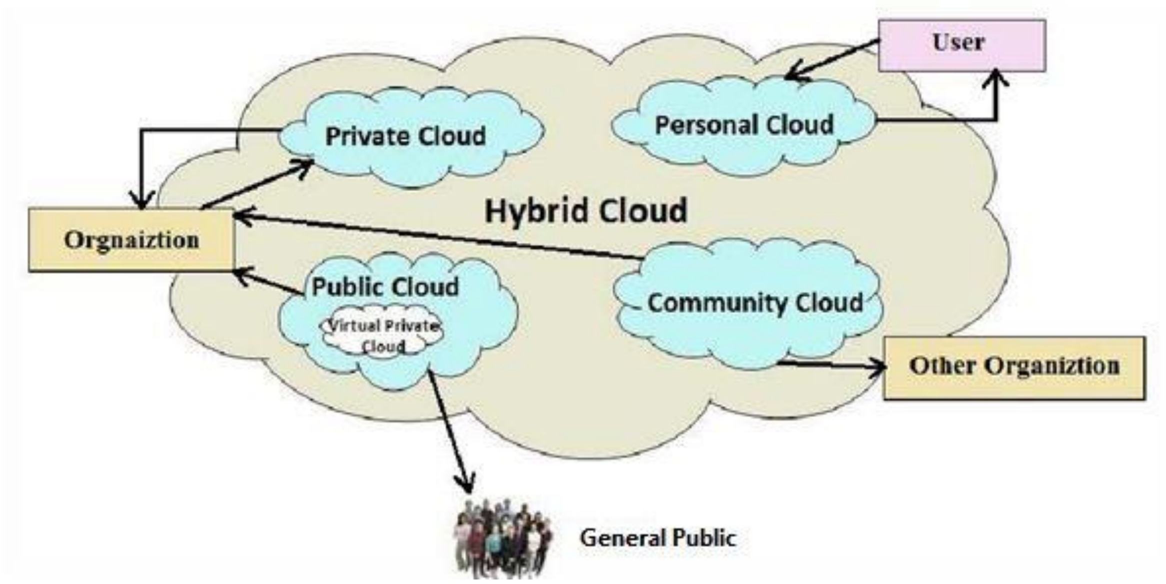


Figure 2.1: Deployment Model Adapted (Al-Khashab, 2014)

Public Cloud

Public cloud can be used and accessed by any consumers or large industry group. In this model, cloud service provider has the full ownership of public cloud with own policy. The characteristics of public cloud is less secure, reduce time and cost, and it doesn't need to maintenance it (Boampong & Wahsheh, 2012).

Private Cloud

A private cloud can be used and accessed by the specific user. The cloud infrastructure is managed by the organization or a third party. In this model, cloud service provider has control over the infrastructure. Storage infrastructure associated is limited to a single company without shared with any others. The characteristics of private cloud are improving security, flexibility, and transparency (Boampong & Wahsheh, 2012).

Community Cloud

Community cloud can be accessed by a group of the organization. In this model, cloud infrastructure is shared by several users that have the same policy and security requirements. It can be managed by one of the organizations in the community or by a third party (Boampong & Wahsheh, 2012).

Hybrid Cloud

In a hybrid cloud, the cloud infrastructure is a combination of two or more cloud deployment types. This model has few limitations such as lack of flexibility, security. The benefits of using this model type are to optimize organization's resources, reduce cost while maintaining privacy and increase security (Boampong & Wahsheh, 2012).

Virtual Private Cloud

The virtual private cloud (VPC) is a combination of the private and the public cloud. VPC represents a perfect balance between control on the private cloud and flexibility on the public cloud. The advantage of using virtual private cloud the connection is secured through VPN; control the security policies on the cloud (Boampong & Wahsheh, 2012).

Personal Cloud

The personal cloud is the realization of different types of cloud deployments in which any type of cloud deployment can become like a personal cloud. It is considered exceptional cloud services because it provides an ideal solution for the secure sharing of compute and storage resources. It classified into three types such as online desktop, online storage, and web base application. Each one of these types is characterized by free up in resources, processing power, and also in the web base application. In the personal cloud, any device with an internet connection can consider as a personal device via a web browser (Na et al., 2010).

2.3.3 Cloud Services

Cloud computing is a rented software that hosted in a shared environment. It is a computing model in which virtualized resources are provided as a service over the Internet. The important feature of cloud computing is cloud service that offers the services through numerous delivery model. Figure 2.2 shows the architecture of cloud computing is classified into three different services models such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

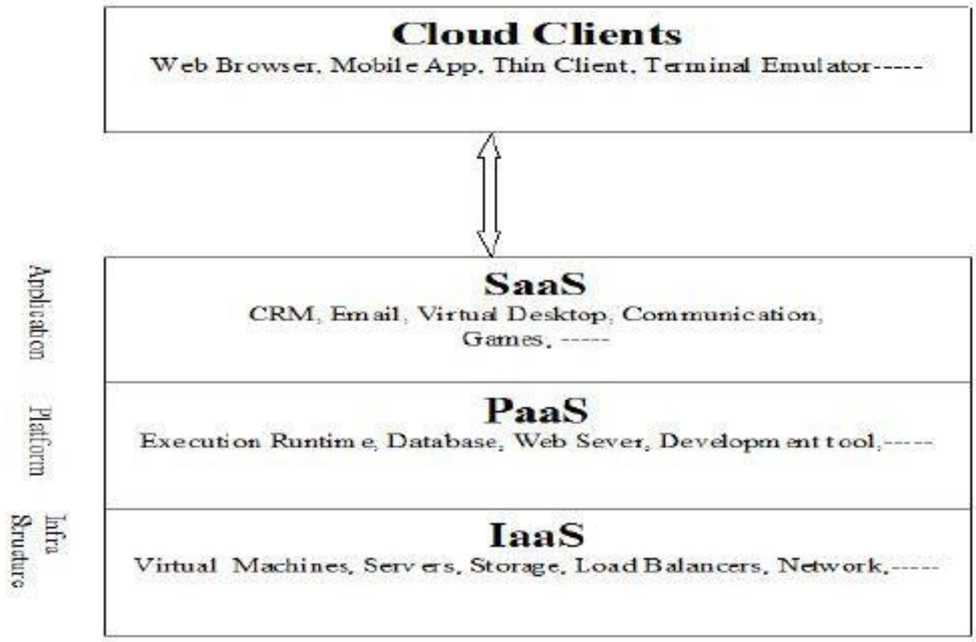


Figure 2.2: Services of Cloud Computing Adapted (Alam & Shakil, 2015)

Infrastructure as a Service (IaaS)

IaaS provides access to a fundamental resource over the network such as virtual machine to enable cloud platforms and application to operate. It allows users to run their operating system through virtualization software that are offered by the service provider. Virtualization in IaaS provides scalable deployment, installation, and maintenance of software. In this model, users have full control over server infrastructure and also it provides only basic security (SO, 2011).

Platform as a Service (PaaS)

PaaS is a set of software and development tools on the provider's servers. PaaS support application hosting environment, possess development infrastructure including programming environment, tools, and configuration. PaaS Provides flexibility and all facilities to support building and delivering web application services. The advantage of

PasS, users can build and deploy their application without installing any tools on their device, users are not obligated to use an expensive hardware or software to develop the applications offered in the cloud (SO, 2011).

Software as a Service (SaaS)

SaaS is a new of software development. In SaaS, the application is hosted as a service so, it allows users to access their application through the network without the need to install and run any software on their device. SaaS provides software over the network, so users can rent software and run the application for pay- per - use instead of owning it. SaaS is a more restrictive model than IaaS, which constrains users to use an existing set of services, rather than deploying it (SO, 2011).

2.4 Security of Database Cloud

Cloud storage is a service of cloud computing that allows users to move their data to the cloud. The cloud database provides Database as a Service (DaaS) as a paradigm of cloud service model. DaaS is a new model of data management, so it managed by a cloud administrator and allows users to create, store, retrieve their data at the host site.

The benefit of using DaaS is service provider allow users for pay per use without the need to purchase an expensive software. With DaaS users can access their data from anywhere on the network, therefore; DaaS is considered as a low cost of data storage. Data privacy is considered a significant issue for any database user. With cloud computing, the data must be protected because it will be shared with unauthorized access. The important characteristic of the cloud is providing proper security that represents data security, privacy, availability, and integrity (Sunitha & Prashanth, 2010).

The important characteristic of the cloud computing poses numerous security issues. Important things to protect data from an unauthorized user based on security requirement are listed below (Youssef & Alageel, 2012).

- Availability

  The availability means cloud's user can use, modify, and access their recourse at any time from any location.

- Integrity

  The integrity means the protection of data from unauthorized users and ensures the data is modified only by authorized access.

- Confidentiality

  Confidentiality means that only authorized user can access the sensitive or critical data.

A Database has become an important component of cloud computing. It can be defined as any form of structured, queryable storage that is hosted in the cloud. Cloud database means different things to the different user (Al-hamami & Al-khashab, 2014).

The data is stored on multiple dynamic servers, rather than on the dedicated servers used in traditional networked data storage. When storing database, the user sees a virtual server, therefore; its provide scalability, high availability, optimized resource allocation and multi-tenancy and (Ru et al., 2014).

Cloud multi-tenancy nature and outsourcing of sensitive data, critical application, and cloud infrastructure causes the security and privacy problems. The data should be encrypted before sending to the cloud to prevent unauthorized access to the original data. Searching for encrypted data has many challenges that need to be solved such as access control, security, performance query optimization and key management. Encryption is considered the biggest concern that is used to encrypt data and to prevent unauthorized access. There are two types of the encryption technique that preserve the operations on data such as Homomorphic encryption and Order Preserving Encryption technique (Yogamangalam & Sriram, 2013).

To protect the cloud database must be encrypted. The encryption in the cloud will cause a problem. Many encryption techniques are not preserving the order of data. There are two encryption techniques are used to preserve the order such as Homomorphic and Order Preserve Encryption technique.

Homomorphic encryption technique is use to preserve the mathematical functionality. The encryption scheme is very useful to protect data from attackers. Encryption is important for constructing Privacy Preserving protocols, because it's a fundamental issue in the cloud computing application.

This issue provides important features for cloud customers to protect their sensitive data. The homomorphic encryption is described a special property of an encryption scheme. That property permits users to perform computation on the ciphertext without decrypting it or knowing the keys (Stehlé &Steinfeld, 2010).

The homomorphic approach divided into two types: Fully Homomorphic Encryption (FHE) and Partially Homomorphic Encryption (PHE). FHE support all arithmetic operations over encrypted data (ciphertext) without decrypt it. Homomorphic is considered as partially if it offers one of arithmetic operation such as additive or multiplicative. There are several examples of PHE such as RSA, ElGamal Cryptosystem, Paillier Cryptosystem (Gentry, 2009).

The main operation of the database will need the indexing, for example, Binary search and preserve the order is important for indexing. This thesis investigated the order preserving encryption technique to preserve the order.

2.4.1 Order Preserving Encryption

The problem of search queries on encrypted data leads to use an Order Preserving Encryption (OPE) scheme that is an important method to solve search problem and support all logical operations. It is a deterministic encryption scheme which means its encryption function preserves numerical ordering of the plaintexts.

OPE is a technique used for encrypting data to preserve the order and make efficient comparison operations on the encrypted data without decrypt the operands. The security problem of OPE is not adequate and also there is some leak of information. OPE is used for processing SQL queries over encrypted data because it can perform order operations on ciphertext in the same way as plaintext. OPE scheme solves the problem of searching for encrypted data partially, but it leaks some information. The three step of OPE constriction are: model the input and target distributions as linear splines, transform the plaintext into uniformly distributed database, and transforming the database into the cipher database but in this case, the database are distributed according to the target distri bution (Agrawal et al.,2004).

OPE is a deterministic symmetric key encryption scheme. It was used to ensure that the Ciphertext preserve the order of the plaintext. There are several types of OPE as follows.

OPE scheme generates a sequence of the random number. It was used to encrypt an integer value by adding the random number to it. This encryption technique was considered as inefficient because it can be only used in a static system when the data has been inserted to the database (Bebek, 2002).

Generalized Order Preserving Encryption (GOPE) scheme adopts a general mathematical object as a ciphertext. The difference between the OPE scheme and GOPE scheme where the ciphertext is a number in the OPE scheme while the ciphertext in the GOPE is a mathematical object. GOPE scheme requires a special comparison algorithm to compare between the ciphertext in the OPE and the GOPE (Boldyreva et al., 2011).

Digit Order Preserving Encryption (DOPE) scheme used to preserve the order of data by using a group of key agents. This scheme enables the distributed encryption to assure that the OPE encryption key is not known by any entity in the system. DOPE has been deployed the key agents between the database and the users. In DOPE scheme, the master key is shared with the key agents where each key agent holds a different encryption key. This scheme separates the key agents with a distinct key by using any OPE scheme. The security problem of DOPE is the key agents can see the plain digit therefore it discloses a part of the sensitive data. The adversary can use the key to compromise the same digit for every data in the database if it compromises the database and one of the key agents (Xiao et al., 2012).

2.4.2 Polynomial Function

A polynomial function is a function such as a quadratic, a cubic, a quartic, and so on, involving only non-negative integer powers. Each polynomial function has a degree; the degree of this function is the highest value for *n* where C_n is not equal to zero. A polynomial of *n* degree is a function of the form: $F(x) = C_nX^n + C_{n-1} X^{n-1}+\ldots+ C_1X+C_0$.

This function is used to hide encryption keyword in search over encrypted data. Search query describes as the coefficient of the polynomial function because the coefficient is used to prevent the adversary from learning data. The number of coefficients is used to measure the security level. The high number of coefficients means the high security leve ls (Ren, et al., 2014).

Polynomial function is used to encrypt the integer value by using the iterative operations. The polynomial function used as the encryption function. The coefficients of the polynomial function used as a key to restrict the users from discovered the key even if discovered the encryption function.Ozsoyoglu et al. investigated the encryption functi on such as the polynomial function applied to each an integer value. They used the coefficients of the polynomial as a key with the range [1-25] (Ozsoyoglu et al., 2003).

Inside the OPE technique, there are several encryption function. These functions are Single Encryption Function, Multiple Encryption Function, Nonlossy Multiple Encryption Function, and Generating and Encryption Function. Single Encryption Function used a single polynomial function as the encryption function. This function considered the degree of security is the number of constants used by the function. Single Encryption function was used to find the inverse of the *n* degree of the polynomial function. Multiple Encryption Function used *n* function of the polynomial and each function has its inverse. This function used to find a sequence of encryption function. The Multiple Encryption Function used the constant of the polynomial as the key and it applies the sequence of function in such a way that the output of the $f_i(x)$ becomes the input of the $f_{i+1}(x)$. This function applies the inverse of each function in the sequence in reverse order (Ozsoyoglu et al., 2003).

2.5 Summary

All related work discussed important points in the cloud computing. They discussed the main idea of the cloud and the advantage of using it. Some of the previous work presented the different type of cloud deployment and cloud services models which is very useful for the researchers.

They discussed the security issues and how to protect data from unauthorized access by using different encryption techniques. Then, they explained an efficient technique that is used for searching over encrypted data and explained how the user can store and retrieve the data without losing it.

This chapter presented different types of encryption technique that is used to preserve operations on data such as Homomorphic and Order Preserving Encryption technique. Based on the related work, this thesis investigated OPEs function and adopted the polynomial function with several parameters to study the effect of these parameters on the performance and security level of OPE.

Chapter Three

OPE Comparative Analysis

3.1 Introduction

Recently, cloud computing has become a topic of great interest but it has some problem. One of that problem is the security which affect the data privacy. The data must be encrypted before storing in outsourcing (cloud), the encryption provides a strong guarantee to protect data so that, the information will be disclosed from unauthorized used.

Due to the many researchers who have studied the field of search over encrypted data. The problem of search over encrypted data, the encryption technique may not work well when a user wants to retrieve files that contain certain keywords. Search over encrypted data became a fundamental issue of great interest in the cloud computing era. Therefore; the critical data must be encrypted before outsourcing to the cloud servers in order to guarantee data confidentiality.

This thesis shed a light on the field of search over encrypted data. The order of data must be preserve when searching over encrypted data. OPE is one of encryption technique that used to preserve the order of data. OPE solves partially the problem of searching over encrypted data, but it leaks some information. It has been adopted the polynomial function with several parameters. These parameters are: degree of the polynomial function, range of coefficients, key sizes, and data types/ sizes.

The purpose of each parameter as follow: this thesis used the degree of polynomial function because it has been decided the security level of the polynomial function. It has been used the range of coefficients depended on the Özsoyoglu et al. approach; they used the coefficients of the polynomial function as a key with the range of coefficients [1-2^5]. Therefore; this thesis has been used five different ranges of coefficients.

This thesis used different key sizes depended on the Popa et al. approach; they added the random noise to increase the ambiguity. Most encryption technique used the maximum key size equal 64 bit, for example Block Cipher Encryption technique used the maximum key size equal 64 and the Advanced Encryption Standard used the key size equal 128 bits but it is divided into two blocks. Each block equal 64 bits, therefore; this thesis has been used the following different key sizes: maximum key size equal 16 bits, minimum key size equal 16 bits, and the key size between maximum and minimum. This thesis has been used three different data types to study the effect of the data type on the performance of OPE.

The steps of the methodology is used to conduct a comparative analysis of the performance of OPE to find the high performance with high security level by using OPEs function. The methodology is a combination of descriptive and quantitative research. It was mainly based on studying and implementing OPE technique (polynomial function) and observing the performance and security level with several parameters. It has used quantitative research for doing many experiments and analyzing the performance of the system. When we start building the experiments, the idea was to run all the combinations to have up to 50 degrees of the polynomial function with several parameters while at the same time the capacity of our computer does not run the huge degree of the polynomial with several parameter. Hence, this thesis used the degree of polynomial function till degree 12. Finally, we found that the number of the experiment was very huge. Because of that, we divided the experiments into four parts. Figure 3.1 shows the proposed model consists of the three phases.

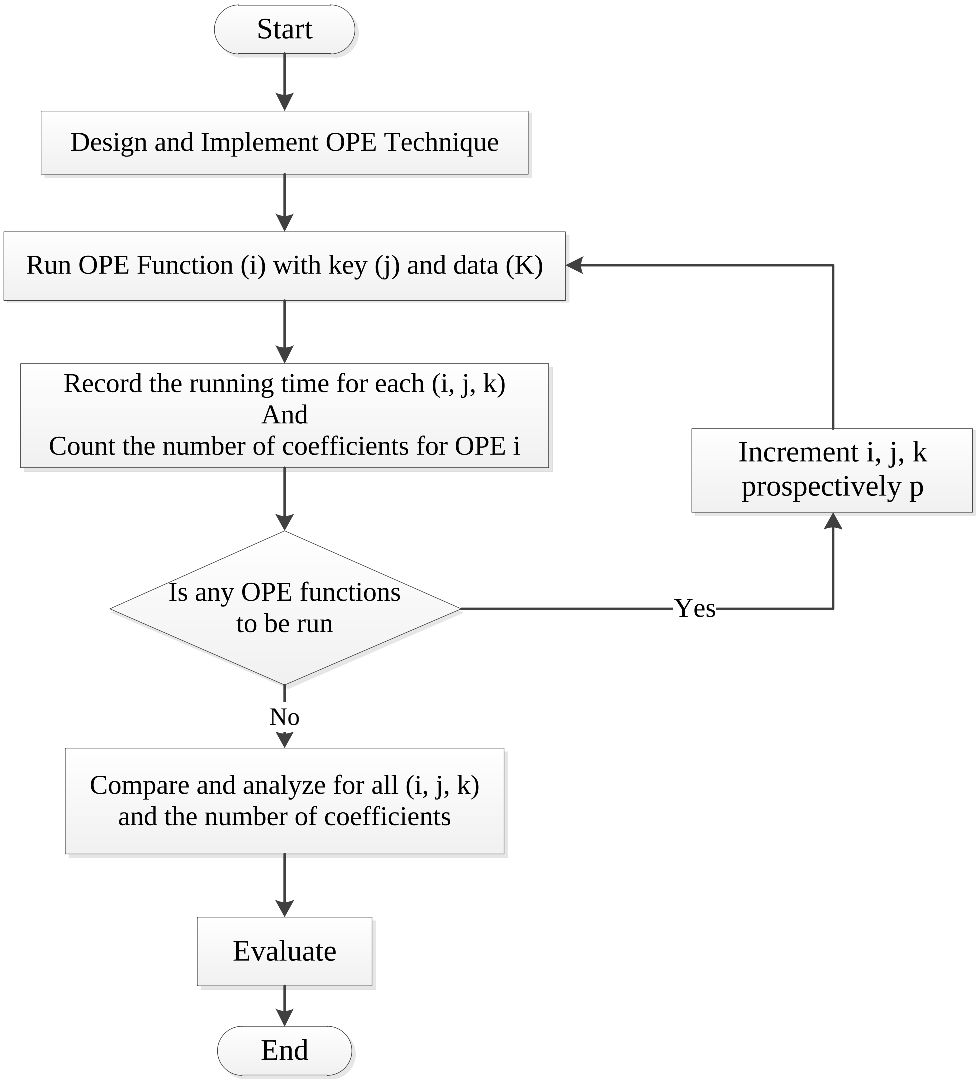


Figure 3.1: The Flowchart of the Methodology

- **Studying Phase**

In this phase, it has been studied the type of OPEs function especially polynomial functions with several parameters.

- **Design and Implementation Phase**

In this phase, it has been collected the data from Northwind database then, studied the OPE technique and adopted polynomial functions. This thesis used the polynomial function because its security level was approved and proposed by (Ozsoyoglu et al., 2003). Ozsoyoglu et al considered the coefficients as a key. They stated that the security level for each polynomial function monitored by counting their coefficients, the less number of coefficients means less security level.

This thesis has been designed and implemented the OPE technique (polynomial function) with several parameters. Then run the OPE technique will all the parameters and recorded the performance (the execution time for each the parameters). It has been computed the execution time of each experiment by using two types of counter. The first counter was used to compute the accumulated time of the experiment. The second counter was used to compute the average time of different data size inside the experiment. This thesis compared and analyzed all the results for all combinations depending on the gain of security and the gain of the performance.

- **Evaluation Phase**

In this phase, it has been evaluated the results according to the performance and security level for OPE function. The performance has been measured by using tools such as set begin-time and set end time, the performance (execution time) measured by millisecond. The security level has been measured by counting the coefficients of the polynomial function. The less number of coefficients means less security level. The efficiency has been computed as trade-off between performance and security level. The optimal efficiency level would be in the situation of minimum loss in the performance with high gain of security.

3.2 System Description

This thesis was described how to study the performance of order preserving encryption technique. Based on the methodology, we divided our work into four procedures to study the effect of using different parameters with regards to performance and security level. Figure 3.2 illustrates the system description in details.

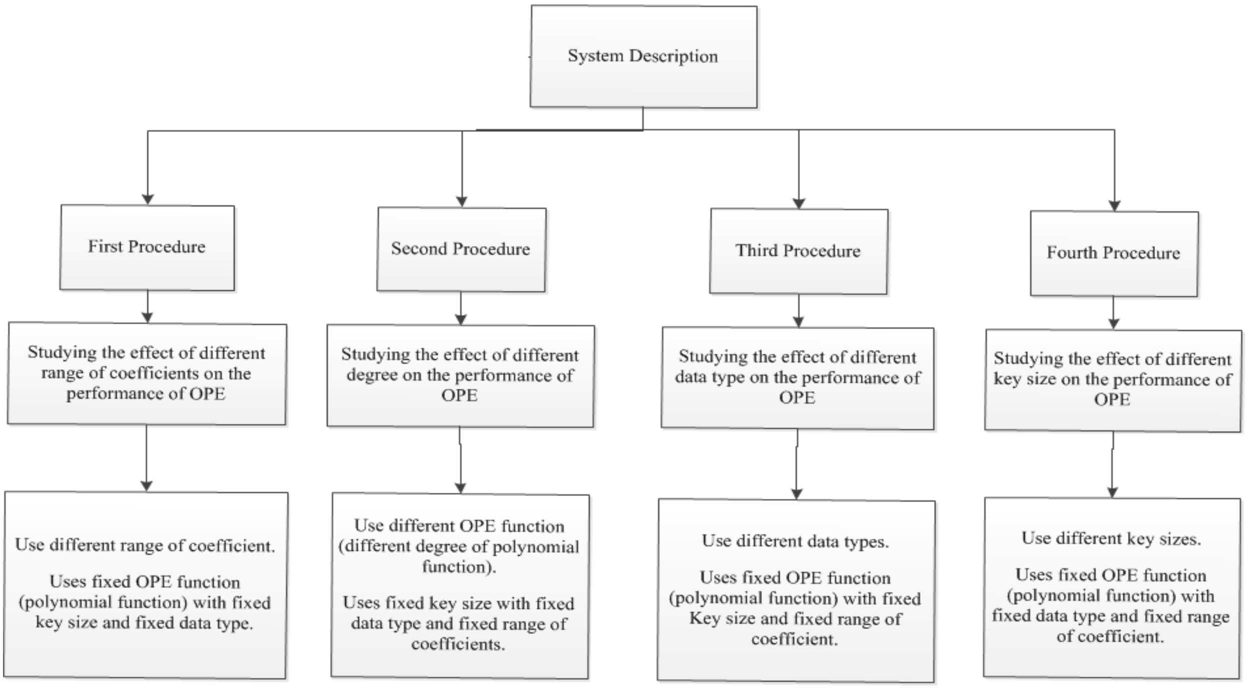


Figure 3.2: System Description

3.2.1 First Procedure

This procedure studies the effect of different range of coefficients on the performance of OPE. It was used five different ranges of coefficients with fixed parameters. These ranges are: range 0-100 with step 10, range 100-1000 with step 100, range 1000-10000 with step 1000, range 10000-100000 with step 10000, range 100000-1000000 with step 100000. The fixed parameters are: degree of polynomial function for example degree 1, keys size for example key size equal 16 bits, and fixed data types for example integer data type.

It studies the result of each parameter to find which range achieves the high performance with the optimal security level.

3.2.2 Second Procedure

This procedure studies the effect of different degree of polynomial function on the performance of OPE. It was used nine different degrees of polynomial function with fixed parameters. These degrees are: degree 1, degree 2, degree 3, degree 4, degree 5, degree 6, degree 8, degree 10, and degree 12. The fixed parameters are: range of coefficient for example range 0-100 with step 10, keys size for example key size equal 32 bits, and fixed data type for example integer data type.

It studies the results of each parameter to find which degree achieves the high performance with an optimal security level.

3.2.3 Third Procedure

This procedure studies the effect of different data types on the performance of OPE. It was used three different data types with fixed parameters. These data types are: integer, string, and both (alphanumeric). The fixed parameters are: degree of polynomial function for example degree 5, range of coefficient for example range 100-1000 with step 100, and key size equal 64 bits.

It studies the result of each parameter to find which data type achieves the high performance with an optimal security level.

3.2.4 Fourth Procedure

This procedure studies the effect of different key sizes on the performance of OPE. It was used three different key sizes with fixed parameters. These key sizes are: 16 bits, 32 bits, and 64 bits. The fixed parameters are: degree of polynomial function for example degree 8, range of coefficient for example range 1000-10000 with step 1000, and fixed data type for example integer data type

It studies the result of each parameter to find which key size achieves the high performance with an optimal security level.

3.3 The Main Algorithm

This algorithm is used to compare the performance of order preserving encryption using four procedures. Each algorithm is called depending on the desired case of comparison as listed below:-

- Case A is called algorithm one. It used for different ranges of coefficient with fixed data type, fixed key size, and fixed degree of polynomial function.
- Case B is called algorithm two. It used for different degrees of polynomial function with fixed data type, fixed key size, and fixed range of coefficient.
- Case C is called algorithm three. It used for different data types with fixed degree of polynomial function, fixed key size, and fixed range of coefficient.
- Case D is called algorithm four. It used for different key sizes with fixed degree of polynomial function, fixed data type, and fixed range of coefficient.

Algorithm 1: Main Algorithm

```
Algorithm: main algorithm
{
Switch(char) {                                    // Char: desired case
Case(A): call range of coefficient algorithm
break;
Case(B): call degree of polynomial algorithm
break;
Case(C): call data type algorithm
break;
Case(D): call key size algorithm
break;
}
```

Range of Coefficient Algorithm

This algorithm is used to compare the performance of OPE by using different ranges of coefficient with fixed data type, fixed key size, and fixed degree of polynomial function to study the effect of these ranges on the performance of OPE. These ranges were:

Range one: [0-100], step 10

Range two: [0-1000], step 100

Range three: [0-10000], step 1000

Range four: [0-100000], step 10000

Range five: [0-1000000], step 100000

This algorithm records the time of the result and compare between each other. It has been determined which range of coefficient can achieve a good performance of OPE. The steps of this algorithm showing as a flowchart in figure 3.3.

