Performance Analysis for Fully and Partially Homomorphic Encryption Techniques

تحليل الاداء لتقنيات التشفير المتماثلة الكاملة والجزئية

Prepared by

Raya A. Al-Shibib

Supervisor

Prof. Ahmad Kayed

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

Department of Computer Science

Faculty of Information Technology

Middle East University

Amman, Jordan

May, 2016

AUTHORIZATION STATEMENT

I, Raya A. Al-Shibib, authorize the Middle East University to provide hard copies or soft copies of my thesis to libraries, institutions or individuals upon their request.

Name: **Raya A. Al-Shibib**

Signature: 

Data: 13 / 6 / 2016

أقرار تفويض

أنا ريا عادل الشبيب، أفوض جامعة الشرق الأوسط للدراسات العليا بتزويد نسخ من رسالتي ورقياً وإلكترونياً للمكتبات أو المنظمات أو الهيئات والمؤسسات المعنية بالأبحاث والدراسات العليا عند طلبها.

الأسم : ريا عادل الشبيب

التاريخ: 13/6/2016

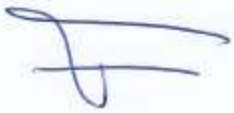
Examination Committee Decision

This is to certify that the thesis entitled "Performance Analysis For Fully And Partially Homomorphic Encryption Techniques" was successfully defended and approved on 24 /5 /2016

Examination Committee Members Signature

(Supervisor)

Prof. Ahmad Kayed



(Head of the Committee and Internal Committee Members)

Dr. Mudhafar Munir Al-Jarrah

(External Committee Members)

Dr. Mohammed Ahmad Alia

ACKNOWLEDGMENT

Thanks to God I have completed this thesis praise and thanks for his blessings.

I would like to thank Prof. Ahmed Kayed for support and assistance and gave me valuable information for the completion of this thesis.

I would like to thank my friends, they always standing with me through the good Times and hard times.

Dedication

I dedicate this thesis to my mother who standing beside me and my Father God's mercy. I hope to reach my research into the world to benefit from it and be ongoing charity to dear my father.

TABLE OF CONTENTS

CHAPTER TWO
BACKGROUND

CHAPTER THREE
Performance analysis for fully and partially homomorphic encryption techniques

CHAPTER FOUR
Experimental & Result

CHAPTER FIVE
Conclusion & Future work

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviations	Meaning
AES	Advanced Encryption Standard
ATV	Alt-Lopez, Tromer and Vaikuntanathan
CPU	Central Processing Unit
CCA1	Circadian Clock Associated 1
DES	Data Encryption Standard
DRM	Digital Rights Management
ECC	Elliptic Curve Cryptography
FHE	Fully Homomorphic Encryption
HE	Homomorphic Encryption
IT	Information Technology
IaaS	Infrastructure as a Service
IP Sec	Internet Protocol Security
LFSR	Linear Feedback Shift Register
LTV	Lopez-Alt, Tromer and Vaikuntanathan
MB	Megabyte
NTRU	Nth Degree Truncated polynomial Ring Unit
OPE	Ordering Preserving Encryption
PHE	Partially Homomorphic Encryption
PDA	Patent Ductus Arteriosus
PaaS	Platform as a Service

RSA	Rivest, Shamir, and Adleman
SDS	Secure Data Sharing
SHA	Secure Hash Algorithm
SMC	Secure Multiparty Computation
SSL	Secure Sockets Layer
SaaS	Software as a Service
SHES	Somewhat Homomorphic Encryption Scheme
TB	Terabytes
TLS	Transport Layer Security

Performance Analysis for Fully and Partially Homomorphic Encryption Techniques

By: Raya A. AL-Shibib

Supervisor: prof. Ahmed Kayed

Abstract

Recently, the use of cloud computing for the storage of business and personal data has shown a great increase, due to many factors such as ease of access anywhere anytime, reduced storage cost, hardware independence, and the availability of reliable network infrastructure. However, the cloud computing approach faces serious security challenges and threats that need to be investigated and discussed. One of the main methods to keep the data safe from attacks by adversaries is the encryption technique, which is considered a practical and an appropriate method for protecting data in the cloud.

This thesis investigates the main encryption techniques that deal with allowing mathematical operations to be performed on data in the cloud without the need for decryption. It has investigated two types of Partial Homomorphic Encryption techniques (Paillier and RSA) and Fully Homomorphic Encryption (Nth Degree Truncated Polynomial Ring Unit) with two operations of NTRU technique (Additive NTRU and Multiplication NTRU, and the combination between them.

The research work studied several parameters that affect cloud security based on these techniques. The selected parameters are investigated with the aim of distinguishing the impact of these parameters on the performance of the Homomorphic and NTRU techniques in terms of security and execution time. This thesis has identified two parameters that have an effect on encryption security

level and execution performance; these are key size and data size. It has tested four key sizes (32 bits, 64 bits, 128 bits and 256 bits) and data sizes ranging from one to nine digits.

The research results show that the optimal key size is 256 bits for RSA and NTRU (additive, multiplication) which give a high security level and minimal performance loss. Also, the research results show that the optimal key size is 128 bits for Paillier and NTRU (combined) which give a high security level and minimal performance loss. For the optimal data size, the results show that for the RSA method, the optimal data size is five digits, which gives minimal performance-loss (4%).

Based on the execution time factor, the research finds out that the Additive PHE (Paillier) performs better than the Additive NTRU, the Multiplication PHE (RSA) performs better than the Multiplication NTRU, and finally the two PHEs together perform better than the two NTRUs.

Keywords: Cloud Computing, Cloud Security, Fully Homomorphic Encryption, Partially Homomorphic Encryption

تحليل الاداء لتقنيات التشفير المتماثلة الكاملة والجزئية

اعداد

ريا عادل الشبيب

إشراف

الأستاذ الدكتور أحمد الكايد

الملخص

في الآونة الاخيرة، تعتبر مفاهيم امن الحوسبة السحابية من المواضيع الهامة التي لابد من معالجتها ومناقشتها في مجال تكنولوجيا المعلومات. ان السبب الرئيسي لزيادة استخدام الحوسبة السحابية هو تقديم الخدمات للمنظمات والافراد من قبل اطراف خارجية وعلى خوادم بعيدة عنها. وبأمكان اي زبون الوصول الى هذه الخدمات من اي مكان وفي اي وقت.

أحد أهم التحديات في بيئة الحوسبة السحابية هو مفهوم الأمن. هناك العديد من الطرق للحفاظ على بيانات المستخدم الهامة، تقنيات التشفير تعتبر الطريقة المناسبة لتوفير الأمن مناسب لهذه البيانات.

تتحقق هذه الاطروحة في تقنيات التشفير الرئيسية التي تحافظ على العمليات الحسابية. والتحقيق فيها نوعين من تقنيات التشفير التماثلي الجزئي (Paillier and RSA) والتشفير التماثلي الكامل (Nth Degree Truncated polynomial Ring Unit) مع اثنين من عمليات تقنية (NTRU) (NTRU and Additive Multiplication NTRU) والدمج مابينهما . قام الباحث بدراسة العديد من المعايير التي تؤثر على أمن السحابة استنادا على هذه التقنيات.

قام الباحث بدراسة معايير لتحديد تأثير هذه المعايير على اداء تقنيات (Homomorphic and NTRU). هذه الاطروحة حددت المعايير من خلال حجم المفتاح وحجم البيانات. وقد اختبرت اربع انواع من احجام المفاتيح وهي (32 bits, 64 bits, 128 bits and 256 bits) وتسعة ارقام من حجم البيانات مابين (1-9).

وجد الى ان حجم المفتاح الامثل هو (256) بت لRSA وNTRU (additive, Multiplication) والذي يعطي مستوى أمان عالي مع خسارة اداء قليله. ايضا, الاطروحة وجدت ان حجم المفتاح الامثل هو 128 بت لPaillier و NTRU (both) الذي يعطي مستوى أمان عالي مع خسارة اداء قليلة. وحجم البيانات الامثل هو لRSA هو (5 ارقام) مع خسارة اداء قليله (4%).

اعتماداً على وقت التنفيذ، توصل الباحث الى ان حساب الوقت في Additive PHE (Paillier) هي افضل من Additive NTRU، الوقت في Multiplication PHE (RSA) هي افضل من Multiplication NTRU، واخيرا فأن الوقت في Both PHE هو افضل من Both NTRU.

الكلمات المفتاحية: الحوسبة السحابية، أمن السحابة، التشفير المتماثل الكاملة، التشفير المتماثل الجزئية

Chapter One

Introduction

1.1 Introduction

Cloud computing is a new concept in technology field; it utilizes the internet and the remote servers in order to provide services and resources to the individual and organizations. The cloud concept enables consumers to access services and resources online through the internet, from anywhere at any time. Thus, the consumers are able to decrease the cost of hardware deployment, software licenses system maintenance and so on.

Cloud computing is gaining more popularity due to its architectural design and characteristics. The cloud computing is a new business model which lacks security issue, although many users selecting to use it. So, they being reconsidered of effectiveness and efficiency of traditional protection mechanisms which includes centralization of security, data and process segmentation, redundancy and high availability (Zissis & Lekkas, 2012).

The relying on digital technologies is increasing in cloud computing world, thus, the customers are concerned about the security of sensitive data and feel the deprivation of control over their data. All these points lead researchers to select a major challenge in cloud computing which represent in security issue, the security issue is utilizing to prevent unauthorized accesses and eliminate possibilities of data corruption through finding appropriate security methods before moving any data to the cloud (Sarwar, & Khan, 2013).

The enterprises and organizations mostly concerned about the security, privacy, confidentiality and availability of their data. This is reflected through great attention from different researchers for using method anonymity in modern encryption technique fields. Therefore, the problem is that a user would be unable to leverage the characteristic of the cloud to carry out computation on data without decrypting it, or loading it entirely back to the user for computation (Sarwar, & Khan, 2013).

The encryption efficiency is very important to data safety. There are many techniques, which would allow operation on data without knowing the actual content, can important to different areas such as technique Homomorphic encryption. The homomorphic technique is allowing any party to publicly transform a collection of cipher-texts which resulted from some plain-texts, without the party knowing the plain-texts themselves (Stehlé & Steinfeld, 2010).

The Homomorphic encryption provides the ability for encrypting data; it is classified into two types: Fully Homomorphic Encryption (FHE) and Partially Homomorphic Encryption (PHE) based on the characteristics them. the partially techniques is implemented in different forms such as additive form, for example paillier homomorphic cryptosystem, and Multiplication form, for example RSA homomorphic cryptosystem (Rohloff & Cousins, 2014 and Tebaa, et al, 2012).

In another hand, The Nth Degree Truncated polynomial Ring Unit (NTRU) technique can improve communication efficiency with enhancing data security; it provides substantially better performance. NTRU encryption requires fewer servers while still protecting (encrypting) all data. The secure encryption is reducing chances of costly data breaches, improves privacy and compliance and saves money by reducing the need for some intrusion detection systems and other security solutions.

There are some studies on how to use NTRU technique with homomorphic encryption, but there still a need to do more research on implementing NTRU with PHE; The NTRU scheme was demand different parameters to support homomorphic evaluation. Thus, the researchers attempt to focus of the impact of the new parameter setting to access the security level (Doröz, et al, 2014).

1.2 Problem Statement

Today, cloud computing is considered a business model to provide services. Therefore, it must be secure by strong methods. The security defined as one of the important problems for cloud database. The encryption technique is presented as a solution to solve the security problem. At the same time, applying encryption techniques on database creates new problems where many mathematical operations can't be done over encrypted data. Homomorphic (Partially or Fully) Encryption (PHE or FHE) techniques solve this problem but decrease the database performance as well as the security level. This research will compare the effect of several parameters on the Nth Degree Truncated Polynomial Ring Unit (NTRU) and PHEs, it will choose two types of PHEs (paillier, RSA) and choose NTRU.

1.3 Research Questions

This research will answer the following questions:

1- How to apply NTRU and PHE?

2- How to compare of NTRU with several Homomorphic Encryptions?

3- What are the parameters that will affect the performance of PHEs as well as NTRU?

1.4 Limitations and Scope

This thesis aimed to compare NTRU technique and PHEs technique on the cloud data. It studies two parameters (Data size and Key size). It limit the encryption into three techniques, these are NTRU, RSA and Paillier. The proposed model is attempt proofing the effect of each parameter via PHE and NTRU operations on the performance.

1.5 Objective

The goal of this research is to study the maintaining confidentiality of the existing data in the cloud network through utilizing homomorphic technique, and identify the role of the NTRU, then comparing it with the performance of PHEs.

The objective of this research is to identify the following:

• The effect of several parameters on NTRU and PHE through comparison between them.

• Explain the impact of these parameters on NTRU and PHEs.

• Study how to improve the operations of NTRU and PHE with all parameters.

• Find the optimal performance with maintaining the security level and data flexibility for using encryption techniques in a practical way in cloud computing environment.

1.6 Contribution

The contribution is summarized to help the data owner inside cloud for encrypting data through realizing different parameters, as the following:

- The security assessment on the cloud computing is determined by the security parameters that effect on the cloud security.
- The homomorphic encryption is an important idea in cloud computing. Therefore, analyzing the performance of the homomorphic encryption and NTRU on cloud data is considered a very important.
- Many cases studied the efficiency for several parameters as details for security level and data flexibility on the performance.

- The efficiency is selected as optimal to increment the performance with a high gain of security and data flexibility.

1.7 Motivation

The cloud computing is a technology which has demonstrated its prosperity by decreasing the expense, ease of use, adaptability to add and partner new services in work, the likelihood of changing the way of work inside of a couple of hours and minimal effort contrasted with conventional innovation. There is still an issue with the execution and security, encrypted data is required in cloud to protect sensitive information, in this research our motivation is to find some solutions to increase the performance and maintain the security level and data flexibility.

1.8 Methodology

The proposed methodology based on descriptive approach and quantitative approach. It implemented to find optimal performance of NTRU and PHE through many experiments. This methodology is describing their experiments to study the effect of NTRU and PHE which considered as descriptive approach. Then compares the results, it will work on several experiments to determine what the parameters that affect the PHE and NTRU.

1.9 Thesis Outline

This thesis includes five chapters:-

Chapter one declares the main problem in cloud computing from the researcher perspective which is representing in security.

Chapter two explains the main concepts of the cloud computing and presented some of the literature reviews that related with this thesis.

Chapter three presents the proposed methodology that has been followed in practical part of this thesis and represents it in algorithm and flowcharts.

Chapter four presents a number of experiments through interfaces and evaluates each experiment by results.

Chapter five summarizes the main conclusions of this thesis and presented some recommendations for future.

Chapter Two

Background and Literature review

2.1Background

Recently, many organizations and individuals users are increasingly realizing the benefits of putting their applications and data into the cloud. Thus, they depending on the cloud as a new technology may lead to gains in efficiency and effectiveness in developing and deployment, saving the cost of purchasing and maintaining the infrastructure.

There are various definitions of cloud computing, but the important definition is made by NIST which declared "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes the availability and is composed of five essential characteristics, three service models, and four deployment models." (Chen, & Zhao, 2012).

Cloud computing represents the popular computing paradigm. it is a new paradigm for hosting and delivering services over the Internet, now proving a necessity for utility computing services. Cloud computing services can be provided by cloud vendors through data centers. These vendors provide the cloud computing as solutions available, where a pool of virtualized and dynamically scalable computing power, storage, platforms, and services are delivered on demand to clients over the Internet in a pay as you go manner (Abu Sharkh, et al, 2013).

One of the biggest problems in adoption of cloud computing is security issue like any new technology. The security measures, taken by the cloud vendor, are usually transparent to the organizations and individual users. The presence of large numbers of users that use the cloud computing is further aggravating from concerns, thus, there are various studies in the literature discussing the security issues of the cloud computing (Ali, et al., 2015).

2.1.1 Deployment Cloud Computing

The customer should be choosing one of the deployment cloud types, depending on their characteristics, to provide a suitable adopting in cloud computing environment. Four cloud deployment models have been defined in the Cloud models offered, namely, a public, private, community and hybrid cloud, as show in Figure (2-1), (Dillon, et al 2010, and Chang, et al., 2013):-

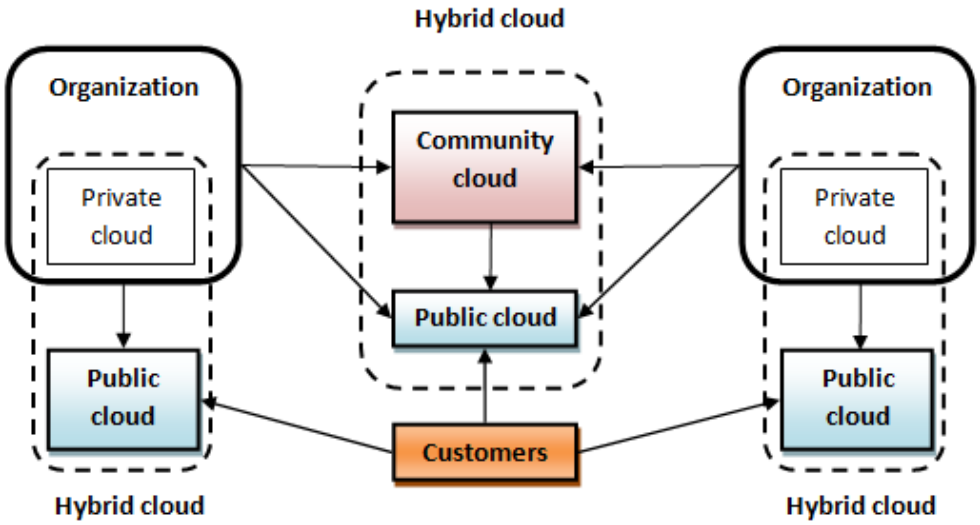


Figure (2-1): Cloud Computing Deployment Model (Youssef, 2012).

- Public Cloud

Public cloud considered as a form of popular cloud deployment model, it is a model which allows users' access to the cloud data by using mainstream web browsers. The organizations and individual users are using this type to save costs and time without obligations of deployment and maintenance. It is typically based on a pay-per-use model, similar to a prepaid electricity metering system which is flexible enough to cater for spikes in demand for cloud optimization.

Public clouds are less secure than the other cloud models, it is including data loss and conflicts concerning legal and ethical issues because the general cloud users can access to the resource

cloud. Many popular cloud services are public clouds such as Amazon, Google App Engine, and Force.com.

- Private Cloud

Private cloud considered as a specific to the single organization and managed by the organization or a third party. The private cloud resources are provided by the cloud vendor, it pooled together and available for cloud users to share or uses, thus only internal users can accessibility to these resources.

The private cloud is different than the public cloud in that all cloud resources and applications are managed by the organization itself, similar to intranet functionality. This type is more suitable for organizations because it characterized security, compliance, and regulatory requirements, and provides more enterprise control over deployment and use.

- Community cloud

Community cloud considered as a private cloud, but it offers more alternative architecture. The community cloud arises for replacing vendor clouds by developing the underutilized resources from the user. These resources are shared by many organizations that have the same task such as security requirement, policy, values and concerns. An example of a community cloud is the educational cloud that used by universities and institutes around the world to provide education and research services.

- Hybrid Cloud

Hybrid cloud considered as a private cloud linked to one or more external cloud services through use part public cloud and part private, centrally managed, provisioned as a single unit. This cloud infrastructure is a combination of more than one clouds deployment types that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability.