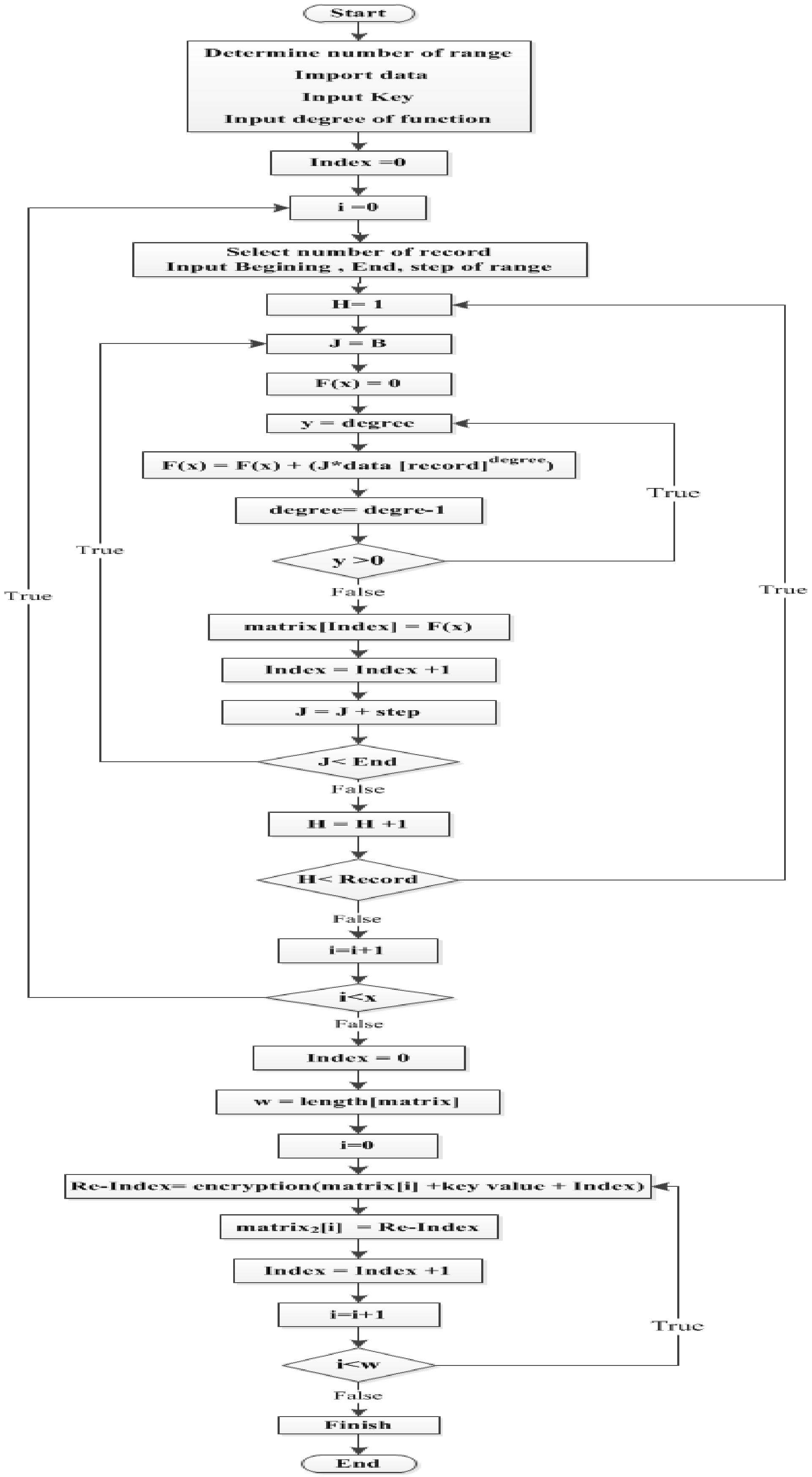


Figure 3.3: Flowchart of Range of Coefficient Algorithm

Algorithm 2: Range of Coefficient Algorithm

```
Algorithm: Range of Coefficient Algorithm
{
X ← no. of range that is uses in this procedure
Data ← Import data from Northwind database with selected data type
Degree ← Input the degree of function
Key size ← Input size of key
Key value ← Rand_key size (Key value)
Index ← 0
For i= 0 to x                    // range [0-100], range [0-1000], range [0-1000000]
Record ← no. of record that is selected from Northwind database (import data)
Beginning ← Input the first value from the range of coefficient
End ← Input the last value of the range of coefficient
Step ←Input the number of increment to the first value until reach to the last value of range
For H =1: record                          // loop of record number
For J= B: Step: End
F(x) = 0
For Y= degree: -1: 0
F(x) ← f(x) + (J *data [record]^degree)
End for                                   // end the loop Y of functions
Matrix [Index] ← f(x)            // this matrix use to save the result of each step of range
Index = Index + 1
End for                                   // end the loop J of range with step
End for                                   // end the loop H of the no. of range
End for                                   // end the loop i of the record
Index ← 0
W ← length (matrix)
For i= 0 to w
Re-Index ← Matrix [i]
Re-Index ← encryption (Matrix [i] + Key value + Index)
Matrix2 [i] ← Re-Index
Index ← Index +1
End for                              // end the loop that is use to save the new index
Return Matrix2
}
```

Degree of Polynomial Algorithm

This algorithm is used to compare the performance of OPE by using different degrees of polynomial function with fixed data type, fixed key size, and fixed range of coefficient to study the effect of these degrees on the performance of OPE. The form of polynomial function represented as follows: $F(x) = C_nX^n + C_{n-1} X^{n-1} + \ldots + C_1X + C_0$.

This algorithm records the time of the result and compare between each other. It has been determined which degree can achieve a good performance of OPE. The steps of this algorithm showing as a flowchart in figure 3.4.

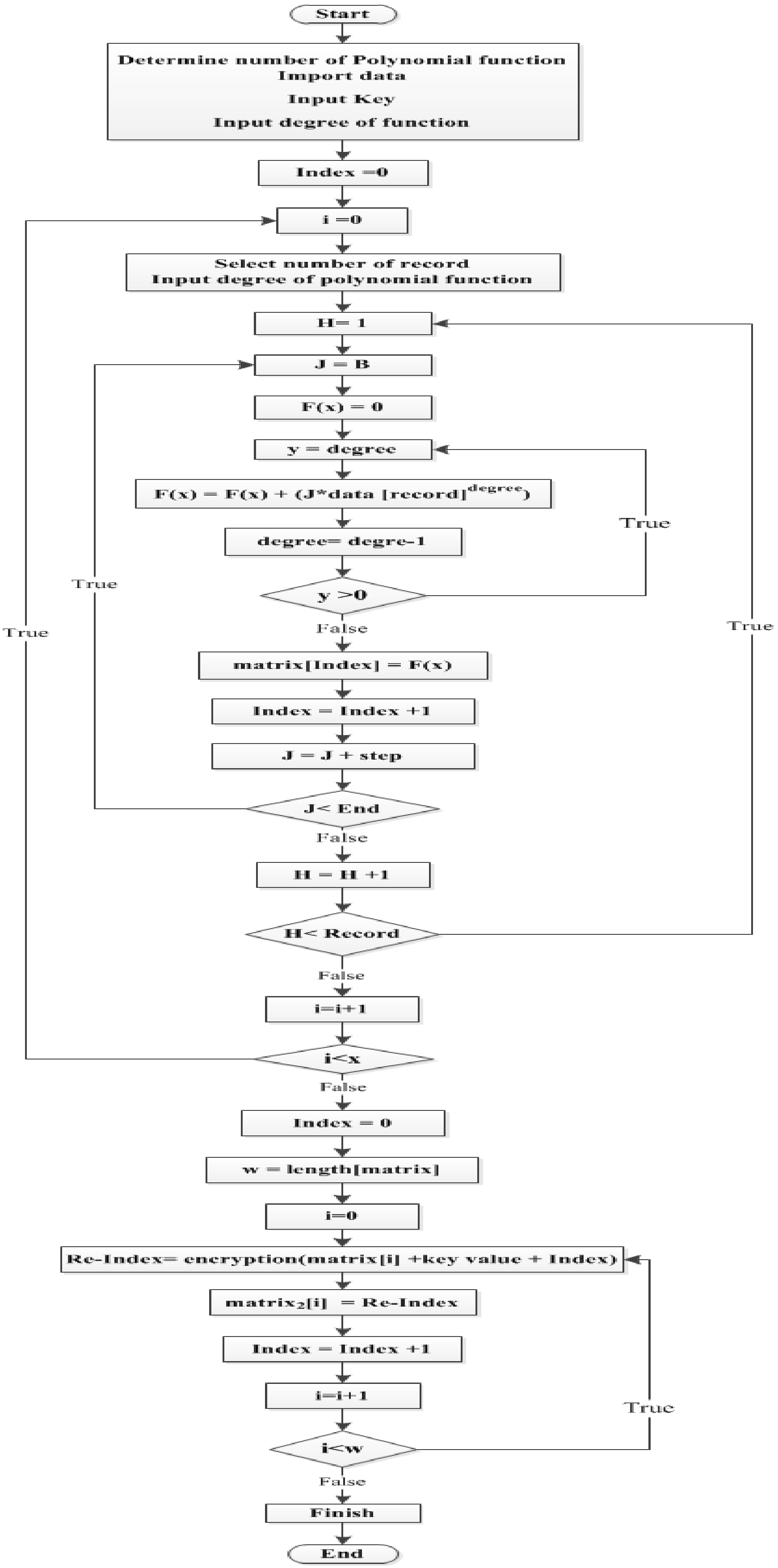


Figure 3.4: Flowchart of Degree of Polynomial Algorithm

Algorithm 3: Degree of Polynomial Algorithm

```
Algorithm: Degree of Polynomial Algorithm
{
X ← no. of functions that is uses in this procedure
Data ← Import data from Northwind database with selected data type
Key size ← Input size of key
Key value ← Rand key size (Key value)
Index ← 0
Beginning ← Input the first value from the range of coefficient
End ← Input the last value of the range of coefficient
Step ←Input the number of increment to the first value until reach to the last value of range
For i= 0 to x
Record ← no. of record that is selected from Northwind database (import data)
Degree ← Input the degree of function
For H =1: record                                  // loop of record number
For J= B: Step: End
F(x) = 0
For Y= degree: -1: 0
F(x) ← f(x) + (J *data [record]degree)
End for                                           // end the loop Y of functions
Matrix [Index] ← f(x)          // this matrix use to save the result of each step of range
Index = Index +1
End for                                     // end the loop J of range with step
End for                                     // end the loop H of the no. of range
End for                                     // end the loop i of the record
Index ← 0
W ← length (matrix)
For i= 0 to w
Re-Index ← Matrix [i]
Re-Index ← encryption (Matrix [i] + Key value + Index)
Matrix2 [i] ← Re-Index
Index ← Index +1
End for                               // end the loop that is use to save the new index
Return Matrix2
}
```

Data Type Algorithm

This algorithm is used to compare the performance of OPE by using different data types with fixed degree of polynomial function, fixed key size, and fixed range of coefficient to study the effect of these types on the performance of OPE. This algorithm records the time of the result and compare between each other. It has been determined which data type can achieve a good performance of OPE. The steps of this algorithm showing as a flowchart in figure 3.5.

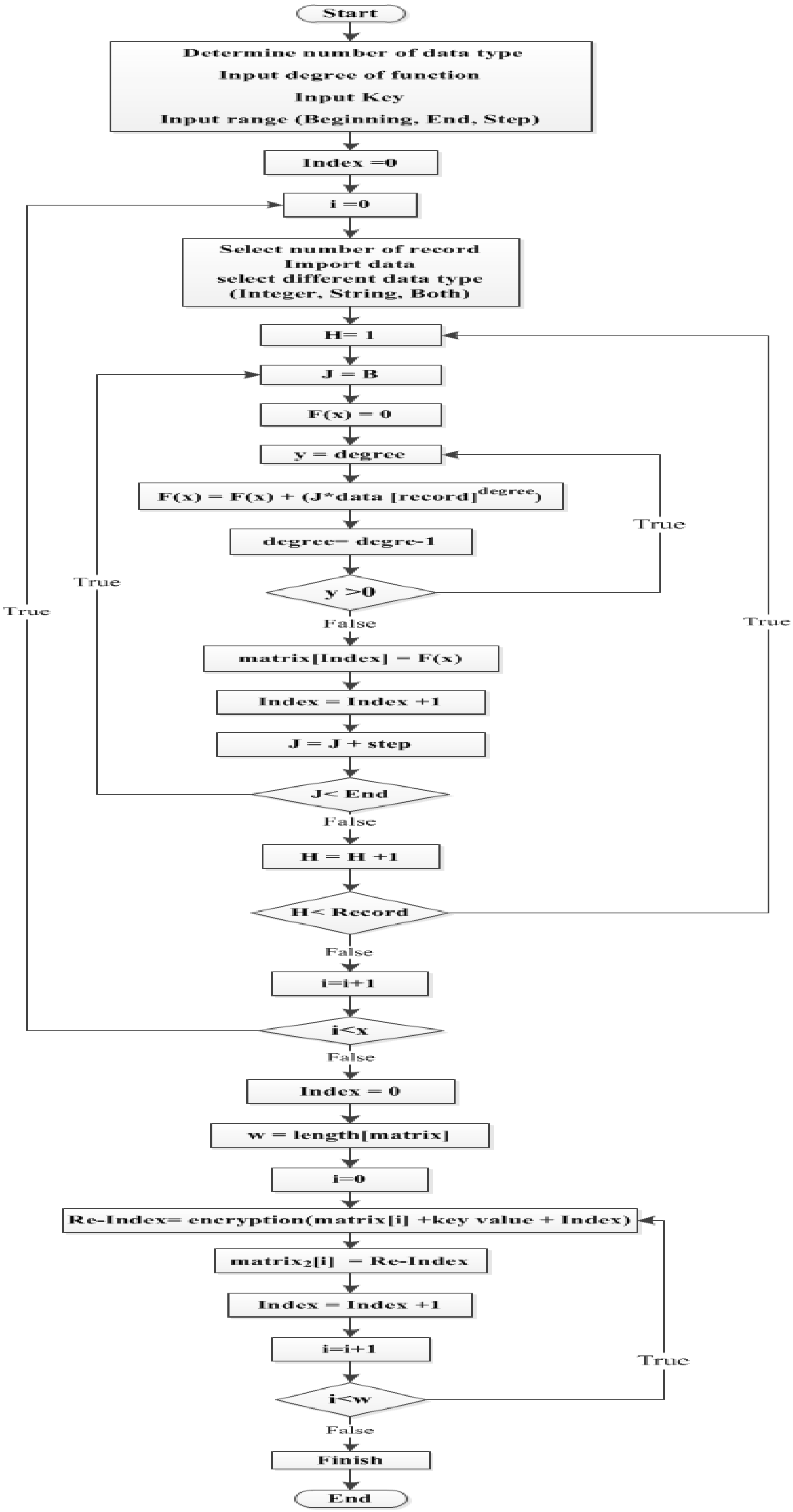


Figure 3.5: Flowchart of Data Type Algorithm

Algorithm 4: Data Type Algorithm

```
Algorithm: Data Type Algorithm
{
X ← no. of data type that is uses in this procedure
Degree ← Input the degree of function
Key size ← Input size of key
Key value ← Rand key size (Key value)
Index ← 0
Beginning ← Input the first value from the range of coefficient
End ← Input the last value of the range of coefficient
Step ←Input the number of increment to the first value until reach to the last value of range
For i= 0 to x
Data ← Import data from Northwind database with selected data type
Record ← no. of record that is selected from Northwind database (import data)
For H =1: record                          // loop of record number
For J= B: Step: End
F(x) = 0
For Y= degree: -1: 0
F(x) ← f(x) + (J *data [record]^degree)
End for // end the loop Y of functions
Matrix [Index] ← f(x)          // this matrix use to save the result of each step of range
Index = Index +1
End for                             // end the loop J of range with step
End for                            // end the loop H of the no. of range
End for                           // end the loop i of the record
Index ← 0
W ← length (matrix)
For i= 0 to w
Re-Index ← Matrix [i]
Re-Index ← encryption (Matrix [i] + Key value + Index)
Matrix2 [i] ← Re-Index
Index ← Index +1
End for                      // end the loop that is use to save the new index
Return Matrix2
}
```

Key Size Algorithm

This algorithm is used to compare the performance of OPE by using different sizes of key with fixed range of coefficient, fixed data type, and fixed degree of polynomial function to study the effect of different sizes of key the performance of OPE.

This algorithm records the time of the results and compare between each other. It has been determined which key size can achieve an optimal performance of OPE. The steps of this algorithm showing as a flowchart in figure 3.6.

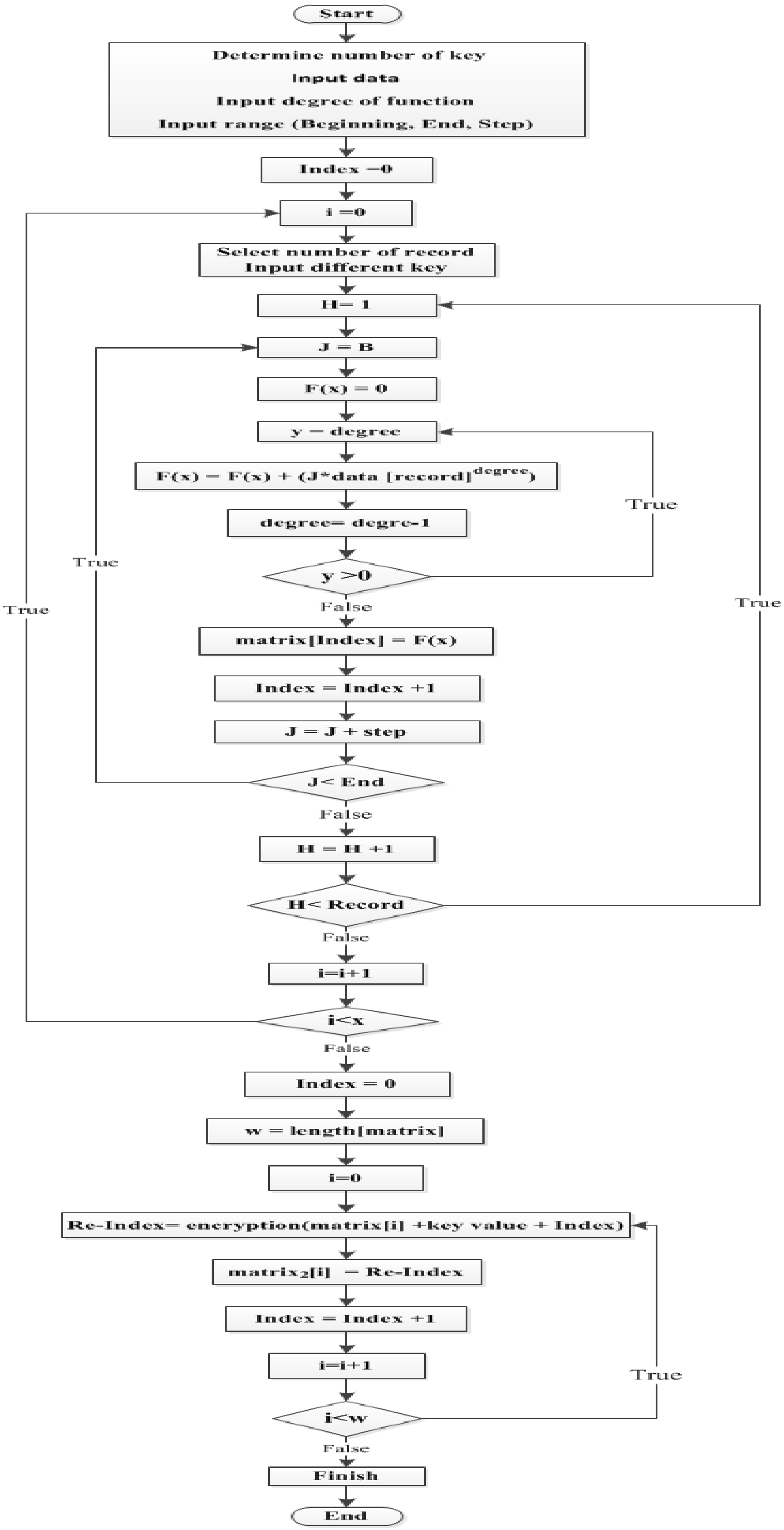


Figure 3.6: Flowchart of Key Size Algorithm

Algorithm 5: Key Size Algorithm

Algorithm: Key Size Algorithm
{
X ← no. of keys that is uses in this procedure
Data ← Import data from Northwind database with selected data type
Degree ← Input the degree of function
Beginning ← Input the first value from the range of coefficient
End ← Input the last value of the range of coefficient
Step ←Input the number of increment to the first value until reach to the last value of range
For i= 0 to x
Key size ← Input size of key
Key value ← Rand $_{\text{key size}}$ (Key value)
Index ← 0
Record ← no. of record that is selected from Northwind database (import data)
For H =1: record **// loop of record number**
For J= B: Step: End
F(x) = 0
For Y= degree: -1: 0
F(x) ← f(x) + (J *data [record]$^{\text{degree}}$)
End for **// end the loop Y of functions**
Matrix [Index] ← f(x) **// this matrix use to save the result of each step of range**
Index = Index +1
End for **// end the loop J of range with step**
End for **// end the loop H of the no. of range**
End for **// end the loop i of the record**
Index ← 0
W ← length (matrix)
For i= 0 to w
Re-Index ← Matrix [i]
Re-Index ← encryption (Matrix [i] + Key value + Index)
Matrix2 [i] ← Re-Index
Index ← Index +1
End for **// end the loop that is use to save the new index**
Return Matrix2
}

Chapter Four

Results and Analysis

4.1 Overview

This chapter explains in details the experimental results and analysis. It is organized in four sections. Section 4.2 introduces the chapter. Section 4.3 explains implementation software. Section 4.4 explains the evaluation metrics. Section 4.5 explains the experiments results and discusses their analysis.

4.2 Brief

The results of this thesis are to find the effect of using the polynomial function with several parameters (Degree of polynomial, Range of coefficients, Key sizes, Data types/ sizes) on the performance of OPE. It has been implemented and designed the software to compare and analyze the performance of OPE. Several experiments have been run with several parameters and recorded the execution time for each data item. This thesis computed the time for each experiment to study the performance of OPE. There were two counters used to compute the time in each experiment. The first counter was used to compute the accumulated time of the experiment. The second counter was used to compute the average time of different data sizes in the same experiment. The results of the experiment were compared and analyzed depending on the performance and security level of OPE.

4.3 Implementation Software

This thesis has been designed and implemented a software to compare and analyze the performance of OPE using VB.net version 2010 as programming language. The software imported the data from Northwind database with the size of 200 records. We found that when importing a large size of the database and ran all the parameters, the experiment have been taken extended time because the combination of the parameters was enormous and the experiment has not ended. In this thesis, Northwind database is not important and is not a parameter. Thus, it has been used the database to run several experiments and to find the outcomes of these experiments. Therefore; we decided to choose the 14 records from Northwind database that have all types of data (integer, string, and both) and different data sizes (2 bytes, 3bytes, 4 bytes, 5 bytes, and 6 bytes). It has been used the size 14 records from 200 records of the database to facilitate studying the effect of several parameters on the performance of OPE.

The following figures displays the interfaces of the implementation software that shows how it has been used the polynomial function with several parameters. The parameter should be initialized before running the software. Figure 4.1 shows these parameters and shows how to import the data with the size of 14 records from Northwind database. The core of the implementation code is listed in Appendix E. The full implementation is available via the email[1].

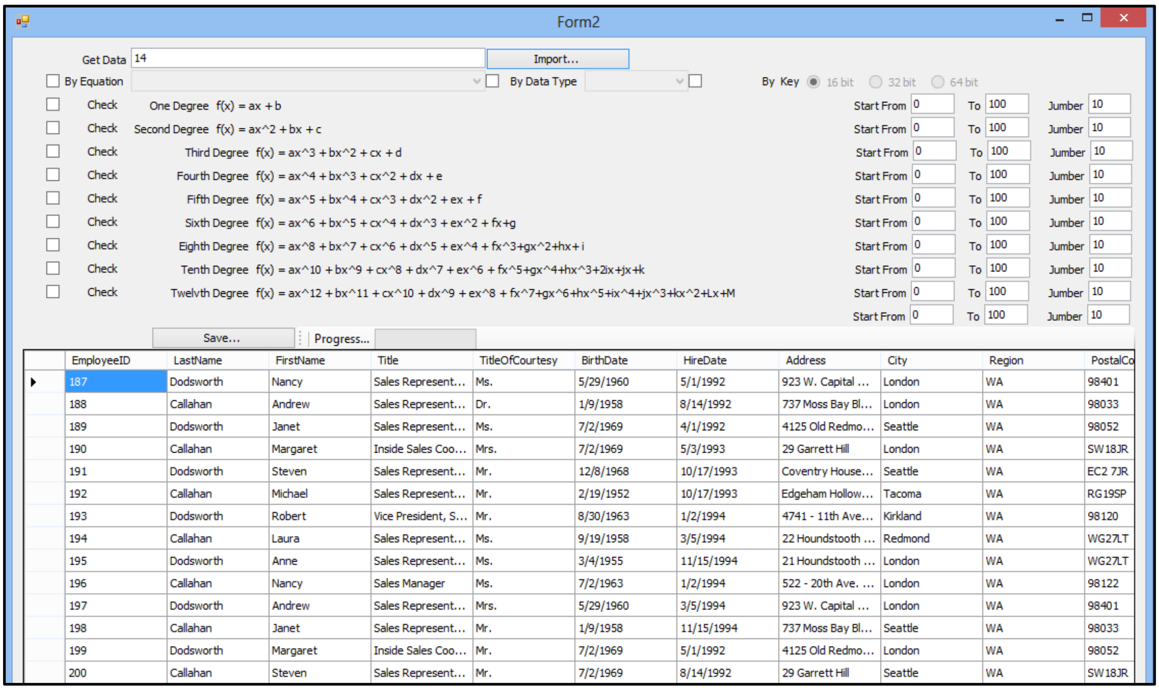


Figure 4.1: Main Interface of Our Proposed Software

[1] Email: Hadeel_alkazaz@yahoo.com

Figure 4.2 shows the use of nine different degrees of polynomial function with selected range, selected key sizes, and selected data types.

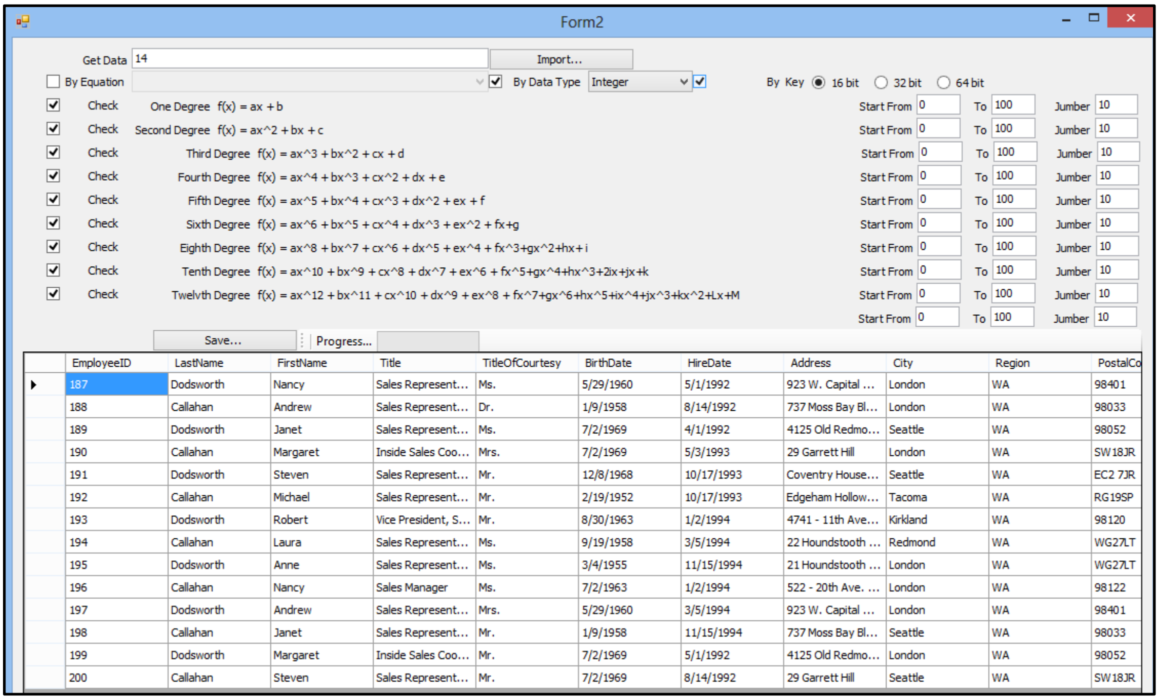


Figure 4.2: The Interface of Using Different Degree of Polynomial Function

Figure 4.3 shows the use of five different ranges of coefficients with selected degree, selected key sizes, and selected data types.

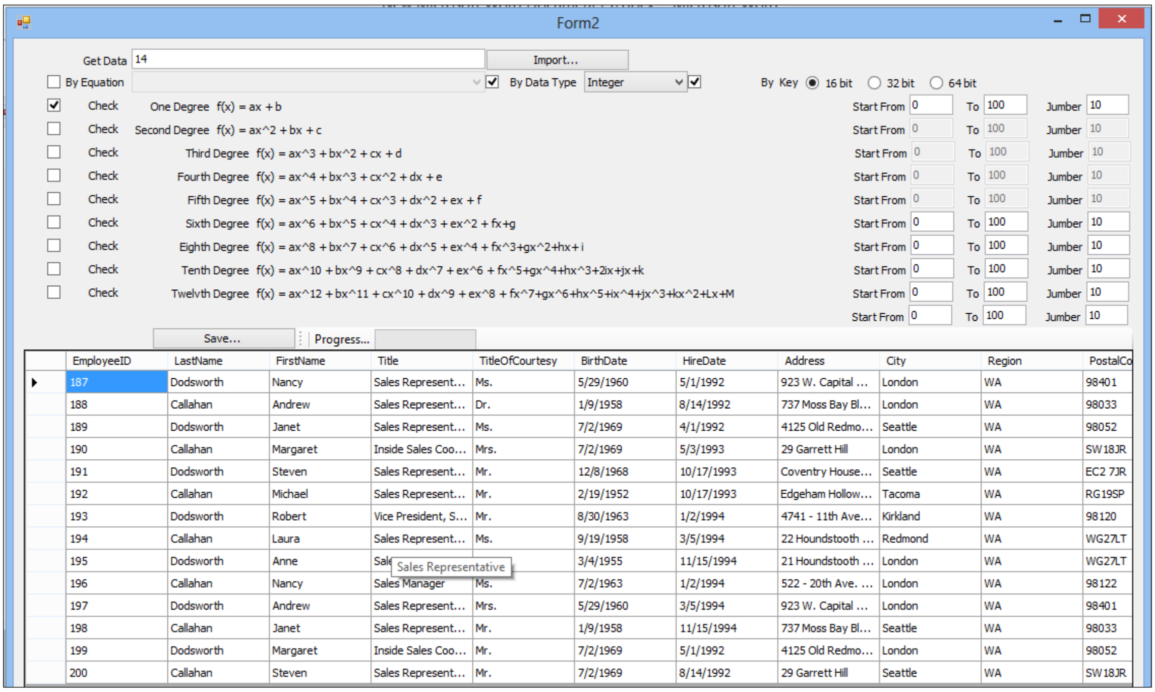


Figure 4.3: The Interface of Using Different Range of Coefficients

Figure 4.4 shows the use of three different key sizes with selected range, selected degree, and selected data types.

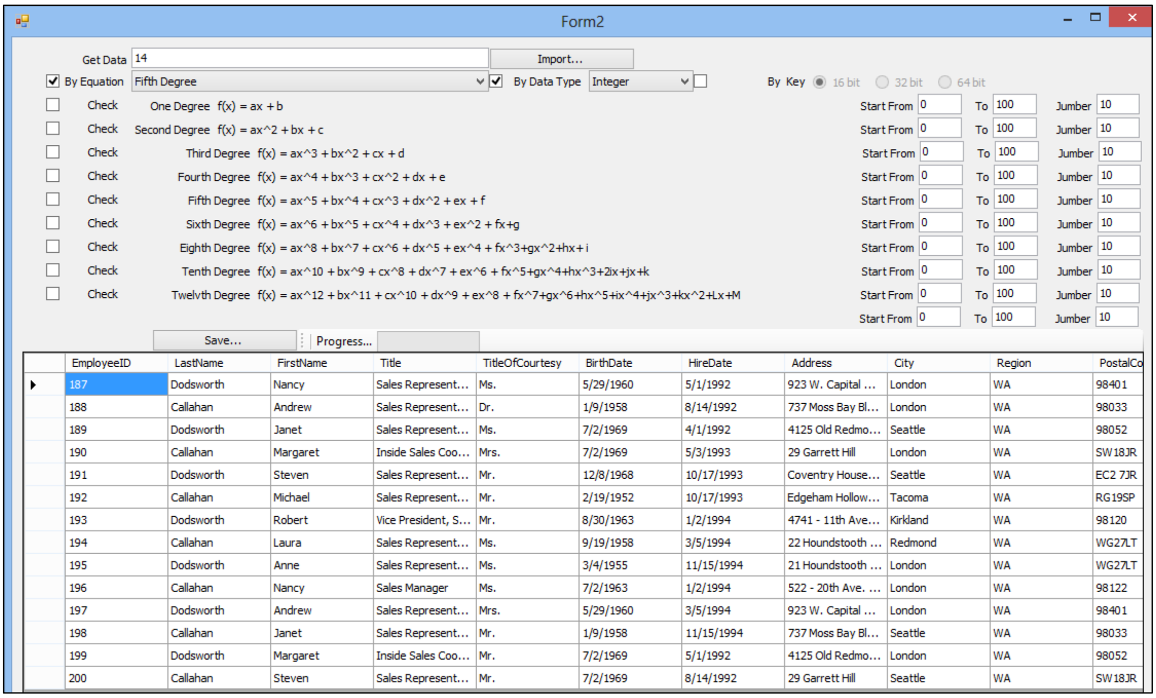


Figure 4.4: The Interface of Using Different Key Sizes

Figure 4.5 shows the use of three different data types with selected range, selected degree, and selected key sizes.

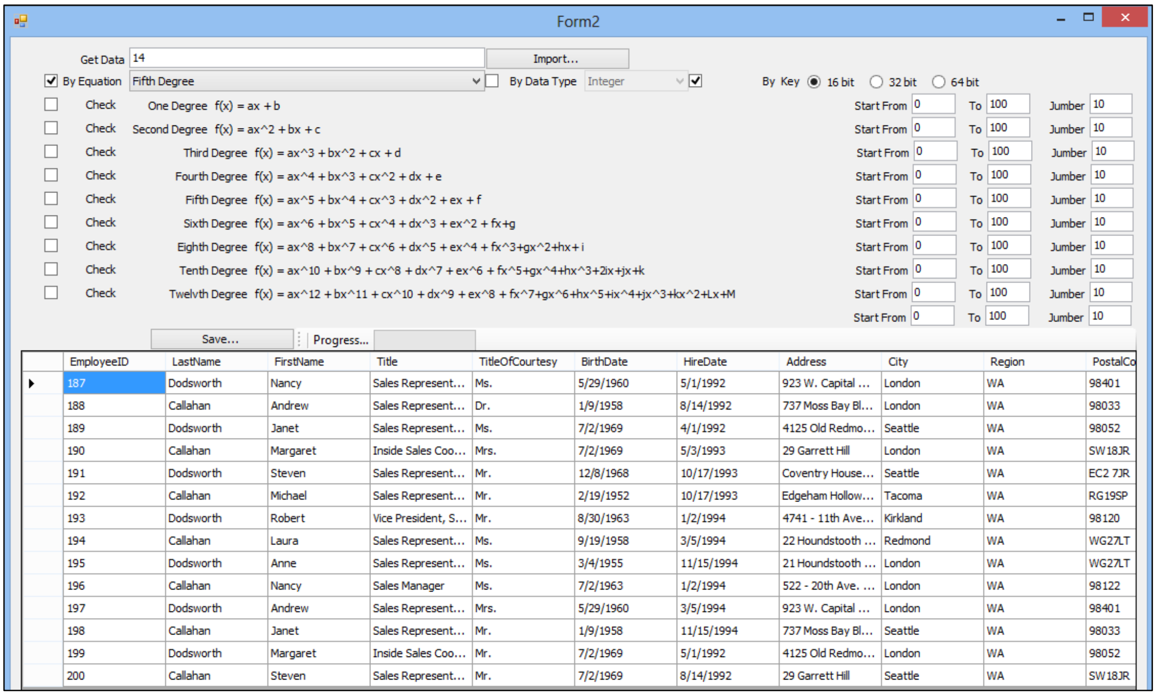


Figure 4.5: The Interface of Using Different Data Types/ Sizes

4.4 Evaluation Metrics

This chapter has been used four evaluation metrics to evaluate and analyze the results of the experiments.

- **The Relative Error**

  This metric is used to find the relative error of the performance time in different degree of the polynomial function. It was used to compute the total of the deviation for the degree of the polynomial function divided by the total performance time for all degrees. The calculation of the Relative Error (deviation) by:

$$\text{The Relative Error} = \frac{\text{the total of deviation}}{\text{the total performance time}} * 100\% \quad \textbf{........ (4.1)}$$

- **Gain Security Level**

This metric is used to find the gain of the security level as a percentage. The gain of the security calculated as the difference between the maximum of the parameters (degree of the polynomial, range of the coefficients, key sizes) and the minimum of these parameters divided by the maximum of the parameters. It has been computed the gain of the security to determine which one of these parameters achieved the high security level.

$$\text{Gain Security} = \frac{\text{maximum of paramter} - \text{minumum of parameters}}{\text{maximum of paramter}} * 100\% \ldots \textbf{\textit{(4.2)}}$$

- **Loss of Performance**

This metric is used to find the loss of the performance in each experiment to determine which parameter can achieves a high performance. It has been computed as a percentage of the performance. The loss of the performance calculated as the difference between the maximum time (performance time) of the parameters (degree of the polynomial, range of the coefficients, key sizes) and the minimum time (performance time) of these parameters divided by the maximum of the parameters.

$$\text{Loss of Performance} = \frac{\text{time the maximum} - \text{time the minimum}}{\text{time the maximum}} * 100\% \ldots\ldots\ldots\ldots \textbf{\textit{(4.3)}}$$

- **Encryption (Re- Index)**

This thesis has been used the encryption formula that is combined the two approaches Özsoyoglu et al. and Popa et al. to study the effect of the key size over security level as well as the performance time (execution time).

$$Encryption\ (Re\text{-}Index) = Function_{value} + Key_{value} + Original\ Index \quad \textbf{…….. (4.4)}$$

4.5 Experimental Results and Analysis

This section has been presented the experimental results and analysis into four parts. These parts are: the effect of several degrees of polynomial function, the effect of several ranges of coefficients, the effect of several key sizes, and the effect of several data types. It has been explained how to identify the efficiency between performance and security level. Inside each experiment, there were many different experiments depending on several parameters that have been selected in each experiment. All of these parts will be discussed in details in following sub-sections.

4.5.1 The effect of using different degree of polynomial function

This experiment has been used different degrees of polynomial function with several parameters to study the effect of these degrees on the performance of OPE. Nine different degrees of polynomial were selected. These degrees were: degree 1, degree 2, degree 3, degree 4, degree 5, degree 6, degree 8, degree 10, and degree 12. The selected range of coefficients was the range 0-100 with step 10. Also three different key sizes have been selected (16 bit, 32 bit, and 64 bit), and three different data types (integer, string, both) with four different data sizes (3 byte, 4 byte, 5 byte, and 6 byte). The degree will decide the security level of the polynomial function. It has been computed the accumulated time of each degree to find which one of these degrees achieves a high performance with high security level. The average time of each data sizes has been computed in this experiment. For this section, we have done six types of experiments that used different degree of polynomial to study the effect of these degrees on the performance of OPE. For the sake of brevity, this thesis will explain only one group of experiments (Experiment 4.5.1) and will list only the summary of other group of experiments. More samples of the experiment results are listed in Appendix A. All results for all experiments are available via the email. In the following only experiment 4.5.1 will be explained in details to study the effect of using different degrees with selected parameters.

Experiment 4.5.1

This experiment has been used nine different degrees of the polynomial function with selected integer data type, the key size equal 16 bit, and range 0-100 with step 10 to study the effect of these degrees on the performance of OPE. It has been computed the accumulated time of each degree to find the optimal polynomial degree. The efficient degree is the degree which gives the highest security level with minimum loss of performance (execution time).

Table 4.1 and figure 4.6 shows the performance time of the nine degrees. The degree 12 had the highest performance time (14.38) and degree 1 had the lowest performance time (9.88). This means that degree 1 achieved a high performance but a low security level since the highest polynomial degree has the highest security level. The security level for each degree had been monitored by counting their coefficients. As expected, the high number of coefficients means high security level. This experiment has been found the optimal point as a trade-off between security level and performance.

Table 4.1: Results of Different Polynomial Function with (Integer, 16 bit, Range 0-100)

DataType	KeySize	Range	EquationID	PerformanceTime
Integer	16 bit	0-100	Degree 1	9.88
Integer	16 bit	0-100	Degree 2	10.14
Integer	16 bit	0-100	Degree 3	11.79
Integer	16 bit	0-100	Degree 4	10.73
Integer	16 bit	0-100	Degree 5	12.24
Integer	16 bit	0-100	Degree 6	12.84
Integer	16 bit	0-100	Degree 8	13.15
Integer	16 bit	0-100	Degree 10	12.94
Integer	16 bit	0-100	Degree 12	14.38

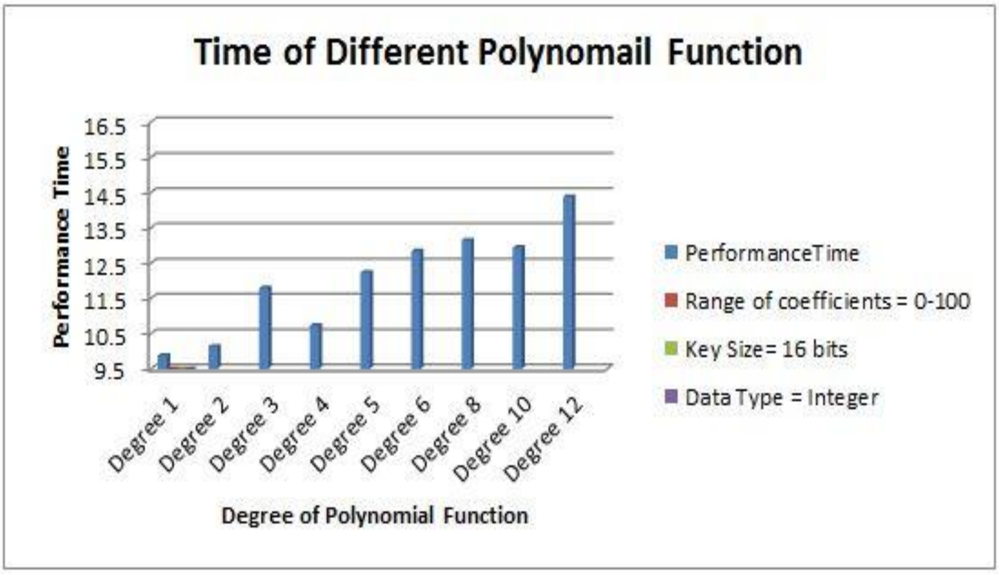


Figure 4.6: Different Polynomial Function

As expected that whenever the degree of polynomial increased, the performance time will be increased. It has been used this as a controlling factor for the experiment. It has been noticed that as shown in table 4.1 and figure 4.6 that the performance time is increased for almost all degrees except for degree 4 and 10 of the polynomial function.

The total time for all degrees is 108 while the deviation for degree 4 and 10 at maximum is 2. Equation (4.1) has been used to calculate the Relative Error (deviation).

$$\text{The Relative Error} = \frac{\text{the total of deviation}}{108} * 100\% \quad = 4/108*100= 4\%$$

It has been found that the error is less than 4 % in all experiments.

From table 4.1 and figure 4.6, we derived table 4.2 that is used to compare between the security and the performance in different degrees of the polynomial function. This thesis has been used two equations to compute the gain of security and the loss of performance. Table 4.2 has been compared several cases to find the optimal outcomes.

Table 4.2: Compare between Security and Performance in Different Degree

Row	Degree	Gain Security Level	Loss of Performance
1	Increased the degree from 1 to 12	91%	31%
2	Increased the degree from 1 to 4	75%	7%
3	Increased the degree from 5 to 8	37%	6%
4	Increased the degree from 5 to 10	50%	5%
5	Increased the degree from 5 to 12	58%	14%

In this experiment, degree 12 of the polynomial function has the maximum security level and the degree 1 of the polynomial has the less security level. The degree decides the security level of the polynomial function. We suppose to compute the percentage of the gain of the security as the different between the maximum degree of the polynomial and the minimum degree of the polynomial divided by the maximum degree of the polynomial.

For example, row 1 in table 4.2 computes the gain of the security by using equation (4.2) and computes the loss of the performance by using equation (4.3).

Gain Security Level = $\frac{\text{degree 12} - \text{degree1}}{\text{degree 12}} * 100\%$ = 91 %

Loss of Performance = $\frac{\text{Time of degree 12 – Time of degree 1}}{\text{Time of degree 12}} * 100\%$

$$= \frac{14.38 - 9.88}{14.38} * 100\% \quad = 31\%$$

Other examples are listed in table 4.2 which illustrates the results of the gain of security and the loss of the performance. Also these are explained below.

- When the degree of the polynomial function increased from 1 to 12, we will gain 91% security level with 31% loss of performance.
- When the degree of the polynomial function increased from 1 to 4, we will gain 75% security level with 7% loss of performance.
- When the degree of the polynomial function increased from 5 to 8, we will gain 37% security level with 6% loss of performance.
- When the degree of the polynomial function increased from 5 to 10, we will gain 50% security level with 5% loss of performance.
- Furthermore, when the degree of the polynomial function increased from 5 to 12, we will gain 58% security level with 14 % loss of performance.

The results of table 4.2 conclude that when the degree of the polynomial function is increased from 1 to 4, the security level will gain 75% with 7% loss of performance. This experiment has found that degree 4 of the polynomial function is the optimal choice.

In experiment 4.5.1, there are different sizes of data inside each degree. Table 4.3 and figure 4.7 shows the different data sizes with different degrees of the polynomial function. In this experiment, it has been computed the average time of each size of data (3 byte, 4 byte, 5 byte, and 6 bytes) in different degree of polynomial function (degree1, degree5, degree8, and degree12).

Table 4.3: Results of Different Data Sizes in Different Polynomial Function

DataType	KeySize	Range	DataSize	Degree 1	Degree 5	Degree 8	Degree 12	Avg. TDDsize
Integer	16 bit	0-100	3 byte	0.34	0.43	0.501	0.506	0.44
Integer	16 bit	0-100	4 byte	0.34	0.44	0.45	0.52	0.43
Integer	16 bit	0-100	5 byte	0.34	0.41	0.51	0.507	0.44
Integer	16 bit	0-100	6 byte	0.39	0.48	0.46	0.51	0.46

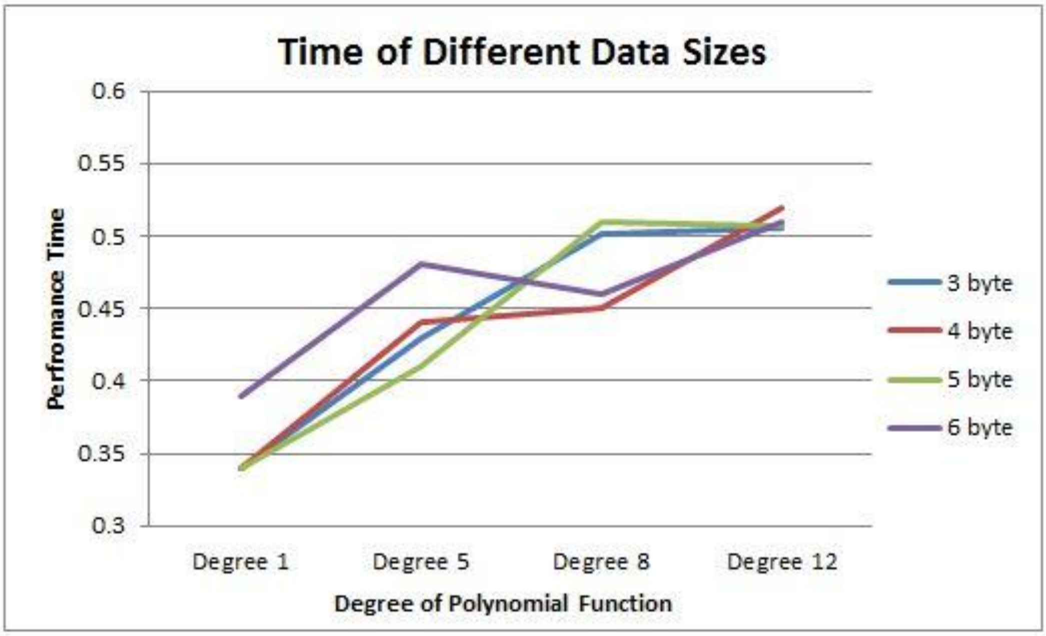


Figure 4.7: Different Data Sizes with Different Polynomial Function

From table 4.3 and figure 4.7, we derived table 4.4 that is used to compare between the security and the performance with different data sizes. Table 4.4 has been compared several cases to find the optimal outcomes.

Table 4.4: Compare between Security and Performance with Different Data Sizes

Data Size		Gain Security Level	Loss of Performance
3 byte	1. Increased the size of data from 3 to 6 byte in Degree 1	50%	12%
	2. Increased the degree from 1 to 5	80%	20%
	3. Increased the degree from 1 to 8	87%	32%
	4. Increased the degree from 1 to 12	91%	32%
4 byte	1. Increased the degree from 1 to 5	80%	22%
	2. Increased the degree from 1 to 8	87%	24%
	3. Increased the degree from 1 to 12	91%	34%
5 byte	1. Increased the degree from 1 to 5	80%	20%
	2. Increased the degree from 1 to 8	87%	32%
	3. Increased the degree from 1 to 12	91%	32%
6 byte	1. Increased the degree from 1 to 5	80%	18%
	2. Increased the degree from 1 to 8	87%	15%
	3. Increased the degree from 1 to 12	91%	23%

It has been computed the gain of security and the loss of the performance with different data sizes as a percentage and then compared between them. From table 4.4, we noticed the following:-

1. Data size (3 byte)
   - Increasing the size of data in degree 1of the polynomial from 3 to 6 bytes will gaining 50% security level with 12% losing of performance.
   - When the degree increased from 1 to 5 of the polynomial function, we will gain 80% security level with 20% loss of performance.
   - When the degree increased from 1 to 8 of the polynomial function, we will gain 87% security level with 32% loss of performance.
   - When the degree increased from 1 to 12 of the polynomial function, we will gain 91% security level with 32% loss of performance.

2. Data size (4 byte)
   - When the degree increased from 1 to 5 of the polynomial function, we will gain 80% security level with 22% loss of performance.
   - When the degree increased from 1 to 8 of the polynomial function, we will gain 87% security level with 24% loss of performance.
   - When the degree increased from 1 to 12 of the polynomial function, we will gain 91% security level with 34% loss of performance.
3. Data size (5 byte)
   - When the degree increased from 1 to 5 of the polynomial function, we will gain 80% security level with 20% loss of performance.
   - When the degree increased from 1 to 8 of the polynomial function, we will gain 87% security level with 32% loss of performance.
   - When the degree increased from 1 to 12 of the polynomial function, we will gain 91% security level with 32% loss of performance.
4. Data size (6 byte)
   - When the degree increased from 1 to 5 of the polynomial function, we will gain 80% security level with 18% loss of performance.
   - When the degree increased from 1 to 8 of the polynomial function, we will gain 87% security level with 15% loss of performance.
   - When the degree increased from 1 to 12 of the polynomial function, we will gain 91% security level with 23% loss of performance.

The results of table 4.4 conclude that when the degree is from 1 to 8 of the polynomial function will gain 87% security level with loss 15% performance. This experiment has been found that degree 8 with data size (6 bytes) is the optimal outcomes.

4.5.2 The effect of using different range of coefficients

This experiment has been used different range of coefficients with several parameters to study the effect of these ranges on the performance of OPE. Five different ranges of coefficients were selected. These ranges were range 0-100 with step 10, range 100-1000 with step 100, range 1000-10000 with step 1000, range 10000-100000 with step 10000, and range 100000-1000000 with step 100000. The selected degrees of polynomial function were degree1 and degree 10. Also three different key sizes have been selected (16 bit, 32 bit, 64 bit), and three different data types (integer, string, both) with different data sizes (3 byte, 4 byte, 5 byte, and 6 byte).

It has been computed the accumulated time of each range to find which one of these ranges achieves a high performance with high security level. The average time of each data sizes has been computed in this experiment. For this section, we have done six types of experiments that is used different range of coefficients to study the effect of these ranges on the performance of OPE. For the sake of brevity, this thesis will explain only one group of experiments (i.e. Experiment 4.5.2) and will list only the summary of other group of experiments. More samples of the experiment results are listed in Appendix B. All results for all experiments are available via the email. In the following only experiment 4.5.2 will be explained in details to study the effect of using different range of coefficients with selected parameters.

Experiment 4.5.2

This experiment has been used the same parameters in the previous experiment but in this time, we used the different range of coefficient instead of different degrees of the polynomial. This experiment was used five different ranges of coefficients with selected degree 1 of the polynomial function, integer data type, the key size equal 16 bit to study the effect of these ranges on the performance of OPE. It has been computed the accumulated time of each range to find the optimal range of coefficient. The efficient range is the range which gives the highest security level with the minimum loss of performance (execution time).

Table 4.5 and figure 4.8 shows the performance time of the five ranges. The range 10000-100000 had the highest performance time (11.08) and the range 100-1000 had the lowest performance time (9.94). This means that the range 100-1000 achieved a high performance. The security level for each range has been monitored by the number of coefficients. The less number of coefficients means that the less security level. This experiment has been found the optimal point as trade-off between security level and performance.

Table 4.5: Results of Different Ranges of Coefficients

EquationID	DataType	KeySize	Range of Coefficients	Step	PerformanceTime
Degree 1	Integer	16 bit	0 - 100	10	10.54
Degree 1	Integer	16 bit	100 - 1000	100	9.94
Degree 1	Integer	16 bit	1000 - 10000	1000	10.26
Degree 1	Integer	16 bit	10000 - 100000	10000	11.08
Degree 1	Integer	16 bit	100000 - 1000000	100000	11.01

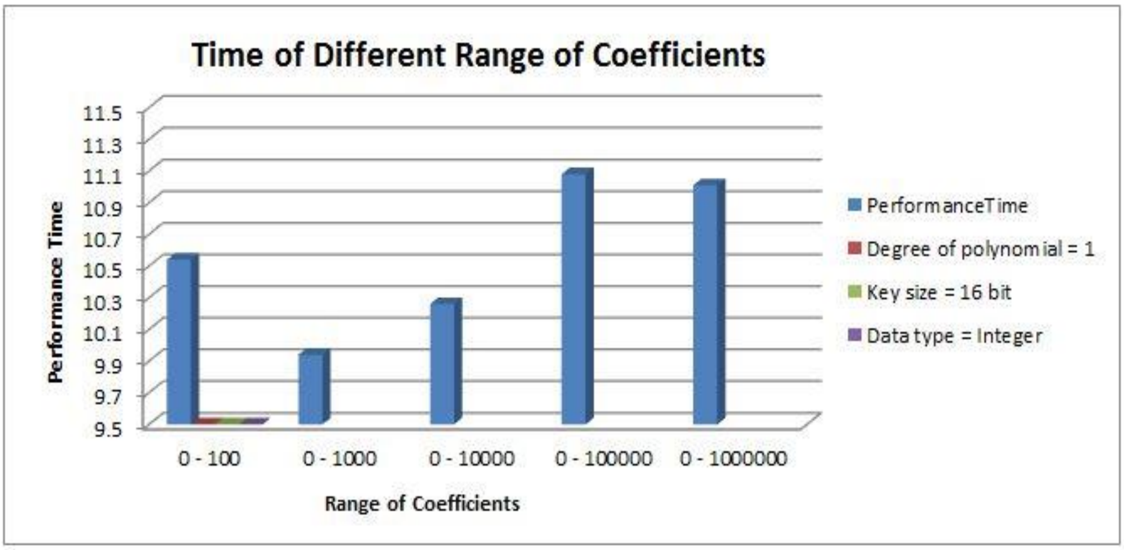


Figure 4.8: Different Ranges of Coefficient

As expected that whenever the range of coefficient increased, the performance time will be increased. It has been used this as a controlling factor for this experiment. It has been noticed that as shown in table 4.5 that the performance time is increased for all ranges except for range 100-1000 and 100000-1000000. The total time for all ranges is 53 while the deviation for range 1000 and 1000000 at maximum is 2. Equation (4.1) has been used to calculate the Relative Error (deviation).

The Relative Error = $\frac{\text{the total of deviation}}{53} * 100\%$ = 4/53*100 = 8 %

It has been found that the error is less than 8 % in all experiments.

From table 4.5 and figure 4.8, we derived table 4.6 that is used to compare between the security and the performance in different ranges of coefficients. Table 4.6 has been compared several cases to find the optimal choice.

Table 4.6: Compare between Security and Performance in Different Range

Row	Range	Gain Security Level	Loss of Performance
1	Increased the range from 100 to 100,000	99%	4%
2	Increased the range from 100 to 1000,000	99%	4%
3	Increased therange from 1000 to 10,000	90%	3%
4	Increased therange from 1000 to 100,000	99%	10%
5	Increased the range from 1000 to 1000,000	99%	9%

In this experiment, the range of coefficients decides the security level. The range 1000000 has the maximum security level and range 1000 has the minimum security level. From table 4.6, we noticed that the gain of security is always between 90% and 99% because this thesis was used the large range of coefficients. This experiment focused on the loss of performance and it has been considered that the minimum loss of performance is the optimal range of coefficient.

For example, row 1 in table 4.6 computes the gain of the security by using equation (4.2) and computes the gain of the performance by using equation (4.3).

$$\text{Gain Security Level} = \frac{\text{range } 100{,}000 - \text{range } 100}{\text{range } 100{,}000} * 100\% = 99\ \%$$

$$\text{Loss of Performance} = \frac{\text{Time of range } 100{,}000 - \text{Time of range } 100}{\text{Time of range } 100{,}000} * 100\%$$

$$= \frac{11.08 - 10.54}{11.08} * 100\% = 4\%$$

Other examples are listed in table 4.6 which illustrates the results of the gain of security and the loss of the performance. Also these are explained below.

- When the range of coefficient increased from 100 to 100,000, we will gain 99% security level with 4% loss of performance.

- When the range of coefficient increased from 100 to 1000,000, we will gain 99% security level with 4% loss of performance.
- When the range of coefficient increased from 1000 to 10,000, we will gain 90% security level with 3% loss of performance.
- When the range of coefficient increased from 1000 to 100,000, we will gain 99% security level with 10% loss of performance.
- Furthermore, when the range of coefficient increased from 1000 to 1000,000, we will gain 99% security level with 9% loss of performance.

The results of table 4.6 conclude that the range of coefficient is increased from 1000 to 10000, we will gain 90% security level with 3% loss of performance. This experiment has been found that range 10000 is the best optimal choice.

In experiment 4.5.2, there are four different data sizes inside each range. Table 4.7 and figure 4.9 shows the different data sizes with different range of coefficients. The average time of each size of data has been computed. These sizes were: 3 bytes, 4 bytes, 5 bytes, and 6 bytes in different range of coefficients.

Table 4.7: Results of Different Data Sizes in different Range of Coefficients

DataType	KeySize	DataSize	Range 0- 100	Range 100 - 1000	Range 1000 - 10000	Range 10000- 100000	Range 100000 -1000000	Avg. TDDsize
Integer	16 bit	3 byte	0.71	0.68	0.66	0.73	0.74	0.704
Integer	16 bit	4 byte	0.69	0.66	0.69	0.73	0.74	0.702
Integer	16 bit	5 byte	0.68	0.65	0.72	0.78	0.71	0.708
Integer	16 bit	6 byte	0.78	0.71	0.73	0.79	0.78	0.75

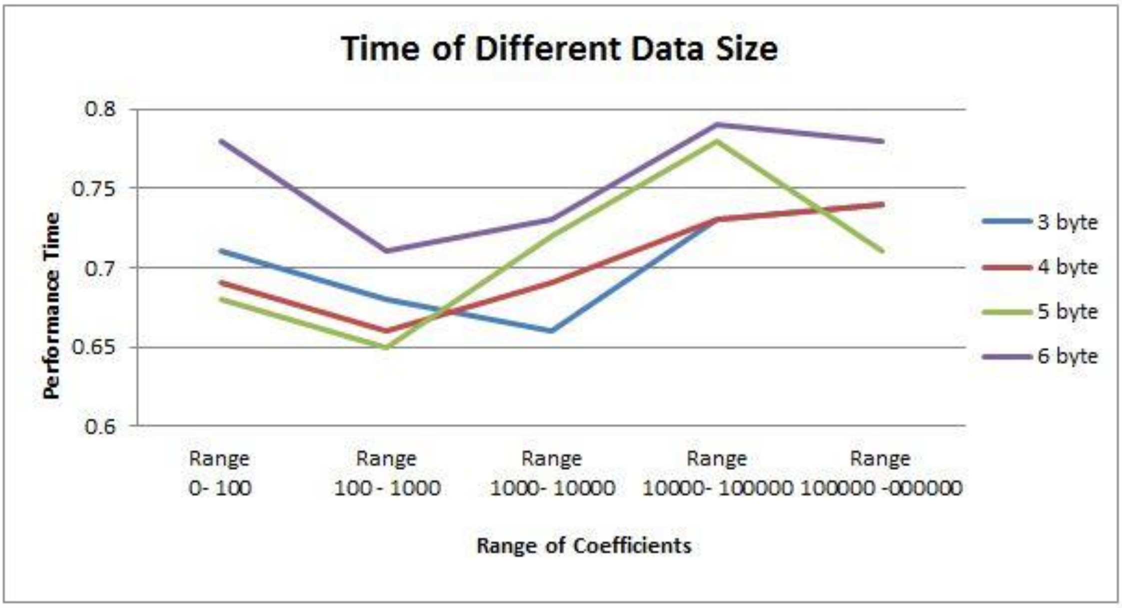


Figure 4.9: Different Data Sizes in Different Range of Coefficients

From table 4.7 and figure 4.9, we derived the table 4.8 that is used to compare between the security and the performance with different data sizes. Table 4.8 has been compared several cases to find the optimal choice.

Table 4.8: Compare between Security and Performance with Different Data Sizes

Data Size		Gain Security Level	Loss of Performance
3 byte	1.Increased thesize of data from 3 to 6 byte in Range 0 -100	50%	8%
	2.Increased the range from (0 - 100) to (10000 -100,000)	75%	2%
	3.Increased the range from (0 - 100) to (100,000 -1000,000)	80%	4%
	4.Increased the range from (100 - 1000) to (10000 -100,000)	50%	6%
	5.Increased the range from (1000 - 10000) to (10000 - 100,000)	25%	9%
4 byte	1.Increased the range from (0 -100) to (10000 -100,000)	75%	5%
	2.Increased the range from (0 -100) to (100,000 -1000,000)	80%	6%
	3.Increased the range from (100 -1000) to (10000 -100,000)	50%	9%
	4.Increased the range from (1000 -10000) to (10000 -100,000)	25%	5%
5 byte	1.Increased the range from (0 -100) to (10000 -100,000)	75%	12%
	2.Increased the range from (0 -100) to (100,000 -1000,000)	80%	4%
	3.Increased the range from (100 -1000) to (10000 -100,000)	50%	16%
	4.Increased the range from (1000 -10000) to (10000 -100,000)	25%	7%
6 byte	2.Increased the range from (0 -100) to (100,000 -1000,000)	75%	0%
	3.Increased the range from (100 -1000) to (10000 -100,000)	50%	10%
	4.Increased the range from (1000 -10000) to (10000 -100,000)	25%	7%

It has been computed the gain of security by using equation (4.2) and the loss of the performance by using equation (4.2) with different data sizes as a percentage and then compared between them. From table 4.8, we noticed the following.