The hybrid cloud is suitable for organizations wishing to reduce costs, whilst maintaining privacy and data security. It provides more secure control other than types of the data and applications. The organizations use this type model in order to optimize their resources for increasing their core competencies. They make that by margining out peripheral business functions onto the cloud, while controlling core activities on-premise through private cloud.

2.1.2 Services Cloud Computing

The architecture of cloud computing has extensively used the following three service models according to the delivery models, namely Infrastructure as a Service (IaaS), Software as a Service (SaaS) and Platform as a Service (PaaS) (Ramgovind, et al, 2010, and Youssef, 2012):-

- Infrastructure as a Service (IaaS)

Cloud consumers directly use IT infrastructures such as processing, storage, networks, and other fundamental computing resources that provided in the IaaS cloud. IaaS uses virtualization technology to convert physical resources into logical resources, it can be dynamically provisioned and released by customers as needed.

In the Infrastructure as a Service type, it is a single tenant cloud layer where the cloud customers used resources at a pay-per-use fee, and the cloud vendor's dedicated these resources are only

shared with contracted clients. In this case, the IaaS type has minimized the need for huge initial investment in computing hardware such as servers, networking devices and processing power. The IaaS completely abstracted the hardware beneath it and allowed users to consume infrastructure as a service without any complexities. The examples of IaaS are Amazon's EC2 and Drop Box.

- Platform-as-a-Service (PaaS)

Platform-as-a-Service (PaaS) is one layer above IaaS on the stack and abstracts away everything up to OS, middleware, etc. It is defined as a set of software and development tools hosted on the provider's servers, and development platform supporting the full software lifecycle which allows cloud customers to develop cloud services and applications directly on the PaaS cloud.

The PaaS customers are using PaaS services transfer even more costs from capital investment to operational expenses, thus it is considered as a supports collaborative work between members of a project team. In the same time, the additional constraints and possibly some degree of lock-in posed by the additional functionality layers. The examples of PaaS are Google AppEngine and windows Azure.

- Software-as-a-Service (SaaS)

Software-as-a-Service is defined as a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network from various clients such as web browser, PDA, etc. by application users.

SaaS operates on the virtualized and associated with pay-per-use costing model when software applications are leased out to contracted organizations by specialized SaaS vendors It is most often implemented to provide business software functionality to enterprise customers at a low cost

SaaS providers may host the software in their own data centers or with co-location providers, or may themselves be outsourced to IaaS providers. The architecture of SaaS-based applications is

specifically designed to support many concurrent users at the same time. Examples of SaaS include SalesForce.com, Google Mail, Google Docs, and so forth.

2.1.3 Properties of Cloud Computing Services

Cloud computing consist of all the IT components (hardware, software, networking, and services) that necessary to enable development and transmission of cloud services via the internet or a private network. So, it has many properties as shown below: (Tebaa, & ELhaj, 2014):

Flexibility: it is considered one of an essential cloud property. The cloud is ability defined of a given infrastructure dynamically adapt for scale.

Ability to adapt: It must allow the cloud self-service administration to reduce human intervention in the administration.

Quality of Service: It allows using several measures such as the number of operations per second and response time.

The guarantee: it provided a guarantee for user and service to specify what the resources are rather than identified stations to meet of service.

High availability: The replicated data in different centers reliable is being a process in virtual infrastructure, and in the second data center is connecting two at essence the formation of the cloud.

Cost reduction: Pay per use is meaning that every process utilized only the basis of which payment.

Environmental orientation: recourses description the strict necessity to reduce the energy consumption of IT, beyond the economically, these reductions allow the ecological energy reduction footprint of the company.

2.1.4 Cloud Security

The security concept is considered the biggest problem in any new technology. Especially in the cloud computing, it is utilizing storage service on a remote location, the consumers are generally unaware of what happens to their data. Thus, the organizations and customers need to evaluate the cloud security before migrating to any type of the cloud (Sarwar, et al., 2013).

Security concepts in cloud environment are similar to any security issue in traditional IT environment. The cloud computing includes groups of technologies, operating systems, storage, networking and virtualization. Thus, the security issue in the cloud has played the most important role in hindering cloud computing acceptance to the cloud customers (Chen & Zhao, 2012 and SO, 2011).

There are different features of cloud computing such as multitenant characteristic, service delivery and deploy models, pooled computing resources and so on. All these features when compared with the traditional IT environment, it seems the cloud may face different risks and introduced new security challenges that require novel techniques (Chen & Zhao, 2012 and SO, 2011).

The main challenges in the cloud are representing that the users are concerned about the security of sensitive data, because the users have less control and limited access rights over their data. In

this case, the most attention of users cloud is to preventing unauthorized accesses and eliminates possibilities of data corruption (Sarwar, et al., 2013).

Any customers before adoption the cloud must trust the cloud security environment, in order to have confidence that their data will be protected and its integrity maintained, and looking for some level of assurance that appropriate security measures are indeed been properly implemented in the daily operations of the cloud infrastructure (Chen & Zhao, 2012. and SO, 2011).

There are many encryption techniques such as:

Ordering Preserving Encryption (OPE) encryption scheme which maintain the order of data, Homomorphic Encryption (HE) is encryption scheme which maintain the mathematical operations .This research will focus of Homomorphic Encryption (Liu, et al., 2014).

2.1.5 Homomorphic Encryption Technique

Homomorphic Encryption used to perform operations on encrypted data without knowing the private key, through computations to be carried out on cipher-text and obtain an encrypted result which decrypted gives the result of operations performed on the plaintext Homomorphic encryption is two shapes: "fully" or "partially" depend on size of data. Where is 1- 256 Byte in partially, it is 4 MB- 73TB in fully. (Li, et al., 2015 and Stehlé & Steinfeld, 2010).

2.1.6 Partial Homomorphic Encryption (PHE)

Partial homomorphic encryption is the most important type of homomorphic technique, it performs the computation on some of mathematical operations and has high efficiency for practical applications such as the Paillier scheme can perform evaluations in milliseconds level, but still has some of defects.

Most of PHE schemes support one type of operation. On the other hand, the PHE supports more than one operation, but still there exist constrain (Hu, 2013).

This research is choosing some of PHE technique, as shown Table (2-1).

Table (2-1): Some of Partially Homomorphic Encryption Schemes (Hu, 2013)

Scheme	Homomorphism	Computation
Textbook RSA	Multiplicative	Mod. Exp. in $\boldsymbol{Z_{pq}}$
Paillier Scheme	Additive	Mod. Exp. in $\boldsymbol{Z_{(pq)^2}}$
Benaloh	Additive	Mod. Exp. in $\boldsymbol{Z_{pq}}$
Paillier ECC variations	Additive	Scalar-point mult. in elliptic curves
Naccache-Stern	Additive	Mod. Exp. in $\boldsymbol{Z_{pq}}$
Okamoto-Uchiyama	Additive	Mod. Exp. in $\boldsymbol{Z_{(p)^2 q}}$
Kawachi-Tanaka-Xagawa	Additive	Lattice Algebra

Paillier Cryptosystem

In 1999, the French researcher which invented Paillier Cryptosystem; it is considered one type of public key cryptography (Paillier, 1999).

This technique of public key is used asymmetric key algorithm, which means the key to encryption is a different key to decryption. Any user has two keys for cryptography, the first key is called a public key used distributed user to receive message, and the second is called a private key for each user to open secret message. The Paillier Cryptosystem has two keys to be corresponding and related by mathematically but the private key cannot be feasibly because it is a private of user, as shown details of Paillier formula in chapter three (Paillier, 1999).

2.6.1.2 RSA Cryptosystem

RSA algorithm (represented the first char from names authors - Rivest, Shamir and Adleman) is helping to save data communication security because it used the most popular asymmetric cryptographic algorithm (Rahman, et al., 2012).

The algorithm is based on two main encryption processes. The first is using a public key, which works on received the encrypted data without recovery to original. The second is using a private key, where it is recovered data to original by it. Today, RSA is used in web browsers, email programs and mobile phones, as shown details of RSA formula in chapter three (Rahman, et al., 2012).

2.1.7 NTRU

Nth Degree Truncated Polynomial Ring Units (NTRU) is considered very efficient for encryption and decryption, where it is a faster and cheap key generation for creating. Additional, it used low memory for applying program in device.

NTRU Cryptography is the smallest, fastest and strongest available. It is well suited to all types of systems including full scale servers down to the smallest of embedded systems. NTRU is the public key cryptosystem not based on factorization or discrete logarithmic problems, as shown details of NTRU formula in chapter three (Mol, & Yung, 2008).

NTRU is a lattice-based alternative to RSA and Elliptic Curve Cryptography (ECC) and is based on the shortest vector problem in a lattice. NTRU is represented as following: (R=Z[X]/(X^N-1)). As well as being able to protect systems from today's attacks, the NTRU algorithm is future-proof and will resist attacks by quantum computers when they come available. NTRU should be implemented anywhere that requires public key encryption including SSL, TLS, IPSec and others (Mol, & Yung, 2008).

This thesis discussed how represented cipher-texts in model implementation, defined NTRU and PHE scheme. Proposed parameters will discuss by selecting the NTRU and PHE to provide a practical secure framework on cloud computing, and then discussing the experimental results from PHE scheme implemented and NTRU.

2.2 Literature review

- Tebaa, et al., 2012 focused on cloud computing security challenges and declared it's also an issue to many researchers; first priority was to focus on security which is the biggest concern of organizations that are considering a move to the cloud. The cloud computing security based on fully Homomorphic Encryption is a new concept of security which enables providing results of calculations on encrypted data without knowing the raw data on which the calculation was carried out, with respect of the data confidentiality (Tebaa, et al, 2012).

  They stated that, the application of FHE is an important stone in Cloud Computing security; the authors could outsource the calculations on confidential data to the cloud server, keeping the secret key that can decrypt the result of calculation. The implementation of this paper analyzing the performance of the existing HE cryptosystems, the authors working on a virtual platform through focusing on the size of the public key and its impact on the size of the encrypted message, the server delay of the request treatment according to the size of the encrypted message, the result decrypting time of the request according to the cipher-text size sent by the server.

- Samanthula, et al., 2012 presented encrypting the data by the data owner and then outsourcing it to the cloud seems to be a reasonable approach. This paper proposed an efficient and Secure Data Sharing (SDS) framework that prevents information leakage from a previously revoked user rejoining the system. This framework is implemented by using HE and proxy re-encryption schemes that prevents the leakage of unauthorized data when a revoked user rejoins the system (Samanthula, et al., 2012).

  They stated that, the proposed framework is secure under the security definition of Secure Multiparty Computation (SMC) and also is a generic approach - any additive Homomorphic

Encryption and proxy re-encryption schemes can be used as the underlying sub-routines. The SDS framework is presented to prevent the information leakage in the case of collusion between a user and the cloud; it is based upon distributing the encrypted data and authorization tokens corresponding to each data recorded between two clouds.

- Huang, et al., 2013 presented a secure and privacy-preserving digital rights management (DRM) scheme using HE in cloud computing, which allows content provider to outsource encrypt contents to centralized content server and allows user to consume contents with the license issued by license server. Further, we provide a secure content key distribution scheme based on additive Homomorphic probabilistic public key encryption and proxy re-encryption, which prevents the malicious employees of license server from issuing license to unauthorized user without letting other parties know. In addition, it achieves privacy-preserving by allowing users to stay anonymous towards the key server and service provider. The analysis and comparison results indicate that the proposed scheme has high efficiency and security (Huang, et al., 2013).

- Majithia, & Singh, 2013 tried to providing the security to the cloud network and data, various algorithms are used to secure data send by a mobile phone using an android platform on a Cloud Network. These algorithms represented in encryption and decryption methods in such a way that eavesdroppers or hackers cannot read it, but that authorized parties can (Majithia, & Singh, 2013).

  In this paper, NTRU algorithm is implemented on cloud network using an android platform. Data input is the data or message send by an android user in bytes. Encryption time has been

measured in milliseconds. The whole time is taken by NTRU algorithm to encrypt the message or time taken to change plaintext into ciphertext to send over cloud network.

This paper shows results of parameters like Encryption Time, Decryption Time and throughput when NTRU, a public key encryption algorithm is implemented on cloud network for an android platform. From these results, the performance of NTRU algorithm has been analyzed and providing stronger security level than other algorithms. NTRU provided better result so it will improve the current security level, speed and provide reliable message at receiver end with respect to key generation, encryption and decryption.

- Majithia, & Singh, 2013 declared the cloud computing as emerging computing paradigm and many of the organizations are moving toward the cloud but lacking due to security reasons. To provide suitable security to the cloud network and data, numbers of algorithms are used for encryption and decryption for cloud network. Encryption is the process of encoding messages in such a way that eavesdroppers or hackers cannot read it, but that authorized parties can (Majithia, & Singh, 2013).

  Various algorithms are used to secure data send by a mobile phone using an android platform on a Cloud Network. this paper implemented NTRU algorithm on cloud network by using an android platform, also, it analyzes the comparison between NTRU public key based algorithms, RSA public key based algorithm, DES secret key based algorithm.

  The results produced by this comparison between algorithms explain that NTRU algorithm is faster and providing stronger security level than other algorithms. NTRU provided better result so it will improve the current security level, speed and provide reliable message at receiver end with respect to key generation, encryption and decryption.

- Ma, 2013 presented a study the Multi-key Fully Homomorphic Encryption (FHE) scheme developed by Lopez-Alt, Tromer and Vaikuntanathan, it is abbreviated as the LTV scheme that is more accessible to non-experts. The LTV scheme is based on NTRU, a public-key cryptosystem using lattice-based cryptography, and it encrypts each single bit of data into one corresponding polynomial. This paper provides serious mathematical proofs and detailed explanations on some implicit mathematical steps that were not addressed by the authors of the original scheme. And it includes the background research on NTRU encryption scheme, the presentation of the FHE scheme in a single-key version (Ma, 2013).

  In addition to, it analyzed the security of the LTV scheme, and it includes an implementation of proposed scheme in a lower-level language such as Java or C/C++. As the results of this paper, the experiments work are a more accessible version of the original scheme with serious mathematical proofs and a Sage package that implements the basic scheme and some real-world applications such as an n-bit Adder. The Sage package is posted in Sage Interact Community website.

- Doröz, et al., 2014 explain variant of NTRU proposed by Stehle and Steinfeld was recently extended into a full-edged multi-key Fully Homomorphic Encryption scheme by Alt-Lopez, Tromer and Vaikuntanathan (ATV). It is presented a customized leveled implementation of the NTRU based ATV Homomorphic Encryption scheme and developed a customized implementation of the ATV scheme. This paper analyzed noise growth for increasing circuit depths and developed a simple formula that allows one to determine parameter sizes to support arbitrary depth circuits efficiently. Furthermore, it specialized the modulus in a way that allows us to drastically reduce the public key size while retaining the ability to apply modulus reduction and switching through the levels of evaluation. The reduced public key size makes it possible

to evaluate deep circuits such as the AES block cipher on common (non-server) computing platforms with a reasonable amount of memory (Doröz, et al., 2014).

This paper presents a generic bit-sliced implementation of the ATV scheme that embodies a number of optimizations. To assess the performance of the scheme, the Homomorphically of this paper evaluates the full 10 round AES circuit in 31 hours with 2048 message slots resulting in 55 sec per AES block evaluation time.

- Rohloff & Cousins, 2014 worked to design, implement and evaluate a Fully Homomorphic Encryption (FHE) scheme; the FHE scheme is an NTRU-like cryptosystem, with additional support for efficient key switching and modulus reduction operations to reduce the frequency of operations. Cipher-texts in proposed scheme are represented as matrices of 64-bit integers (Rohloff and Cousins, 2014).

  They stated that, the basis of this design is a layered software services stack to provide high-level FHE operations supported by lower-level lattice-based primitive implementations running on a computing substrate. This paper implemented and evaluated the FHE scheme to run on a commodity CPU-based computing environment. the experimental results is showing the FHE implementation that provides at least an order of magnitude improvement in runtime as compared to recent publicly known evaluation results of other FHE software implementations.

- Dahab, et al., 2015 described adaptive key recovery attacks on NTRU-based SHE schemes which considered as somewhat homomorphic encryption schemes. ciphertexts produced by an FHE scheme can be operated on in such a way that we obtain a ciphertext that corresponds to the addition or multiplication of the respective plaintexts (Dahab, et al, 2015).

  Adaptive key recovery attacks on homomorphic encryption seem to be realistic in certain scenarios, so they are potentially a serious problem in practice. Given access to a decryption oracle, the attack allows us to compute the private key for all parameter choices; such attacks show that one must be very careful about the use of homomorphic encryption in practice.

  Many FHE schemes have as public value an encryption of the private key bits, which can be sent to the decryption oracle before the challenge, which makes such schemes insecure against CCA1 adversaries. Indeed, almost every somewhat homomorphic construction proposed till now in the literature is vulnerable to an attack of this type. Hence, our result adds to a body of literature that shows that building CCA1-secure homomorphic schemes is not trivial.

- Steinfeld, 2014 Presented the general concepts about The NTRU public-key cryptosystem, and surveyed recent developments in both the security analysis and applications of the NTRU cryptosystem and its variants. the author is focusing on recent exciting developments in both the security analysis and applications of NTRU, that believe should make the NTRU system even more attractive for study and development in future than the traditional reasons, and suggest new motivation and directions for studying NTRU's mathematical underpinnings (Steinfeld, 2014).

Some of these developments motivate the study of new computational problems on polynomial rings, whereas others help to unify the field of lattice-based cryptography, by showing that the security of the NTRU system can be based on the same foundations as more recent lattice-based schemes. This paper reviews two recent novel variants of the NTRU system, which allow powerful new functionality to be added to the basic cryptosystem. The first application is an NTRU-based Fully Homomorphic Encryption (FHE) scheme, which allows useful computation on encrypted messages, and the second is a construction for NTRU-based multi-linear maps, which open the door to another class of applications including non-interactive multiparty key agreement.

- Çetin, et al., 2015 introduced survey a number of classical sorting algorithms which are capable of efficiently sorting encrypted data without the secret key, then show that some are more suitable than others. These algorithms are obtained by focusing on the multiplicative depth of the sorting circuit alongside the more traditional metrics such as number of comparisons and number of iterations. The reduced depth allows much reduced noise growth and thereby makes it possible to select smaller parameter sizes in somewhat homomorphic encryption instantiations resulting in greater efficiency savings (Çetin, et al, 2015).

  The authors proposed two depth optimized sorting algorithms for efficient homomorphic evaluation. Circuit depth is intimately related to the parameter sizes in leveled homomorphic encryption implementations and therefore directly affects the overall performance of the homomorphic circuit evaluation. Existing sorting algorithms are not optimized for homomorphic evaluation. The authors instantiate a Somewhat Homomorphic Encryption Scheme (SWHE) based on NTRU, and present an implementation of the proposed sorting

algorithm using this SWHE scheme. The results of this paper confirm theoretical analysis, i.e. that the performance of the proposed sorting algorithm scales favorably as N increases.

- Liu, 2015 proposed several efficient algorithms and architectures for NTRUEncrypt and NTRU based homomorphic encryption system. For NTRUEncrypt system, a new Linear Feedback Shift Register (LFSR) based architecture is firstly presented. Then two new architectures are proposed for computation of NTRU based fully homomorphic encryption system. One architecture uses LFSR with a novel design of the modular multiplication unit, and the other proposed architecture is systolic array based which uses two types of PEs (Liu, 2015).

  The new architecture takes advantage of large number of zero coefficients of the input; a parameter selection optimization for hardware is also introduced. A novel design of the modular arithmetic unit is proposed to reduce the critical path delay. The implementation results have shown that the proposed design outperforms all the existing works in terms of area-delay product. Further enhancement on the efficiency of the LFSR based architecture is proposed by using extended LFSR architecture. The implementation result shows that the proposed design uses slightly larger resource but has much faster speed.

2.3 Summary

There are many types of research presented in the cloud computing area, this thesis focused on the research that presented in the field of cloud security by implementing Partial Homomorphic and NTRU techniques.

In this thesis, chapter two explains different methods for implementing Partial Homomorphic and NTRU techniques for protection the database cloud and effects each of these techniques on the performance and security. Then, it focuses on the most important points in this research for using it. Based on the notes extracted from this research, the researcher suggested the proposed model that implemented in chapter three and extracting results in chapter four.

Chapter Three

Performance Analysis for Fully and Partially Homomorphic Encryption Technique

3.1 Introduction

Cloud computing presents many advantages to the organizations and individual users, these advantages presents in organizing and provisioning computational resources. At the same time, the security issue has emerged as a significant problem to increased deployment and adoption of cloud computing.

The security issue considered as one of the challenges that facing the user. This thesis investigates the compare between two encryption methods in the database cloud by studying the different ratio of some encryption technologies. The researcher attempted to show that the methodology have highly efficient performance and provably secure.

This thesis studied the compared between encryption techniques which represent in Homomorphic and NTRU techniques. In addition to, it studied the impact of parameters on the performance based on descriptive approach and quantitative approach.

In this research, the design of model is applying for improvement different type of PHE techniques, each technique is depending on different parameters for calculating time. Where is evaluated difference between using NTRU and PHE through comparing all results.

the researcher implements several steps through using many experiments to provide appropriate security level and data flexibility with performance, it using two types of PHE algorithm, additive (paillier) and Multiplication (RSA), and using NTRU technique. all the steps studied the difference between NTRU and PHE through using several parameters and what proportion of the different between parameters, as shown in Figure (3-1).

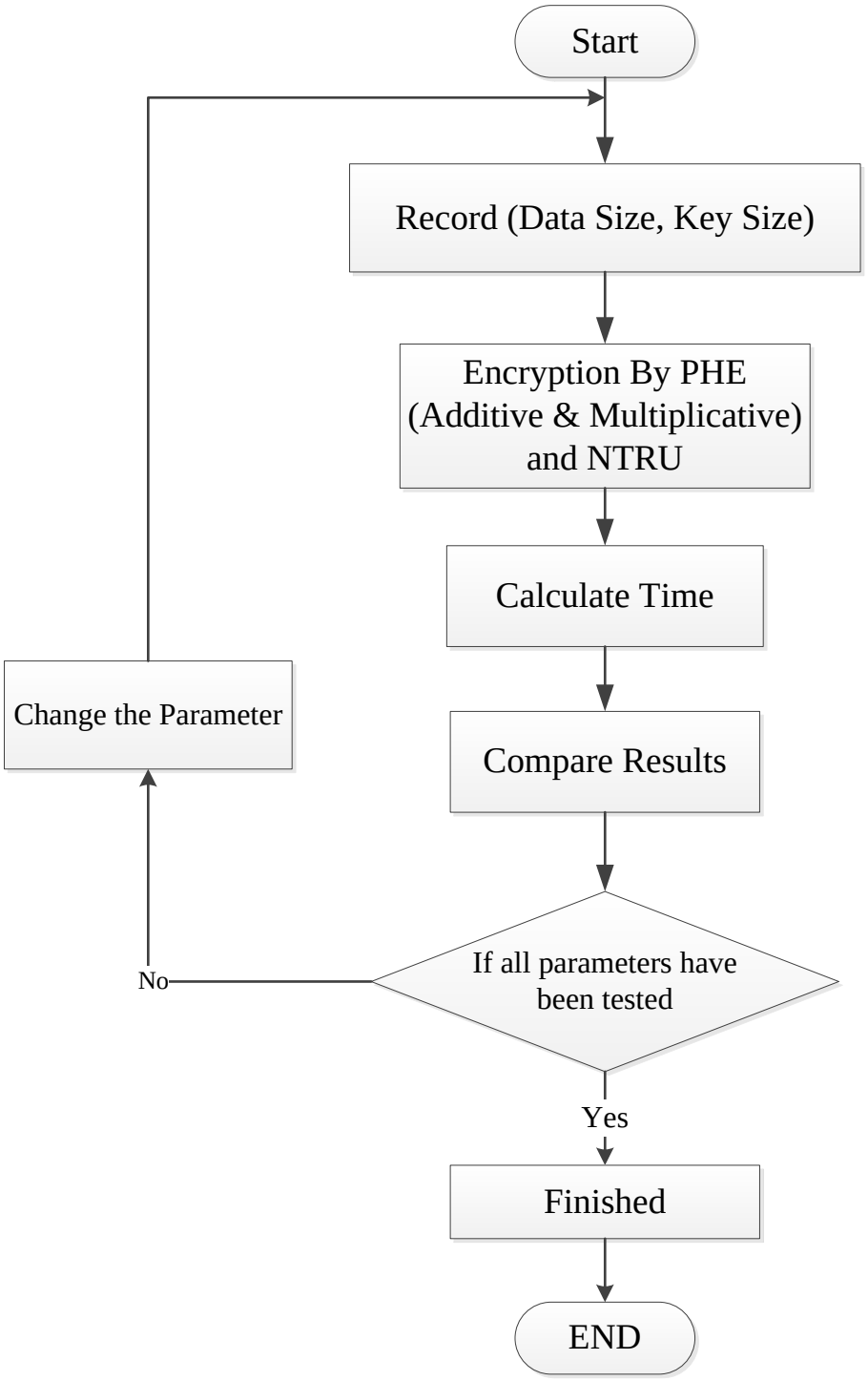


Figure (3-1): Main Algorithm

3.2 Description Methodology

The researcher investigates the encryption techniques that perform to obtain encrypted results without knowing the raw data. This technique is homomorphic encryption considered very important for database to keep the data encrypted for this reason. The aim of this study is to reach the best performance while maintaining the level of security and data flexibility by using different parameters which change standards in every process implemented where is extracted and compared the results with each other through the recording time.

Each operation is conducted and then determines each the parameter to achieve best performance for homomorphic encryption (PHE) and using NTRU technique to calculate time of the operation, and the extent of the impact this technique(NTRU) and (PHE) in terms of performance, the proposed parameters are shown below:-

3.2.1 Operation

The researcher attempted to expand the work of Homomorphic technique on the cloud database by utilizing two types algorithms of PHE (Additive and Multiplication Encryption) and NTRU technique to demonstrate its impact on them:-

A. Homomorphic Algorithm

There are many types of PHE techniques depending on the properties of several standard public key encryption schemes, such as Multiplication, Additive and XOR and so on.

1- Paillier Encryption Technique:

The proposed model is facilitating confidential data aggregation through additive homomorphic encryption. The additive homomorphic property of public-key cryptosystems is applied by utilizing Paillier Encryption technique. The formula Paillier Cryptosystem is as following, (Tebaa, et al, 2012):-

Mathematical Expressions

- Choose two large prime numbers p and *q* randomly and independently of each other, $\gcd(pq,(p-1)(q-1))=1$
- Compute $n=p*q$
- $\lambda=\operatorname{lcm}(p-1,q-1)$.
- $\mu=(L(g^{\lambda}\bmod n^2))^{-1}\bmod n$
- Select random integer g where *g=n+1*.
- The public (encryption) key is (n,g).
- The private (decryption) key is (λ,μ).

Encryption Equation

- $c=g^m\cdot r^n\bmod n^2$

Where select random r where $r\in\mathbb{Z}_n^*$

Decryption Equation

- $m=L(c^{\lambda}\bmod n^2)\cdot\mu\bmod n$

2- RSA Encryption Technique:

The proposed model is exploiting Multiplication homomorphic property of RSA algorithm; it is public-key cryptosystems that support the homomorphic operation of Multiplication modulo. The formula RSA Cryptosystem is as following, (Tebaa, et al, 2012):-

Mathematical Expressions

- Compute $n = p^* q$
- Compute $\varphi(n) = (p - 1)(q - 1)$
- Choose an integer e such that $1 < e < \varphi(n)$ and $gcd(e, \varphi(n)) = 1$;

  i.e., d and $\varphi(n)$ are co-prime.
- Determine d as $d.e \equiv 1 \pmod{\varphi(n)}$
- The public (encryption) key is (e,n).
- The private (decryption) key is (d).

Encryption Equation

- $c \equiv m^e \mod n$

Decryption Equation

- $m = c^d \mod n$.

B. NTRU Algorithm

The NTRU technique is a practical lattice-based cryptosystem. The proposed model present NTRU technique with somewhat homomorphic encryption schemes (Paillier and RSA), then compare the time for each of them. The formula NTRU Cryptosystem is as following, (Majithia, & Singh, 2013):-

Mathematical Expressions

- Choose random parameter (n, p ,q, d ,r), chose q larger than p
- Compute f(x), g(x) as two small polynomials function
- Compute $f_q(x) = f(x)^{-1} \ mod \ (q)$
- Compute $f_p(x) = f(x)^{-1} \ mod \ (p)$
- Compute $h(x) = p * (fq) * g \ (mod \ q)$

Encryption Equation

- $E = r*h + m \ (mod \ q)$

Decryption Equation

- $a = f * e \ (mod \ q)$
- $b = a \ (mod \ p)$
- $c = f_p * b \ (mod \ p)$

C. Key Size

Key size is considered a measured bits used for cryptography. Each method in cryptography is dependent on used key size that is represented the security of an algorithm, because it is measured for cryptographic strong and faster method.

Most key algorithms in common use are designed to have security equal to their key length. For instance, Triple DES has a key size of 168 bits yet gives at most 112 bits of security, since an assault of multifaceted nature 112 is known. This property of Triple DES is not a weakness provided 112 bits of security is adequate for an application (Barker, et al., 2012).

Several different types of keys are defined. The keys are identified according to their classification as public or symmetric keys, and as to their use. For public and private key-agreement keys, their status as static or ephemeral keys is also specified. Asymmetric-key algorithms with this property are known; elliptic curve cryptography come the closest with an effective security of roughly half its key length.

The researcher studies the effect of different key sizes to compare the performance of using NTRU and PHE. Therefore, each process used different key size with fixed data size.it has determined which key size can achieve the best performance of NTRU and PHE, these key sizes are (32 bits, 64 bits, 128 bits and 256 bits).

D. Data Type & Size.

The researcher studies the effect of different data size to compare the performance of using NTRU and PHE. Therefore, each process used different key size with fixed key size.it has been determined which data size can achieve the best performance of NTRU and PHE; the researcher used (1-9) digits of data size from type integer.

3.3 Scenario of Algorithm

The proposed algorithm is used to compare the performance of partial homomorphic technique and NTRU by using two procedures, as explained below:

3.3.1 Fixed Data Size with Multi-Key Size Procedure

In this scenario, the algorithm is used to study the effect performance of NTRU with two types of PHE technique by using fixed data size with different key size. It was used four different key sizes with fixed data size. These key sizes are: 32 bits, 64 bits, 128 bits and 256 bits.

3.3.1.1 Calculated Time of Multi-Key by Using Paillier Formula

The proposed model is importing fixed data size and choosing one of the different key sizes. It is implemented these parameters through paillier algorithm and calculated time, then saving time to be used for comparison later. This algorithm records the time from each experiment and compare between them, as shown in Figure (3-2).

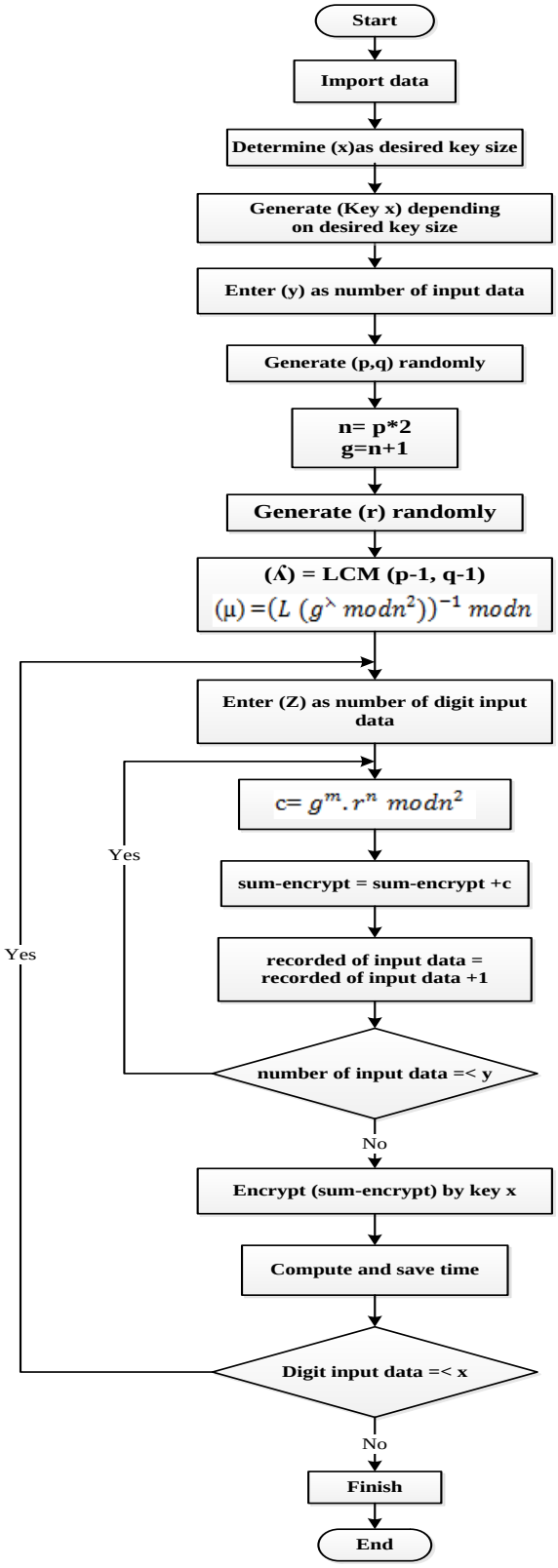


Figure (3-2): Paillier Flowchart for Fixed Data Size

Algorithm:

Step1: Start

Step2: Import data

Step3: Enter variable (x) as size of key used

Step4: Determine (y) as number of input data

Step5: Generate variable (p & q) randomly as large prime number

Step6: Compute (n) where $n = p*q$

Step7: Compute (g) where $g=n+1$.

Step8: Generate variable (r) randomly.

Step9: Compute ( ) where $\lambda = LCM\ (p-1, q-1)$

Step10: Compute (μ) $=(L\ (g^{\lambda}\ mod n^2))^{-1}\ mod n$

Step11: Generate (Key $_x$) depending on desired key size

Step12: Encrypt (M) as plain text where c= $g^m . r^n\ mod n^2$

Step13: Compute sum-encrypt where *sum-encrypt = sum-encrypt +c*

Step14: Incremental record of input data by one.

Step15: Until number of input data is less than (y) then go to step (12)

Else Go to step (16)

Step16: Encrypt (sum-encrypt) by (Key $_x$)

Step17: Compute time

Step18: Saving time

Step19: Until number of key size is less than (x) Then Go to step (11)

Else Go to step (20)

Step20: Finish

3.3.1.2 Calculated Time of Multi-Key by Using RSA Formula

The proposed model is importing fixed data size and choosing one of the different key sizes. It is implemented these parameters through RSA algorithm and calculated time, then saving time to be used for comparison later. This algorithm records the time from each experiment and compare between them, as shown in Figure (3-3).

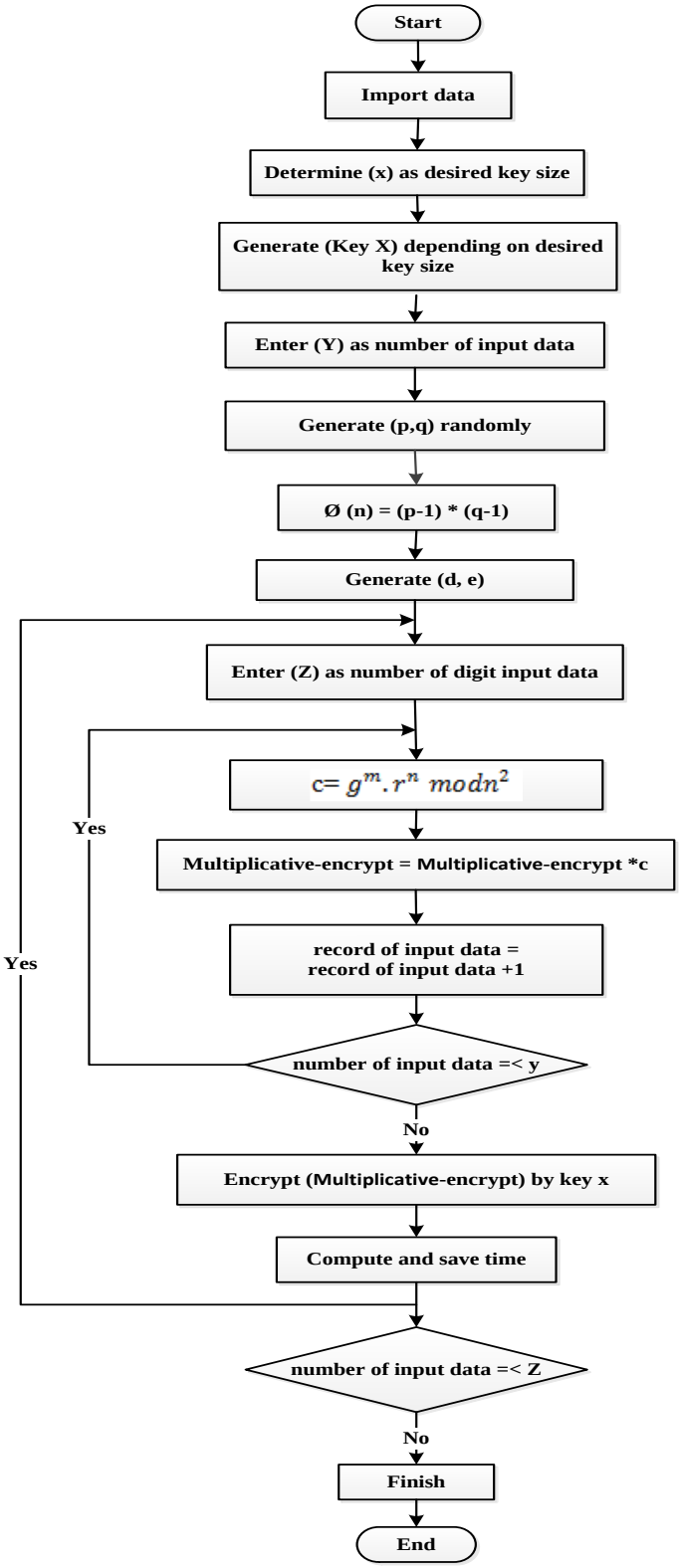


Figure (3-3): RSA Flowchart for Fixed Data Size

Algorithm:

Step1: Start

Step2: Import data

Step3: Enter variable (x) as size of key used

Step4: Determine (y) as number of input data

Step5: Generate variable (p & q) randomly as large prime number

Step6: Compute *Ø (n) = (p-1) (q-1).*

Step7: Generate variable (e) as *((1<e< Ø (n) and gcd (e, Ø (n)) =1).*

Step8: Generate variable (d) as *(d.e=1 mod (Ø (n))).*

Step9: Generate (Key $_x$) depending on desired key size

Step10: Encrypt (M) as plain text where c= $g^m . r^n \ mod n^2$

Step11: Compute multiplicative-encrypt where multiplicative-encrypt = multiplicative-encrypt*c

Step12: Incremental record of input data by one

Step13: Until number of input data is less than (y) then go to step (10)

Else Go to step (14)

Step14: Encrypt (multiplicative-encrypt) by (Key $_x$)

Step15: Compute time

Step16: Saving time

Step17: Until number of key size is less than (x) go to step (9)

Else Go to step (18)

Step18: Finish

3.3.1.3 Calculated Time of Multi-Key by Using Additive NTRU Formula

The proposed model is importing fixed data size and choosing one of the different key sizes. It is implemented these parameters through Additive-NTRU algorithm and calculated time, then saving time to be used for comparison later. This algorithm records the time from each experiment and compare between them, as shown in Figure (3-4).