1. Data size (3 byte)
   - Increasing the size of data in range (0-100) of coefficients from 3 to 6 bytes will gaining 50% security level with 8% losing of performance.
   - When the range of coefficients increased from (0- 100) to (10000 – 100,000), we will gain 75% security level with 2% loss of performance.
   - When the range increased from (0- 100) to (100,000 –1000, 000), we will gain 80% security level with 4% loss of performance.
   - When the range increased from (100- 1000) to (10000- 100,000), we will gain 50% security level with 6% loss of performance.

- When the range increased from (1000- 10000) to (10000- 100,000), we will gain 25% security level with 9% loss of performance.

2. Data size (4 byte)
   - When the range increased from (0-100) to (10000 –100,000), we will gain 75% security level with 5% loss of performance.
   - When the range increased from (0-100) to (100,000- 1000, 000), we will gain 80% security level with 6% loss of performance.
   - When the range increased from (100-1000) to (10000- 100,000), we will gain 50% security level with 9% loss of performance.
   - When the range increased from (1000-10000) to (10000-100,000), we will gain 25% security level with 5% loss of performance.
3. Data size (5 byte)
   - When the range increased from (0-100) to (10000–100,000), we will gain 75% security level with 12% loss of performance.
   - When the range increased from (0-100) to (100,000–1000, 000), we will gain 80% security level with 4% loss of performance.
   - When the range increased from (100-1000) to (10000-100,000), we will gain 50% security level with 16% loss of performance.
   - When the range increased from (1000-10000) to (10000-100,000), we will gain 25% security level with 7% loss of performance.
4. Data size (6 byte)
   - When the range increased from (0-100) to (100,000–1000, 000), we will gain 75% security level with 0% loss of performance.
   - When the range increased from (0-100) to (100,000–1000, 000), we will gain 50% security level with 10% loss of performance.

- When the range increased from (1000-10000) to (10000-100,000), we will gain 25% security level with 7% loss of performance.

The results of table 4.8 conclude that the range of coefficients is increase from (0-100) to (100,000–1000, 000) will gain 75% security level with no losing of performance. This experiment has been found that range (100,000 - 1000, 000) with data size (6 bytes) is the best optimal choice.

4.5.3 The effect of using different key size

This experiment has been used different key sizes with several parameters to study the effect of these keys on the performance of OPE. Three different key sizes were selected. These key-sizes were: 16 bits, 32 bits, and 64 bits. For the sake of brevity, only four degrees have been selected. The selected degrees of the polynomial function were degree1, degree5, degree8, and degree12 of the polynomial function. Also three different data types have been selected (integer, string, and both) with different data sizes (3 byte, 4 byte, 5 byte, 6 byte), and the range of coefficients was the range 0 -100, step 10. The key-size will decide the security level of the encryption function. The higher key-size will give a higher security level.

Keys and keys-sizes have essential role in deciding the security level for OPEs. There were two approaches for using the keys to enhance the security of OPEs. Therese were (Özsoyoglu et al., 2003) and (Popa et al., 2009). In Özsoyoglu et al. approach, they used the polynomial function as the encryption function. They used the coefficients of the polynomial function as a key with the range of coefficient [1-2^5]. However, Popa et al. enhanced the OPE security by adding a random noise to increase the ambiguity which consequently enhance the security level.

This thesis combined the above two approaches to study the effect of key-size over security level as well as the performance (execution time). It has been used an encryption formula that combined the two approaches (Equation 4.4). In Equation 4.4, the function value represents the value of the polynomial for several inputs with different degrees and different coefficients. The effect of this function will represent effect of Özsoyoglu et al. approach. In addition to equation 4.4, Key $_{value}$ has been generated randomly for each key size. The key value represents the Popa et al. approach. Many experiments have been done using Equation 4.4 and the execution time has been recorded. The new index will be used re-index of existing data in the database.

$$\textbf{\textit{Encryption (Re-Index)}} = \textbf{\textit{Function}}_{value} + \textbf{\textit{Key}}_{value} + \textbf{\textit{Original}}_{Index}$$

The value of the re-index must be an integer value to facilitate the search over encrypted data. It has been computed the accumulated time of each key to find which one of these keys achieves a high performance with high security level. The average time of each data sizes has been computed in this group of experiments. We have done several experiments using different key sizes (three sizes), different data types, and different polynomial degrees to study the effect of these keys on the performance of OPE. For the sake of brevity, this thesis will explain only one group of experiments (i.e. Experiment 4.5.3) and will list only the summary of other group of experiments. More samples of the experiment results are listed in Appendix C. All results for all experiments are available via the email. In the following, only experiment 4.4.3 will be explained in details to study the effect of using different key-sizes with selected parameters.

Experiment 4.5.3

This experiment has been used the same parameters in the experiment 4.5.1 and experiment 4.5.2 but in this time, we used the different key sizes instead of different degrees of polynomial and different range of coefficient. This experiment used three different key sizes with selected integer data type, degree 1 of polynomial function, and selected range of coefficient 0-100 with step 10 to study the effect of three different key sizes on the performance of OPE. It has been computed the accumulated time of each key size to find the optimal key size.

Table 4.9 and figure 4.10 shows the performance time of the three key sizes. The key size equal 16 bit had the lowest performance time (8.76) and the key size equal 64 bit had the highest performance time (12.08). As expected, the large size of key means high security level. This experiment has been found the optimal point as a trade- off between security level and performance.

Table 4.9: Results of Different Key Sizes

EquationID	DataType	Range	KeySize	PerformanceTime
Degree 1	Integer	0-100	16 bit	8.76
Degree 1	Integer	0-100	32 bit	9.73
Degree 1	Integer	0-100	64 bit	12.08

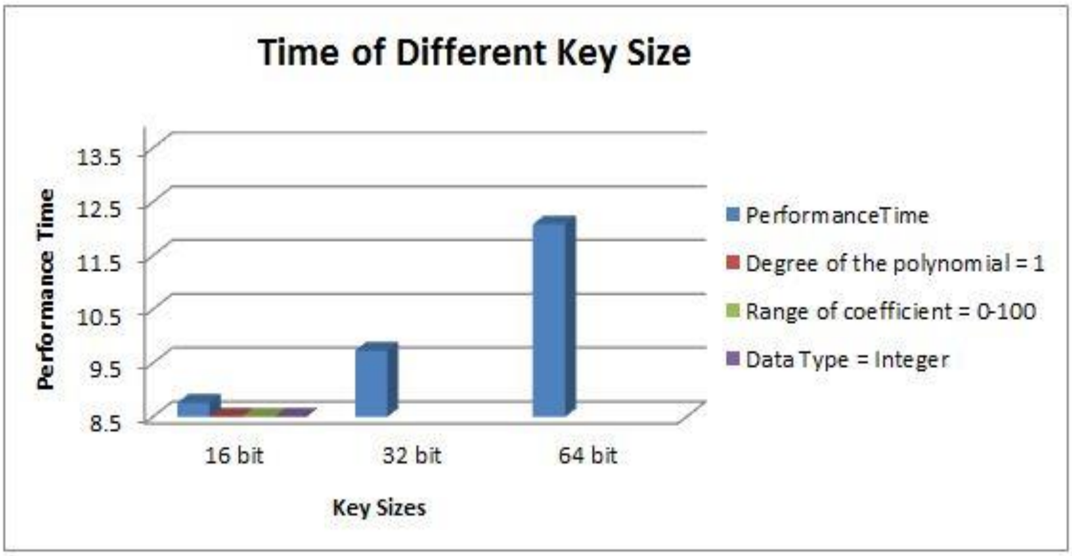


Figure 4.10: Results of Different Key Sizes

From table 4.9 and figure 4.10, we derived table 4.10 that is used to compare between the security and the performance in different key sizes. Table 4.10 has been compared several cases to find the best optimal choice.

Table 4.10: Compare between Security and Performance in Different Key Size

Row	Key Size	Gain Security Level	Loss of Performance
1	Increased the Key from 16 bit to 32 bit	50%	9%
2	Increased the Key from 16 bit to 64 bit	75%	27%
3	Increased the Key from 32 bit to 64 bit	50%	19%

In this experiment, key size equal 64 bits has the maximum security level and the key size equal 16 bits has the minimum security level. We suppose to compute the percentage of the gain of the security as the different between the maximum key size and the minimum key size divided by the maximum key size.

For example, row1 in table 4.10 computes the gain of the security by using equation (4.2) and computes the loss of the performance by using equation (4.3).

$$\text{Gain Security Level} = \frac{\text{key } 32 - \text{key } 16}{\text{key } 32} * 100\% = 50\ \%$$

$$\text{Loss of Performance} = \frac{\text{Time of key } 32 - \text{Time of key } 16}{\text{Time of key } 32} * 100\%$$

$$= \frac{9.73 - 8.76}{9.73} * 100\% = 9\%$$

Other examples are listed in table 4.10 which illustrates the results of the gain of security and the loss of the performance.

- When the key size increased from size equal 16 bit to size equal 32 bit, we will gain 50% security level with 9% loss of performance.

- When the key size increased from size equal 16 bit to size equal 64 bit, we will gain 75% security level with 27% loss of performance.
- When the key size increased from size equal 32 bit to size equal 64 bit, we will gain 50% security level with 19% loss of performance.

The results of table 4.10 conclude that when the key size is increased from size equal 16 bits to size equal 32 bits; the security level will gain 50% with loss 9% performance. This experiment has been found that key size equal 32 bit is the best optimal choice.

In experiment 4.5.3, there are different sizes of data inside each key size. Table 4.11 and figure 4.11 shows the different data sizes with different key sizes. It has been computed the average time of each size of data (3 bytes, 4 bytes, 5 bytes, and 6 bytes) in different key size (16 bits, 32 bits, and 64 bits).

Table 4.11: Results of Different Data Sizes in Different Key Sizes

EquationID	DataType	Range	DataSize	Key 16 bit	Key 32 bit	Key 64 bit	Avg. TDDsize
Degree 1	Integer	0-100	3 byte	0.31	0.32	0.49	0.37
	Integer	0-100	4 byte	0.31	0.37	0.39	0.35
	Integer	0-100	5 byte	0.31	0.408	0.38	0.36
	Integer	0-100	6 byte	0.37	0.33	0.34	0.34

Figure 4.11: Different Data Sizes in Different Key Sizes

From table 4.11 and figure 4.11, we derived table 4.12 that is used to compare between the security and the performance with different data sizes. Table 4.12 has been compared several cases to find the optimal choice.

Table 4.12: Compare between Security and Performance with Different Data Sizes

Data Size		Gain Security Level	Loss of Performance
3 byte	1. Increased the size of data from 3 to 6 byte in Key size16 bit	50%	16%
	2. Increased the key size from 16 bit to 32 bit	50%	3%
	3. Increased the key size from 16 bit to 64 bit	75%	36%
	4. Increased the key size from 32 bit to 64 bit	50%	43%
4 byte	1. Increased the key size from 16 bit to 32 bit	50%	16%
	2. Increased the key size from 16 bit to 64 bit	75%	20%
	3. Increased the key size from 32 bit to 64 bit	50%	5%
5 byte	1. Increased the key size from 16 bit to 32 bit	50%	22%
	2. Increased the key size from 16 bit to 64 bit	75%	18%
6 byte	1. Increased the key size from 32 bit to 64 bit	50%	2%

It has been computed the gain of the security by using equation (4.2) and the loss of the performance by using equation (4.2) with different data sizes as a percentage and then compared between them. From table 4.12, we noticed the following:-

1. Data size (3 byte)
    - Increase the size of data in key size equal 16 1 from (3 byte) to (6 byte) will gain 50% security level with 16% loss of performance.
    - Increase the key size from key size equal 16 bit to key size equal 32 bit will gain 50% security level with 3% loss of performance.
    - Increase the key size from key size equal 16 bit to key size equal 64 bit will gain 75% security level with 36% loss of performance.
    - Increase the key size from key size equal 32 bit to key size equal 64 bit will gain 50% security level with 34% loss of performance.
2. Data size (4 byte)
    - Increase the key size from key size equal 16 bit to key size equal 32 bit will gain 50% security level with 16% loss of performance.
    - Increase the key size from key size equal 16 bit to key size equal 64 bit will gain 75% security level with 20% loss of performance.
    - Increase the key size from key size equal 32 bit to key size equal 64 bit will gain 50% security level with 5% loss of performance.
3. Data size (5 byte)
    - Increase the key size from key size equal 16 bit to key size equal 32 bit will gain 50% security level with 22% loss of performance.
    - Increase the key size from key size equal 16 bit to key size equal 64 bit will gain 75% security level with 18% loss of performance.

4. Data size (6 byte)

    - Increase the key size from key size equal 32 bit to key size equal 64 bit will gain 50% security level with 2% loss of performance.

The results of table 4.12 conclude that when using data size (6 byte) and the key size is increased from size equal 32 bit to size equal 64 bit, the security level will gain 50% with loss 2% performance. This experiment has been found that key size equal 64 bit is the best optimal choice.

4.5.4 The effect of using different data type

This experiment has used different data types with several parameters to study the effect of these types on the performance of OPE. Three different data types were selected. These types were: integer, string, both (integer and string). The selected degrees of the polynomial function were degree 1, degree 5, degree 8, and degree 12 of the polynomial function. Also three different key sizes have been selected (16 bit, 32 bit, 64 bit), and the range of coefficient was the range 0-100, step 10. It has been computed the accumulated time for each data type to find which one of these types achieves a high performance with high security level. The average time of each data size has been computed in this experiment. From results of using different data types, we noticed the following:

- In the integer data type, there were four different data sizes: 3 bytes, 4 bytes , 5bytes, and 6 bytes. In the string and both data type, there are five different data sizes: 2 byte, 3 byte, 4 byte, 5 byte, and 6 bytes.
- In order to find the effect of string and both data type, we need to convert it to number. Using the concatenation of the ASCII for each character as a

value in the polynomial function has caused many overflow problems. Therefore; this thesis has been used the summation of ASCII for each character instead of the concatenation to prevent the overflow problems.

- Converting the string or alphanumeric data types to the ASCII code using the summation of the ASCII codes will cause the size of data to be 3 bytes. This is due to that the maximum data size was 6 characters with maximum ASCII codes summation 999 (3 bytes).
- The results showed that the data type affect the performance of OPE but it is not significant.

This thesis has been done several experiments using different data types (three types), different key sizes, and different polynomial function. It has been studied the effect of these types on the performance of OPE with several data types and several polynomial degrees. For the sake of brevity, this thesis will explain only one group of experiments (i.e. Experiment 4.5.4) and will list only the summary of other group of experiments. More samples of the experiment results are listed in Appendix D. All results for all experiments are available via the email. In the following only experiment 4.5.4 will be explained in details to study the effect of using different types with selected parameters.

Experiment 4.5.4

This experiment used the same parameters in the experiment 4.5.1, experiment 4.5.2 and experiment 4.5.3 but in this time, we used the different data types instead of different degrees of polynomial, different range of coefficient and different key sizes to study the effect of three different data types on the performance of OPE. This experiment used three different data types with selected degree1 of polynomial function, range of coefficient (range 0-100, step 10) and the key size equal 16 bit. It has been computed the accumulated time to find the optimal outcomes of the data type.

Table 4.13 and figure 4.12 shows the performance time of the three types of data. The integer data type had the lowest performance time (8.48) and the both data type had the highest performance time (11.14). As expected, the integer data type achieves a good performance and the both data type achieves the worst performance. This experiment has been found the optimal point between security level and performance.

Table 4.13: Results of Different Data Types

EquationID	KeySize	Range	DataType	PerformanceTime
Degree 1	16 bit	0 -100	Integer	8.48
Degree 1	16 bit	0 -100	String	10.55
Degree 1	16 bit	0 -100	Both	11.14

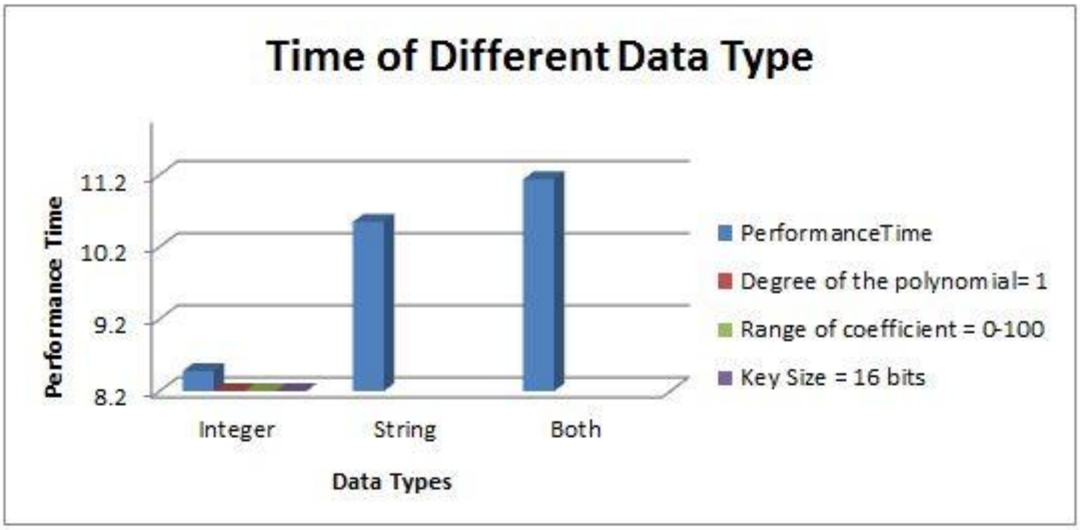


Figure 4.12: Results of Different Data Types

In experiment 4.4.4, there are three different data types. Table 4.14 and figure 4.13 shows the different data sizes in the integer data type with different degree of the polynomial function. It has been computed the average time of each size of data (3 byte, 4 byte, 5 byte, 6 byte) in different degrees of the polynomial function.

Table 4.14: Results of Different Data Sizes with Integer Data Type

DataType	Key Size	Range	DataSize	Degree 1	Degree 5	Degree 8	Degree 12	Avg. TDDSize
Integer	16 bit	0 -100	3 byte	0.29	0.35	0.37	0.37	0.34
Integer	16 bit	0 -100	4 byte	0.3	0.36	0.37	0.36	0.34
Integer	16 bit	0 -100	5 byte	0.29	0.35	0.39	0.36	0.34
Integer	16 bit	0 -100	6 byte	0.36	0.4	0.43	0.41	0.4

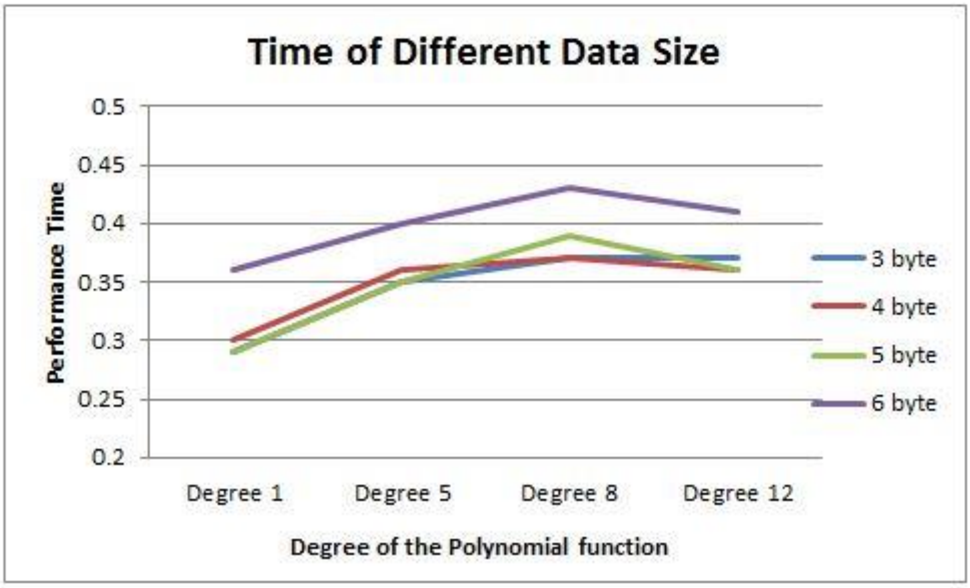


Figure 4.13: Results of Different Data sizes with Integer Data Type

From table 4.14 and figure 4.13, we derived table 4.15 that is used to compare between the security and the performance with different data sizes. Table 4.15 has been compared several cases to find the optimal outcomes.

Table 4.15: Compare between Security and Performance with Different Data Sizes

Row	Data Size		Gain Security Level	Loss of Performance
1	3 byte	1. Increased the size of data from 3 to 6 byte in Degree 1	50%	19%
2		2. Increased the degree from 1 to 5	80%	17%
3		3. Increased the degree from 1 to 8	87%	21%
4		4. Increased the degree from 1 to 12	91%	21%
5				
6	4 byte	1. Increased the degree from 1 to 5	80%	16%
7		2. Increased the degree from 1 to 8	87%	18%
8		3. Increased the degree from 1 to 12	91%	16%
9				
10	5 byte	1. Increased the degree from 1 to 5	80%	17%
11		2. Increased the degree from 1 to 8	87%	25%
12		3. Increased the degree from 1 to 12	91%	19%
13				
14	6 byte	1. Increased the degree from 1 to 5	80%	10%
15		2. Increased the degree from 1 to 8	87%	16%
16		3. Increased the degree from 1 to 12	91%	12%

For example, row1 in table 4.15 computes the gain of the security by using equation (4.2) and the gain of the performance by using equation (4.3).

$$\text{Gain Security Level} = \frac{\text{data size 6} - \text{data size 3}}{\text{data size 6}} * 100\% = 50\ \%$$

$$\text{Loss of Performance} = \frac{\text{Time of data size 6} - \text{Time of data size 3}}{\text{Time of data size 6}} * 100\%$$

$$= \frac{0.36 - 0.29}{0.36} * 100\% = 19\%$$

Other examples are listed in tables 4.15 which illustrate the results of the gain security and the gain of the performance. Also these are explained below.

1. Data size (3 byte)

- Increasing the size of data in degree 1 from 3 to 6 bytes will gaining 50% security level with 19% losing of performance.
- When the degree increased from 1 to 5, will gain 80% security level with 17% loss of performance.
- When the degree increased from 1 to 8, we will gain 87% security level with 21% loss of performance.
- When the degree increased from 1 to 12, we will gain 91% security level with 21% loss of performance.

2. Data size (4 byte)

- When the degree increased from 1 to 5, we will gain 80% security level with 16% loss of performance.
- When the degree increased from 1 to 8, we will gain 87% security level with 18% loss of performance.
- When the degree increased from 1 to 12, we will gain 91% security level with 16% loss of performance.

3. Data size (5 byte)

- When the degree increased from 1 to 5, we will gain 80% security level with 17% loss of performance.
- When the degree increased from 1 to 8, we will gain 87% security level with 25% loss of performance.
- When the degree increased from 1 to 12, we will gain 91% security level with 19% loss of performance.

4. Data size (6 byte)

- When the degree increased from 1 to 5, we will gain 80% security level with 10% loss of performance.
- When the degree increased from 1 to 8, we will gain 87% security level with 16% loss of performance.
- When the degree increased from 1 to 12, we will gain 91% security level with 12% loss of performance.

The results of table 4.15 conclude that the data sizes of the integer data type are not significant. It has been found that when using data size (6 bytes) and increasing the degree of the polynomial function from 1 to 5 will gain 80% security level with loss 10% performance. Therefore; the data size (6 bytes) is the optimal outcomes compared with others.

Table 4.16 and figure 4.14 shows the results of string data type when use different degrees of the polynomial function. The string data type has been converted to the ASCII code by using the summation number of the ASCII codes. Thus, the size of data will be 3 bytes instead of different size of data. This is due to that the maximum data size was 6 characters and the maximum summation of the ASCII code was 999 (3 bytes). The results showed that the string data type affects the performance of OPE but it is not significant.

Table 4.16: Results of Different Data Sizes in String Data Type

DataType	Key Size	Range	DataSize	Degree 1	Degree 5	Degree 8	Degree 12	Avg. TDDSize
String	16 bit	0 -100	3 byte	0.37	0.37	0.38	0.38	0.37

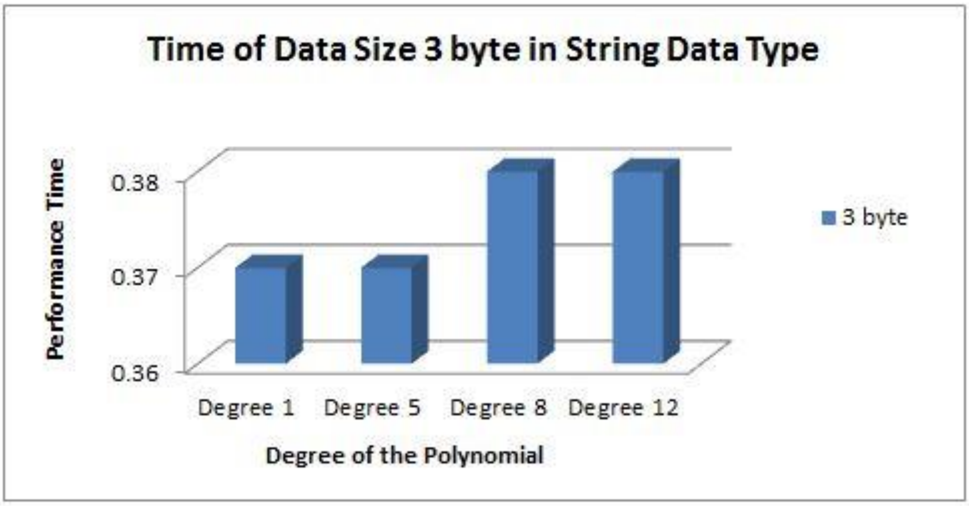


Figure 4.14: Results of Different Data Sizes in String Data Type

Table 4.17 and figure 4.15 shows the results of both data type when use different degrees of the polynomial function. The both data type has been converted to the ASCII code by using the summation number of the ASCII codes. Thus, the size of data will be 3 bytes instead of different size of data. This is due to that the maximum data size was 6 characters and the maximum summation of the ASCII code was 999 (3 bytes). The results showed that the both data type affects the performance of OPE but it is not significant.

Table 4.17: Results of Different Data Sizes in Both Data Type

DataType	Key Size	Range	DataSize	Degree 1	Degree 5	Degree 8	Degree 12	Avg. TDDSize
Both	16 bit	0 -100	3 byte	0.38	0.41	0.42	0.41	0.4

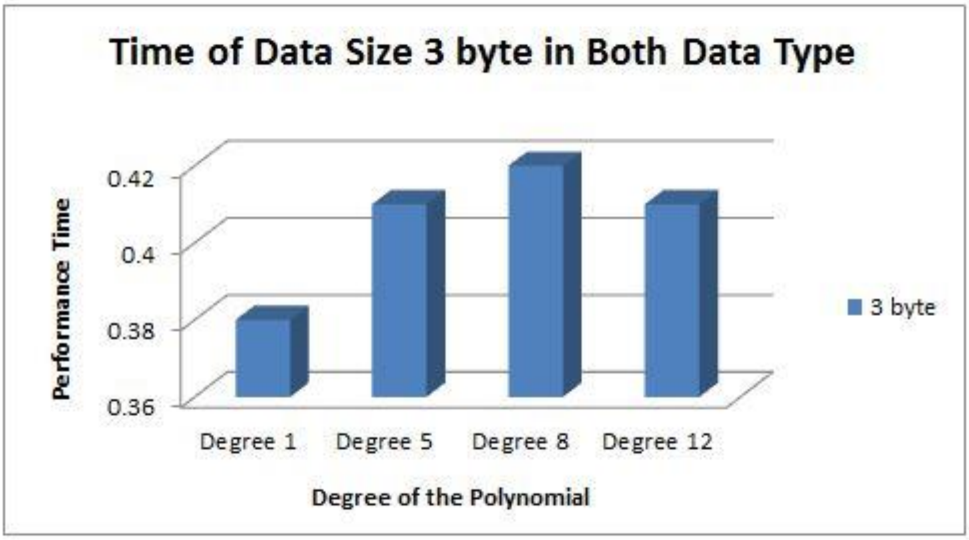


Figure 4.15: Results of Different Data Sizes in Both

Chapter Five

Conclusion and Future Work

5.1 Conclusion

This thesis concludes its finding in identifying the excellent analysis that can give us the maximum security level with minimum execution time. The finding of this research was to study the performance and security level of the polynomial function with several parameters (degree of the polynomial, range of coefficients, key sizes, data types/sizes).

This thesis contributes in identify:

- The parameters that affect the performance and security of OPEs.
- The efficiency for many cases for several parameters as a trade-off between performance and security level.
- The optimal efficiency as the minimum loss in the performance with a high gain of security.

It has been built many experiments depending on the problem of OPE to achieve the main goal of this research. This thesis accomplished many experiments to study the effect of several parameters on the performance of OPE. It was found the optimal point as a trade-off between security level and performance. The thesis has been computed the gain of security and the gain of the performance as a percentage and then compared between them. The results of the experiment will be presented as follows:

- Effect of using different degrees of polynomial function
  - This thesis found that when increased the degree of the polynomial from 1 to 4, we will gain 75% security level with 7% loss performance.
  - This thesis has been found that degree 4 of the polynomial function is the optimal outcomes.
- Effect of using different ranges of coefficients
  - This thesis has been found that increasing the range of coefficients from 1000 to 10000 will gaining 90% security level with 3% losing performance.

- This thesis has been found that range of coefficients 10000 is the optimal outcomes.

- Effect of using different key sizes

- This thesis has been found that increasing the size of key from size equal 16 bits to size equal 32 bits will gaining 50 % security level with 9% losing performance.
- This thesis has been found that key size equal 32 bits is the optimal outcomes.

- Effect of using different data types

- This thesis has been found that the data types affect the performance of OPE but it is not significant.

This thesis has been implemented and designed a software to compare and analyze the performance of OPE. It has been run the software by using all the parameters and recorded the running time for each parameter. It has been computed the time for each experiment to study the performance of OPE. The results of the experiment has been compared and analyzed depending on loss of the performance and the gain security level of OPE.

5.2 Future Work

The comparative analysis for the performance of OPE in this thesis give strong basis for a number of interesting directions for future work, which will lead to improve the security level and the performance of OPE. We plan to study the effect of using non-polynomial function on the performance. **Additionally,** using database operation such as select, insert, delete, and update to compute the re-index time and to enhance the performance and security level of OPE. **Finally**, enhance the OPE technique to reduce the leakage of information.