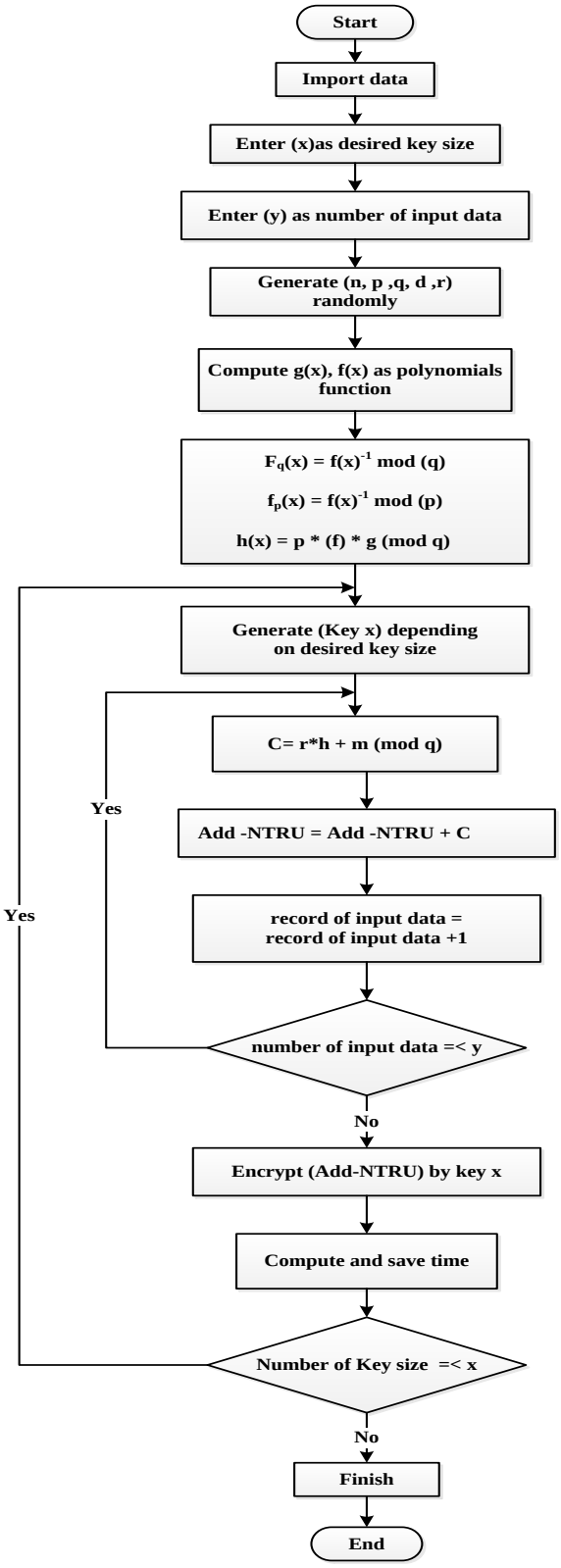


Figure (3-4): Additive NTRU Flowchart for Fixed Data Size

Algorithm:

Step 1: Start

Step 2: Import data

Step3: Enter variable (x) as size of key used

Step4: Determine (y) as number of input data

Step5: Generate variables (n, p ,q, d ,r) randomly

Step6: Compute g(x) as polynomials function

Step7: Compute f(x) as polynomials function

Step8: Compute $f_q(x) = f(x)^{-1} \bmod (q)$

Step9: Compute $f_p(x) = f(x)^{-1} \bmod (p)$

Step10: Compute $h(x) = p * (fq) * g \pmod q$

Step11: Generate (Key_x) depending on desired key size

Step12: Encryption (M) as plain text where $C = r*h + m \pmod q$

Step13: Compute Add-NTRU where $Add\text{-}NTRU = Add\text{-}NTRU + c$

Step14: Incremental record of input data by one

Step15: Until number of input data is less than (y) then Go to Step(12)

Else Go to Step (16)

Step16: Encrypt (Add-NTRU) by (Key_x)

Step17: Compute time

Step18: Saving time

Step19: until number of key size is less than (x) then Go to Step(11)

Else Go to Step(20)

Step20: Finish

3.3.1.4 Calculated Time of Multi-Key by Using Multiplication NTRU Formula

The proposed model is importing fixed data size and choosing one of the different key sizes. It is implemented these parameters through Multiplication -NTRU algorithm and calculated time, then saving time to be used for comparison later. This algorithm records the time from each experiment and compare between them, as shown in Figure (3-5).

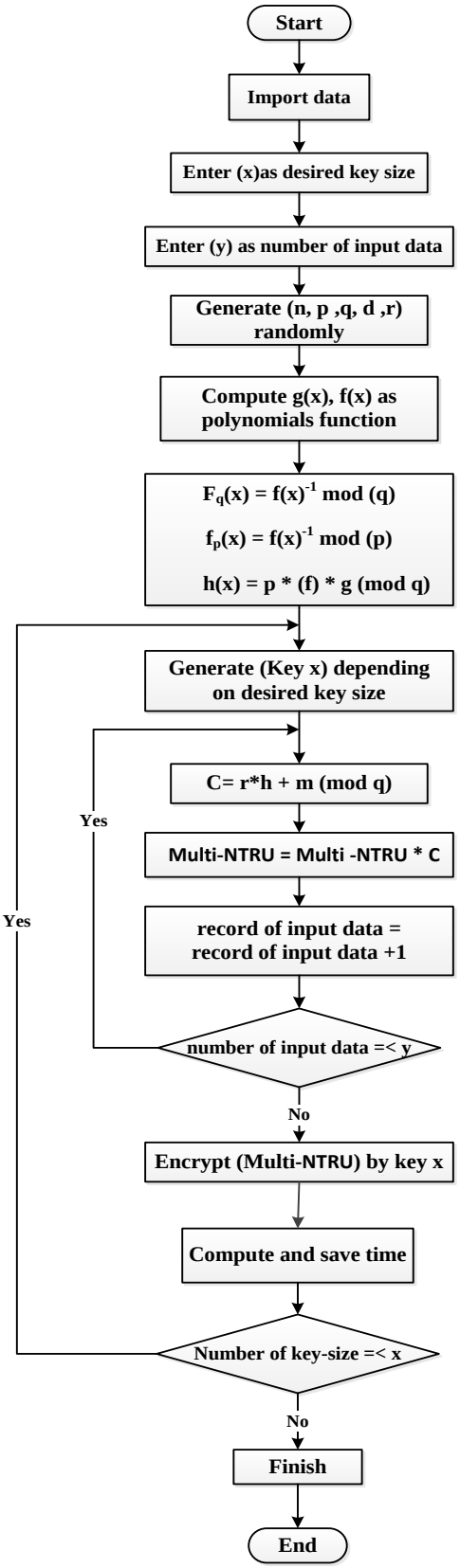

Start
Import data
Enter (x)as desired key size
Enter (y) as number of input data
Generate (n, p ,q, d ,r) randomly
Compute g(x), f(x) as polynomials function
$F_q(x) = f(x)^{-1} \bmod (q)$
$f_p(x) = f(x)^{-1} \bmod (p)$
$h(x) = p * (f) * g \pmod q$
Generate (Key x) depending on desired key size
C= r*h + m (mod q)
Multi-NTRU = Multi -NTRU * C
record of input data = record of input data +1
number of input data =< y
Yes
No
Yes
Encrypt (Multi-NTRU) by key x
Compute and save time
Number of key-size =< x
No
Finish
End

Figure (3-5): Multiplication NTRU Flowchart for Fixed Data Size

Algorithm:

Step 1: Start

Step 2: Import data

Step3: Enter variable (x) as size of key used

Step4: Determine (y) as number of input data

Step5: Generate variables (n, p ,q, d ,r) randomly

Step6: Compute g(x) as polynomials function

Step7: Compute f(x) as polynomials function

Step8: Compute $f_q(x) = f(x)^{-1} \ mod \ (q)$

Step9: Compute $f_p(x) = f(x)^{-1} \ mod \ (p)$

Step10: Compute $h(x) = p * (fq) * g \ (mod \ q)$

Step11: Generate (Key_x) depending on desired key size

Step12: Encryption (M) as plain text where $C = r*h + m \ (mod \ q)$

Step13: Compute Multiplicative-NTRU where *Multiplicative-NTRU = Multi-NTRU * c*

Step14: Incremental record of input data by one

Step15: Until number of input data is less than (y) then Go to Step(12)

Else Go to Step (16)

Step16: Encrypt (Multiplicative -NTRU) by (Key_x)

Step17: Compute time

Step18: Saving time

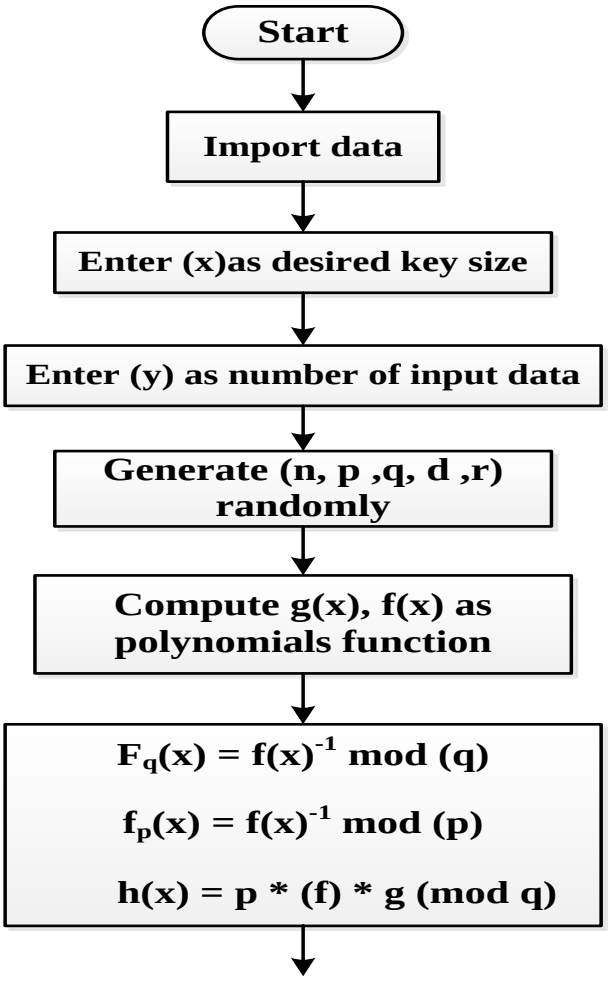
Step19: until number of key size is less than (x) then Go to Step(11)

Else Go to Step(20)

Step20: Finish

3.3.1.5 Calculated Time of Multi-Key by Using Both NTRU Formula

The proposed model is importing fixed data size and choosing one of the different key sizes. It is implemented these parameters through both NTRU formula in the same time (Additive & Multiplication) and calculated total time, then saving time to be used for comparison later. This algorithm records the time from each experiment and compare between them, as shown in Figure (3-6).



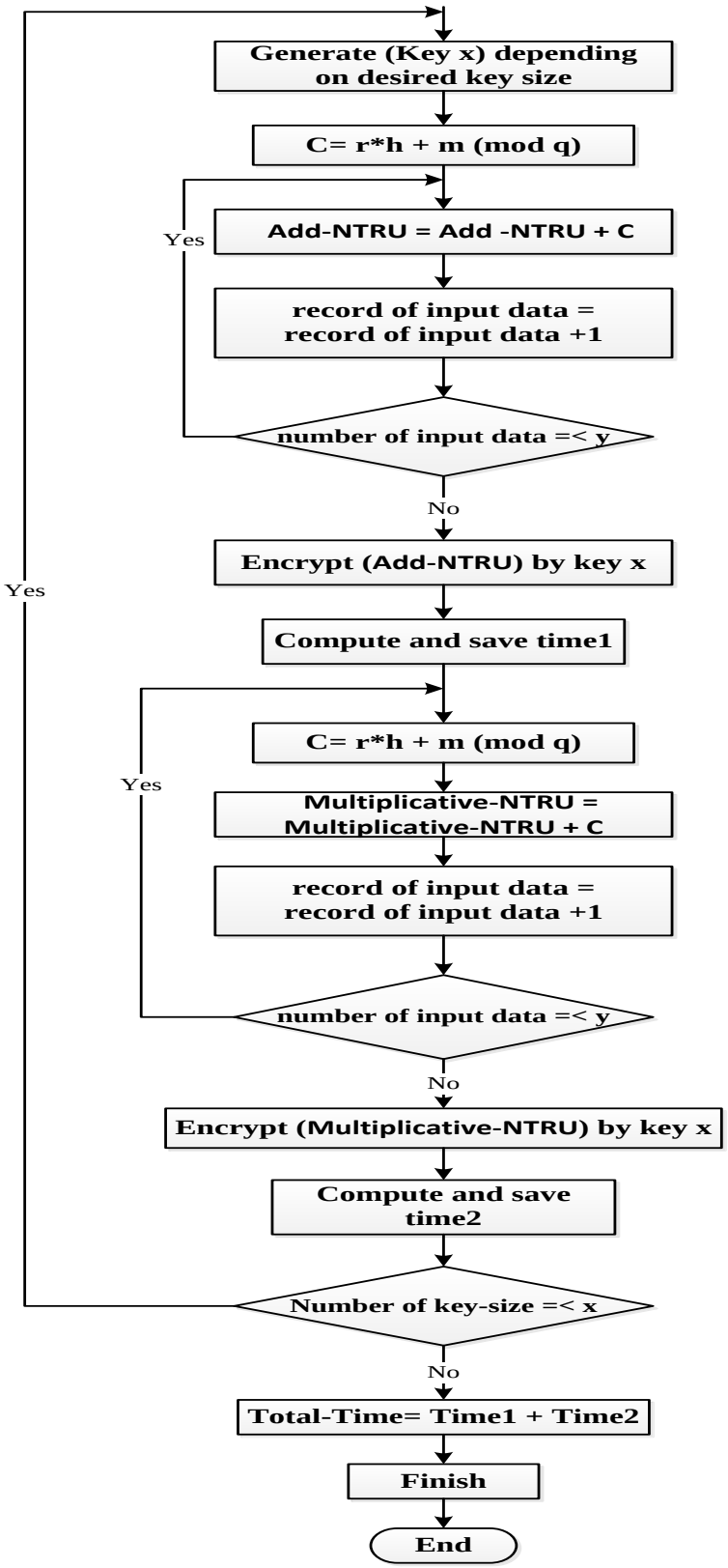


Figure (3-6): Both-NTRU Flowchart for Fixed Data Size

Algorithm:

Step 1: Start

Step 2: Import data

Step3: Enter variable (x) as size of key used

Step4: Determine (y) as number of input data

Step5: Generate variables (n, p, q, d, r) randomly

Step6: Compute g(x) as polynomials function

Step7: Compute f(x) as polynomials function

Step8: Compute $f_q(x) = f(x)^{-1} \bmod (q)$

Step9: Compute $f_p(x) = f(x)^{-1} \bmod (p)$

Step10: Compute $h(x) = p * (fq) * g \pmod q$

Step11: Generate (Key_x) depending on desired key size

Step12: Encryption (M) as plain text where $C = r*h + m \pmod q$

Step13: Compute Add-NTRU where *Add-NTRU = Add-NTRU + c*

Step14: Incremental record of input data by one

Step15: Until number of input data is less than (y) then Go to Step(12)

Else Go to Step (16)

Step16: Encrypt (Add-NTRU) by (Key_x)

Step17: Compute time1

Step18: Saving time1

Step19: Encryption (M) as plain text where $C = r*h + m \pmod q$

Step20: Compute Multi -NTRU where *Multiplicative-NTRU = Multiplicative-NTRU * c*

Step21: Incremental record of input data by one

Step22: Until number of input data is less than (y) then Go to Step(19)

Else Go to Step (23)

Step23: Encrypt (Multiplicative-NTRU) by (Key_x)

Step24: Compute time2

Step25: Saving time2

Step26: *Total-Time = Time1 + Time2*.

Step27: until number of key size is less than (x) then Go to Step(11)

Else Go to Step (28)

Step28: Finish

3.3.2 Fixed Key and Multi-Data Size Procedure

In this scenario, the algorithm is used to study the effect the performance of NTRU with two types of PHE technique by using different data size with fixed key size. It was used nine different data sizes with fixed key size. The ranges of data sizes are (1-9) digits.

3.3.2.1 Calculated Time of Multi-Data Size by Using Paillier Formula

The proposed model is importing fixed key size and choosing one of the different data sizes. It is implemented these parameters through paillier algorithm and calculated time, then saving time to be used for comparison later. This algorithm records the time from each experiment and compare between them, as shown in Figure (3-7).