References

References

- Agrawal, R., Kiernan, J., Srikant, R., & Xu, Y. (2004). Order preserving encryption for numeric data. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data (pp. 563-574). ACM.
- Alam, M., & Shakil, K. A. (2015). Recent Developments in Cloud Based Systems: State of Art. arXiv preprint arXiv:1501.01323.
- Al-Hamami A. & Al-Khashab, R. (2014). Providing Availability, Performance, and Scalability By Using Cloud Database. International Journal of Advanced Computer Technology, 3(8).
- Al-Khashab, R. (2014). An Authentication Model for Cloud Computing Application.Germany.LAMBERT Academic publishing.
- Alvar, T. A., & Atan, R. (2012). Service availability and accessibility of requirements usingclustering in cloud environment. International Journal of New Computer Architectures and their Applications (IJNCAA), 2(3), 457-463.
- Bebek, G. (2002). Anti-tamper database research: Inference control techniques. Case Western Reserve University, Technical Report EECS, 433.
- Boampong, P. A., & Wahsheh, L. A. (2012). Different facets of security in the cloud. In Proceedings of the 15th Communications and Networking Simulation Symposium (p. 5). Society for Computer Simulation International.
- Boldyreva, A., Chenette, N., & O'Neill, A. (2011). Order-preserving encryption revisited: Improved security analysis and alternative solutions. In Advances in Cryptology–CRYPTO 2011 (pp. 578-595). Springer Berlin Heidelberg.
- Boldyreva, A., Chenette, N., Lee, Y., & O'neill, A. (2009). Order-preserving symmetric encryption. In Advances in Cryptology-EUROCRYPT 2009 (pp. 224-241). Springer Berlin Heidelberg.

- Dillon, T., Wu, C., & Chang, E. (2010). Cloud computing: issues and challenges. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on* (pp. 27-33).
- Gentry, C. (2009). A fully homomorphic encryption scheme (Doctoral dissertation, Stanford University) in September, 2009.
- Huth, A., & Cebula, J. (2011). The basics of cloud computing. *United States Computer*.
- Jadeja, Y., & Modi, K. (2012). Cloud computing-concepts, architecture and challenges. In Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on (pp. 877-880). IEEE.
- Liu, Q., Wang, G., & Wu, J. (2009). An efficient privacy preserving keyword search scheme in cloud computing. In Computational Science and Engineering, 2009. CSE'09. International Conference on (Vol. 2, pp. 715-720). IEEE.
- Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing—The business perspective. *Decision Support Systems*,*51*(1), 176-189.
- Mell, P., & Grance, T. (2011). The NIST definition of cloud computing, National Institute of Standards and Technology.
- Mohanty, M. K. (2013). Secure Data Storage on the Cloud using Homomorphic Encryption, Doctoral dissertation, National Institute of Technology Rourkela – 769 008, India.
- Na, S. H., Park, J. Y., & Huh, E. N. (2010). Personal cloud computing security framework. In Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific (pp. 671-675). IEEE.

- Özsoyoglu, G., Singer, D. A., & Chung, S. S. (2003). Anti-Tamper Databases: Querying Encrypted Databases. In DBSec (pp. 133-146).
- Popa, R. A., Li, F. H., & Zeldovich, N. (2013). An ideal-security protocol for order-preserving encoding. In Security and Privacy (SP), 2013 IEEE Symposium on (pp. 463-477). IEEE.
- Ramgovind, S., Eloff, M. M., & Smith, E. (2010). The management of security in cloud computing. In *Information Security for South Africa (ISSA), 2010* (pp. 1-7). IEEE.
- Ren, Y., Chen, Y., Yang, J., & Xie, B. (2014). Privacy-preserving ranked multi-keyword search leveraging polynomial function in cloud computing. In *Global Communications Conference (GLOBECOM), 2014 IEEE* (pp. 594-600). IEEE.
- Ru, J., Grundy, J., & Keung, J. (2014). Software engineering for multi-tenancy computing challenges and implications. In Proceedings of the International Workshop on Innovative Software Development Methodologies and Practices on (pp. 1-10). ACM.
- SO, K. (2011). Cloud computing security issues and challenges. International Journal of Computer Networks, 3(5).
- Stehlé, D., & Steinfeld, R. (2010). Faster fully homomorphic encryption. InAdvances in Cryptology-ASIACRYPT 2010 (pp. 377-394). Springer Berlin Heidelberg.
- Sunitha, K., & Prashanth, S. K. (2010).Enhancing Privacy in Cloud Service Provider Using Cryptographic Algorithm. IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN, 2278-0661, 12(5).
- Xiao, L., Yen, I. L., & Huynh, D. T. (2012). Extending Order Preserving Encryption for Multi-User Systems. IACR Cryptology ePrint Archive, 2012, 192.

- Yogamangalam, R., & Sriram, V. S. (2013). A Review on Security Issues in Cloud Computing. Journal of Artificial Intelligence, 6(1), 1-7.
- Youssef, A. E., & Alageel, M. (2012). A Framework for A Framework for Secure Cloud ure Cloud ure Cloud Computing Computing Computing.
- Zhang, S., Zhang, S., Chen, X., & Huo, X. (2010). Cloud computing research and development trend. In *Future Networks, 2010. ICFN'10. Second International Conference on* (pp. 93-97).

Appendix

The results of the experiment are very huge. This thesis has been presented a sample of the results. For full results, you can contact the author via the email [2]

A. Results of using different degree of polynomial function

Original Index	EquationID	InputDataX	DataSize	DataType	Range of Coefficent	Step	FunctionValue	KeySize	KeyValue	PerformanceTime	TimePerRange	ReIndex
1	1	428789	6	Integer	0,0	10	0	16	880	0.105647	0.105647	881
2	1	428789	6	Integer	10,10	10	4287900	16	880	0.184384	0.184384	4288782
3	1	428789	6	Integer	20,20	10	8575800	16	880	0.244858	0.244858	8576683
4	1	428789	6	Integer	30,30	10	12863700	16	880	0.308731	0.308731	12864584
5	1	428789	6	Integer	40,40	10	17151600	16	880	0.368352	0.368352	17152485
6	1	428789	6	Integer	50,50	10	21439500	16	880	0.427156	0.427156	21440386
7	1	428789	6	Integer	60,60	10	25727400	16	880	0.485458	0.485458	25728287
8	1	428789	6	Integer	70,70	10	30015300	16	880	0.544238	0.544238	30016188
9	1	428789	6	Integer	80,80	10	34303200	16	880	0.602629	0.602629	34304089
10	1	428789	6	Integer	90,90	10	38591100	16	880	0.661811	0.661811	38591990
11	1	428789	6	Integer	100,100	10	42879000	16	880	0.924212	0.924212	42879891
12	1	46559	5	Integer	0,0	10	0	16	880	1.07542	0.0558537	892
13	1	46559	5	Integer	10,10	10	465600	16	880	1.1353	0.115735	466493
14	1	46559	5	Integer	20,20	10	931200	16	880	1.19349	0.173929	932094
15	1	46559	5	Integer	30,30	10	1396800	16	880	1.25202	0.232455	1397695
16	1	46559	5	Integer	40,40	10	1862400	16	880	1.31056	0.290998	1863296
17	1	46559	5	Integer	50,50	10	2328000	16	880	1.37158	0.352018	2328897
18	1	46559	5	Integer	60,60	10	2793600	16	880	1.43073	0.41117	2794498
19	1	46559	5	Integer	70,70	10	3259200	16	880	1.48945	0.46989	3260099
20	1	46559	5	Integer	80,80	10	3724800	16	880	1.54907	0.529502	3725700
21	1	46559	5	Integer	90,90	10	4190400	16	880	1.60723	0.587662	4191301
22	1	46559	5	Integer	100,100	10	4656000	16	880	1.66666	0.647097	4656902
23	1	465	3	Integer	0,0	10	0	16	880	1.74278	0.0411772	903
24	1	465	3	Integer	10,10	10	4660	16	880	1.8023	0.100696	5564
25	1	465	3	Integer	20,20	10	9320	16	880	1.86139	0.159786	10225
26	1	465	3	Integer	30,30	10	13980	16	880	1.92065	0.219049	14886

[2] Email address: Hadeel_alkazaz@yahoo.com

27	1	465	3	Integer	40,40	10	18640	16	880	1.98053	0.278925	19547
28	1	465	3	Integer	50,50	10	23300	16	880	2.03942	0.337813	24208
29	1	465	3	Integer	60,60	10	27960	16	880	2.09978	0.398172	28869
30	1	465	3	Integer	70,70	10	32620	16	880	2.16195	0.460346	33530
31	1	465	3	Integer	80,80	10	37280	16	880	2.2187	0.517098	38191
32	1	465	3	Integer	90,90	10	41940	16	880	2.27758	0.575978	42852
33	1	465	3	Integer	100,100	10	46600	16	880	2.33849	0.636883	47513
34	1	2344	4	Integer	0,0	10	0	16	880	2.4174	0.0430563	914
35	1	2344	4	Integer	10,10	10	23450	16	880	2.47592	0.101567	24365
36	1	2344	4	Integer	20,20	10	46900	16	880	2.53611	0.161767	47816
37	1	2344	4	Integer	30,30	10	70350	16	880	2.59431	0.219963	71267
38	1	2344	4	Integer	40,40	10	93800	16	880	2.65359	0.279242	94718
39	1	2344	4	Integer	50,50	10	117250	16	880	2.7129	0.338554	118169
40	1	2344	4	Integer	60,60	10	140700	16	880	2.77159	0.397241	141620
41	1	2344	4	Integer	70,70	10	164150	16	880	2.83173	0.45738	165071
42	1	2344	4	Integer	80,80	10	187600	16	880	2.89116	0.516807	188522
43	1	2344	4	Integer	90,90	10	211050	16	880	2.95068	0.576335	211973
44	1	2344	4	Integer	100,100	10	234500	16	880	3.01015	0.635803	235424
45	1	452	3	Integer	0,0	10	0	16	880	3.08722	0.0420123	925
46	1	452	3	Integer	10,10	10	4530	16	880	3.1462	0.101	5456
47	1	452	3	Integer	20,20	10	9060	16	880	3.20566	0.160458	9987
48	1	452	3	Integer	30,30	10	13590	16	880	3.26566	0.220461	14518
49	1	452	3	Integer	40,40	10	18120	16	880	3.32606	0.280858	19049
50	1	452	3	Integer	50,50	10	22650	16	880	3.4051	0.359898	23580
51	1	452	3	Integer	60,60	10	27180	16	880	3.48256	0.437356	28111
52	1	452	3	Integer	70,70	10	31710	16	880	3.54294	0.497737	32642
53	1	452	3	Integer	80,80	10	36240	16	880	3.60218	0.556975	37173
54	1	452	3	Integer	90,90	10	40770	16	880	3.66146	0.616257	41704
55	1	452	3	Integer	100,100	10	45300	16	880	3.72139	0.676183	46235
56	1	3453	4	Integer	0,0	10	0	16	880	3.79945	0.0420743	936
57	1	3453	4	Integer	10,10	10	34540	16	880	3.85771	0.10033	35477
58	1	3453	4	Integer	20,20	10	69080	16	880	3.91777	0.160388	70018
59	1	3453	4	Integer	30,30	10	103620	16	880	3.97655	0.219173	104559
60	1	3453	4	Integer	40,40	10	138160	16	880	4.03759	0.280214	139100
61	1	3453	4	Integer	50,50	10	172700	16	880	4.0976	0.340225	173641
62	1	3453	4	Integer	60,60	10	207240	16	880	4.15753	0.400156	208182
63	1	3453	4	Integer	70,70	10	241780	16	880	4.22752	0.470143	242723
64	1	3453	4	Integer	80,80	10	276320	16	880	4.28778	0.530399	277264
65	1	3453	4	Integer	90,90	10	310860	16	880	4.34594	0.588562	311805
66	1	3453	4	Integer	100,100	10	345400	16	880	4.40786	0.650482	346346
67	1	15964	5	Integer	0,0	10	0	16	880	4.48435	0.0410672	947
68	1	15964	5	Integer	10,10	10	159650	16	880	4.54487	0.101583	160598
69	1	15964	5	Integer	20,20	10	319300	16	880	4.60479	0.16151	320249
70	1	15964	5	Integer	30,30	10	478950	16	880	4.66504	0.221761	479900
71	1	15964	5	Integer	40,40	10	638600	16	880	4.72473	0.281446	639551
72	1	15964	5	Integer	50,50	10	798250	16	880	4.78492	0.341635	799202
73	1	15964	5	Integer	60,60	10	957900	16	880	4.84341	0.400132	958853
74	1	15964	5	Integer	70,70	10	1117550	16	880	4.90375	0.460464	1118504
75	1	15964	5	Integer	80,80	10	1277200	16	880	4.96342	0.520133	1278155
76	1	15964	5	Integer	90,90	10	1436850	16	880	5.02457	0.581289	1437806
77	1	15964	5	Integer	100,100	10	1596500	16	880	5.08562	0.642336	1597457
78	1	465	3	Integer	0,0	10	0	16	880	5.16312	0.0414929	958
79	1	465	3	Integer	10,10	10	4660	16	880	5.22246	0.100834	5619
80	1	465	3	Integer	20,20	10	9320	16	880	5.28254	0.160912	10280

81	1	465	3	Integer	30,30	10	13980	16	880	5.34204	0.220418	14941
82	1	465	3	Integer	40,40	10	18640	16	880	5.40251	0.28089	19602
83	1	465	3	Integer	50,50	10	23300	16	880	5.46294	0.341314	24263
84	1	465	3	Integer	60,60	10	27960	16	880	5.52403	0.402401	28924
85	1	465	3	Integer	70,70	10	32620	16	880	5.58312	0.461497	33585
86	1	465	3	Integer	80,80	10	37280	16	880	5.64231	0.520686	38246
87	1	465	3	Integer	90,90	10	41940	16	880	5.70249	0.580864	42907
88	1	465	3	Integer	100,100	10	46600	16	880	5.76218	0.640557	47568
89	1	2344	4	Integer	0,0	10	0	16	880	5.84029	0.0413603	969
90	1	2344	4	Integer	10,10	10	23450	16	880	5.90036	0.101431	24420
91	1	2344	4	Integer	20,20	10	46900	16	880	5.96118	0.162251	47871
92	1	2344	4	Integer	30,30	10	70350	16	880	6.02105	0.222122	71322
93	1	2344	4	Integer	40,40	10	93800	16	880	6.08415	0.285219	94773
94	1	2344	4	Integer	50,50	10	117250	16	880	6.14457	0.345648	118224
95	1	2344	4	Integer	60,60	10	140700	16	880	6.20479	0.40586	141675
96	1	2344	4	Integer	70,70	10	164150	16	880	6.26423	0.465302	165126
97	1	2344	4	Integer	80,80	10	187600	16	880	6.32571	0.526788	188577
98	1	2344	4	Integer	90,90	10	211050	16	880	6.38595	0.587022	212028
99	1	2344	4	Integer	100,100	10	234500	16	880	6.4466	0.64767	235479
100	1	452	3	Integer	0,0	10	0	16	880	6.5235	0.0410372	980
101	1	452	3	Integer	10,10	10	4530	16	880	6.58336	0.100899	5511
102	1	452	3	Integer	20,20	10	9060	16	880	6.644	0.161536	10042
103	1	452	3	Integer	30,30	10	13590	16	880	6.70444	0.221975	14573
104	1	452	3	Integer	40,40	10	18120	16	880	6.76424	0.281778	19104
105	1	452	3	Integer	50,50	10	22650	16	880	6.825	0.342538	23635
106	1	452	3	Integer	60,60	10	27180	16	880	6.88466	0.402195	28166
107	1	452	3	Integer	70,70	10	31710	16	880	6.94388	0.461412	32697
108	1	452	3	Integer	80,80	10	36240	16	880	7.00291	0.520445	37228
109	1	452	3	Integer	90,90	10	40770	16	880	7.06236	0.579893	41759
110	1	452	3	Integer	100,100	10	45300	16	880	7.12721	0.644743	46290
111	1	5467	4	Integer	0,0	10	0	16	880	7.20926	0.0450305	991
112	1	5467	4	Integer	10,10	10	54680	16	880	7.26729	0.103065	55672
113	1	5467	4	Integer	20,20	10	109360	16	880	7.32825	0.164017	110353
114	1	5467	4	Integer	30,30	10	164040	16	880	7.3888	0.22457	165034
115	1	5467	4	Integer	40,40	10	218720	16	880	7.44697	0.28274	219715
116	1	5467	4	Integer	50,50	10	273400	16	880	7.50637	0.342146	274396
117	1	5467	4	Integer	60,60	10	328080	16	880	7.56625	0.402023	329077
118	1	5467	4	Integer	70,70	10	382760	16	880	7.62726	0.463027	383758
119	1	5467	4	Integer	80,80	10	437440	16	880	7.68655	0.522323	438439
120	1	5467	4	Integer	90,90	10	492120	16	880	7.74958	0.585352	493120
121	1	5467	4	Integer	100,100	10	546800	16	880	7.80981	0.645578	547801
122	1	349957	6	Integer	0,0	10	0	16	880	7.88761	0.0420718	1002
123	1	349957	6	Integer	10,10	10	3499580	16	880	7.94682	0.10128	3500583
124	1	349957	6	Integer	20,20	10	6999160	16	880	8.00719	0.161648	7000164
125	1	349957	6	Integer	30,30	10	10498740	16	880	8.06657	0.221028	10499745
126	1	349957	6	Integer	40,40	10	13998320	16	880	8.12841	0.28287	13999326
127	1	349957	6	Integer	50,50	10	17497900	16	880	8.19031	0.344763	17498907
128	1	349957	6	Integer	60,60	10	20997480	16	880	8.25035	0.404812	20998488
129	1	349957	6	Integer	70,70	10	24497060	16	880	8.30927	0.463729	24498069
130	1	349957	6	Integer	80,80	10	27996640	16	880	8.36911	0.523564	27997650
131	1	349957	6	Integer	90,90	10	31496220	16	880	8.43002	0.584481	31497231
132	1	349957	6	Integer	100,100	10	34995800	16	880	8.51307	0.66753	34996812
133	1	3355	4	Integer	0,0	10	0	16	880	8.59222	0.0421666	1013
134	1	3355	4	Integer	10,10	10	33560	16	880	8.65274	0.102689	34574

135	1	3355	4	Integer	20,20	10	67120	16	880	8.71225	0.162194	68135
136	1	3355	4	Integer	30,30	10	100680	16	880	8.77154	0.221485	101696
137	1	3355	4	Integer	40,40	10	134240	16	880	8.8317	0.28165	135257
138	1	3355	4	Integer	50,50	10	167800	16	880	8.89132	0.341271	168818
139	1	3355	4	Integer	60,60	10	201360	16	880	8.95097	0.400919	202379
140	1	3355	4	Integer	70,70	10	234920	16	880	9.01072	0.460663	235940
141	1	3355	4	Integer	80,80	10	268480	16	880	9.07167	0.52162	269501
142	1	3355	4	Integer	90,90	10	302040	16	880	9.13225	0.582194	303062
143	1	3355	4	Integer	100,100	10	335600	16	880	9.19222	0.642163	336623
144	1	5176	4	Integer	0,0	10	0	16	880	9.27175	0.0430217	1024
145	1	5176	4	Integer	10,10	10	51770	16	880	9.33211	0.103383	52795
146	1	5176	4	Integer	20,20	10	103540	16	880	9.39189	0.163165	104566
147	1	5176	4	Integer	30,30	10	155310	16	880	9.45294	0.224214	156337
148	1	5176	4	Integer	40,40	10	207080	16	880	9.51673	0.288002	208108
149	1	5176	4	Integer	50,50	10	258850	16	880	9.57818	0.349457	259879
150	1	5176	4	Integer	60,60	10	310620	16	880	9.63898	0.41025	311650
151	1	5176	4	Integer	70,70	10	362390	16	880	9.69981	0.471088	363421
152	1	5176	4	Integer	80,80	10	414160	16	880	9.75968	0.530951	415192
153	1	5176	4	Integer	90,90	10	465930	16	880	9.82094	0.592218	466963
154	1	5176	4	Integer	100,100	10	517700	16	880	9.88018	0.651459	518734
155	2	428789	6	Integer	0,0,0	10	0	16	880	0.0441076	0.0441076	1035
156	2	428789	6	Integer	10,10,10	10	1838604353110	16	880	0.105054	0.043103	1838604354146
157	2	428789	6	Integer	20,20,20	10	3677208706220	16	880	0.166519	0.0419876	3677208707257
158	2	428789	6	Integer	30,30,30	10	5515813059330	16	880	0.227901	0.0435533	5515813060368
159	2	428789	6	Integer	40,40,40	10	7354417412440	16	880	0.289236	0.0430312	7354417413479
160	2	428789	6	Integer	50,50,50	10	9193021765550	16	880	0.354612	0.0475848	9193021766590
161	2	428789	6	Integer	60,60,60	10	11031626118660	16	880	0.415803	0.0426913	11031626119701

162	2	428789	6	Integer	70,70,70	10	12870230471770	16	880	0.476033	0.0424318	12870230472812
163	2	428789	6	Integer	80,80,80	10	14708834824880	16	880	0.539915	0.0446285	14708834825923
164	2	428789	6	Integer	90,90,90	10	16547439177990	16	880	0.598797	0.0420065	16547439179034
165	2	428789	6	Integer	100,100,100	10	18386043531100	16	880	0.661031	0.0429154	18386043532145
166	2	46559	5	Integer	0,0,0	10	0	16	880	0.740338	0.0425661	1046
167	2	46559	5	Integer	10,10,10	10	21677870410	16	880	0.800817	0.0420545	21677871457
168	2	46559	5	Integer	20,20,20	10	43355740820	16	880	0.861121	0.0417229	43355741868
169	2	46559	5	Integer	30,30,30	10	65033611230	16	880	0.923402	0.0438333	65033612279
170	2	46559	5	Integer	40,40,40	10	86711481640	16	880	0.984989	0.0434577	86711482690
171	2	46559	5	Integer	50,50,50	10	108389352050	16	880	1.04575	0.0421009	108389353101
172	2	46559	5	Integer	60,60,60	10	130067222460	16	880	1.10715	0.0429367	130067223512
173	2	46559	5	Integer	70,70,70	10	151745092870	16	880	1.1678	0.0428161	151745093923
174	2	46559	5	Integer	80,80,80	10	173422963280	16	880	1.22875	0.0423033	173422964334
175	2	46559	5	Integer	90,90,90	10	195100833690	16	880	1.29122	0.0440004	195100834745
176	2	46559	5	Integer	100,100,100	10	216778704100	16	880	1.35451	0.0452726	216778705156
177	2	465	3	Integer	0,0,0	10	0	16	880	1.4363	0.0459135	1057
178	2	465	3	Integer	10,10,10	10	2166910	16	880	1.49797	0.0437164	2167968
179	2	465	3	Integer	20,20,20	10	4333820	16	880	1.55858	0.0431428	4334879
180	2	465	3	Integer	30,30,30	10	6500730	16	880	1.62017	0.0431166	6501790
181	2	465	3	Integer	40,40,40	10	8667640	16	880	1.68039	0.0429154	8668701
182	2	465	3	Integer	50,50,50	10	10834550	16	880	1.74334	0.0434906	10835612
183	2	465	3	Integer	60,60,60	10	13001460	16	880	1.80406	0.042504	13002523
184	2	465	3	Integer	70,70,70	10	15168370	16	880	1.86656	0.0430919	15169434
185	2	465	3	Integer	80,80,80	10	17335280	16	880	1.9279	0.0430677	17336345
186	2	465	3	Integer	90,90,90	10	19502190	16	880	1.98912	0.0423267	19503256
187	2	465	3	Integer	100,100,100	10	21669100	16	880	2.04958	0.0426929	21670167
188	2	2344	4	Integer	0,0,0	10	0	16	880	2.12973	0.0430529	1068
189	2	2344	4	Integer	10,10,10	10	54966810	16	880	2.19429	0.0457189	54967879
190	2	2344	4	Integer	20,20,20	10	109933620	16	880	2.25771	0.0450321	109934690
191	2	2344	4	Integer	30,30,30	10	164900430	16	880	2.31975	0.0440309	164901501
192	2	2344	4	Integer	40,40,40	10	219867240	16	880	2.38517	0.0464074	219868312
193	2	2344	4	Integer	50,50,50	10	274834050	16	880	2.44586	0.0423809	274835123
194	2	2344	4	Integer	60,60,60	10	329800860	16	880	2.50716	0.0426342	329801934
195	2	2344	4	Integer	70,70,70	10	384767670	16	880	2.56812	0.0426227	384768745
196	2	2344	4	Integer	80,80,80	10	439734480	16	880	2.62916	0.0430267	439735556
197	2	2344	4	Integer	90,90,90	10	494701290	16	880	2.69086	0.0437434	494702367
198	2	2344	4	Integer	100,100,100	10	549668100	16	880	2.75222	0.0435817	549669178
199	2	452	3	Integer	0,0,0	10	0	16	880	2.83211	0.0429639	1079
200	2	452	3	Integer	10,10,10	10	2047570	16	880	2.89289	0.043128	2048650
201	2	452	3	Integer	20,20,20	10	4095140	16	880	2.95303	0.0424084	4096221
202	2	452	3	Integer	30,30,30	10	6142710	16	880	3.01407	0.0440304	6143792
203	2	452	3	Integer	40,40,40	10	8190280	16	880	3.07554	0.0436273	8191363
204	2	452	3	Integer	50,50,50	10	10237850	16	880	3.13663	0.0436876	10238934
205	2	452	3	Integer	60,60,60	10	12285420	16	880	3.19853	0.0432258	12286505
206	2	452	3	Integer	70,70,70	10	14332990	16	880	3.25901	0.0423452	14334076
207	2	452	3	Integer	80,80,80	10	16380560	16	880	3.32035	0.043213	16381647
208	2	452	3	Integer	90,90,90	10	18428130	16	880	3.38022	0.0422503	18429218
209	2	452	3	Integer	100,100,100	10	20475700	16	880	3.44208	0.0430993	20476789
210	2	3453	4	Integer	0,0,0	10	0	16	880	3.51887	0.0420291	1090
211	2	3453	4	Integer	10,10,10	10	119266630	16	880	3.57924	0.0420316	119267721
212	2	3453	4	Integer	20,20,20	10	238533260	16	880	3.65315	0.0440111	238534352
213	2	3453	4	Integer	30,30,30	10	357799890	16	880	3.71481	0.0430008	357800983
214	2	3453	4	Integer	40,40,40	10	477066520	16	880	3.77867	0.0439007	477067614
215	2	3453	4	Integer	50,50,50	10	596333150	16	880	3.83923	0.0427734	596334245

216	2	3453	4	Integer	60,60,60	10	715599780	16	880	3.90062	0.043766	715600876
217	2	3453	4	Integer	70,70,70	10	834866410	16	880	3.96219	0.0441261	834867507
218	2	3453	4	Integer	80,80,80	10	954133040	16	880	4.02594	0.0461676	954134138
219	2	3453	4	Integer	90,90,90	10	1073399670	16	880	4.08848	0.0436872	1073400769
220	2	3453	4	Integer	100,100,100	10	1192666300	16	880	4.14999	0.0434179	1192667400
221	2	15964	5	Integer	0,0,0	10	0	16	880	4.22844	0.0420311	1101
222	2	15964	5	Integer	10,10,10	10	2548652610	16	880	4.28839	0.0420311	2548653712
223	2	15964	5	Integer	20,20,20	10	5097305220	16	880	4.35082	0.0433744	5097306323
224	2	15964	5	Integer	30,30,30	10	7645957830	16	880	4.4122	0.0430091	7645958934
225	2	15964	5	Integer	40,40,40	10	10194610440	16	880	4.47604	0.0446096	10194611545
226	2	15964	5	Integer	50,50,50	10	12743263050	16	880	4.53694	0.0431272	12743264156
227	2	15964	5	Integer	60,60,60	10	15291915660	16	880	4.59925	0.0440276	15291916767
228	2	15964	5	Integer	70,70,70	10	17840568270	16	880	4.66044	0.0426551	17840569378
229	2	15964	5	Integer	80,80,80	10	20389220880	16	880	4.77113	0.0456927	20389221989
230	2	15964	5	Integer	90,90,90	10	22937873490	16	880	4.83357	0.0439417	22937874600
231	2	15964	5	Integer	100,100,100	10	25486526100	16	880	4.89543	0.0439515	25486527211
232	2	465	3	Integer	0,0,0	10	0	16	880	4.97549	0.0437119	1112
233	2	465	3	Integer	10,10,10	10	2166910	16	880	5.03822	0.0444253	2168023
234	2	465	3	Integer	20,20,20	10	4333820	16	880	5.10065	0.0442053	4334934
235	2	465	3	Integer	30,30,30	10	6500730	16	880	5.16352	0.0447189	6501845
236	2	465	3	Integer	40,40,40	10	8667640	16	880	5.22472	0.0432249	8668756
237	2	465	3	Integer	50,50,50	10	10834550	16	880	5.28602	0.0430714	10835667
238	2	465	3	Integer	60,60,60	10	13001460	16	880	5.34753	0.0439766	13002578
239	2	465	3	Integer	70,70,70	10	15168370	16	880	5.40933	0.0439766	15169489
240	2	465	3	Integer	80,80,80	10	17335280	16	880	5.47374	0.0448991	17336400
241	2	465	3	Integer	90,90,90	10	19502190	16	880	5.53561	0.0438563	19503311
242	2	465	3	Integer	100,100,100	10	21669100	16	880	5.59959	0.0450744	21670222
243	2	2344	4	Integer	0,0,0	10	0	16	880	5.68061	0.0450337	1123
244	2	2344	4	Integer	10,10,10	10	54966810	16	880	5.74206	0.0420307	54967934
245	2	2344	4	Integer	20,20,20	10	109933620	16	880	5.86834	0.0436346	109934745
246	2	2344	4	Integer	30,30,30	10	164900430	16	880	5.92872	0.0431239	164901556
247	2	2344	4	Integer	40,40,40	10	219867240	16	880	5.99052	0.044032	219868367
248	2	2344	4	Integer	50,50,50	10	274834050	16	880	6.05142	0.0428132	274835178
249	2	2344	4	Integer	60,60,60	10	329800860	16	880	6.11349	0.0451294	329801989
250	2	2344	4	Integer	70,70,70	10	384767670	16	880	6.17303	0.0417335	384768800
251	2	2344	4	Integer	80,80,80	10	439734480	16	880	6.23503	0.0437521	439735611
252	2	2344	4	Integer	90,90,90	10	494701290	16	880	6.29521	0.0428686	494702422
253	2	2344	4	Integer	100,100,100	10	549668100	16	880	6.35589	0.0427849	549669233
254	2	452	3	Integer	0,0,0	10	0	16	880	6.43766	0.0460239	1134
255	2	452	3	Integer	10,10,10	10	2047570	16	880	6.50139	0.0454911	2048705
256	2	452	3	Integer	20,20,20	10	4095140	16	880	6.56288	0.0438765	4096276
257	2	452	3	Integer	30,30,30	10	6142710	16	880	6.62436	0.0436732	6143847
258	2	452	3	Integer	40,40,40	10	8190280	16	880	6.68476	0.0427475	8191418
259	2	452	3	Integer	50,50,50	10	10237850	16	880	6.74685	0.0440041	10238989
260	2	452	3	Integer	60,60,60	10	12285420	16	880	6.80763	0.0430251	12286560
261	2	452	3	Integer	70,70,70	10	14332990	16	880	6.87017	0.0445563	14334131
262	2	452	3	Integer	80,80,80	10	16380560	16	880	7.00947	0.0613768	16381702
263	2	452	3	Integer	90,90,90	10	18428130	16	880	7.07496	0.0471611	18429273
264	2	452	3	Integer	100,100,100	10	20475700	16	880	7.13615	0.042713	20476844
265	2	5467	4	Integer	0,0,0	10	0	16	880	7.21518	0.0437906	1145
266	2	5467	4	Integer	10,10,10	10	298935570	16	880	7.27791	0.0446052	298936716
267	2	5467	4	Integer	20,20,20	10	597871140	16	880	7.34395	0.0470294	597872287
268	2	5467	4	Integer	30,30,30	10	896806710	16	880	7.40299	0.0420275	896807858
269	2	5467	4	Integer	40,40,40	10	1195742280	16	880	7.46426	0.0430316	1195743429

270	2	5467	4	Integer	50,50,50	10	1494677850	16	880	7.52686	0.0440304	1494679000
271	2	5467	4	Integer	60,60,60	10	1793613420	16	880	7.58769	0.0429031	1793614571
272	2	5467	4	Integer	70,70,70	10	2092548990	16	880	7.64953	0.0434466	2092550142
273	2	5467	4	Integer	80,80,80	10	2391484560	16	880	7.71036	0.0429589	2391485713
274	2	5467	4	Integer	90,90,90	10	2690420130	16	880	7.77241	0.0440127	2690421284
275	2	5467	4	Integer	100,100,100	10	2989355700	16	880	7.83596	0.04545	2989356855
276	2	349957	6	Integer	0,0,0	10	0	16	880	7.91697	0.0438371	1156
277	2	349957	6	Integer	10,10,10	10	1224702518070	16	880	7.97908	0.0441293	1224702519227
278	2	349957	6	Integer	20,20,20	10	2449405036140	16	880	8.0578	0.0604015	2449405037298
279	2	349957	6	Integer	30,30,30	10	3674107554210	16	880	8.15587	0.0530349	3674107555369
280	2	349957	6	Integer	40,40,40	10	4898810072280	16	880	8.21948	0.0450312	4898810073440
281	2	349957	6	Integer	50,50,50	10	6123512590350	16	880	8.27725	0.0424593	6123512591511
282	2	349957	6	Integer	60,60,60	10	7348215108420	16	880	8.34025	0.0434413	7348215109582
283	2	349957	6	Integer	70,70,70	10	8572917626490	16	880	8.40192	0.0434758	8572917627653
284	2	349957	6	Integer	80,80,80	10	9797620144560	16	880	8.46558	0.0440907	9797620145724
285	2	349957	6	Integer	90,90,90	10	11022322662630	16	880	8.52972	0.0460309	11022322663795
286	2	349957	6	Integer	100,100,100	10	12247025180700	16	880	8.59335	0.0466238	12247025181866
287	2	3355	4	Integer	0,0,0	10	0	16	880	8.71884	0.0596613	1167
288	2	3355	4	Integer	10,10,10	10	112593810	16	880	8.78157	0.0447579	112594978
289	2	3355	4	Integer	20,20,20	10	225187620	16	880	8.84291	0.0434712	225188789
290	2	3355	4	Integer	30,30,30	10	337781430	16	880	8.90549	0.0442985	337782600
291	2	3355	4	Integer	40,40,40	10	450375240	16	880	8.96752	0.0438415	450376411
292	2	3355	4	Integer	50,50,50	10	562969050	16	880	9.03031	0.0441733	562970222
293	2	3355	4	Integer	60,60,60	10	675562860	16	880	9.0925	0.0433366	675564033
294	2	3355	4	Integer	70,70,70	10	788156670	16	880	9.15405	0.0436396	788157844
295	2	3355	4	Integer	80,80,80	10	900750480	16	880	9.29877	0.0849473	900751655
296	2	3355	4	Integer	90,90,90	10	1013344290	16	880	9.36982	0.0520345	1013345466

297	2	3355	4	Integer	100,100,100	10	1125938100	16	880	9.43767	0.0470314	1125939277
298	2	5176	4	Integer	0,0,0	10	0	16	880	9.51902	0.044218	1178
299	2	5176	4	Integer	10,10,10	10	267961530	16	880	9.58164	0.0440243	267962709
300	2	5176	4	Integer	20,20,20	10	535923060	16	880	9.64542	0.0436104	535924240
301	2	5176	4	Integer	30,30,30	10	803884590	16	880	9.70731	0.0435135	803885771
302	2	5176	4	Integer	40,40,40	10	1071846120	16	880	9.77003	0.0437118	1071847302
303	2	5176	4	Integer	50,50,50	10	1339807650	16	880	9.8321	0.0435841	1339808833
304	2	5176	4	Integer	60,60,60	10	1607769180	16	880	9.89626	0.0447899	1607770364
305	2	5176	4	Integer	70,70,70	10	1875730710	16	880	9.95832	0.0448317	1875731895
306	2	5176	4	Integer	80,80,80	10	2143692240	16	880	10.0218	0.0439754	2143693426
307	2	5176	4	Integer	90,90,90	10	2411653770	16	880	10.086	0.046019	2411654957
308	2	5176	4	Integer	100,100,100	10	2679615300	16	880	10.149	0.0440349	2679616488
309	3	428789	6	Integer	0,0,0,0	10	0	16	880	0.0452131	0.0452131	1189
310	3	428789	6	Integer	10,10,10,10	10	788373321965684	16	880	0.108059	0.108059	119788373321965
311	3	428789	6	Integer	20,20,20,20	10	157674664393137	16	880	0.235609	0.235609	119257674664393
312	3	428789	6	Integer	30,30,30,30	10	236511996589705	16	880	0.312248	0.312248	119436511996589
313	3	428789	6	Integer	40,40,40,40	10	315349328786273	16	880	0.383295	0.383295	119615349328786
314	3	428789	6	Integer	50,50,50,50	10	394186660982842	16	880	0.45134	0.452341	119794186660982
315	3	428789	6	Integer	60,60,60,60	10	47302399317941	16	880	0.515674	0.515674	119973023993179
316	3	428789	6	Integer	70,70,70,70	10	551861325375979	16	880	0.587722	0.587722	120151861325375
317	3	428789	6	Integer	80,80,80,80	10	630698657572547	16	880	0.657768	0.657768	120330698657572
318	3	428789	6	Integer	90,90,90,90	10	709535989769116	16	880	0.723763	0.723763	120509535989769
319	3	428789	6	Integer	100,100,100,100	10	788373321965684	16	880	0.792504	0.792504	120688373321965
320	3	46559	5	Integer	0,0,0,0	10	0	16	880	0.887566	0.0520341	1200
321	3	46559	5	Integer	10,10,10,10	10	10092999684192	16	880	0.951608	0.116076	120200929996841
322	3	46559	5	Integer	20,20,20,20	10	20185999368384	16	880	1.01497	0.179435	120401859993683
323	3	46559	5	Integer	30,30,30,30	10	30278999052576	16	880	1.09002	0.254485	120602789990525

B. Results of using Different Ranges of Coefficients

Original Index	EquationID	InputDataX	DataSize	DataType	Range of Coefficent	Step	FunctionValue	KeySize	KeyValue	PerformanceTime	TimePerRange	ReIndex
1	1	428789	6	Integer	0,0	10	0	16	656	0.109072	0.109072	657
2	1	428789	6	Integer	10,10	10	4287900	16	656	0.202134	0.202134	4288558
3	1	428789	6	Integer	20,20	10	8575800	16	656	0.260172	0.260172	8576459
4	1	428789	6	Integer	30,30	10	12863700	16	656	0.31721	0.31721	12864360
5	1	428789	6	Integer	40,40	10	17151600	16	656	0.37725	0.37725	17152261
6	1	428789	6	Integer	50,50	10	21439500	16	656	0.438291	0.438291	21440162
7	1	428789	6	Integer	60,60	10	25727400	16	656	0.531353	0.531353	25728063
8	1	428789	6	Integer	70,70	10	30015300	16	656	0.604401	0.604401	30015964
9	1	428789	6	Integer	80,80	10	34303200	16	656	0.679451	0.679451	34303865
10	1	428789	6	Integer	90,90	10	38591100	16	656	0.751499	0.751499	38591766
11	1	428789	6	Integer	100,100	10	42879000	16	656	0.821545	0.821545	42879667
12	1	46559	5	Integer	0,0	10	0	16	656	0.957636	0.0410278	668
13	1	46559	5	Integer	10,10	10	465600	16	656	1.01868	0.102068	466269
14	1	46559	5	Integer	20,20	10	931200	16	656	1.08072	0.16411	931870
15	1	46559	5	Integer	30,30	10	1396800	16	656	1.13976	0.223148	1397471
16	1	46559	5	Integer	40,40	10	1862400	16	656	1.2038	0.28719	1863072
17	1	46559	5	Integer	50,50	10	2328000	16	656	1.26284	0.34623	2328673
18	1	46559	5	Integer	60,60	10	2793600	16	656	1.32088	0.404268	2794274
19	1	46559	5	Integer	70,70	10	3259200	16	656	1.38592	0.469312	3259875
20	1	46559	5	Integer	80,80	10	3724800	16	656	1.44896	0.532354	3725476
21	1	46559	5	Integer	90,90	10	4190400	16	656	1.51601	0.599399	4191077
22	1	46559	5	Integer	100,100	10	4656000	16	656	1.58905	0.673447	4656678

23	1	465	3	Integer	0,0	10	0	16	656	1.68912	0.0560372	679
24	1	465	3	Integer	10,10	10	4660	16	656	1.76217	0.129085	5340
25	1	465	3	Integer	20,20	10	9320	16	656	1.83021	0.197131	10001
26	1	465	3	Integer	30,30	10	13980	16	656	1.89426	0.261173	14662
27	1	465	3	Integer	40,40	10	18640	16	656	1.9543	0.321213	19323
28	1	465	3	Integer	50,50	10	23300	16	656	2.01234	0.379252	23984
29	1	465	3	Integer	60,60	10	27960	16	656	2.07137	0.438291	28645
30	1	465	3	Integer	70,70	10	32620	16	656	2.13241	0.499331	33306
31	1	465	3	Integer	80,80	10	37280	16	656	2.19145	0.558371	37967
32	1	465	3	Integer	90,90	10	41940	16	656	2.27651	0.643427	42628
33	1	465	3	Integer	100,100	10	46600	16	656	2.34656	0.713474	47289
34	1	2344	4	Integer	0,0	10	0	16	656	2.43261	0.0450297	690
35	1	2344	4	Integer	10,10	10	23450	16	656	2.49365	0.10607	24141
36	1	2344	4	Integer	20,20	10	46900	16	656	2.5617	0.174116	47592
37	1	2344	4	Integer	30,30	10	70350	16	656	2.63675	0.249166	71043
38	1	2344	4	Integer	40,40	10	93800	16	656	2.7098	0.322214	94494
39	1	2344	4	Integer	50,50	10	117250	16	656	2.77884	0.391259	117945
40	1	2344	4	Integer	60,60	10	140700	16	656	2.85389	0.46631	141396
41	1	2344	4	Integer	70,70	10	164150	16	656	2.91594	0.528351	164847
42	1	2344	4	Integer	80,80	10	187600	16	656	2.97598	0.588391	188298
43	1	2344	4	Integer	90,90	10	211050	16	656	3.03301	0.645428	211749
44	1	2344	4	Integer	100,100	10	234500	16	656	3.08905	0.701466	235200
45	1	452	3	Integer	0,0	10	0	16	656	3.1661	0.0420283	701
46	1	452	3	Integer	10,10	10	4530	16	656	3.22514	0.101067	5232
47	1	452	3	Integer	20,20	10	9060	16	656	3.29319	0.169113	9763
48	1	452	3	Integer	30,30	10	13590	16	656	3.35323	0.229153	14294
49	1	452	3	Integer	40,40	10	18120	16	656	3.41427	0.290193	18825
50	1	452	3	Integer	50,50	10	22650	16	656	3.47631	0.352234	23356
51	1	452	3	Integer	60,60	10	27180	16	656	3.53935	0.415275	27887
52	1	452	3	Integer	70,70	10	31710	16	656	3.6104	0.486323	32418
53	1	452	3	Integer	80,80	10	36240	16	656	3.68445	0.560372	36949
54	1	452	3	Integer	90,90	10	40770	16	656	3.75749	0.633421	41480
55	1	452	3	Integer	100,100	10	45300	16	656	3.82554	0.701466	46011
56	1	3453	4	Integer	0,0	10	0	16	656	3.90959	0.0450297	712
57	1	3453	4	Integer	10,10	10	34540	16	656	3.97164	0.107071	35253
58	1	3453	4	Integer	20,20	10	69080	16	656	4.03568	0.171114	69794
59	1	3453	4	Integer	30,30	10	103620	16	656	4.09672	0.232156	104335
60	1	3453	4	Integer	40,40	10	138160	16	656	4.15476	0.290193	138876
61	1	3453	4	Integer	50,50	10	172700	16	656	4.2118	0.347231	173417
62	1	3453	4	Integer	60,60	10	207240	16	656	4.26983	0.405269	207958
63	1	3453	4	Integer	70,70	10	241780	16	656	4.34689	0.482321	242499
64	1	3453	4	Integer	80,80	10	276320	16	656	4.40893	0.544361	277040
65	1	3453	4	Integer	90,90	10	310860	16	656	4.47297	0.608405	311581
66	1	3453	4	Integer	100,100	10	345400	16	656	4.53601	0.671446	346122
67	1	15964	5	Integer	0,0	10	0	16	656	4.63608	0.0540355	723
68	1	15964	5	Integer	10,10	10	159650	16	656	4.71313	0.131087	160374

69	1	15964	5	Integer	20,20	10	319300	16	656	4.78818	0.206136	320025
70	1	15964	5	Integer	30,30	10	478950	16	656	4.85822	0.276183	479676
71	1	15964	5	Integer	40,40	10	638600	16	656	4.92227	0.340226	639327
72	1	15964	5	Integer	50,50	10	798250	16	656	4.98031	0.398264	798978
73	1	15964	5	Integer	60,60	10	957900	16	656	5.04035	0.458304	958629
74	1	15964	5	Integer	70,70	10	1117550	16	656	5.09738	0.515342	1118280
75	1	15964	5	Integer	80,80	10	1277200	16	656	5.15842	0.576382	1277931
76	1	15964	5	Integer	90,90	10	1436850	16	656	5.21746	0.635421	1437582
77	1	15964	5	Integer	100,100	10	1596500	16	656	5.2775	0.695461	1597233
78	1	465	3	Integer	0,0	10	0	16	656	5.35956	0.0450301	734
79	1	465	3	Integer	10,10	10	4660	16	656	5.4276	0.113075	5395
80	1	465	3	Integer	20,20	10	9320	16	656	5.49164	0.177117	10056
81	1	465	3	Integer	30,30	10	13980	16	656	5.56169	0.247164	14717
82	1	465	3	Integer	40,40	10	18640	16	656	5.63174	0.317211	19378
83	1	465	3	Integer	50,50	10	23300	16	656	5.70679	0.393261	24039
84	1	465	3	Integer	60,60	10	27960	16	656	5.78784	0.473314	28700
85	1	465	3	Integer	70,70	10	32620	16	656	5.85889	0.544362	33361
86	1	465	3	Integer	80,80	10	37280	16	656	5.92693	0.612407	38022
87	1	465	3	Integer	90,90	10	41940	16	656	5.98697	0.672447	42683
88	1	465	3	Integer	100,100	10	46600	16	656	6.05202	0.73749	47344
89	1	2344	4	Integer	0,0	10	0	16	656	6.13207	0.0450301	745
90	1	2344	4	Integer	10,10	10	23450	16	656	6.19411	0.107071	24196
91	1	2344	4	Integer	20,20	10	46900	16	656	6.25915	0.172114	47647
92	1	2344	4	Integer	30,30	10	70350	16	656	6.31919	0.232154	71098
93	1	2344	4	Integer	40,40	10	93800	16	656	6.38424	0.297198	94549
94	1	2344	4	Integer	50,50	10	117250	16	656	6.45328	0.366243	118000
95	1	2344	4	Integer	60,60	10	140700	16	656	6.51432	0.427284	141451
96	1	2344	4	Integer	70,70	10	164150	16	656	6.58437	0.49733	164902
97	1	2344	4	Integer	80,80	10	187600	16	656	6.66042	0.573381	188353
98	1	2344	4	Integer	90,90	10	211050	16	656	6.73747	0.650432	211804
99	1	2344	4	Integer	100,100	10	234500	16	656	6.81052	0.72348	235255
100	1	452	3	Integer	0,0	10	0	16	656	6.89458	0.0450305	756
101	1	452	3	Integer	10,10	10	4530	16	656	6.95362	0.104069	5287
102	1	452	3	Integer	20,20	10	9060	16	656	7.01165	0.162108	9818
103	1	452	3	Integer	30,30	10	13590	16	656	7.06969	0.220146	14349
104	1	452	3	Integer	40,40	10	18120	16	656	7.12973	0.280186	18880
105	1	452	3	Integer	50,50	10	22650	16	656	7.19077	0.341226	23411
106	1	452	3	Integer	60,60	10	27180	16	656	7.24981	0.400265	27942
107	1	452	3	Integer	70,70	10	31710	16	656	7.30885	0.459305	32473
108	1	452	3	Integer	80,80	10	36240	16	656	7.3789	0.529351	37004
109	1	452	3	Integer	90,90	10	40770	16	656	7.49798	0.64843	41535
110	1	452	3	Integer	100,100	10	45300	16	656	7.57002	0.720478	46066
111	1	5467	4	Integer	0,0	10	0	16	656	7.66209	0.0520349	767
112	1	5467	4	Integer	10,10	10	54680	16	656	7.73814	0.128085	55448
113	1	5467	4	Integer	20,20	10	109360	16	656	7.81118	0.201134	110129
114	1	5467	4	Integer	30,30	10	164040	16	656	7.87923	0.269179	164810

115	1	5467	4	Integer	40,40	10	218720	16	656	7.94127	0.33122	219491
116	1	5467	4	Integer	50,50	10	273400	16	656	8.00031	0.390259	274172
117	1	5467	4	Integer	60,60	10	328080	16	656	8.06335	0.453301	328853
118	1	5467	4	Integer	70,70	10	382760	16	656	8.12139	0.51134	383534
119	1	5467	4	Integer	80,80	10	437440	16	656	8.18043	0.570379	438215
120	1	5467	4	Integer	90,90	10	492120	16	656	8.24047	0.630418	492896
121	1	5467	4	Integer	100,100	10	546800	16	656	8.30051	0.690459	547577
122	1	349957	6	Integer	0,0	10	0	16	656	8.39958	0.0580385	778
123	1	349957	6	Integer	10,10	10	3499580	16	656	8.46162	0.12008	3500359
124	1	349957	6	Integer	20,20	10	6999160	16	656	8.53467	0.193128	6999940
125	1	349957	6	Integer	30,30	10	10498740	16	656	8.60871	0.267177	10499521
126	1	349957	6	Integer	40,40	10	13998320	16	656	8.68476	0.343228	13999102
127	1	349957	6	Integer	50,50	10	17497900	16	656	8.76081	0.419278	17498683
128	1	349957	6	Integer	60,60	10	20997480	16	656	8.83286	0.491326	20998264
129	1	349957	6	Integer	70,70	10	24497060	16	656	8.89891	0.557371	24497845
130	1	349957	6	Integer	80,80	10	27996640	16	656	8.95895	0.617411	27997426
131	1	349957	6	Integer	90,90	10	31496220	16	656	9.01899	0.67745	31497007
132	1	349957	6	Integer	100,100	10	34995800	16	656	9.08203	0.740491	34996588
133	1	3355	4	Integer	0,0	10	0	16	656	9.16308	0.043028	789
134	1	3355	4	Integer	10,10	10	33560	16	656	9.22412	0.104069	34350
135	1	3355	4	Integer	20,20	10	67120	16	656	9.28817	0.168112	67911
136	1	3355	4	Integer	30,30	10	100680	16	656	9.3482	0.228151	101472
137	1	3355	4	Integer	40,40	10	134240	16	656	9.40925	0.289192	135033
138	1	3355	4	Integer	50,50	10	167800	16	656	9.46829	0.348232	168594
139	1	3355	4	Integer	60,60	10	201360	16	656	9.53933	0.419278	202155
140	1	3355	4	Integer	70,70	10	234920	16	656	9.61638	0.496329	235716
141	1	3355	4	Integer	80,80	10	268480	16	656	9.69143	0.571379	269277
142	1	3355	4	Integer	90,90	10	302040	16	656	9.76148	0.641426	302838
143	1	3355	4	Integer	100,100	10	335600	16	656	9.83253	0.712473	336399
144	1	5176	4	Integer	0,0	10	0	16	656	9.91658	0.0440288	800
145	1	5176	4	Integer	10,10	10	51770	16	656	9.97662	0.10407	52571
146	1	5176	4	Integer	20,20	10	103540	16	656	10.0347	0.162108	104342
147	1	5176	4	Integer	30,30	10	155310	16	656	10.0937	0.221147	156113
148	1	5176	4	Integer	40,40	10	207080	16	656	10.1547	0.282187	207884
149	1	5176	4	Integer	50,50	10	258850	16	656	10.2148	0.342228	259655
150	1	5176	4	Integer	60,60	10	310620	16	656	10.2748	0.402267	311426
151	1	5176	4	Integer	70,70	10	362390	16	656	10.3369	0.464309	363197
152	1	5176	4	Integer	80,80	10	414160	16	656	10.3989	0.52635	414968
153	1	5176	4	Integer	90,90	10	465930	16	656	10.4669	0.594394	466739
154	1	5176	4	Integer	100,100	10	517700	16	656	10.542	0.669444	518510

C. Results of using Different Key Sizes

Original Index	EquationID	InputDataX	DataSize	DataType	Range of Coefficient	Step	FunctionValue	KeySize	KeyValue	PerformanceTime	TimePerRange	ReIndex
1	1	428789	6	Integer	0,0	10	0	16	1136	0.0929376	0.0929376	1137
2	1	428789	6	Integer	10,10	10	4287900	16	1136	0.2283	0.2283	4289038
3	1	428789	6	Integer	20,20	10	8575800	16	1136	0.282679	0.282679	8576939
4	1	428789	6	Integer	30,30	10	12863700	16	1136	0.335354	0.335354	12864840
5	1	428789	6	Integer	40,40	10	17151600	16	1136	0.388017	0.388017	17152741
6	1	428789	6	Integer	50,50	10	21439500	16	1136	0.442535	0.442535	21440642
7	1	428789	6	Integer	60,60	10	25727400	16	1136	0.495873	0.495873	25728543
8	1	428789	6	Integer	70,70	10	30015300	16	1136	0.55058	0.55058	30016444
9	1	428789	6	Integer	80,80	10	34303200	16	1136	0.604901	0.604901	34304345
10	1	428789	6	Integer	90,90	10	38591100	16	1136	0.657781	0.657781	38592246
11	1	428789	6	Integer	100,100	10	42879000	16	1136	0.712144	0.712144	42880147
12	1	46559	5	Integer	0,0	10	0	16	1136	0.784443	0.0363917	1148
13	1	46559	5	Integer	10,10	10	465600	16	1136	0.837903	0.0898515	466749
14	1	46559	5	Integer	20,20	10	931200	16	1136	0.891004	0.142952	932350
15	1	46559	5	Integer	30,30	10	1396800	16	1136	0.94434	0.196288	1397951
16	1	46559	5	Integer	40,40	10	1862400	16	1136	0.997326	0.249275	1863552
17	1	46559	5	Integer	50,50	10	2328000	16	1136	1.05245	0.304395	2329153
18	1	46559	5	Integer	60,60	10	2793600	16	1136	1.1057	0.357648	2794754
19	1	46559	5	Integer	70,70	10	3259200	16	1136	1.15865	0.410598	3260355
20	1	46559	5	Integer	80,80	10	3724800	16	1136	1.21295	0.464902	3725956
21	1	46559	5	Integer	90,90	10	4190400	16	1136	1.26635	0.5183	4191557
22	1	46559	5	Integer	100,100	10	4656000	16	1136	1.31931	0.571256	4657158

23	1	465	3	Integer	0,0	10	0	16	1136	1.38937	0.037723	1159
24	1	465	3	Integer	10,10	10	4660	16	1136	1.44255	0.0909058	5820
25	1	465	3	Integer	20,20	10	9320	16	1136	1.50243	0.15078	10481
26	1	465	3	Integer	30,30	10	13980	16	1136	1.56441	0.212766	15142
27	1	465	3	Integer	40,40	10	18640	16	1136	1.62652	0.274872	19803
28	1	465	3	Integer	50,50	10	23300	16	1136	1.67955	0.327903	24464
29	1	465	3	Integer	60,60	10	27960	16	1136	1.73567	0.384029	29125
30	1	465	3	Integer	70,70	10	32620	16	1136	1.78951	0.437864	33786
31	1	465	3	Integer	80,80	10	37280	16	1136	1.84346	0.491815	38447
32	1	465	3	Integer	90,90	10	41940	16	1136	1.89904	0.547399	43108
33	1	465	3	Integer	100,100	10	46600	16	1136	1.95216	0.600512	47769
34	1	2344	4	Integer	0,0	10	0	16	1136	2.02241	0.0379147	1170
35	1	2344	4	Integer	10,10	10	23450	16	1136	2.07595	0.0914477	24621
36	1	2344	4	Integer	20,20	10	46900	16	1136	2.13189	0.147396	48072
37	1	2344	4	Integer	30,30	10	70350	16	1136	2.18593	0.201432	71523
38	1	2344	4	Integer	40,40	10	93800	16	1136	2.23896	0.254458	94974
39	1	2344	4	Integer	50,50	10	117250	16	1136	2.29369	0.30919	118425
40	1	2344	4	Integer	60,60	10	140700	16	1136	2.34735	0.36285	141876
41	1	2344	4	Integer	70,70	10	164150	16	1136	2.40126	0.416757	165327
42	1	2344	4	Integer	80,80	10	187600	16	1136	2.4555	0.471007	188778
43	1	2344	4	Integer	90,90	10	211050	16	1136	2.50875	0.524247	212229
44	1	2344	4	Integer	100,100	10	234500	16	1136	2.56239	0.577894	235680
45	1	452	3	Integer	0,0	10	0	16	1136	2.63251	0.0379349	1181

46	1	452	3	Integer	10,10	10	4530	16	1136	2.68943	0.0948559	5712
47	1	452	3	Integer	20,20	10	9060	16	1136	2.7423	0.147732	10243
48	1	452	3	Integer	30,30	10	13590	16	1136	2.79585	0.201282	14774
49	1	452	3	Integer	40,40	10	18120	16	1136	2.84995	0.255375	19305
50	1	452	3	Integer	50,50	10	22650	16	1136	2.90383	0.309262	23836
51	1	452	3	Integer	60,60	10	27180	16	1136	2.95734	0.362768	28367
52	1	452	3	Integer	70,70	10	31710	16	1136	3.01005	0.415474	32898
53	1	452	3	Integer	80,80	10	36240	16	1136	3.06364	0.469068	37429
54	1	452	3	Integer	90,90	10	40770	16	1136	3.11843	0.523861	41960
55	1	452	3	Integer	100,100	10	45300	16	1136	3.17389	0.579319	46491
56	1	3453	4	Integer	0,0	10	0	16	1136	3.24482	0.0379994	1192
57	1	3453	4	Integer	10,10	10	34540	16	1136	3.29858	0.0917622	35733
58	1	3453	4	Integer	20,20	10	69080	16	1136	3.35182	0.144995	70274
59	1	3453	4	Integer	30,30	10	103620	16	1136	3.40679	0.199967	104815
60	1	3453	4	Integer	40,40	10	138160	16	1136	3.45996	0.253134	139356
61	1	3453	4	Integer	50,50	10	172700	16	1136	3.51444	0.307618	173897
62	1	3453	4	Integer	60,60	10	207240	16	1136	3.56804	0.361223	208438
63	1	3453	4	Integer	70,70	10	241780	16	1136	3.62289	0.416066	242979
64	1	3453	4	Integer	80,80	10	276320	16	1136	3.67652	0.469699	277520
65	1	3453	4	Integer	90,90	10	310860	16	1136	3.73018	0.523355	312061
66	1	3453	4	Integer	100,100	10	345400	16	1136	3.78336	0.576544	346602
67	1	15964	5	Integer	0,0	10	0	16	1136	3.85447	0.0374545	1203
68	1	15964	5	Integer	10,10	10	159650	16	1136	3.90819	0.0911817	160854
69	1	15964	5	Integer	20,20	10	319300	16	1136	3.96217	0.145155	320505
70	1	15964	5	Integer	30,30	10	478950	16	1136	4.0166	0.199589	480156
71	1	15964	5	Integer	40,40	10	638600	16	1136	4.0695	0.252492	639807
72	1	15964	5	Integer	50,50	10	798250	16	1136	4.12496	0.307948	799458
73	1	15964	5	Integer	60,60	10	957900	16	1136	4.20314	0.386126	959109
74	1	15964	5	Integer	70,70	10	1117550	16	1136	4.25761	0.440599	1118760
75	1	15964	5	Integer	80,80	10	1277200	16	1136	4.31282	0.495807	1278411
76	1	15964	5	Integer	90,90	10	1436850	16	1136	4.36659	0.549579	1438062
77	1	15964	5	Integer	100,100	10	1596500	16	1136	4.42078	0.603769	1597713
78	1	465	3	Integer	0,0	10	0	16	1136	4.49169	0.0381546	1214
79	1	465	3	Integer	10,10	10	4660	16	1136	4.54698	0.0934454	5875
80	1	465	3	Integer	20,20	10	9320	16	1136	4.6004	0.146861	10536
81	1	465	3	Integer	30,30	10	13980	16	1136	4.65555	0.202008	15197
82	1	465	3	Integer	40,40	10	18640	16	1136	4.7084	0.254867	19858
83	1	465	3	Integer	50,50	10	23300	16	1136	4.76433	0.310793	24519
84	1	465	3	Integer	60,60	10	27960	16	1136	4.81742	0.363882	29180
85	1	465	3	Integer	70,70	10	32620	16	1136	4.87126	0.417724	33841
86	1	465	3	Integer	80,80	10	37280	16	1136	4.92595	0.472413	38502
87	1	465	3	Integer	90,90	10	41940	16	1136	4.98	0.526459	43163
88	1	465	3	Integer	100,100	10	46600	16	1136	5.03401	0.580469	47824
89	1	2344	4	Integer	0,0	10	0	16	1136	5.10421	0.0378348	1225
90	1	2344	4	Integer	10,10	10	23450	16	1136	5.15837	0.0929913	24676
91	1	2344	4	Integer	20,20	10	46900	16	1136	5.21302	0.146642	48127