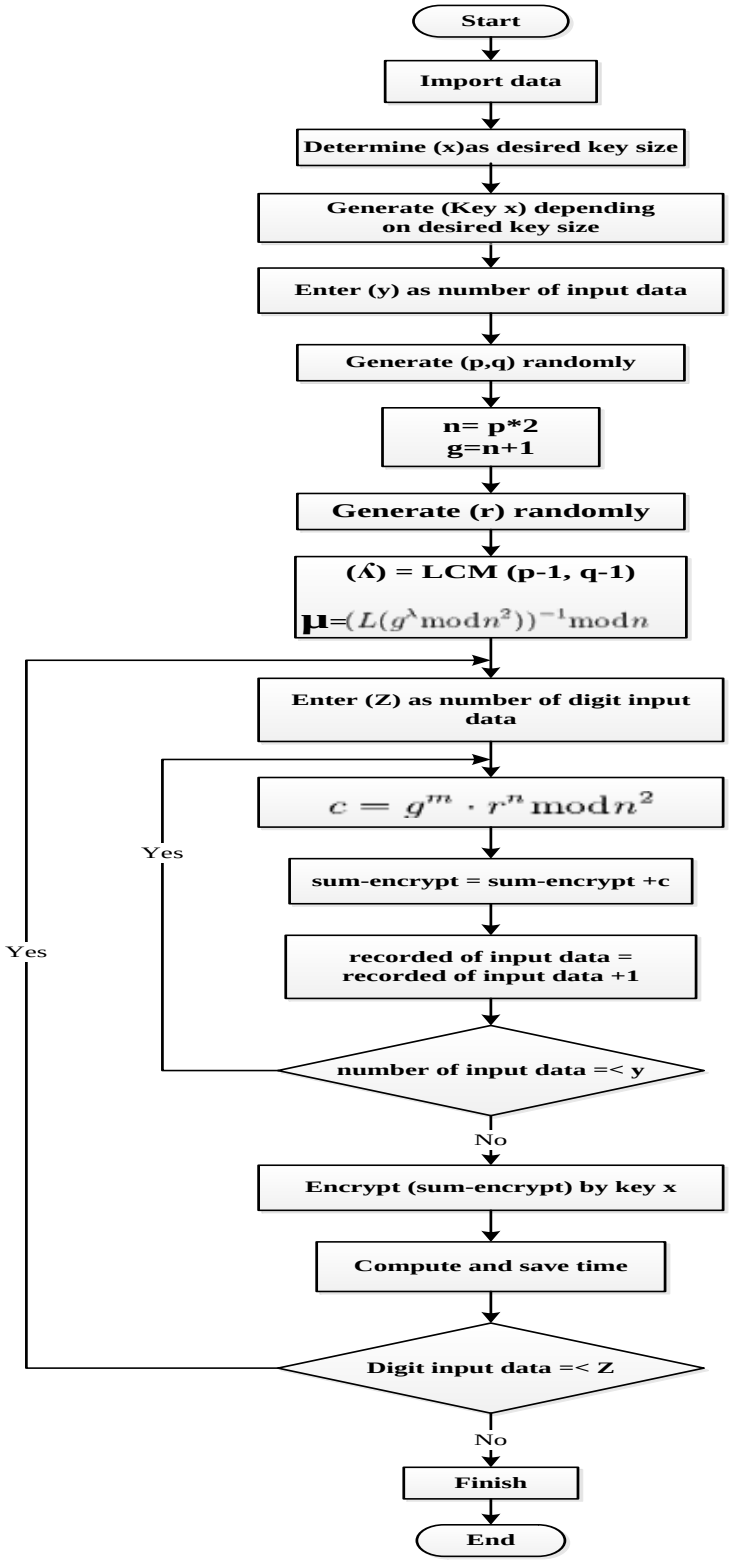


Figure (3-7): Paillier Flowchart for Fixed Key Size

Algorithm:

Step1: Start

Step2: Import data

Step3: Determine (x) as key size

Step4: Generate (Key $_x$) depending on desired key size

Step5: Determine (y) as number of input data

Step6: Generate variable (p & q) randomly as large prime number

Step7: Compute (n) where $n = p*q$

Step8: Compute (g) where $g=n+1$.

Step9: Generate variable (r) randomly.

Step10: Compute ( ) where $\lambda = LCM\ (p-1, q-1)$

Step11: Compute (μ) = $(L(g^{\lambda} \bmod n^2))^{-1} \bmod n$

Step12: Enter (z) as number of digit input data.

Step13: Encrypt (M) as plain text where $c = g^m \cdot r^n \bmod n^2$

Step14: Compute sum-encrypt where *sum-encrypt = sum-encrypt +c*

Step15: Incremental record of input data by one

Step16: Until number of input data is less than (y) then go to step (13)

Else Go to step (17)

Step17: Encrypt (sum-encrypt) by (Key $_x$)

Step18: Compute time

Step19: Saving time

Step20: Until number of digit input data is less than (z) Then go to step (12)

Else Go to step (21)

Step21: Finish

3.3.2.2 Calculated Time of Multi-Data Size by Using RSA Formula

The proposed model is importing fixed key size and choosing one of the different data sizes. It is implemented these parameters through RSA algorithm and calculated time, then saving time to be used for comparison later. This algorithm records the time from each experiment and compare between them, as shown in Figure (3-8).

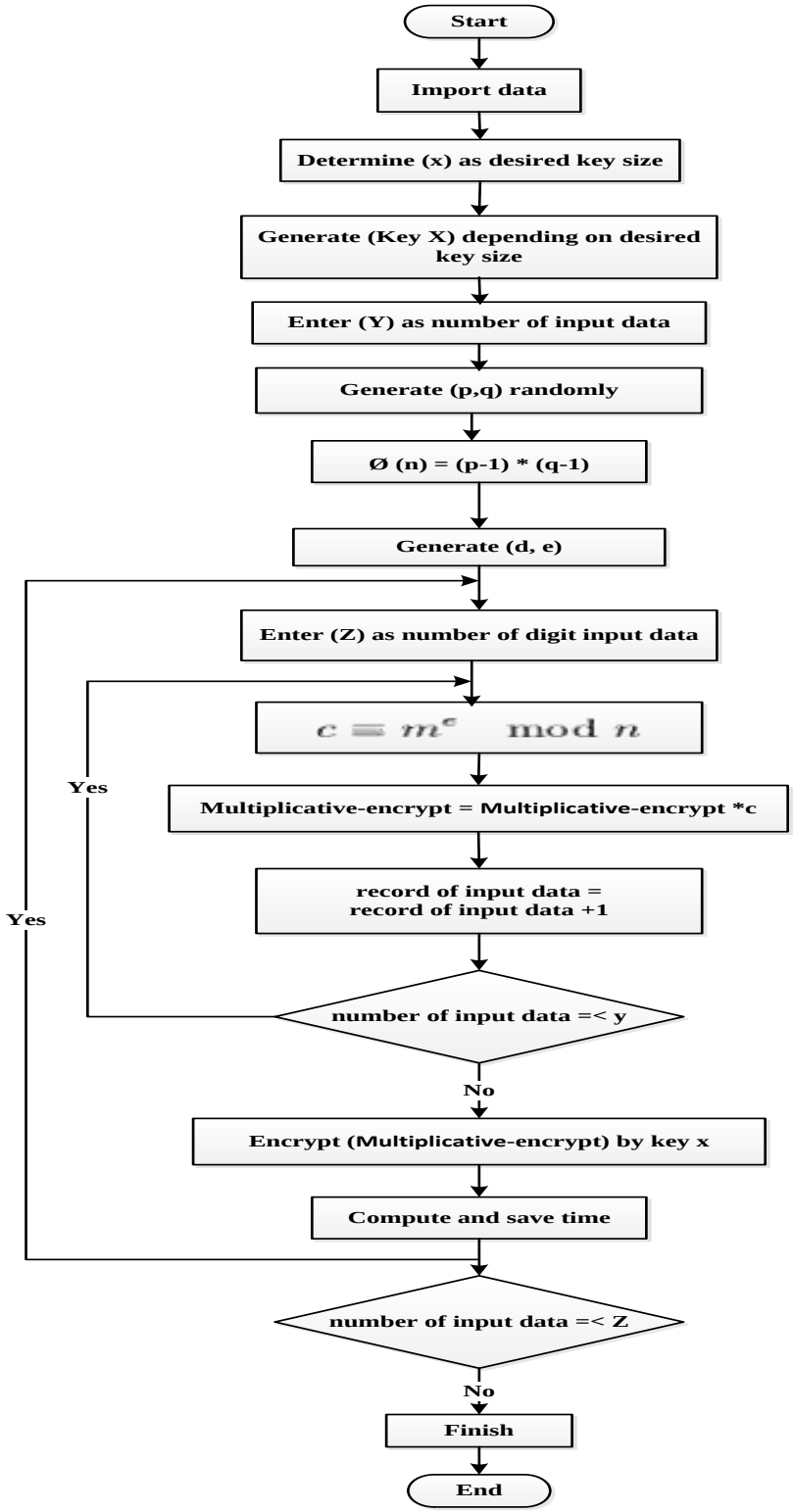


Figure (3-8): RSA Flowchart for Fixed Key Size

Algorithm:

Step1: Start

Step2: Import data

Step3: Determine (x) as key size

Step4: Generate (Key $_x$) depending on desired key size

Step5: Enter (y) as number of input data

Step6: Generate variable (p & q) randomly as large prime number

Step7: Compute *Ø (n) = (p-1) (q-1).*

Step8: determine variable (e) as *((1<e< Ø (n) and gcd (e, Ø (n)) =1).*

Step9: determine variable (d) as *(d.e=1 mod (Ø (n))).*

Step10: Enter (z) as number of digit input data

Step11: Encrypt (m) as plain text where $c \equiv m^e \mod n$

Step12: Compute Multiplicative-encrypt where *Multiplicative-encrypt =*

*Multiplicative-encrypt*c*

Step13: Incremental record of input data by one

Step14: Until number of input data is less than (y) then go to step (11)

Else Go to step (15)

Step15: Encrypt (Multiplicative-encrypt) by (Key $_x$)

Step16: Compute time

Step17: Saving time

Step18: Until number of digit input data is less than (Z) go to step (10)

Else Go to step (19)

Step19: Finish

3.3.2.3 Calculated Time of Multi-Data Size by Using Additive NTRU Formula

The proposed model is importing fixed key size and choosing one of the different data sizes. It is implemented these parameters through Additive-NTRU algorithm and calculated time, then saving time to be used for comparison later. This algorithm records the time from each experiment and compare between them, as shown in Figure (3-9).

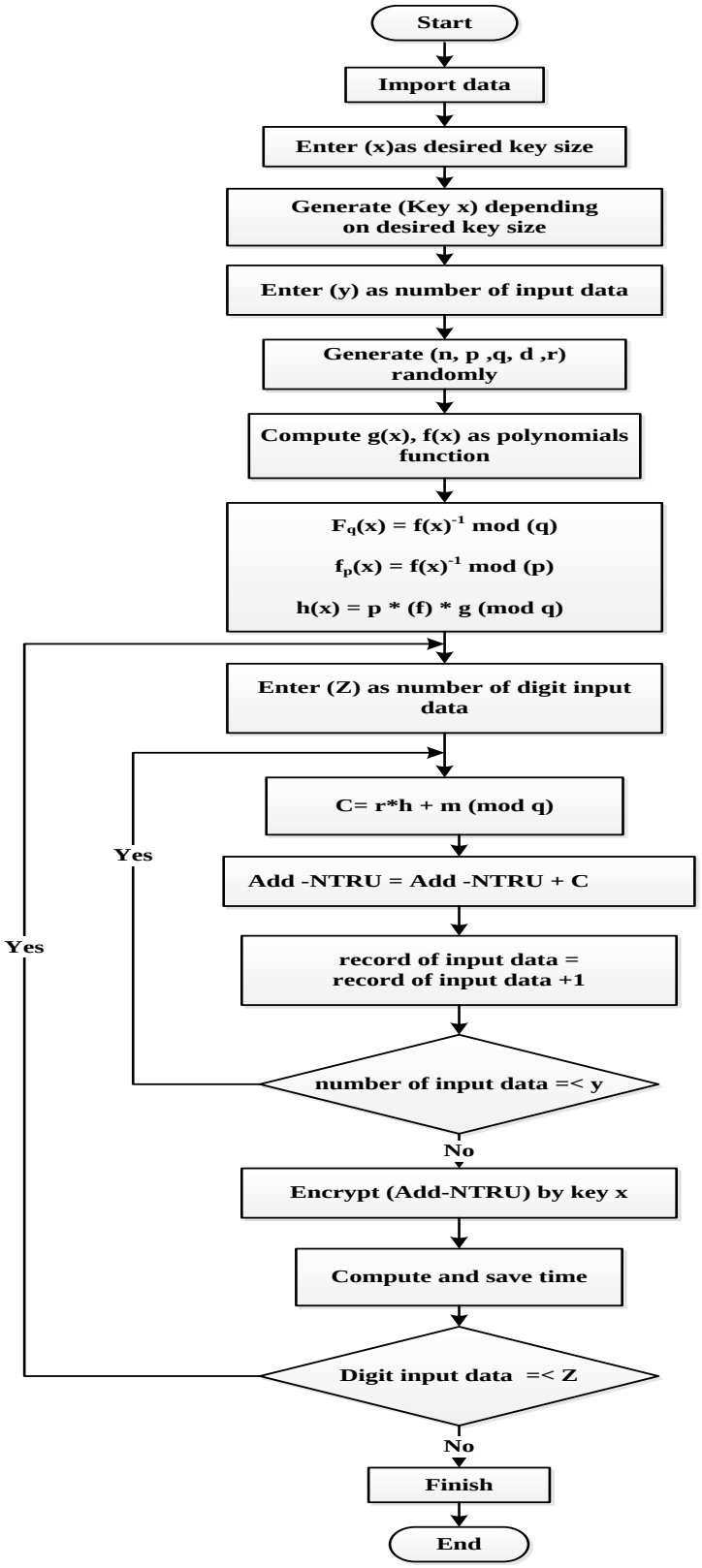


Figure (3-9): Additive NTRU Flowchart for Fixed Key Size

Algorithm:

Step 1: Start

Step 2: Import data

Step3: Determine variable (x) as size of key used

Step4: Generate (Key $_x$) depending on desired key size

Step5: Determine (y) as number of input data

Step6: Generate variables (n, p ,q, d ,r) randomly

Step7: Compute g(x) as polynomials function

Step8: Compute f(x) as polynomials function

Step9: Compute $f_q(x) = f(x)^{-1} \ mod \ (q)$

Step10: Compute $f_p(x) = f(x)^{-1} \ mod \ (p)$

Step11: Compute $h(x) = p * (f) * g \ (mod \ q)$

Step12: Enter (z) as number of digit input data

Step13: Encryption (M) as plain text where $C = r*h + m \ (mod \ q)$

Step14: Compute Add-NTRU where $Add\text{-}NTRU = Add\text{-}NTRU + c$

Step15: Incremental record of input data by one

Step16: Until number of input data is less than (y) then Go to Step(13)

Else Go to Step (17)

Step17: Encrypt (Add-NTRU) by (Key $_x$)

Step18: Compute time

Step19: Saving time

Step20: until number of key size is less than (z) then Go to Step(12)

Else Go to Step (21)

Step21: Finish

3.3.2.4 Calculated Time of Multi-Data Size by Using Multiplication NTRU Formula

The proposed model is importing fixed key size and choosing one of the different data sizes. It is implemented these parameters through Multiplication -NTRU algorithm and calculated time, then saving time to be used for comparison later. This algorithm records the time from each experiment and compare between them, as shown in Figure (3-10).

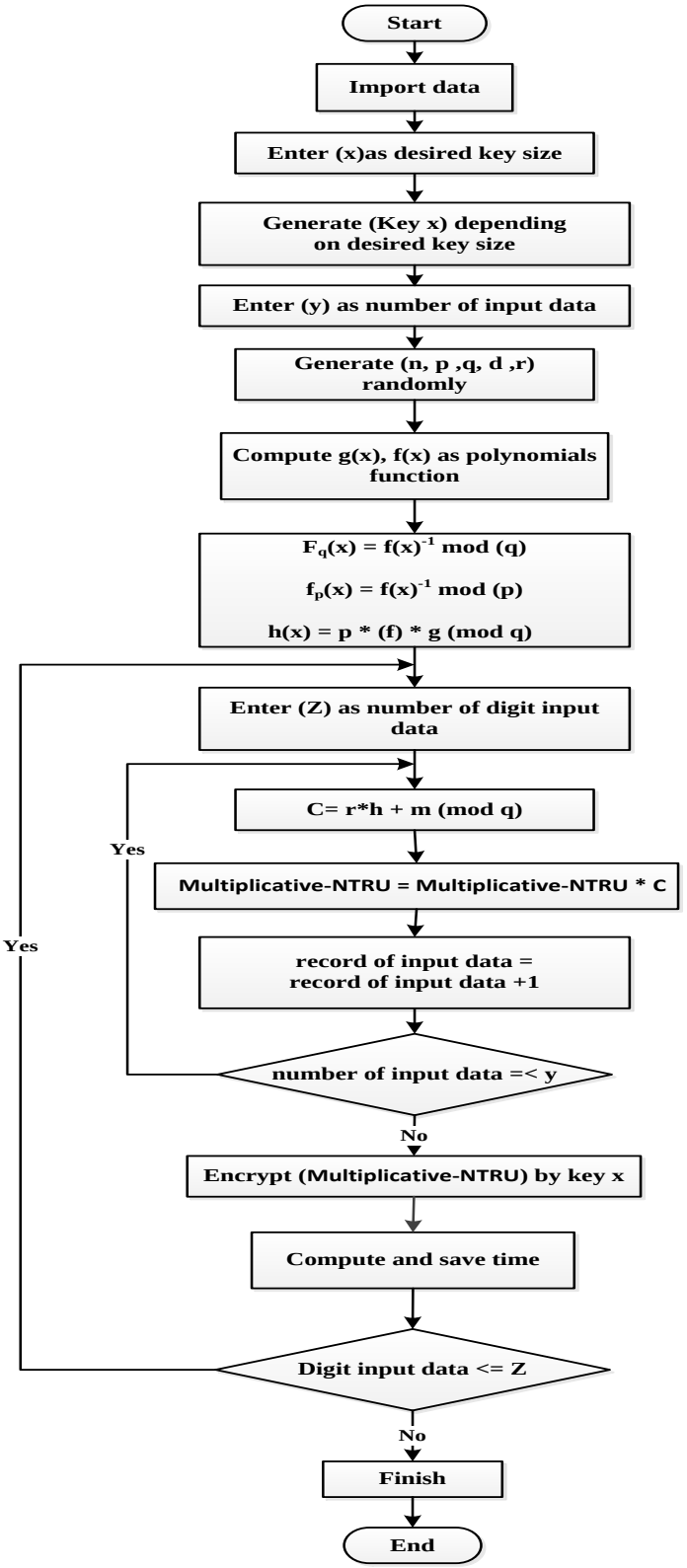


Figure (3-10): Multiplication NTRU Flowchart for Fixed Key Size

Algorithm:

Step 1: Start

Step 2: Import data

Step3: Determine variable (x) as size of key used

Step4: Generate (Key $_x$) depending on desired key size

Step5: Determine (y) as number of input data

Step6: Generate variables (n, p ,q, d ,r) randomly

Step7: Compute g(x) as polynomials function

Step8: Compute f(x) as polynomials function

Step9: Compute $f_q(x) = f(x)^{-1} \ mod \ (q)$

Step10: Compute $f_p(x) = f(x)^{-1} \ mod \ (p)$

Step11: Compute $h(x) = p * (f) * g \ (mod \ q)$

Step12: Enter (z) as number of digit input data

Step13: Encryption (m) as plain text where $C = r*h + m \ (mod \ q)$

Step14: Compute Multiplicative-NTRU where

$$Multiplicative\text{-}NTRU = Multiplicative\text{-}NTRU * c$$

Step15: Incremental record of input data by one

Step16: Until number of input data is less than (y) then Go to Step(13)

Else Go to Step (17)

Step17: Encrypt (Multiplicative-NTRU) by (Key $_x$)

Step18: Compute time

Step19: Saving time

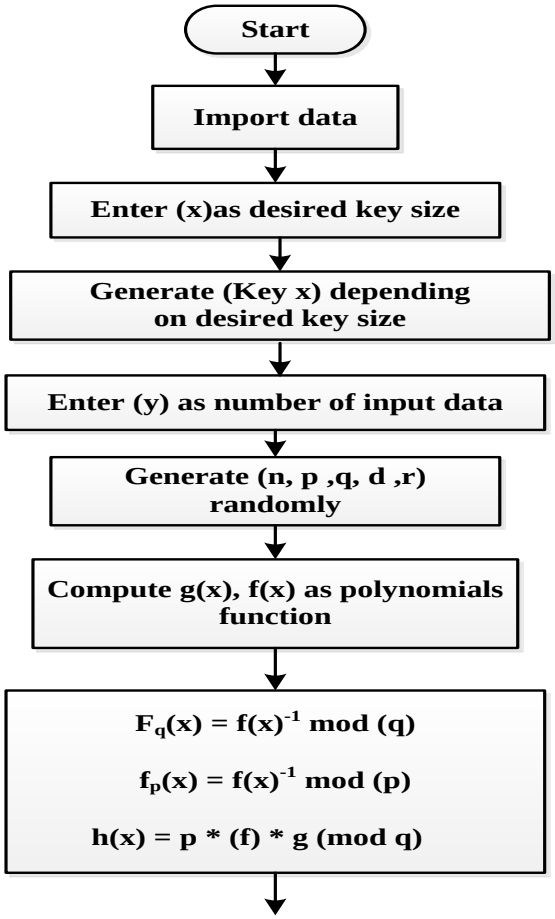
Step20: until number of key size is less than (Z) then Go to Step (12)

Else Go to Step (21)

Step21: Finish.

3.3.2.5 Calculated Time of Multi-Data Size by Using Both NTRU Formula

The proposed model is importing fixed key size and choosing one of the different data sizes. It is implemented these parameters through both NTRU formula in the same time (Additive & Multiplicative) and calculated total time, then saving time to be used for comparison later. This algorithm records the time from each experiment and compare between them, as shown in Figure (3-11).



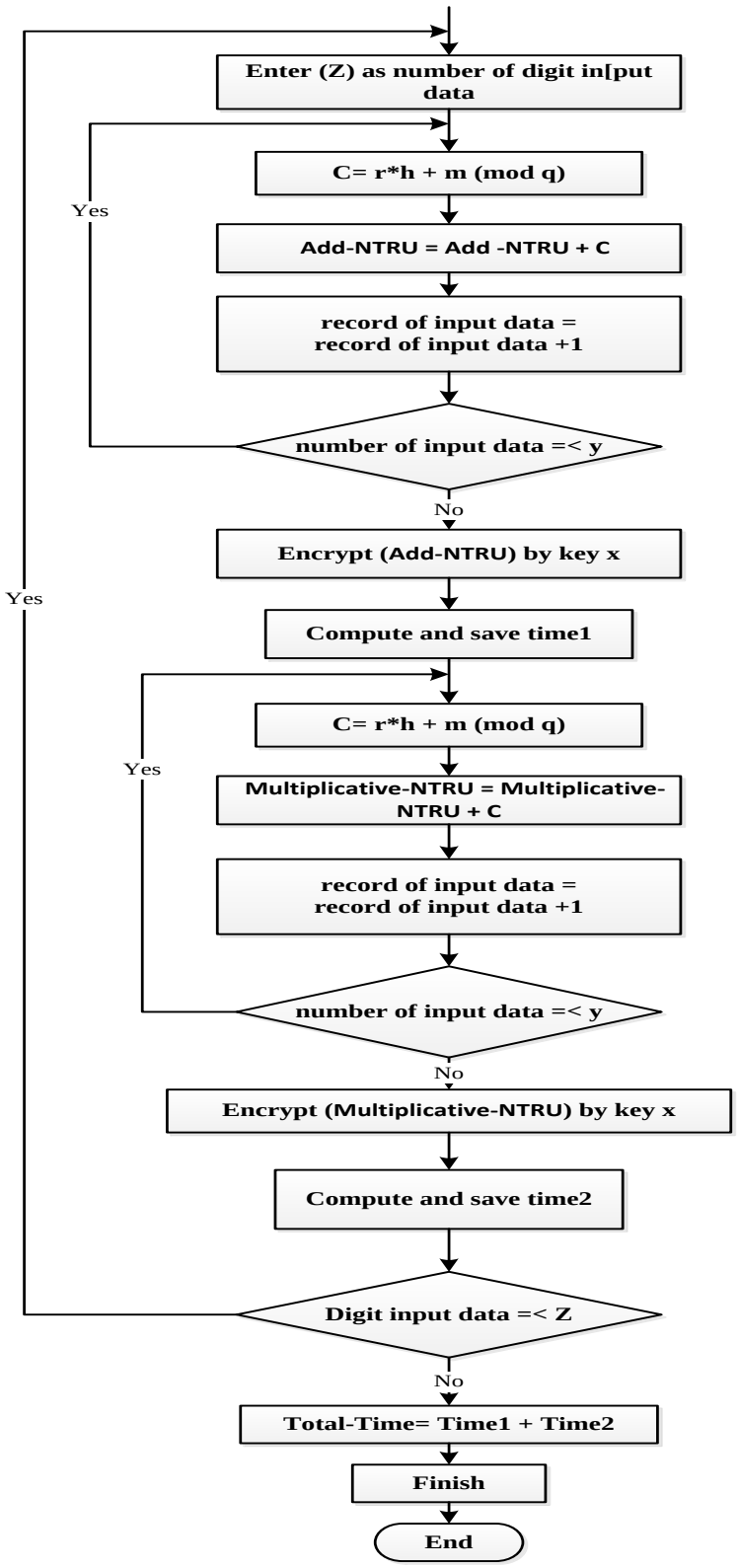


Figure (3-11): Both NTRU Flowcharts for Fixed Key Size

Algorithm:

Step 1: Start

Step 2: Import data

Step3: Enter variable (x) as size of key used

Step4: Generate (Key $_x$) depending on desired key size

Step5: Determine (y) as number of input data

Step6: Generate variables (n, p ,q, d ,r) randomly

Step7: Compute g(x) as polynomials function

Step8: Compute f(x) as polynomials function

Step9: Compute $f_q(x) = f(x)^{-1} \bmod (q)$

Step10: Compute $f_p(x) = f(x)^{-1} \bmod (p)$

Step11: Compute $h(x) = p * (f) * g \pmod q$

Step12: Enter (z) as number of digit input data

Step13: Encryption (m) as plain text where $C = r*h + m \pmod q$

Step14: Compute Add-NTRU where $Add\text{-}NTRU = Add\text{-}NTRU + c$

Step15: Incremental record of input data by one

Step16: Until number of input data is less than (y) then Go to Step(13)

Else Go to Step (17)

Step17: Encrypt (Add-NTRU) by (Key $_x$)

Step18: Compute time1

Step19: Saving time1

Step20: Encryption (m) as plain text where $C = r*h + m \pmod q$

Step21: Compute Multiplicative-NTRU where *Multiplicative-NTRU = Multi -NTRU * c*

Step22: Incremental record of input data by one

Step23: Until number of input data is less than (y) then Go to Step(20)

Else Go to Step (24)

Step24: Encrypt (Multiplicative-NTRU) by (Key_x)

Step25: Compute time2.

Step26: Saving time2.

Step27: *Total-Time = Time1 + Time2.*

Step28: until number of key size is less than (z) then Go to Step(12)

Else Go to Step (29)

Step29: Finish

Chapter Four

Experimental And Results

4.1 Introduction

Cloud computing is one of the attractive concepts in the technology field. It is providing more services depending on users demand. These services making computing resources are available to all users anytime and from anywhere. However, all the advantages that offered by the cloud computing, but there are many concerns about the migrating to the cloud.

The security issue aimed to protect the cloud database to each user by using various encryption methods. This thesis focused on the implemented of Homomorphic Encryption method on the Cloud Computing security.

The researcher builds a model to utilize two types of PHE (Paillier and RSA) and the NTRU encryption scheme, this model is evaluation primitives including additive, Multiplication of partial homomorphic encryption and two types of operations over NTRU (additive NTRU & Multiplication NTRU) computation, as shown in Figure (4-1).

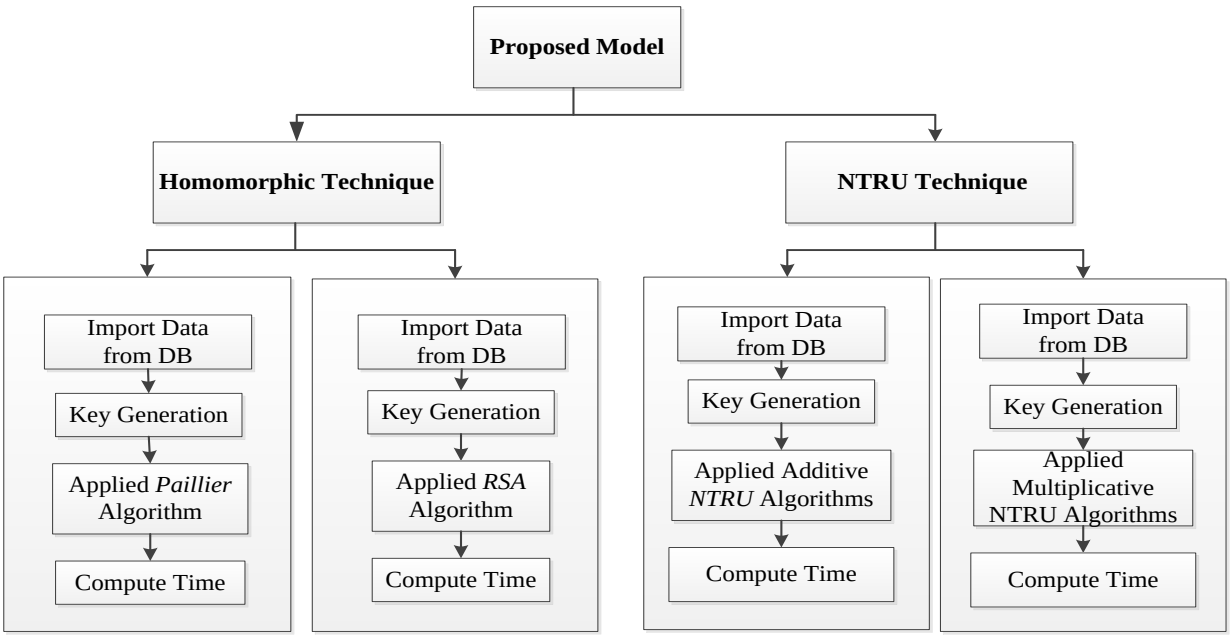


Figure (4-1): Proposed Model

4.2 Tools For Experimental

This thesis has designed and implemented software to compare the performance of PHE & NTRU using VB.net version 2010 as language programming. The software imported the data from Microsoft Access 2010 with the size of (500) records.

In this thesis, it has been used the database to run several experiments and to find the outcomes of these experiments. The researcher implemented fixed data type which represent in Integer type, different size of keys which represent in (32, 64, 128 and 256) bits and different data sizes which represent in (from 1 digit to 9 digits).

The experiments have taken long time because the combinations of the parameters were enormous. Where has used the 500 records of the database to facilitate studying the effect of several parameters on the performance of PHE and NTRU.

4.3 Evaluation Measures

The researcher evaluates the proposed model by using two measures; these measures evaluate the homomorphic and NTRU by calculating percentage of gaining security level to the Performance loss.

1- Security gain measure:

The researcher used the security gain measure to determine which one of the proposed parameters (key size, or data size) is optimal to investigate the best security and data flexibility with performance of the proposed experimental.

The measure is calculated through divided by the large value of desired parameter on the small value of desired parameter as a percentage

$$\text{Security gain} = \frac{\text{large value of parameter}}{\text{small value of parameter}} * 100 \quad \ldots\ldots. (4.1)$$

2- Performance loss measure:

The researcher used the performance loss measure to determine which one of the proposed parameters (key size, or data size) is optimal to investigate the best security and data flexibility with performance of the proposed experimental.

The measure is calculated through difference between the maximum time of desired parameter and the minimum time of desired parameter divided by the maximum time of desired parameter as a percentage

$$\text{Performance loss} = \frac{\text{The max } value\ of\ perfomance\ time - \text{The min } value\ of\ perfomance\ time}{\text{The max } value\ of\ perfomance\ time} * 100\% \ldots (4.2)$$

4.4 Performance Evaluation

The researcher designed and implemented a model for presenting alternatives to provide authentication in the cloud environment. This model used to compare the performance between two types of encryption algorithms (Partially Homomorphic Encryption and NTRU techniques). The researcher is importing record of database and run different parameters on these data. The model implemented through two types on PHE (RSA and paillier) and three types of NTRU (Additive NTRU, Multiplication NTRU and Combine between them). The following Figure (4-2) declared the main parameters of implementation model when it is divided in two procedures:-

- Fixed Data Size with Multi-Key Size Procedure
- Fixed Key size with Multi-Data Size Procedure

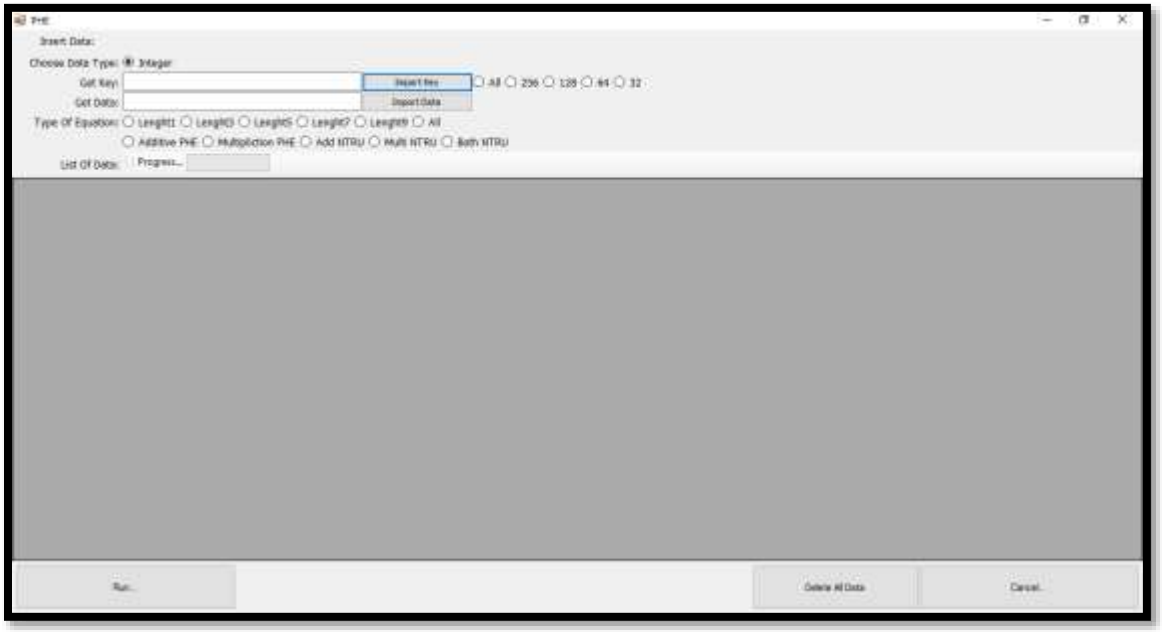


Figure (4-2): Main Interface

4.4.1 Execution Fixed Data Size with Multi-Key Size Procedure

The first procedure implementing different key size on fixed data. These key sizes are: 32 bits, 64 bits, 128 bits and 256 bits. It is divided in five Phases (RSA, Paillier, Additive NTRU, Multiplication NTRU and Both NTRU).

4.4.1.1 Execution Time of Multi-Key size by Using Paillier Formula

Figure (4-3) shows the use of Paillier equation on all types of key size (32, 64, 128 and 256) with selected fixed data size (length three).

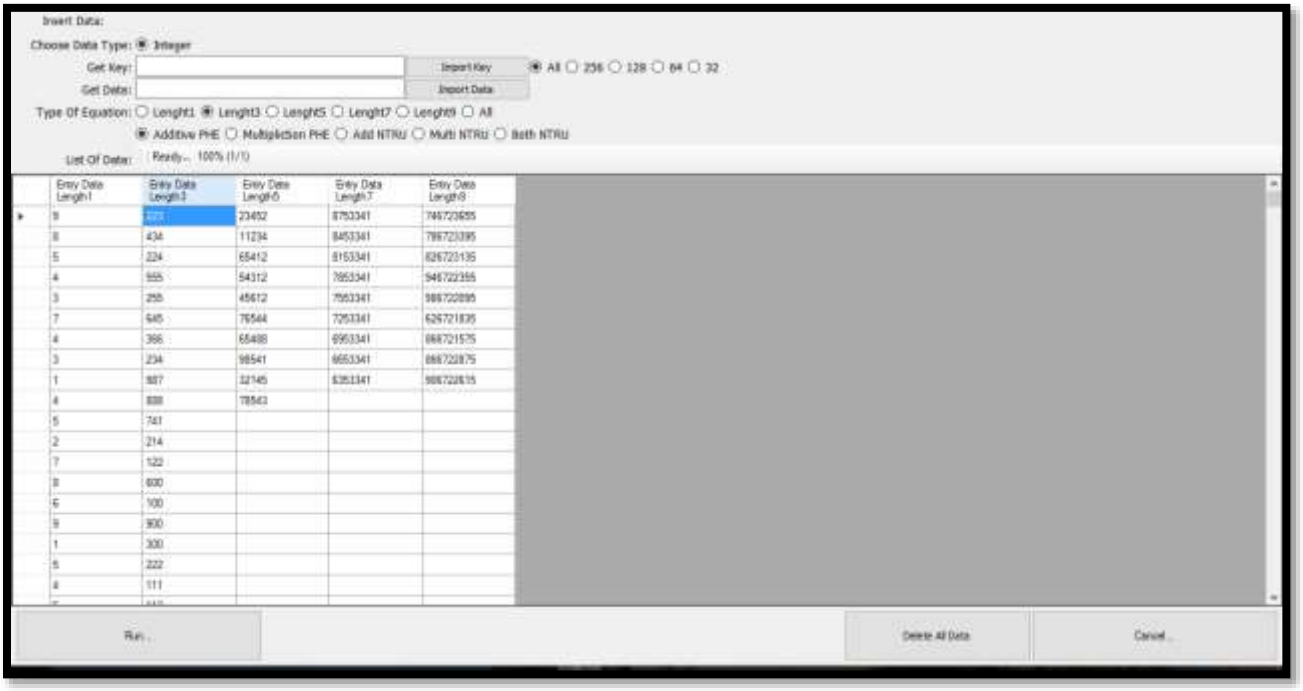


Figure (4-3): Execute Paillier (Multi Key Size)

The result of previous experiment has shown in Table (4-1) and Figure (4-4). The aim of this experiment is to find the optimal key size when importing (500) records as the number of input data, then testing each key size on the length three through Paillier equations. The difference between each time of each key is determined the best key size to give appropriate security level with minimum Performance loss.