92	1	2344	4	Integer	30,30	10	70350	16	1136	5.26866	0.202284	71578
93	1	2344	4	Integer	40,40	10	93800	16	1136	5.32409	0.257714	95029
94	1	2344	4	Integer	50,50	10	117250	16	1136	5.3783	0.311925	118480
95	1	2344	4	Integer	60,60	10	140700	16	1136	5.43379	0.36741	141931
96	1	2344	4	Integer	70,70	10	164150	16	1136	5.48774	0.421369	165382
97	1	2344	4	Integer	80,80	10	187600	16	1136	5.54206	0.475685	188833
98	1	2344	4	Integer	90,90	10	211050	16	1136	5.59596	0.529583	212284
99	1	2344	4	Integer	100,100	10	234500	16	1136	5.64984	0.583467	235735
100	1	452	3	Integer	0,0	10	0	16	1136	5.72033	0.0375818	1236
101	1	452	3	Integer	10,10	10	4530	16	1136	5.77458	0.091827	5767
102	1	452	3	Integer	20,20	10	9060	16	1136	5.82977	0.14702	10298
103	1	452	3	Integer	30,30	10	13590	16	1136	5.88392	0.201171	14829
104	1	452	3	Integer	40,40	10	18120	16	1136	5.93947	0.256723	19360
105	1	452	3	Integer	50,50	10	22650	16	1136	5.99391	0.31116	23891
106	1	452	3	Integer	60,60	10	27180	16	1136	6.04803	0.36528	28422
107	1	452	3	Integer	70,70	10	31710	16	1136	6.10225	0.419495	32953
108	1	452	3	Integer	80,80	10	36240	16	1136	6.15631	0.473562	37484
109	1	452	3	Integer	90,90	10	40770	16	1136	6.21052	0.52777	42015
110	1	452	3	Integer	100,100	10	45300	16	1136	6.2668	0.584046	46546
111	1	5467	4	Integer	0,0	10	0	16	1136	6.3374	0.0383844	1247
112	1	5467	4	Integer	10,10	10	54680	16	1136	6.39349	0.0944802	55928
113	1	5467	4	Integer	20,20	10	109360	16	1136	6.44795	0.148931	110609
114	1	5467	4	Integer	30,30	10	164040	16	1136	6.50244	0.203429	165290

115	1	5467	4	Integer	40,40	10	218720	16	1136	6.55602	0.257009	219971
116	1	5467	4	Integer	50,50	10	273400	16	1136	6.61287	0.313855	274652
117	1	5467	4	Integer	60,60	10	328080	16	1136	6.67571	0.3767	329333
118	1	5467	4	Integer	70,70	10	382760	16	1136	6.73568	0.43667	384014
119	1	5467	4	Integer	80,80	10	437440	16	1136	6.79188	0.492869	438695
120	1	5467	4	Integer	90,90	10	492120	16	1136	6.84568	0.546663	493376
121	1	5467	4	Integer	100,100	10	546800	16	1136	6.90084	0.601828	548057
122	1	349957	6	Integer	0,0	10	0	16	1136	6.97326	0.039983	1258
123	1	349957	6	Integer	10,10	10	3499580	16	1136	7.02695	0.0936773	3500839
124	1	349957	6	Integer	20,20	10	6999160	16	1136	7.08162	0.14835	7000420
125	1	349957	6	Integer	30,30	10	10498740	16	1136	7.13607	0.202791	10500001
126	1	349957	6	Integer	40,40	10	13998320	16	1136	7.19092	0.257647	13999582
127	1	349957	6	Integer	50,50	10	17497900	16	1136	7.24452	0.311244	17499163
128	1	349957	6	Integer	60,60	10	20997480	16	1136	7.30007	0.366793	20998744
129	1	349957	6	Integer	70,70	10	24497060	16	1136	7.35573	0.422457	24498325
130	1	349957	6	Integer	80,80	10	27996640	16	1136	7.41211	0.47883	27997906
131	1	349957	6	Integer	90,90	10	31496220	16	1136	7.46747	0.534198	31497487
132	1	349957	6	Integer	100,100	10	34995800	16	1136	7.52096	0.587687	34997068
133	1	3355	4	Integer	0,0	10	0	16	1136	7.59194	0.0384119	1269
134	1	3355	4	Integer	10,10	10	33560	16	1136	7.64655	0.0930233	34830
135	1	3355	4	Integer	20,20	10	67120	16	1136	7.70136	0.147831	68391
136	1	3355	4	Integer	30,30	10	100680	16	1136	7.75583	0.202302	101952
137	1	3355	4	Integer	40,40	10	134240	16	1136	7.80998	0.25645	135513
138	1	3355	4	Integer	50,50	10	167800	16	1136	7.86561	0.312087	169074
139	1	3355	4	Integer	60,60	10	201360	16	1136	7.91896	0.365433	202635
140	1	3355	4	Integer	70,70	10	234920	16	1136	7.97476	0.421233	236196
141	1	3355	4	Integer	80,80	10	268480	16	1136	8.02891	0.475381	269757
142	1	3355	4	Integer	90,90	10	302040	16	1136	8.0841	0.53057	303318
143	1	3355	4	Integer	100,100	10	335600	16	1136	8.13884	0.585317	336879
144	1	5176	4	Integer	0,0	10	0	16	1136	8.20944	0.0384562	1280
145	1	5176	4	Integer	10,10	10	51770	16	1136	8.2654	0.0944133	53051
146	1	5176	4	Integer	20,20	10	103540	16	1136	8.32023	0.149246	104822
147	1	5176	4	Integer	30,30	10	155310	16	1136	8.37793	0.206942	156593
148	1	5176	4	Integer	40,40	10	207080	16	1136	8.43212	0.261141	208364
149	1	5176	4	Integer	50,50	10	258850	16	1136	8.48669	0.315708	260135
150	1	5176	4	Integer	60,60	10	310620	16	1136	8.54226	0.37128	311906
151	1	5176	4	Integer	70,70	10	362390	16	1136	8.59598	0.424992	363677
152	1	5176	4	Integer	80,80	10	414160	16	1136	8.6525	0.48152	415448
153	1	5176	4	Integer	90,90	10	465930	16	1136	8.70709	0.536106	467219
154	1	5176	4	Integer	100,100	10	517700	16	1136	8.76274	0.59176	518990
155	1	428789	6	Integer	0,0	10	0	32	1184	0.0394268	0.0394268	1339
156	1	428789	6	Integer	10,10	10	4287900	32	1184	0.0936084	0.0936084	4289240
157	1	428789	6	Integer	20,20	10	8575800	32	1184	0.149158	0.149158	8577141
158	1	428789	6	Integer	30,30	10	12863700	32	1184	0.203856	0.203856	12865042
159	1	428789	6	Integer	40,40	10	17151600	32	1184	0.258395	0.258395	17152943
160	1	428789	6	Integer	50,50	10	21439500	32	1184	0.313449	0.313449	21440844

161	1	428789	6	Integer	60,60	10	25727400	32	1184	0.37081	0.37081	25728745
162	1	428789	6	Integer	70,70	10	30015300	32	1184	0.426767	0.426767	30016646
163	1	428789	6	Integer	80,80	10	34303200	32	1184	0.481376	0.481376	34304547
164	1	428789	6	Integer	90,90	10	38591100	32	1184	0.537284	0.537284	38592448
165	1	428789	6	Integer	100,100	10	42879000	32	1184	0.59214	0.59214	42880349
166	1	46559	5	Integer	0,0	10	0	32	1184	0.663236	0.0387329	1350
167	1	46559	5	Integer	10,10	10	465600	32	1184	0.830162	0.205659	466951
168	1	46559	5	Integer	20,20	10	931200	32	1184	0.981305	0.356803	932552
169	1	46559	5	Integer	30,30	10	1396800	32	1184	1.03709	0.412587	1398153
170	1	46559	5	Integer	40,40	10	1862400	32	1184	1.09228	0.467781	1863754
171	1	46559	5	Integer	50,50	10	2328000	32	1184	1.14608	0.521582	2329355
172	1	46559	5	Integer	60,60	10	2793600	32	1184	1.20123	0.576729	2794956
173	1	46559	5	Integer	70,70	10	3259200	32	1184	1.25735	0.632846	3260557
174	1	46559	5	Integer	80,80	10	3724800	32	1184	1.31188	0.687373	3726158
175	1	46559	5	Integer	90,90	10	4190400	32	1184	1.3694	0.744901	4191759
176	1	46559	5	Integer	100,100	10	4656000	32	1184	1.42477	0.800267	4657360
177	1	465	3	Integer	0,0	10	0	32	1184	1.49641	0.0398701	1361
178	1	465	3	Integer	10,10	10	4660	32	1184	1.55103	0.0944881	6022
179	1	465	3	Integer	20,20	10	9320	32	1184	1.60736	0.150822	10683
180	1	465	3	Integer	30,30	10	13980	32	1184	1.66279	0.206252	15344
181	1	465	3	Integer	40,40	10	18640	32	1184	1.72228	0.265742	20005
182	1	465	3	Integer	50,50	10	23300	32	1184	1.77818	0.321646	24666
183	1	465	3	Integer	60,60	10	27960	32	1184	1.83327	0.376735	29327
184	1	465	3	Integer	70,70	10	32620	32	1184	1.88914	0.432606	33988
185	1	465	3	Integer	80,80	10	37280	32	1184	1.94358	0.487045	38649
186	1	465	3	Integer	90,90	10	41940	32	1184	1.99936	0.542824	43310
187	1	465	3	Integer	100,100	10	46600	32	1184	2.05327	0.596729	47971
188	1	2344	4	Integer	0,0	10	0	32	1184	2.1251	0.0390022	1372
189	1	2344	4	Integer	10,10	10	23450	32	1184	2.17876	0.0926595	24823
190	1	2344	4	Integer	20,20	10	46900	32	1184	2.23652	0.150417	48274
191	1	2344	4	Integer	30,30	10	70350	32	1184	2.29117	0.205068	71725
192	1	2344	4	Integer	40,40	10	93800	32	1184	2.34676	0.260659	95176
193	1	2344	4	Integer	50,50	10	117250	32	1184	2.40226	0.316154	118627
194	1	2344	4	Integer	60,60	10	140700	32	1184	2.45762	0.37152	142078
195	1	2344	4	Integer	70,70	10	164150	32	1184	2.51358	0.427472	165529
196	1	2344	4	Integer	80,80	10	187600	32	1184	2.56902	0.482916	188980
197	1	2344	4	Integer	90,90	10	211050	32	1184	2.62504	0.538935	212431
198	1	2344	4	Integer	100,100	10	234500	32	1184	2.68176	0.595661	235882
199	1	452	3	Integer	0,0	10	0	32	1184	2.75309	0.0387872	1383
200	1	452	3	Integer	10,10	10	4530	32	1184	2.80834	0.0940361	5914
201	1	452	3	Integer	20,20	10	9060	32	1184	2.86628	0.151978	10445
202	1	452	3	Integer	30,30	10	13590	32	1184	2.92212	0.207818	14976
203	1	452	3	Integer	40,40	10	18120	32	1184	2.98543	0.271122	19507
204	1	452	3	Integer	50,50	10	22650	32	1184	3.04647	0.332165	24038
205	1	452	3	Integer	60,60	10	27180	32	1184	3.105	0.390696	28569
206	1	452	3	Integer	70,70	10	31710	32	1184	3.16038	0.446073	33100

207	1	452	3	Integer	80,80	10	36240	32	1184	3.21507	0.500761	37631
208	1	452	3	Integer	90,90	10	40770	32	1184	3.27099	0.556685	42162
209	1	452	3	Integer	100,100	10	45300	32	1184	3.32601	0.6117	46693
210	1	3453	4	Integer	0,0	10	0	32	1184	3.39726	0.0396115	1394
211	1	3453	4	Integer	10,10	10	34540	32	1184	3.45223	0.0945767	35935
212	1	3453	4	Integer	20,20	10	69080	32	1184	3.50798	0.150334	70476
213	1	3453	4	Integer	30,30	10	103620	32	1184	3.56235	0.204699	105017
214	1	3453	4	Integer	40,40	10	138160	32	1184	3.61877	0.261118	139558
215	1	3453	4	Integer	50,50	10	172700	32	1184	3.67499	0.317342	174099
216	1	3453	4	Integer	60,60	10	207240	32	1184	3.72982	0.372174	208640
217	1	3453	4	Integer	70,70	10	241780	32	1184	3.78769	0.430036	243181
218	1	3453	4	Integer	80,80	10	276320	32	1184	3.84437	0.486724	277722
219	1	3453	4	Integer	90,90	10	310860	32	1184	3.90068	0.54303	312263
220	1	3453	4	Integer	100,100	10	345400	32	1184	3.95641	0.598765	346804
221	1	15964	5	Integer	0,0	10	0	32	1184	4.0276	0.0391579	1405
222	1	15964	5	Integer	10,10	10	159650	32	1184	4.08315	0.0947147	161056
223	1	15964	5	Integer	20,20	10	319300	32	1184	4.14078	0.152343	320707
224	1	15964	5	Integer	30,30	10	478950	32	1184	4.19737	0.208933	480358
225	1	15964	5	Integer	40,40	10	638600	32	1184	4.2527	0.264257	640009
226	1	15964	5	Integer	50,50	10	798250	32	1184	4.30906	0.320618	799660
227	1	15964	5	Integer	60,60	10	957900	32	1184	4.36584	0.377399	959311
228	1	15964	5	Integer	70,70	10	1117550	32	1184	4.42155	0.433107	1118962
229	1	15964	5	Integer	80,80	10	1277200	32	1184	4.48139	0.492955	1278613

D. Results of using Different Key Sizes

Original Index	EquationID	InputDataX	DataSize	DataType	Range of Coefficient	Step	FunctionValue	KeySize	KeyValue	PerformanceTime	TimePerRange	ReIndex
1	1	428789	6	Integer	0,0	10	0	16	544	0.0933243	0.0933243	545
2	1	428789	6	Integer	10,10	10	4287900	16	544	0.216607	0.216607	4288446
3	1	428789	6	Integer	20,20	10	8575800	16	544	0.270636	0.270636	8576347
4	1	428789	6	Integer	30,30	10	12863700	16	544	0.324524	0.324524	12864248
5	1	428789	6	Integer	40,40	10	17151600	16	544	0.377232	0.377232	17152149
6	1	428789	6	Integer	50,50	10	21439500	16	544	0.428988	0.428988	21440050
7	1	428789	6	Integer	60,60	10	25727400	16	544	0.480707	0.480707	25727951
8	1	428789	6	Integer	70,70	10	30015300	16	544	0.536841	0.536841	30015852
9	1	428789	6	Integer	80,80	10	34303200	16	544	0.588856	0.588856	34303753
10	1	428789	6	Integer	90,90	10	38591100	16	544	0.640894	0.640894	38591654
11	1	428789	6	Integer	100,100	10	42879000	16	544	0.693511	0.693511	42879555
12	1	46559	5	Integer	0,0	10	0	16	544	0.76465	0.0358124	556
13	1	46559	5	Integer	10,10	10	465600	16	544	0.818709	0.0898717	466157
14	1	46559	5	Integer	20,20	10	931200	16	544	0.870895	0.142057	931758
15	1	46559	5	Integer	30,30	10	1396800	16	544	0.923928	0.195091	1397359
16	1	46559	5	Integer	40,40	10	1862400	16	544	0.975389	0.246552	1862960
17	1	46559	5	Integer	50,50	10	2328000	16	544	1.02842	0.299586	2328561
18	1	46559	5	Integer	60,60	10	2793600	16	544	1.0795	0.350665	2794162
19	1	46559	5	Integer	70,70	10	3259200	16	544	1.12754	0.398698	3259763
20	1	46559	5	Integer	80,80	10	3724800	16	544	1.17827	0.449434	3725364
21	1	46559	5	Integer	90,90	10	4190400	16	544	1.2294	0.500564	4190965
22	1	46559	5	Integer	100,100	10	4656000	16	544	1.27985	0.551017	4656566

23	1	465	3	Integer	0,0	10	0	16	544	1.34542	0.0350218	567
24	1	465	3	Integer	10,10	10	4660	16	544	1.39746	0.0870579	5228
25	1	465	3	Integer	20,20	10	9320	16	544	1.44949	0.139093	9889
26	1	465	3	Integer	30,30	10	13980	16	544	1.49968	0.189278	14550
27	1	465	3	Integer	40,40	10	18640	16	544	1.55071	0.240312	19211
28	1	465	3	Integer	50,50	10	23300	16	544	1.60274	0.292346	23872
29	1	465	3	Integer	60,60	10	27960	16	544	1.6538	0.343398	28533
30	1	465	3	Integer	70,70	10	32620	16	544	1.70472	0.394326	33194
31	1	465	3	Integer	80,80	10	37280	16	544	1.75776	0.447361	37855
32	1	465	3	Integer	90,90	10	41940	16	544	1.80779	0.497394	42516
33	1	465	3	Integer	100,100	10	46600	16	544	1.85883	0.548428	47177
34	1	2344	4	Integer	0,0	10	0	16	544	1.92686	0.0360468	578
35	1	2344	4	Integer	10,10	10	23450	16	544	1.97928	0.0884718	24029
36	1	2344	4	Integer	20,20	10	46900	16	544	2.03163	0.141821	47480
37	1	2344	4	Integer	30,30	10	70350	16	544	2.08469	0.193873	70931
38	1	2344	4	Integer	40,40	10	93800	16	544	2.13747	0.246654	94382
39	1	2344	4	Integer	50,50	10	117250	16	544	2.18985	0.299038	117833
40	1	2344	4	Integer	60,60	10	140700	16	544	2.24233	0.35152	141284
41	1	2344	4	Integer	70,70	10	164150	16	544	2.2946	0.403783	164735
42	1	2344	4	Integer	80,80	10	187600	16	544	2.34784	0.457029	188186
43	1	2344	4	Integer	90,90	10	211050	16	544	2.39978	0.508968	211637
44	1	2344	4	Integer	100,100	10	234500	16	544	2.4538	0.562986	235088
45	1	452	3	Integer	0,0	10	0	16	544	2.52636	0.0392441	589

46	1	452	3	Integer	10,10	10	4530	16	544	2.57916	0.0920397	5120
47	1	452	3	Integer	20,20	10	9060	16	544	2.63521	0.148097	9651
48	1	452	3	Integer	30,30	10	13590	16	544	2.68336	0.196243	14182
49	1	452	3	Integer	40,40	10	18120	16	544	2.7356	0.248485	18713
50	1	452	3	Integer	50,50	10	22650	16	544	2.7843	0.297187	23244
51	1	452	3	Integer	60,60	10	27180	16	544	2.83233	0.345217	27775
52	1	452	3	Integer	70,70	10	31710	16	544	2.88112	0.394007	32306
53	1	452	3	Integer	80,80	10	36240	16	544	2.95248	0.465358	36837
54	1	452	3	Integer	90,90	10	40770	16	544	3.00495	0.517837	41368
55	1	452	3	Integer	100,100	10	45300	16	544	3.07452	0.5874	45899
56	1	3453	4	Integer	0,0	10	0	16	544	3.14405	0.0380581	600
57	1	3453	4	Integer	10,10	10	34540	16	544	3.19633	0.0903405	35141
58	1	3453	4	Integer	20,20	10	69080	16	544	3.24948	0.143484	69682
59	1	3453	4	Integer	30,30	10	103620	16	544	3.30205	0.196063	104223
60	1	3453	4	Integer	40,40	10	138160	16	544	3.35576	0.249766	138764
61	1	3453	4	Integer	50,50	10	172700	16	544	3.40871	0.302718	173305
62	1	3453	4	Integer	60,60	10	207240	16	544	3.46506	0.359068	207846
63	1	3453	4	Integer	70,70	10	241780	16	544	3.52052	0.41453	242387
64	1	3453	4	Integer	80,80	10	276320	16	544	3.57422	0.46823	276928
65	1	3453	4	Integer	90,90	10	310860	16	544	3.62757	0.521578	311469
66	1	3453	4	Integer	100,100	10	345400	16	544	3.68195	0.575961	346010
67	1	15964	5	Integer	0,0	10	0	16	544	3.80971	0.0369127	611
68	1	15964	5	Integer	10,10	10	159650	16	544	3.86271	0.089907	160262

69	1	15964	5	Integer	20,20	10	319300	16	544	3.91459	0.14179	319913
70	1	15964	5	Integer	30,30	10	478950	16	544	3.96806	0.195257	479564
71	1	15964	5	Integer	40,40	10	638600	16	544	4.02025	0.247447	639215
72	1	15964	5	Integer	50,50	10	798250	16	544	4.07351	0.300712	798866
73	1	15964	5	Integer	60,60	10	957900	16	544	4.12707	0.354269	958517
74	1	15964	5	Integer	70,70	10	1117550	16	544	4.18145	0.408652	1118168
75	1	15964	5	Integer	80,80	10	1277200	16	544	4.2356	0.462797	1277819
76	1	15964	5	Integer	90,90	10	1436850	16	544	4.28915	0.516347	1437470
77	1	15964	5	Integer	100,100	10	1596500	16	544	4.34118	0.568381	1597121
78	1	465	3	Integer	0,0	10	0	16	544	4.41131	0.0390289	622
79	1	465	3	Integer	10,10	10	4660	16	544	4.46269	0.0904107	5283
80	1	465	3	Integer	20,20	10	9320	16	544	4.5144	0.142123	9944
81	1	465	3	Integer	30,30	10	13980	16	544	4.56906	0.196782	14605
82	1	465	3	Integer	40,40	10	18640	16	544	4.6281	0.255821	19266
83	1	465	3	Integer	50,50	10	23300	16	544	4.67917	0.306891	23927
84	1	465	3	Integer	60,60	10	27960	16	544	4.7282	0.355921	28588
85	1	465	3	Integer	70,70	10	32620	16	544	4.77779	0.405518	33249
86	1	465	3	Integer	80,80	10	37280	16	544	4.82872	0.456445	37910
87	1	465	3	Integer	90,90	10	41940	16	544	4.88076	0.50848	42571
88	1	465	3	Integer	100,100	10	46600	16	544	4.9323	0.560028	47232
89	1	2344	4	Integer	0,0	10	0	16	544	4.99835	0.0350238	633
90	1	2344	4	Integer	10,10	10	23450	16	544	5.05031	0.0869852	24084
91	1	2344	4	Integer	20,20	10	46900	16	544	5.10157	0.138248	47535

92	1	2344	4	Integer	30,30	10	70350	16	544	5.15298	0.189654	70986
93	1	2344	4	Integer	40,40	10	93800	16	544	5.20401	0.240687	94437
94	1	2344	4	Integer	50,50	10	117250	16	544	5.2565	0.293172	117888
95	1	2344	4	Integer	60,60	10	140700	16	544	5.30853	0.345206	141339
96	1	2344	4	Integer	70,70	10	164150	16	544	5.36092	0.397591	164790
97	1	2344	4	Integer	80,80	10	187600	16	544	5.41295	0.449625	188241
98	1	2344	4	Integer	90,90	10	211050	16	544	5.46443	0.501103	211692
99	1	2344	4	Integer	100,100	10	234500	16	544	5.51646	0.553138	235143
100	1	452	3	Integer	0,0	10	0	16	544	5.58651	0.0390244	644
101	1	452	3	Integer	10,10	10	4530	16	544	5.63831	0.0908249	5175
102	1	452	3	Integer	20,20	10	9060	16	544	5.69008	0.142595	9706
103	1	452	3	Integer	30,30	10	13590	16	544	5.74412	0.196633	14237
104	1	452	3	Integer	40,40	10	18120	16	544	5.79515	0.247665	18768
105	1	452	3	Integer	50,50	10	22650	16	544	5.84593	0.298451	23299
106	1	452	3	Integer	60,60	10	27180	16	544	5.89453	0.347048	27830
107	1	452	3	Integer	70,70	10	31710	16	544	5.94348	0.396001	32361
108	1	452	3	Integer	80,80	10	36240	16	544	5.99239	0.44491	36892
109	1	452	3	Integer	90,90	10	40770	16	544	6.04113	0.493651	41423
110	1	452	3	Integer	100,100	10	45300	16	544	6.09017	0.542684	45954
111	1	5467	4	Integer	0,0	10	0	16	544	6.15714	0.037028	655
112	1	5467	4	Integer	10,10	10	54680	16	544	6.20917	0.0890629	55336
113	1	5467	4	Integer	20,20	10	109360	16	544	6.26184	0.14173	110017
114	1	5467	4	Integer	30,30	10	164040	16	544	6.31387	0.193764	164698
115	1	5467	4	Integer	40,40	10	218720	16	544	6.36591	0.245799	219379
116	1	5467	4	Integer	50,50	10	273400	16	544	6.41794	0.297834	274060
117	1	5467	4	Integer	60,60	10	328080	16	544	6.46939	0.349281	328741
118	1	5467	4	Integer	70,70	10	382760	16	544	6.52143	0.401316	383422
119	1	5467	4	Integer	80,80	10	437440	16	544	6.57336	0.453249	438103
120	1	5467	4	Integer	90,90	10	492120	16	544	6.6294	0.509287	492784
121	1	5467	4	Integer	100,100	10	546800	16	544	6.68343	0.563322	547465
122	1	349957	6	Integer	0,0	10	0	16	544	6.75148	0.0370255	666
123	1	349957	6	Integer	10,10	10	3499580	16	544	6.80437	0.0899164	3500247
124	1	349957	6	Integer	20,20	10	6999160	16	544	6.85582	0.141364	6999828
125	1	349957	6	Integer	30,30	10	10498740	16	544	6.90738	0.19293	10499409
126	1	349957	6	Integer	40,40	10	13998320	16	544	6.9599	0.245446	13998990
127	1	349957	6	Integer	50,50	10	17497900	16	544	7.01351	0.299059	17498571
128	1	349957	6	Integer	60,60	10	20997480	16	544	7.06783	0.353375	20998152
129	1	349957	6	Integer	70,70	10	24497060	16	544	7.12178	0.407324	24497733
130	1	349957	6	Integer	80,80	10	27996640	16	544	7.17509	0.460642	27997314
131	1	349957	6	Integer	90,90	10	31496220	16	544	7.22832	0.513865	31496895
132	1	349957	6	Integer	100,100	10	34995800	16	544	7.2849	0.57045	34996476
133	1	3355	4	Integer	0,0	10	0	16	544	7.35252	0.0360276	677
134	1	3355	4	Integer	10,10	10	33560	16	544	7.40433	0.08784	34238
135	1	3355	4	Integer	20,20	10	67120	16	544	7.45537	0.138873	67799
136	1	3355	4	Integer	30,30	10	100680	16	544	7.50758	0.191087	101360
137	1	3355	4	Integer	40,40	10	134240	16	544	7.56162	0.245124	134921

138	1	3355	4	Integer	50,50	10	167800	16	544	7.61465	0.298159	168482
139	1	3355	4	Integer	60,60	10	201360	16	544	7.67669	0.360198	202043
140	1	3355	4	Integer	70,70	10	234920	16	544	7.72917	0.412673	235604
141	1	3355	4	Integer	80,80	10	268480	16	544	7.7842	0.467709	269165
142	1	3355	4	Integer	90,90	10	302040	16	544	7.83748	0.520988	302726
143	1	3355	4	Integer	100,100	10	335600	16	544	7.8891	0.572607	336287
144	1	5176	4	Integer	0,0	10	0	16	544	7.95665	0.0366676	688
145	1	5176	4	Integer	10,10	10	51770	16	544	8.00968	0.0897022	52459
146	1	5176	4	Integer	20,20	10	103540	16	544	8.06265	0.142671	104230
147	1	5176	4	Integer	30,30	10	155310	16	544	8.11498	0.195002	156001
148	1	5176	4	Integer	40,40	10	207080	16	544	8.16737	0.247393	207772
149	1	5176	4	Integer	50,50	10	258850	16	544	8.21941	0.299428	259543
150	1	5176	4	Integer	60,60	10	310620	16	544	8.27344	0.353462	311314
151	1	5176	4	Integer	70,70	10	362390	16	544	8.32548	0.405498	363085
152	1	5176	4	Integer	80,80	10	414160	16	544	8.37983	0.45985	414856
153	1	5176	4	Integer	90,90	10	465930	16	544	8.43286	0.512884	466627
154	1	5176	4	Integer	100,100	10	517700	16	544	8.48519	0.565208	518398
155	1	301	3	String	0,0	10	0	16	352	0.0377822	0.0377822	507
156	1	301	3	String	10,10	10	3020	16	352	0.092715	0.092715	3528
157	1	301	3	String	20,20	10	6040	16	352	0.146068	0.146068	6549
158	1	301	3	String	30,30	10	9060	16	352	0.198442	0.198442	9570
159	1	301	3	String	40,40	10	12080	16	352	0.256502	0.256502	12591
160	1	301	3	String	50,50	10	15100	16	352	0.31454	0.31454	15612

161	1	301	3	String	60,60	10	18120	16	352	0.369575	0.369575	18633
162	1	301	3	String	70,70	10	21140	16	352	0.425613	0.425613	21654
163	1	301	3	String	80,80	10	24160	16	352	0.475891	0.475891	24675
164	1	301	3	String	90,90	10	27180	16	352	0.529392	0.529392	27696
165	1	301	3	String	100,100	10	30200	16	352	0.586429	0.586429	30717
166	1	233	3	String	0,0	10	0	16	352	0.662479	0.0410274	518
167	1	233	3	String	10,10	10	2340	16	352	0.713514	0.0920619	2859
168	1	233	3	String	20,20	10	4680	16	352	0.766548	0.145097	5200
169	1	233	3	String	30,30	10	7020	16	352	0.828589	0.207138	7541
170	1	233	3	String	40,40	10	9360	16	352	0.885627	0.264175	9882
171	1	233	3	String	50,50	10	11700	16	352	0.939664	0.318213	12223
172	1	233	3	String	60,60	10	14040	16	352	0.990697	0.369245	14564
173	1	233	3	String	70,70	10	16380	16	352	1.05174	0.430285	16905
174	1	233	3	String	80,80	10	18720	16	352	1.11378	0.492326	19246
175	1	233	3	String	90,90	10	21060	16	352	1.17582	0.554369	21587
176	1	233	3	String	100,100	10	23400	16	352	1.22485	0.603402	23928
177	1	160	3	String	0,0	10	0	16	352	1.28966	0.0350254	529
178	1	160	3	String	10,10	10	1610	16	352	1.34832	0.0936883	2140
179	1	160	3	String	20,20	10	3220	16	352	1.40836	0.153728	3751
180	1	160	3	String	30,30	10	4830	16	352	1.4684	0.213769	5362
181	1	160	3	String	40,40	10	6440	16	352	1.52344	0.268805	6973
182	1	160	3	String	50,50	10	8050	16	352	1.57347	0.318839	8584
183	1	160	3	String	60,60	10	9660	16	352	1.63752	0.382882	10195
184	1	160	3	String	70,70	10	11270	16	352	1.69655	0.441919	11806
185	1	160	3	String	80,80	10	12880	16	352	1.75359	0.498958	13417
186	1	160	3	String	90,90	10	14490	16	352	1.8032	0.548572	15028
187	1	160	3	String	100,100	10	16100	16	352	1.86125	0.606612	16639
188	1	458	3	String	0,0	10	0	16	352	1.93829	0.0430271	540
189	1	458	3	String	10,10	10	4590	16	352	1.99533	0.100065	5131
190	1	458	3	String	20,20	10	9180	16	352	2.04669	0.151428	9722
191	1	458	3	String	30,30	10	13770	16	352	2.10341	0.208143	14313
192	1	458	3	String	40,40	10	18360	16	352	2.16245	0.26718	18904
193	1	458	3	String	50,50	10	22950	16	352	2.22149	0.32622	23495
194	1	458	3	String	60,60	10	27540	16	352	2.27452	0.379256	28086
195	1	458	3	String	70,70	10	32130	16	352	2.32656	0.431291	32677
196	1	458	3	String	80,80	10	36720	16	352	2.40461	0.509341	37268
197	1	458	3	String	90,90	10	41310	16	352	2.46765	0.573383	41859
198	1	458	3	String	100,100	10	45900	16	352	2.52369	0.628421	46450
199	1	160	3	String	0,0	10	0	16	352	2.59215	0.0384948	551
200	1	160	3	String	10,10	10	1610	16	352	2.65419	0.100536	2162
201	1	160	3	String	20,20	10	3220	16	352	2.71623	0.162577	3773
202	1	160	3	String	30,30	10	4830	16	352	2.77126	0.217613	5384
203	1	160	3	String	40,40	10	6440	16	352	2.82182	0.268165	6995
204	1	160	3	String	50,50	10	8050	16	352	2.87201	0.318362	8606
205	1	160	3	String	60,60	10	9660	16	352	2.92171	0.368059	10217
206	1	160	3	String	70,70	10	11270	16	352	2.97394	0.420293	11828