Table (4-1): Result Execute Paillier (Multi Key size)

Key Size	Data Size	Execution Time	Total Numbers	Total Enc	Total Dec
32 bit	3 digit	0.80 ms	193658	8507529649	193658
64 bit	3 digit	0.81 ms	387316	2841349071	387316
128 bit	3 digit	0.83 ms	580974	4840229665	580974
256 bit	3 digit	0.95 ms	774632	5190901140	774632

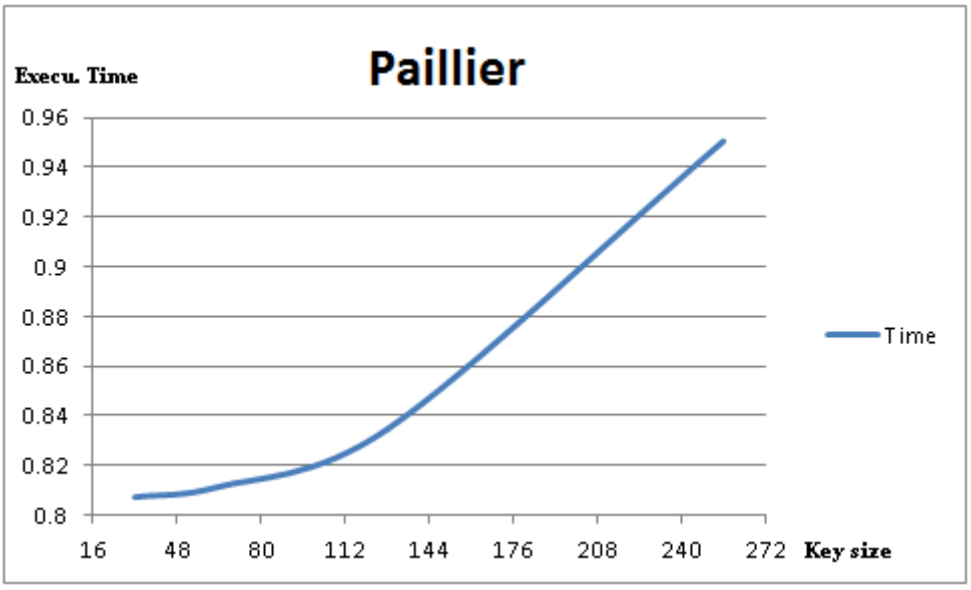


Figure (4-4): Timing Execute Paillier (Multi Key size)

1- Compute security gain for Paillier

From the Table (4-1), the researcher has implemented equation (4-1) that is used to compute the ratio of security gain. It has been used this equation for each key size on fixed data size through Paillier algorithm to find optimal outcomes. For example:-

(64/32) *100% = 200%

2- Compute performance loss for Add PHE (paillier)

From the Table (4-1), the researcher has implemented equation (4-2) that is used to compute the ratio of performance loss. It has been used this equation for each key size on fixed data size through Paillier algorithm to find optimal outcomes. For example:-

32-46 = 81-80/8 *100% = 1%

Table (4-2): Calculate security and Performance Ratio for Paillier (Multi Key size)

Key size	Security gain	Performance loss
32-64	200%	1%
32-128	400%	4%
64-128	200%	2%
32-256	800%	16%
64-256	400%	15%
128-256	200%	13%

In this experiment, the key size (256 bits) has the maximum security gain, while the key size (32 bits) has the minimum security gain. The increasing security level depending on the increasing key size in each experiment, are illustrated in result Table (4-2) and as shown below:-

- The increasing key size (from 32 bits to 256 bits), the ratio resulted in security gain is (800%) which considered the maximum security gain.
- The increasing key size (from 32 bits to 128 bits) or (from 64 bits to 256 bits), the ratio resulted in security gain is (400%).
- The increasing key size (from 32 bits to 64 bits) or (from 64 bits to 128 bits) or (from 128 bits to 256 bits), the ratio resulted in security gain is (200%) which considered the minimum security level.

In addition, the key size used is affected directly on the performance through effects on the time. The key size (256 bits) has the maximum Performance loss while the key size (32 bits) has the minimum Performance loss. The increasing Performance loss depending on the increasing key size in each experiment, as shown below:-

- The increasing key size (from 32 bits to 64 bits), the ratio resulted in the performance loss is (1%) which considered the minimum Performance loss.
- The increasing key size (from 32 bits to 128 bits), the ratio resulted in the performance loss is (4%).
- The increasing key size (from 64 bits to 128 bits), the ratio resulted in the performance loss is (2%).
- The increasing key size (from 32 bits to 256 bits), the ratio resulted in the performance loss is (16%) which considered the maximum Performance loss.
- The increasing key size (from 64 bits to 256 bits), the ratio resulted in the performance loss is (15%).
- The increasing key size (from 128 bits to 256 bits), the ratio resulted in the performance loss is (13%).

4.4.1.2 Execution Time of Multi-Key size by Using RSA Formula

Figure (4-5) shows the use of RSA equation on all types of key size (32, 64, 128 and 256 bits) with selected fixed data size (length three).

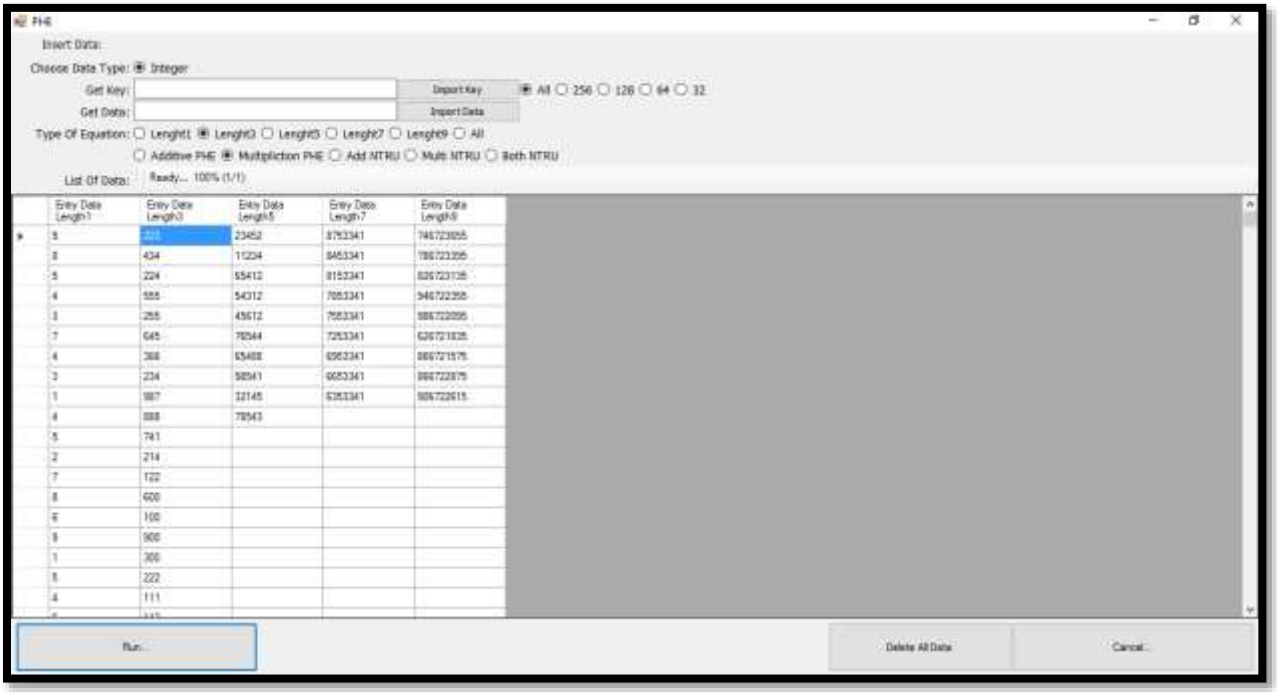


Figure (4-5): Execute Multiplication RSA (Multi Key size)

The result of previous experiment has shown in Table (4-3) and Figure (4-6). The aim of this experiment is to find the optimal key size when importing (500) records as the number of input data, then testing each key size on the length three through RSA equations. The difference between each time of each key is determined the best key size to give appropriate security level with minimum Performance loss.

Table (4-3): Result Execute Multiplication PHE (Multi key size)

Key Size	Data Size	Execution Time	Total Numbers	Total Enc	Total Dec
32 bit	3 digit	1.65 ms	∞	∞	∞
64 bit	3 digit	1.88ms	∞	∞	∞
128 bit	3 digit	1.90ms	∞	∞	∞
256 bit	3 digit	2.64ms	∞	∞	∞

Note: The (∞) is representing the large number in Tables

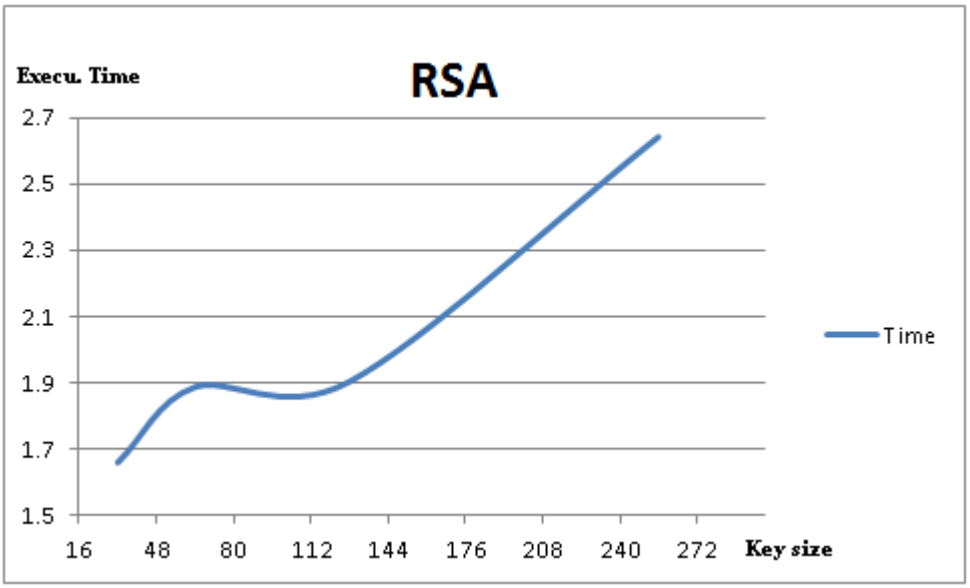


Figure (4-6): Timing Execute Multiplication PHE (Multi Key size)

1- Compute security level for Multiplication (RSA)

From the Table (4-3), the researcher has implemented equation (4-1) that is used to compute the ratio of security level. It has been used this equation for each key size on fixed data size through RSA algorithm to find optimal outcomes. For example:-

(128/32) *100% =400%

2- Compute performance for Multiplication PHE (RSA)

From the Table (4-3), the researcher has implemented equation (4-2) that is used to compute the ratio of performance loss. It has been used this equation for each key size on fixed data size through RSA algorithm to find optimal outcomes. For example:-

32→ 128= 190-165/190 *100% = 13%

Table (4-4): Calculate Security and Performance Ratio for Multiplication PHE (Multi Key size)

Key Size	Security gain	Performance loss
32→ 64	200%	12%
32→ 128	400%	13%
32→ 256	800%	1%
64→ 128	200%	37%
64→ 256	400%	29%
128→ 256	200%	28%

In this experiment, the key size (256 bits) has the maximum security level, while the key size (32 bits) has the minimum security level. The increasing security level depending on the increasing key size in each experiment, are illustrated in result Table (4-4) and as shown below:-

- The increasing key size (from 32 to 256) bits, the ratio resulted in the security level is (800%) which considered the maximum security level.
- The increasing key size (from 32 to 128) bits or (from 64 to 256) bits, the ratio resulted in the security level is (400%).
- The increasing key size (from 32 to 64) bits or (from 64 to 128) bits or (from 128 to 256) bits, the ratio resulted in the security level is (200%) which considered the minimum security level.

The key size used is affected directly on the performance through effects on the time. The key size (256 bits) has the maximum Performance loss while the key size (32 bits) has the minimum Performance loss. The increasing Performance loss depending on the increasing key size in each experiment, as shown below:-

- The increasing key size (from 32 to 64) bits, the ratio resulted in the performance loss is (12%).
- The increasing key size (from 32 to 128) bits, the ratio resulted in the performance loss is (13%).
- The increasing key size (from 32 to 256) bits, the ratio resulted in the performance loss is (1%) which considered the minimum Performance loss.
- The increasing key size (from 64 to 128) bits, the ratio resulted in the performance loss is (37%) which considered the maximum Performance loss.
- The increasing key size (from 64 to 256) bits, the ratio resulted in the performance loss is (29%).
- The increasing key size (from 128 to 256) bits, the ratio resulted in the performance loss is (28%).

4.4.1.3 Execution Time of Multi-Key size by Using Additive NTRU Formula

Figure (4-7) shows the use of Additive NTRU equation on all types of key size (32, 64, 128 and 256) bits with selected fixed data size (length three).

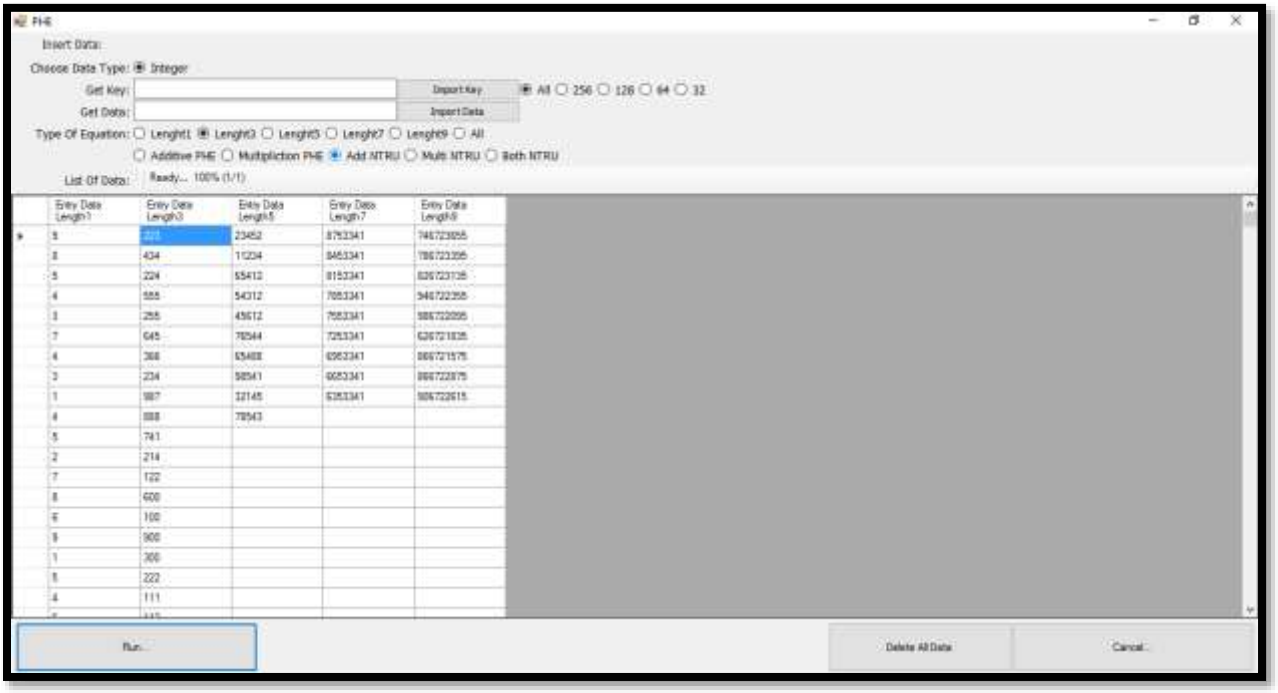


Figure (4-7): Execute Additive NTRU (Multi Key size)

The result of previous experiment has shown in Table (4-5) and Figure (4-8). The aim of this experiment is to find the optimal key size when importing (500) records as the number of input data, then testing each key size on the length three through Additive NTRU equations. The difference between each time of each key is determined the best key size to give appropriate security level with minimum Performance loss.

Table (4-5): Result Execute Additive NTRU (Multi Key size)

Key Size	Data Size	Execution Time	Total Numbers	Total Enc	Total Dec
32 bit	3 digit	0.80 ms	193658	618757524	193658
64 bit	3 digit	1.55ms	387316	795863326	387316
128 bit	3 digit	2.39ms	580974	1626654517	580974
256 bit	3 digit	3.20ms	774632	2463566562	774632

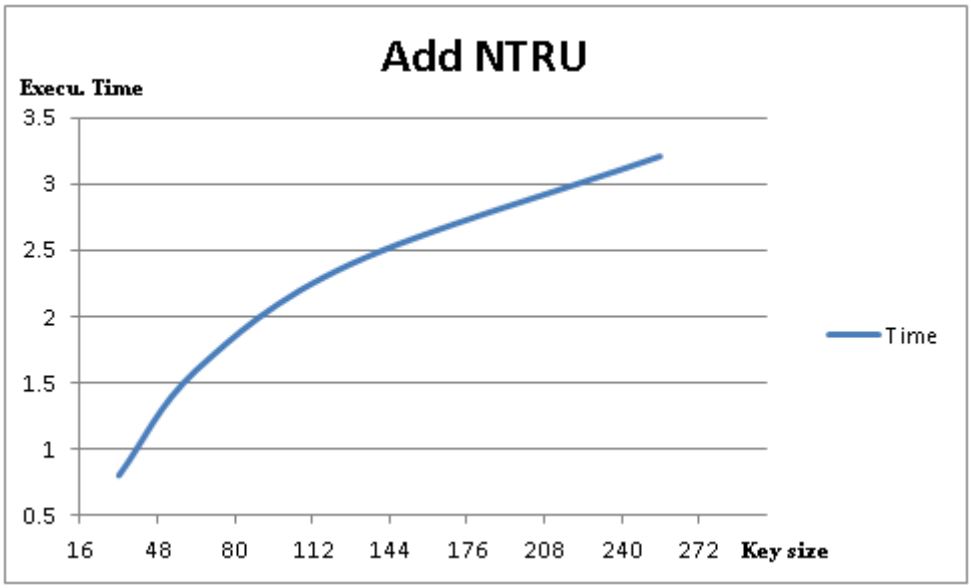


Figure (4-8): Timing Execute Additive NTRU (Multi Key size)

1- Compute security level for Add (NTRU)

From the Table (4-5), the researcher has implemented equation (4-1) that is used to compute the ratio of security level. It has been used this equation for each key size on fixed data size through Additive NTRU algorithm to find optimal outcomes. For example:-

(64/32) *100% = 200%

2- Compute performance for Add (NTRU)

From the Table (4-5), the researcher has implemented equation (4-2) that is used to compute the ratio of performance loss. It has been used this equation for each key size on fixed data size through Additive NTRU algorithm to find optimal outcomes. For example:-

32→ 64= 155-80/155 *100% = 48%

Table (4-6): Calculate Security and Performance Ratio for Additive NTRU (Multi Key size)

Key Size	Security gain	Performance loss
32→ 64	200%	48%
32→ 128	400%	66%
32→ 256	800%	75%
64→ 128	200%	35%
64→ 256	400%	52%
128→ 256	200%	25%

In this experiment, the key size (256 bits) has the maximum security level, while the key size (32 bits) has the minimum security level. The increasing security level depending on the increasing key size in each experiment, are illustrated in result Table (4-6) and as shown below:-

- The increasing key size (from 32 to 256) bits, the ratio resulted in the security level is (800%) which considered the maximum security level.
- The increasing key size (from 32 to 128) bits or (from 64 to 256) bits, the ratio resulted in the security level is (400%).
- The increasing key size (from 32 to 64) bits or (from 64 to 128) bits or (from 128 to 256) bits, the ratio resulted in the security level is (200%) which considered the minimum security level.