207	1	160	3	String	80,80	10	12880	16	352	3.03065	0.477001	13439
208	1	160	3	String	90,90	10	14490	16	352	3.09069	0.53704	15050
209	1	160	3	String	100,100	10	16100	16	352	3.15073	0.597081	16661
210	1	160	3	String	0,0	10	0	16	352	3.21643	0.0360227	562
211	1	160	3	String	10,10	10	1610	16	352	3.27347	0.0930607	2173
212	1	160	3	String	20,20	10	3220	16	352	3.33351	0.153101	3784
213	1	160	3	String	30,30	10	4830	16	352	3.39455	0.214141	5395
214	1	160	3	String	40,40	10	6440	16	352	3.44859	0.268178	7006
215	1	160	3	String	50,50	10	8050	16	352	3.50362	0.323213	8617
216	1	160	3	String	60,60	10	9660	16	352	3.56366	0.383253	10228
217	1	160	3	String	70,70	10	11270	16	352	3.6237	0.443293	11839
218	1	160	3	String	80,80	10	12880	16	352	3.67974	0.499331	13450
219	1	160	3	String	90,90	10	14490	16	352	3.73336	0.552952	15061
220	1	160	3	String	100,100	10	16100	16	352	3.7974	0.616993	16672
221	1	233	3	String	0,0	10	0	16	352	3.87545	0.0430291	573
222	1	233	3	String	10,10	10	2340	16	352	3.93149	0.0990655	2914
223	1	233	3	String	20,20	10	4680	16	352	3.98252	0.1501	5255
224	1	233	3	String	30,30	10	7020	16	352	4.04256	0.210139	7596
225	1	233	3	String	40,40	10	9360	16	352	4.1016	0.269179	9937
226	1	233	3	String	50,50	10	11700	16	352	4.16264	0.330219	12278
227	1	233	3	String	60,60	10	14040	16	352	4.21468	0.382255	14619
228	1	233	3	String	70,70	10	16380	16	352	4.26793	0.435503	16960
229	1	233	3	String	80,80	10	18720	16	352	4.32496	0.492539	19301

230	1	233	3	String	90,90	10	21060	16	352	4.385	0.552578	21642
231	1	233	3	String	100,100	10	23400	16	352	4.44404	0.611617	23983
232	1	160	3	String	0,0	10	0	16	352	4.52109	0.0390261	584
233	1	160	3	String	10,10	10	1610	16	352	4.57245	0.0903783	2195
234	1	160	3	String	20,20	10	3220	16	352	4.63349	0.151419	3806
235	1	160	3	String	30,30	10	4830	16	352	4.69453	0.212459	5417
236	1	160	3	String	40,40	10	6440	16	352	4.74956	0.267496	7028
237	1	160	3	String	50,50	10	8050	16	352	4.80081	0.318744	8639
238	1	160	3	String	60,60	10	9660	16	352	4.85885	0.37678	10250
239	1	160	3	String	70,70	10	11270	16	352	4.91989	0.437821	11861
240	1	160	3	String	80,80	10	12880	16	352	4.98193	0.499862	13472
241	1	160	3	String	90,90	10	14490	16	352	5.03334	0.551275	15083
242	1	160	3	String	100,100	10	16100	16	352	5.08864	0.606573	16694
243	1	233	3	String	0,0	10	0	16	352	5.15965	0.0388553	595
244	1	233	3	String	10,10	10	2340	16	352	5.21672	0.0959221	2936
245	1	233	3	String	20,20	10	4680	16	352	5.27602	0.155232	5277
246	1	233	3	String	30,30	10	7020	16	352	5.33073	0.209932	7618
247	1	233	3	String	40,40	10	9360	16	352	5.3852	0.264405	9959
248	1	233	3	String	50,50	10	11700	16	352	5.43779	0.316993	12300
249	1	233	3	String	60,60	10	14040	16	352	5.49161	0.370821	14641
250	1	233	3	String	70,70	10	16380	16	352	5.61927	0.498475	16982
251	1	233	3	String	80,80	10	18720	16	352	5.67402	0.553225	19323
252	1	233	3	String	90,90	10	21060	16	352	5.72917	0.608379	21664

252	1	233	3	String	90,90	10	21060	16	352	5.72917	0.608379	21664
253	1	233	3	String	100,100	10	23400	16	352	5.78703	0.666237	24005
254	1	362	3	String	0,0	10	0	16	352	5.91489	0.0395696	606
255	1	362	3	String	10,10	10	3630	16	352	5.97178	0.0964639	4237
256	1	362	3	String	20,20	10	7260	16	352	6.02817	0.152853	7868
257	1	362	3	String	30,30	10	10890	16	352	6.08696	0.211639	11499
258	1	362	3	String	40,40	10	14520	16	352	6.24974	0.374427	15130
259	1	362	3	String	50,50	10	18150	16	352	6.3088	0.433478	18761
260	1	362	3	String	60,60	10	21780	16	352	6.36366	0.488341	22392
261	1	362	3	String	70,70	10	25410	16	352	6.41867	0.543353	26023
262	1	362	3	String	80,80	10	29040	16	352	6.5447	0.669386	29654
263	1	362	3	String	90,90	10	32670	16	352	6.60318	0.727862	33285
264	1	362	3	String	100,100	10	36300	16	352	6.6614	0.786088	36916
265	1	408	3	String	0,0	10	0	16	352	6.87927	0.0754087	617
266	1	408	3	String	10,10	10	4090	16	352	6.93425	0.130385	4708
267	1	408	3	String	20,20	10	8180	16	352	6.98723	0.183367	8799
268	1	408	3	String	30,30	10	12270	16	352	7.04063	0.236764	12890
269	1	408	3	String	40,40	10	16360	16	352	7.16735	0.363491	16981
270	1	408	3	String	50,50	10	20450	16	352	7.22286	0.418997	21072
271	1	408	3	String	60,60	10	24540	16	352	7.27841	0.474546	25163
272	1	408	3	String	70,70	10	28630	16	352	7.33474	0.53088	29254
273	1	408	3	String	80,80	10	32720	16	352	7.49374	0.689879	33345
274	1	408	3	String	90,90	10	36810	16	352	7.54965	0.745789	37436
275	1	408	3	String	100,100	10	40900	16	352	7.60456	0.800696	41527
276	1	233	3	String	0,0	10	0	16	352	7.79001	0.0641829	628
277	1	233	3	String	10,10	10	2340	16	352	7.88031	0.154484	2969
278	1	233	3	String	20,20	10	4680	16	352	7.93724	0.211417	5310
279	1	233	3	String	30,30	10	7020	16	352	8.08079	0.354961	7651
280	1	233	3	String	40,40	10	9360	16	352	8.24224	0.516418	9992
281	1	233	3	String	50,50	10	11700	16	352	8.36324	0.637417	12333
282	1	233	3	String	60,60	10	14040	16	352	8.42185	0.696023	14674
283	1	233	3	String	70,70	10	16380	16	352	8.47634	0.750513	17015
284	1	233	3	String	80,80	10	18720	16	352	8.53223	0.806403	19356
285	1	233	3	String	90,90	10	21060	16	352	8.66727	0.941443	21697
286	1	233	3	String	100,100	10	23400	16	352	8.72405	0.99822	24038
287	1	233	3	String	0,0	10	0	16	352	8.79595	0.0390257	639
288	1	233	3	String	10,10	10	2340	16	352	8.85056	0.0936289	2980
289	1	233	3	String	20,20	10	4680	16	352	9.01841	0.261482	5321
290	1	233	3	String	30,30	10	7020	16	352	9.08138	0.324449	7662
291	1	233	3	String	40,40	10	9360	16	352	9.13828	0.38135	10003
292	1	233	3	String	50,50	10	11700	16	352	9.26932	0.512389	12344
293	1	233	3	String	60,60	10	14040	16	352	9.33036	0.57343	14685
294	1	233	3	String	70,70	10	16380	16	352	9.38878	0.63185	17026
295	1	233	3	String	80,80	10	18720	16	352	9.44624	0.689306	19367
296	1	233	3	String	90,90	10	21060	16	352	9.60392	0.846988	21708
297	1	233	3	String	100,100	10	23400	16	352	9.65918	0.902256	24049

298	1	233	3	String	0,0	10	0	16	352	9.73107	0.0393031	650
299	1	233	3	String	10,10	10	2340	16	352	9.84405	0.152282	2991
300	1	233	3	String	20,20	10	4680	16	352	9.90013	0.208357	5332
301	1	233	3	String	30,30	10	7020	16	352	9.95867	0.266896	7673
302	1	233	3	String	40,40	10	9360	16	352	10.0148	0.323013	10014
303	1	233	3	String	50,50	10	11700	16	352	10.1858	0.494073	12355
304	1	233	3	String	60,60	10	14040	16	352	10.2565	0.564692	14696
305	1	233	3	String	70,70	10	16380	16	352	10.3122	0.620413	17037
306	1	233	3	String	80,80	10	18720	16	352	10.4429	0.751115	19378
307	1	233	3	String	90,90	10	21060	16	352	10.5017	0.809912	21719
308	1	233	3	String	100,100	10	23400	16	352	10.5577	0.86592	24060
309	1	262	3	Both	0,0	10	0	16	1568	0.0393056	0.0393056	1877
310	1	262	3	Both	10,10	10	2630	16	1568	0.0964253	0.0964253	4508
311	1	262	3	Both	20,20	10	5260	16	1568	0.154114	0.154114	7139
312	1	262	3	Both	30,30	10	7890	16	1568	0.210685	0.210685	9770
313	1	262	3	Both	40,40	10	10520	16	1568	0.382386	0.382386	12401
314	1	262	3	Both	50,50	10	13150	16	1568	0.44139	0.44139	15032
315	1	262	3	Both	60,60	10	15780	16	1568	0.500106	0.500106	17663
316	1	262	3	Both	70,70	10	18410	16	1568	0.558943	0.558943	20294
317	1	262	3	Both	80,80	10	21040	16	1568	0.61503	0.61503	22925
318	1	262	3	Both	90,90	10	23670	16	1568	0.763017	0.763017	25556
319	1	262	3	Both	100,100	10	26300	16	1568	0.819044	0.819044	28187
320	1	263	3	Both	0,0	10	0	16	1568	0.893281	0.0402445	1888
321	1	263	3	Both	10,10	10	2640	16	1568	0.949688	0.0966515	4529
322	1	263	3	Both	20,20	10	5280	16	1568	1.12096	0.267923	7170
323	1	263	3	Both	30,30	10	7920	16	1568	1.18826	0.335222	9811
324	1	263	3	Both	40,40	10	10560	16	1568	1.24457	0.391535	12452
325	1	263	3	Both	50,50	10	13200	16	1568	1.30177	0.448738	15093
326	1	263	3	Both	60,60	10	15840	16	1568	1.45364	0.600599	17734
327	1	263	3	Both	70,70	10	18480	16	1568	1.51025	0.657214	20375
328	1	263	3	Both	80,80	10	21120	16	1568	1.56669	0.71365	23016
329	1	263	3	Both	90,90	10	23760	16	1568	1.72797	0.874935	25657
330	1	263	3	Both	100,100	10	26400	16	1568	1.7847	0.93166	28298
331	1	264	3	Both	0,0	10	0	16	1568	1.85716	0.0398665	1899
332	1	264	3	Both	10,10	10	2650	16	1568	1.91793	0.100637	4550
333	1	264	3	Both	20,20	10	5300	16	1568	1.97362	0.156324	7201
334	1	264	3	Both	30,30	10	7950	16	1568	2.03405	0.21676	9852
335	1	264	3	Both	40,40	10	10600	16	1568	2.09231	0.275021	12503
336	1	264	3	Both	50,50	10	13250	16	1568	2.14897	0.331683	15154
337	1	264	3	Both	60,60	10	15900	16	1568	2.20501	0.38772	17805
338	1	264	3	Both	70,70	10	18550	16	1568	2.25763	0.440336	20456
339	1	264	3	Both	80,80	10	21200	16	1568	2.3121	0.494807	23107
340	1	264	3	Both	90,90	10	23850	16	1568	2.36908	0.55179	25758
341	1	264	3	Both	100,100	10	26500	16	1568	2.51184	0.694552	28409
342	1	431	3	Both	0,0	10	0	16	1568	2.58714	0.0421198	1910
343	1	431	3	Both	10,10	10	4320	16	1568	2.64659	0.10157	6231

344	1	431	3	Both	20,20	10	8640	16	1568	2.70831	0.163296	10552
345	1	431	3	Both	30,30	10	12960	16	1568	2.84819	0.303176	14873
346	1	431	3	Both	40,40	10	17280	16	1568	2.92398	0.378966	19194
347	1	431	3	Both	50,50	10	21600	16	1568	3.01543	0.470409	23515
348	1	431	3	Both	60,60	10	25920	16	1568	3.07021	0.525194	27836
349	1	431	3	Both	70,70	10	30240	16	1568	3.24707	0.702052	32157
350	1	431	3	Both	80,80	10	34560	16	1568	3.32689	0.781868	36478
351	1	431	3	Both	90,90	10	38880	16	1568	3.39162	0.846602	40799
352	1	431	3	Both	100,100	10	43200	16	1568	3.65825	1.11323	45120
353	1	429	3	Both	0,0	10	0	16	1568	3.997	0.0642382	1921
354	1	429	3	Both	10,10	10	4300	16	1568	4.06452	0.131751	6222
355	1	429	3	Both	20,20	10	8600	16	1568	4.23315	0.300381	10523
356	1	429	3	Both	30,30	10	12900	16	1568	4.31445	0.381689	14824
357	1	429	3	Both	40,40	10	17200	16	1568	4.37279	0.440026	19125
358	1	429	3	Both	50,50	10	21500	16	1568	4.50417	0.571399	23426
359	1	429	3	Both	60,60	10	25800	16	1568	4.56696	0.634198	27727
360	1	429	3	Both	70,70	10	30100	16	1568	4.62938	0.696614	32028
361	1	429	3	Both	80,80	10	34400	16	1568	4.69306	0.760296	36329
362	1	429	3	Both	90,90	10	38700	16	1568	4.83827	0.905499	40630
363	1	429	3	Both	100,100	10	43000	16	1568	4.90154	0.968778	44931
364	1	422	3	Both	0,0	10	0	16	1568	4.98128	0.0452341	1932
365	1	422	3	Both	10,10	10	4230	16	1568	5.03639	0.100337	6163
366	1	422	3	Both	20,20	10	8460	16	1568	5.17949	0.243443	10394

367	1	422	3	Both	30,30	10	12690	16	1568	5.23652	0.300476	14625
368	1	422	3	Both	40,40	10	16920	16	1568	5.29432	0.358268	18856
369	1	422	3	Both	50,50	10	21150	16	1568	5.35071	0.414665	23087
370	1	422	3	Both	60,60	10	25380	16	1568	5.50931	0.573263	27318
371	1	422	3	Both	70,70	10	29610	16	1568	5.56744	0.631388	31549
372	1	422	3	Both	80,80	10	33840	16	1568	5.62776	0.691712	35780
373	1	422	3	Both	90,90	10	38070	16	1568	5.80669	0.870647	40011
374	1	422	3	Both	100,100	10	42300	16	1568	5.8721	0.936053	44242
375	1	260	3	Both	0,0	10	0	16	1568	5.97063	0.0422183	1943
376	1	260	3	Both	10,10	10	2610	16	1568	6.0287	0.10029	4554
377	1	260	3	Both	20,20	10	5220	16	1568	6.08572	0.157308	7165
378	1	260	3	Both	30,30	10	7830	16	1568	6.14269	0.214272	9776
379	1	260	3	Both	40,40	10	10440	16	1568	6.19872	0.270308	12387
380	1	260	3	Both	50,50	10	13050	16	1568	6.2551	0.326685	14998
381	1	260	3	Both	60,60	10	15660	16	1568	6.31152	0.383104	17609
382	1	260	3	Both	70,70	10	18270	16	1568	6.36778	0.439362	20220
383	1	260	3	Both	80,80	10	20880	16	1568	6.42591	0.497495	22831
384	1	260	3	Both	90,90	10	23490	16	1568	6.48549	0.55707	25442
385	1	260	3	Both	100,100	10	26100	16	1568	6.54266	0.614247	28053
386	1	423	3	Both	0,0	10	0	16	1568	6.61762	0.0413813	1954
387	1	423	3	Both	10,10	10	4240	16	1568	6.67443	0.0981939	6195
388	1	423	3	Both	20,20	10	8480	16	1568	6.7317	0.155462	10436
389	1	423	3	Both	30,30	10	12720	16	1568	6.78907	0.212839	14677
390	1	423	3	Both	40,40	10	16960	16	1568	6.84571	0.269471	18918
391	1	423	3	Both	50,50	10	21200	16	1568	6.90424	0.328	23159
392	1	423	3	Both	60,60	10	25440	16	1568	6.96179	0.38555	27400
393	1	423	3	Both	70,70	10	29680	16	1568	7.01866	0.442428	31641
394	1	423	3	Both	80,80	10	33920	16	1568	7.07729	0.501054	35882
395	1	423	3	Both	90,90	10	38160	16	1568	7.13419	0.557956	40123
396	1	423	3	Both	100,100	10	42400	16	1568	7.19255	0.616317	44364
397	1	423	3	Both	0,0	10	0	16	1568	7.26552	0.0409293	1965
398	1	423	3	Both	10,10	10	4240	16	1568	7.32306	0.0984686	6206
399	1	423	3	Both	20,20	10	8480	16	1568	7.38054	0.155953	10447
400	1	423	3	Both	30,30	10	12720	16	1568	7.43865	0.21406	14688
401	1	423	3	Both	40,40	10	16960	16	1568	7.49701	0.272425	18929
402	1	423	3	Both	50,50	10	21200	16	1568	7.55374	0.329155	23170
403	1	423	3	Both	60,60	10	25440	16	1568	7.61197	0.387383	27411
404	1	423	3	Both	70,70	10	29680	16	1568	7.69497	0.470388	31652
405	1	423	3	Both	80,80	10	33920	16	1568	7.7523	0.527713	35893
406	1	423	3	Both	90,90	10	38160	16	1568	7.80867	0.584081	40134
407	1	423	3	Both	100,100	10	42400	16	1568	7.86668	0.642095	44375
408	1	262	3	Both	0,0	10	0	16	1568	7.93899	0.0402717	1976
409	1	262	3	Both	10,10	10	2630	16	1568	7.99538	0.0966627	4607
410	1	262	3	Both	20,20	10	5260	16	1568	8.0528	0.154078	7238
411	1	262	3	Both	30,30	10	7890	16	1568	8.11163	0.21291	9869
412	1	262	3	Both	40,40	10	10520	16	1568	8.16918	0.27046	12500

413	1	262	3	Both	50,50	10	13150	16	1568	8.22744	0.328718	15131
414	1	262	3	Both	60,60	10	15780	16	1568	8.28776	0.389041	17762
415	1	262	3	Both	70,70	10	18410	16	1568	8.34515	0.446428	20393
416	1	262	3	Both	80,80	10	21040	16	1568	8.40297	0.504256	23024
417	1	262	3	Both	90,90	10	23670	16	1568	8.46031	0.561588	25655
418	1	262	3	Both	100,100	10	26300	16	1568	8.51807	0.619352	28286
419	1	262	3	Both	0,0	10	0	16	1568	8.59204	0.0422926	1987
420	1	262	3	Both	10,10	10	2630	16	1568	8.64836	0.0986126	4618
421	1	262	3	Both	20,20	10	5260	16	1568	8.71119	0.161446	7249
422	1	262	3	Both	30,30	10	7890	16	1568	8.77724	0.227488	9880
423	1	262	3	Both	40,40	10	10520	16	1568	8.83537	0.28562	12511
424	1	262	3	Both	50,50	10	13150	16	1568	8.89795	0.348203	15142
425	1	262	3	Both	60,60	10	15780	16	1568	8.95505	0.405302	17773
426	1	262	3	Both	70,70	10	18410	16	1568	9.0151	0.465353	20404
427	1	262	3	Both	80,80	10	21040	16	1568	9.07353	0.523782	23035
428	1	262	3	Both	90,90	10	23670	16	1568	9.12964	0.579888	25666
429	1	262	3	Both	100,100	10	26300	16	1568	9.18826	0.638517	28297
430	1	263	3	Both	0,0	10	0	16	1568	9.2604	0.0400495	1998
431	1	263	3	Both	10,10	10	2640	16	1568	9.31709	0.0967431	4639
432	1	263	3	Both	20,20	10	5280	16	1568	9.37313	0.152777	7280
433	1	263	3	Both	30,30	10	7920	16	1568	9.43313	0.212783	9921
434	1	263	3	Both	40,40	10	10560	16	1568	9.48896	0.268609	12562
435	1	263	3	Both	50,50	10	13200	16	1568	9.54538	0.325027	15203

436	1	263	3	Both	60,60	10	15840	16	1568	9.60381	0.383459	17844
437	1	263	3	Both	70,70	10	18480	16	1568	9.66349	0.443134	20485
438	1	263	3	Both	80,80	10	21120	16	1568	9.72213	0.50178	23126
439	1	263	3	Both	90,90	10	23760	16	1568	9.78386	0.563512	25767
440	1	263	3	Both	100,100	10	26400	16	1568	9.84246	0.622104	28408
441	1	264	3	Both	0,0	10	0	16	1568	9.91543	0.0402946	2009
442	1	264	3	Both	10,10	10	2650	16	1568	9.97213	0.0969919	4660
443	1	264	3	Both	20,20	10	5300	16	1568	10.0323	0.157155	7311
444	1	264	3	Both	30,30	10	7950	16	1568	10.0873	0.212192	9962
445	1	264	3	Both	40,40	10	10600	16	1568	10.1431	0.267946	12613
446	1	264	3	Both	50,50	10	13250	16	1568	10.1981	0.322981	15264
447	1	264	3	Both	60,60	10	15900	16	1568	10.2557	0.380594	17915
448	1	264	3	Both	70,70	10	18550	16	1568	10.3118	0.43663	20566
449	1	264	3	Both	80,80	10	21200	16	1568	10.3685	0.49337	23217
450	1	264	3	Both	90,90	10	23850	16	1568	10.4247	0.549541	25868
451	1	264	3	Both	100,100	10	26500	16	1568	10.4865	0.611362	28519
452	1	431	3	Both	0,0	10	0	16	1568	10.5661	0.0470043	2020
453	1	431	3	Both	10,10	10	4320	16	1568	10.6245	0.105432	6341
454	1	431	3	Both	20,20	10	8640	16	1568	10.6813	0.162169	10662
455	1	431	3	Both	30,30	10	12960	16	1568	10.74	0.220893	14983
456	1	431	3	Both	40,40	10	17280	16	1568	10.7969	0.277817	19304
457	1	431	3	Both	50,50	10	21600	16	1568	10.8558	0.336709	23625
458	1	431	3	Both	60,60	10	25920	16	1568	10.9127	0.393653	27946
459	1	431	3	Both	70,70	10	30240	16	1568	10.9693	0.450208	32267
460	1	431	3	Both	80,80	10	34560	16	1568	11.0249	0.505845	36588
461	1	431	3	Both	90,90	10	38880	16	1568	11.0853	0.566199	40909
462	1	431	3	Both	100,100	10	43200	16	1568	11.1418	0.622717	45230

E. Sample of the implementation code

The researcher divided the implementation code into four sections depending on the proposed solution. For full implementation code, you can contact the author via the email.

```
Private Sub Button3_Click(sender As System.Object, e As System.EventArgs) Handles Button3.Click
    Dim arrList As ArrayList
    'تثبيت نوع البيانات المدخله و المفتاح والمعادنه متغيره
    '   case1
    If CheckBox1.Checked = False And CheckBox2.Checked = True And CheckBox3.Checked = True Then
        If ListDataType.SelectedIndex = 0 And Opt16.Checked = True Then
            LenghtKey = 16
            DataType = "Integer"
            VX = 13
        ElseIf ListDataType.SelectedIndex = 0 And Opt32.Checked = True Then
            LenghtKey = 32
            DataType = "Integer"
            VX = 13
        ElseIf ListDataType.SelectedIndex = 0 And Opt64.Checked = True Then
            LenghtKey = 64
            DataType = "Integer"
            VX = 13
        ElseIf ListDataType.SelectedIndex = 1 And Opt16.Checked = True Then
            LenghtKey = 16
            DataType = "String"
            VX = 11
        ElseIf ListDataType.SelectedIndex = 1 And Opt32.Checked = True Then
            LenghtKey = 32
            DataType = "String"
            VX = 11
        ElseIf ListDataType.SelectedIndex = 1 And Opt64.Checked = True Then
            LenghtKey = 64
            DataType = "String"
            VX = 11
        ElseIf ListDataType.SelectedIndex = 2 And Opt16.Checked = True Then
            LenghtKey = 16
            DataType = "Both"
            VX = 10
```

```
Dim cont As Integer = 0
           Dim DataTypeValue As Integer = 0
           arrList = New ArrayList
           For i = 1 To LenghtKey
               Dim Rand As Integer = New Random().Next(1, 100)
               cont = CInt(cont) + CInt(Rand)
           Next

           If CheckBox4.Checked = True Then
               '-----
               sql = "select * from EquationPerTime order by ID"
               adp = New SqlDataAdapter(sql, cn)
               ds = New DataSet
               adp.Fill(ds, sql)
               dtb = ds.Tables(0)
               '-----------------------
               Dim maxID As Integer
               Try
                   maxID = Val(dtb.Rows(dtb.Rows.Count - 1)("IDPerEquation")) + 1
               Catch ex As Exception
                   maxID = 1
               End Try
               arr = New ArrayList
               ToolStripProgressBar1.Maximum = txtTo1.Text
               ToolStripProgressBar1.Step = jumber1.Text
               ToolStripProgressBar1.Visible = True
               ToolStripProgressBar1.Style = ProgressBarStyle.Continuous
               ToolStripProgressBar1.MarqueeAnimationSpeed = 100
               ToolStripLabel7.Text = "Please wait..."
               startTime = DateTime.Now
               For k = 0 To Me.DataGridView1.Rows.Count - 1
                   If DataType = "Both" Or DataType = "String" Then
                       Dim v As Integer = 0
                       For z = 0 To DataGridView1.Item(VX, k).Value.ToString.Length - 1
                           v = Val(v) + Asc(Me.DataGridView1.Item(VX, k).Value.ToString.Substring(z).ToString.ToUpper)
                       Next
                       VInt = v
                   Else
                       VInt = Me.DataGridView1.Item(VX, k).Value
                   End If
```

```
'تثبيت نوع البيانات المدخله و المعادله والمفتاح متغير
    If CheckBox1.Checked = True And CheckBox2.Checked = True And CheckBox3.Checked = False Then
        Dim EquationType As Integer
        If ListDataType.SelectedIndex = 0 And Combobox1.SelectedIndex = 0 Then
            EquationType = 1
            DataType = "Integer"
            VX = 13
        ElseIf ListDataType.SelectedIndex = 0 And Combobox1.SelectedIndex = 1 Then
            EquationType = 2
            DataType = "Integer"
            VX = 13
        ElseIf ListDataType.SelectedIndex = 0 And Combobox1.SelectedIndex = 2 Then
            EquationType = 5
            DataType = "Integer"
            VX = 13
        ElseIf ListDataType.SelectedIndex = 0 And Combobox1.SelectedIndex = 3 Then
            EquationType = 8
            DataType = "Integer"
            VX = 13
        ElseIf ListDataType.SelectedIndex = 0 And Combobox1.SelectedIndex = 4 Then
            EquationType = 12
            DataType = "Integer"
            VX = 13
        ElseIf ListDataType.SelectedIndex = 1 And Combobox1.SelectedIndex = 0 Then
            EquationType = 1
            DataType = "String"
            VX = 11
        ElseIf ListDataType.SelectedIndex = 1 And Combobox1.SelectedIndex = 1 Then
            EquationType = 2
            DataType = "String"
            VX = 11
        ElseIf ListDataType.SelectedIndex = 1 And Combobox1.SelectedIndex = 2 Then
            EquationType = 5
            DataType = "String"
            VX = 11
        ElseIf ListDataType.SelectedIndex = 1 And Combobox1.SelectedIndex = 3 Then
            EquationType = 8
            DataType = "String"
            VX = 11
        ElseIf ListDataType.SelectedIndex = 1 And Combobox1.SelectedIndex = 4 Then
            EquationType = 12
            DataType = "String"
            VX = 11
        ElseIf ListDataType.SelectedIndex = 2 And Combobox1.SelectedIndex = 0 Then
            EquationType = 1
            DataType = "Both"
            VX = 10
```

```
'تثبيت المفتاح والمعادله ونوع البيانات متغير'
    If CheckBox1.Checked = True And CheckBox2.Checked = False And CheckBox3.Checked = True Then
        Dim EquationType As Integer
        If Combobox1.SelectedIndex = 0 And Opt16.Checked = True Then
            EquationType = 1
            LenghtKey = 16
            ' VX = 13
        ElseIf Combobox1.SelectedIndex = 0 And Opt32.Checked = True Then
            EquationType = 1
            LenghtKey = 32
            '   VX = 13
        ElseIf Combobox1.SelectedIndex = 0 And Opt64.Checked = True Then
            EquationType = 1
            LenghtKey = 64
            '      VX = 11
        ElseIf Combobox1.SelectedIndex = 1 And Opt16.Checked = True Then
            EquationType = 2
            LenghtKey = 16
            '       VX = 11
        ElseIf Combobox1.SelectedIndex = 1 And Opt32.Checked = True Then
            EquationType = 2
            LenghtKey = 32
        ElseIf Combobox1.SelectedIndex = 1 And Opt64.Checked = True Then
            EquationType = 2
            LenghtKey = 64
        ElseIf Combobox1.SelectedIndex = 2 And Opt16.Checked = True Then
            EquationType = 5
            LenghtKey = 16
            '       VX = 11
        ElseIf Combobox1.SelectedIndex = 2 And Opt32.Checked = True Then
            EquationType = 5
            LenghtKey = 32
        ElseIf Combobox1.SelectedIndex = 2 And Opt64.Checked = True Then
            EquationType = 5
            LenghtKey = 64
        ElseIf Combobox1.SelectedIndex = 3 And Opt16.Checked = True Then
            EquationType = 8
            LenghtKey = 16
            '       VX = 11
        ElseIf Combobox1.SelectedIndex = 3 And Opt32.Checked = True Then
            EquationType = 8
            LenghtKey = 32
        ElseIf Combobox1.SelectedIndex = 3 And Opt64.Checked = True Then
            EquationType = 8
            LenghtKey = 64
        ElseIf Combobox1.SelectedIndex = 4 And Opt16.Checked = True Then
```