The key size used is affected directly on the performance through effects on the time. The key size (256 bits) has the maximum performance loss while the key size (32 bits) has the minimum performance loss. The increasing Performance loss depending on the increasing key size in each experiment, as shown below:-

- The increasing key size (from 32 to 64) bits, the ratio resulted in the performance loss is (48%).
- The increasing key size (from 32 to 128) bits, the ratio resulted in the performance loss is (66%).
- The increasing key size (from 32 to 256) bits, the ratio resulted in the performance loss is (75%) which considered the maximum Performance loss.
- The increasing key size (from 64 to 128) bits, the ratio resulted in the performance loss is (35%).
- The increasing key size (from 64 to 256) bits, the ratio resulted in the performance loss is (52%).
- The increasing key size (from 128 to 256) bits, the ratio resulted in the performance loss is (25%) which considered the minimum performance loss.

4.4.1.4 Execution Time of Multi-Key size by Using Multiplication NTRU Formula

Figure (4-9) shows the use of Multiplication NTRU equation on all types of key size (32, 64, 128 and 256) bits with selected fixed data size (length three).

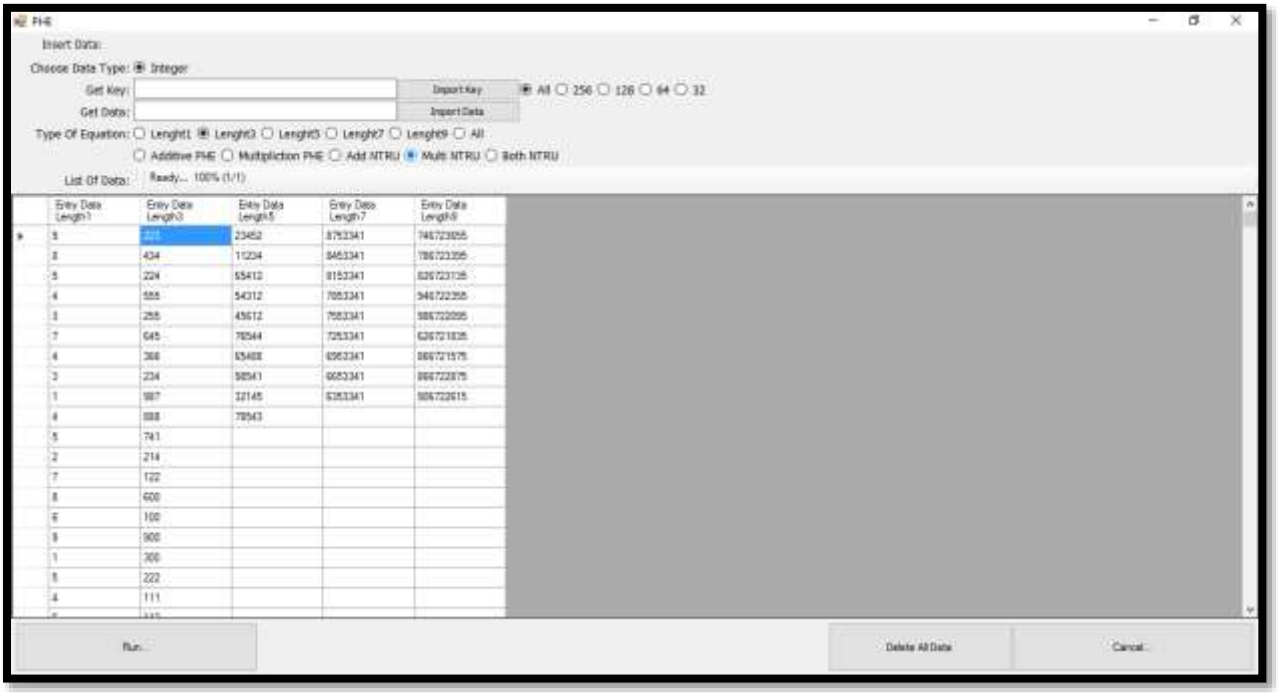


Figure (4-9): Execute Multiplication NTRU (Multi Key size)

The result of previous experiment has shown in Table (4-7) and Figure (4-10). The aim of this experiment is to find the optimal key size when importing (500) records as the number of input data, then testing each key size on the length three through Multiplication NTRU equations. The difference between each time of each key is determined the best key size to give appropriate security level with minimum Performance loss.

Table (4-7): Result Execute Multiplication NTRU (Multi Key size)

Key Size	Data Size	Execution Time	Total Numbers	Total Enc	Total Dec
32 bit	3 digit	1.48 ms	21679168	418126187	21679168
64 bit	3 digit	2.51ms	21679168	1385405542	21679168
128 bit	3 digit	3.64ms	21679168	1385405542	21679168
256 bit	3 digit	5.04ms	21679168	2384293196	21679168

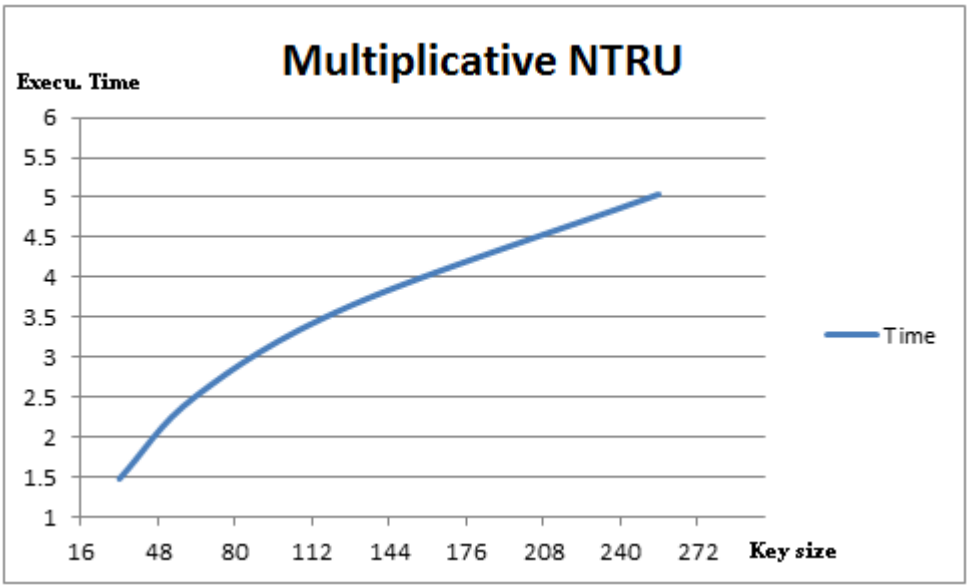


Figure (4-10): Timing Execute Multiplication NTRU (Multi Key size)

1- Compute security level for Multiplication (NTRU)

From the Table (4-7), the researcher has implemented equation (4-1) that is used to compute the ratio of security level. It has been used this equation for each key size on fixed data size through Multiplication NTRU algorithm to find optimal outcomes. For example:-

(64/32) *100% = 200%

2- Compute performance for Multiplication (NTRU)

From the Table (4-7), the researcher has implemented equation (4-2) that is used to compute the ratio of performance loss. It has been used this equation for each key size on fixed data size through Multiplication NTRU algorithm to find optimal outcomes. For example:-

32→ 64= 251-148/251*100% = 41%

Table (4-8): Calculate Security and Performance Ratio for Multiplication NTR (Multi Key size)

Key Size	Security gain	Performance loss
32→ 64	200%	41%
32→ 128	400%	59%
32→ 256	800%	71%
64→ 128	200%	31%
64→ 256	400%	50%
128→ 256	200%	28%

In this experiment, the key size (256 bits) has the maximum security level, while the key size (32 bits) has the minimum security level. The increasing security level depending on the increasing key size in each experiment, are illustrated in result Table (4-8) and as shown below:-

- The increasing key size (from 32 to 256) bits, the ratio resulted in the security level is (800%) which considered the maximum security level.
- The increasing key size (from 32 to 128) bits or (from 64 to 256) bits, the ratio resulted in the security level is (400%).
- The increasing key size (from 32 to 64) bits or (from 64 to 128) bits or (from 128 to 256) bits, the ratio resulted in the security level is (200%) which considered the minimum security level.

The key size used is affected directly on the performance through effects on the time. The key size (256 bits) has the maximum Performance loss while the key size (32 bits) has the minimum Performance loss. The increasing Performance loss depending on the increasing key size in each experiment, as shown below:-

- The increasing key size (from 32 to 64) bits, the ratio resulted in the performance loss is (41%).
- The increasing key size (from 32 to 128) bits, the ratio resulted in the performance loss is (59%).
- The increasing key size (from 32 to 256) bits, the ratio resulted in the performance loss is (71%) which considered the maximum performance loss.
- The increasing key size (from 64 to 128) bits, the ratio resulted in the performance loss is (31%).
- The increasing key size (from 64 to 256) bits, the ratio resulted in the performance loss is (50%).
- The increasing key size (from 128 to 256) bits, the ratio resulted in the performance loss is (28%) which considered the minimum performance loss.

4.4.1.5 Execution Time of Multi-Key size by Using Both NTRU Formula

Figure (4-11) shows the use of Both NTRU (Additive & Multiplication) equation on all types of key size (32, 64, 128 and 256) bits with selected fixed data size (length three).

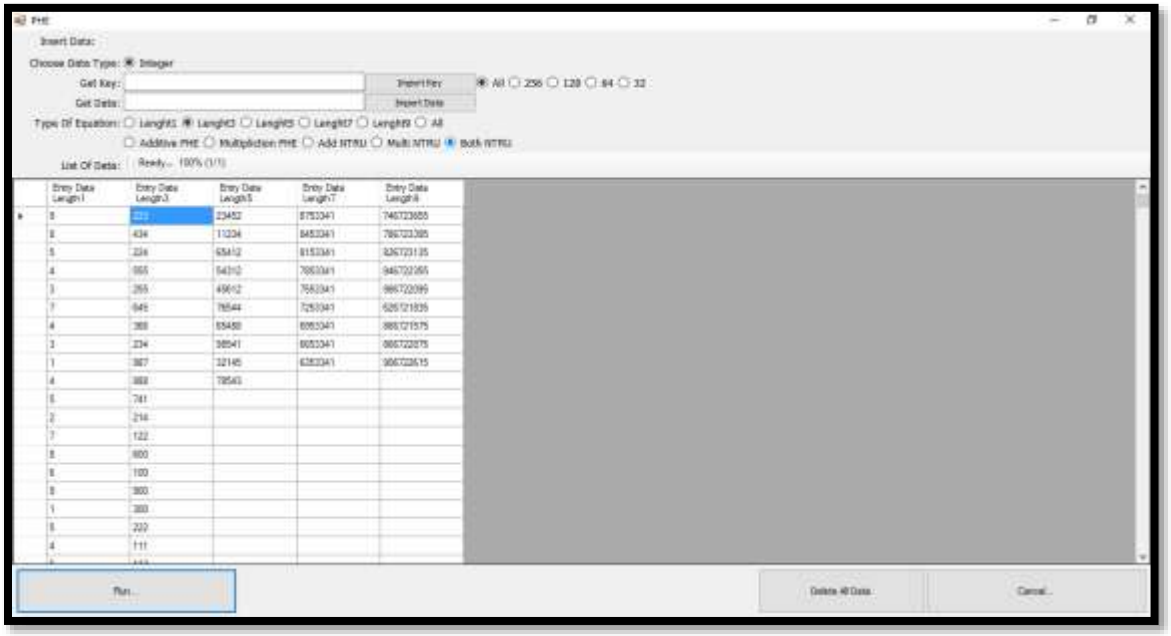


Figure (4-11): Execute Both NTRU (Multi Key size)

The aim of this experiment is to find the optimal key size when importing (500) records as the number of input data, then testing each key size on the length three through Both NTRU equations. The difference between each time of each key is determined the best key size to give appropriate security level with minimum Performance loss.

Table (4-9): Result Execute Both NTRU (Multi Key size)

Key Size	Data Size	Execution Time	Total Numbers	Total Enc	Total Dec
32 bit	3 digit	2.55ms	387317	1.188809	387317
64 bit	3 digit	2.71ms	774633	4.855508	774633
128 bit	3 digit	3.01ms	1161949	6.49908	1161949
256 bit	3 digit	4.74ms	1549265	1.355509	1549265

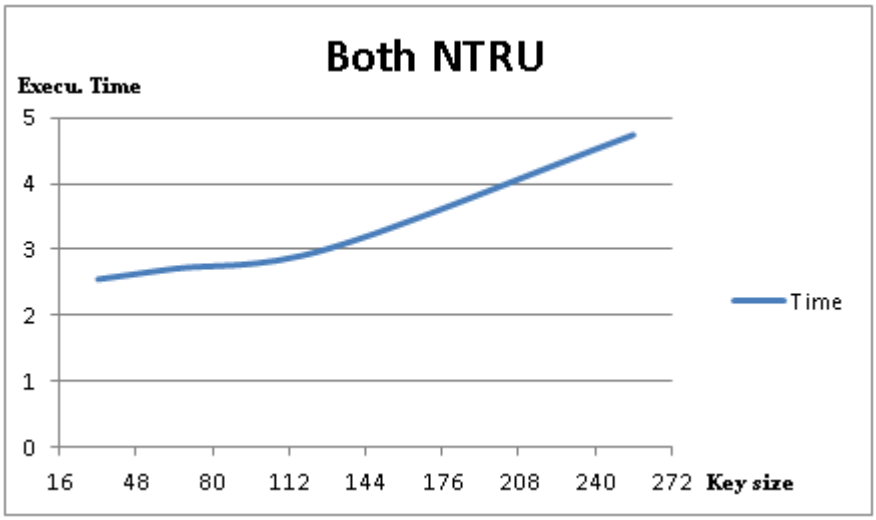


Figure (4-12): Timing Execute Both NTRU (Multi Key size)

1- Compute security level for both(NTRU)

From the Table (4-9), the researcher has implemented equation (4-1) that is used to compute the ratio of security level. It has been used this equation for each key size on fixed data size through Both NTRU algorithm to find optimal outcomes. For example:-

(64/32) *100% = 200%

2- Compute performance for both(NTRU)

From the Table (4-9), the researcher has implemented equation (4-2) that is used to compute the ratio of performance loss. It has been used this equation for each key size on fixed data size through Both NTRU algorithm to find optimal outcomes. For example:-

32→ 64 = 271-255/271*100% = 59%

Table (4-10): Calculate Security and Performance Ratio for Both NTRU (Multi Key size)

Key Size	Security gain	Performance loss
32→ 64	200%	6%
32→ 128	400%	15%
32→ 256	800%	46%
64→ 128	200%	10%
64→ 256	400%	43%
128→ 256	200%	36%

In this experiment, the key size (256 bits) has the maximum security level, while the key size (32 bits) has the minimum security level. The increasing security level depending on the increasing key size in each experiment, are illustrated in result Table (4-10) and as shown below:-

- The increasing key size (from 32 to 256) bits, the ratio resulted in the security level is (800%) which considered the maximum security level.
- The increasing key size (from 32 to 128) bits or (from 64 to 256) bits, the ratio resulted in the security level is (400%).
- The increasing key size (from 32 to 64) bits or (from 64 to 128) bits or (from 128 to 256) bits, the ratio resulted in the security level is (200%) which considered the minimum security level.

The key size used is affected directly on the performance through effects on the time. The key size (256 bits) has the maximum Performance loss while the key size (32 bits) has the minimum Performance loss. The increasing Performance loss depending on the increasing key size in each experiment, as shown below:-

- The increasing key size (from 32 to 64) bits, the ratio resulted in the performance loss is (6%) which considered the minimum Performance loss.
- The increasing key size (from 32 to 128) bits, the ratio resulted in the performance loss is (15%).
- The increasing key size (from 32 to 256) bits, the ratio resulted in the performance loss is (46%) which considered the maximum Performance loss.
- The increasing key size (from 64 to 128) bits, the ratio resulted in the performance loss is (10%).
- The increasing key size (from 64 to 256) bits, the ratio resulted in the performance loss is (43%).
- The increasing key size (from 128 to 256) bits, the ratio resulted in the performance loss is (36%).

4.4.2 Execution Fixed Key size with Multi-Data Size Procedure

The second procedure is implementing different data size on fixed key. The ranges of data sizes are used nine digits (1-9).

4.4.2.1 Execution Time of Multi-Data Size by Using Paillier Formula

Figure (4-13) shows the use of Paillier equation on different sizes of data (from one to nine digits) with selected fixed key size (128 bits).

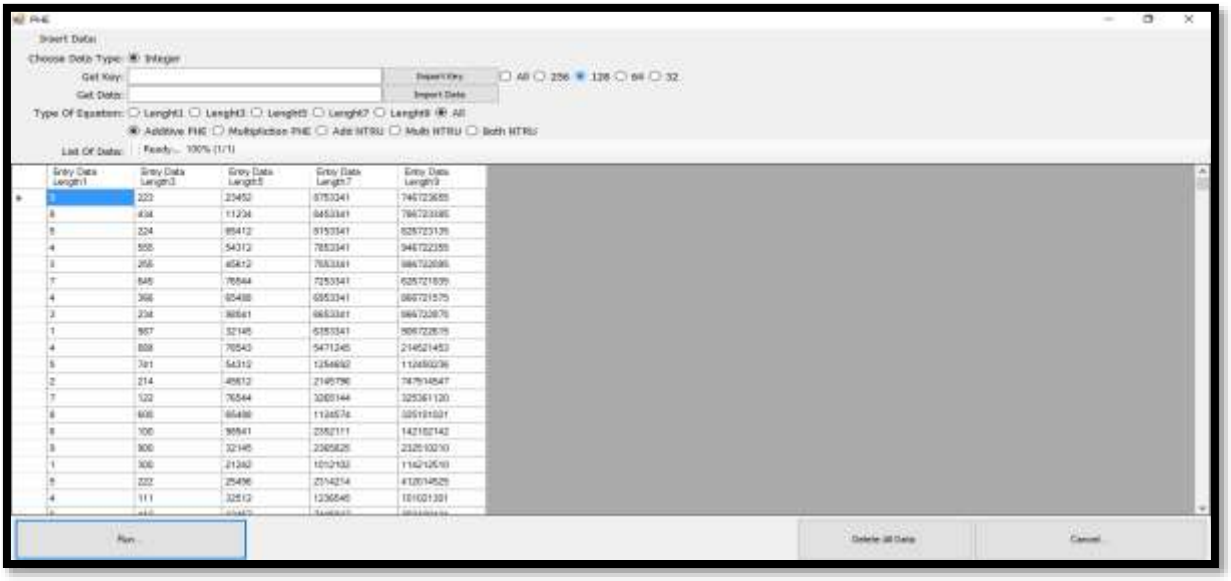


Figure (4-13): Execute Additive PHE (Multi Data size)

The result of previous experiment has shown in Table (4-11) and Figure (4-14). The aim of this experiment is to find the optimal data size on the key size (128 bits) through Paillier equation, when importing (500) records as the number of input data. The difference between each time of each data is determined the best data size to gives data flexibility with minimum Performance loss.

Table (4-11): Result Execute Additive PHE (Multi Data size)

Key Size	Data Size	Execution Time	Total Numbers	Total Enc	Total Dec
128 bit	1 digit	0.81ms	2585	1711917246	2585
128 bit	3 digit	0.83ms	196243	3304279607	196243
128 bit	5 digit	0.84ms	21673962	4001266315	21673962
128 bit	7 digit	0.85ms	2060636052	4087837316	2060636052
128 bit	9 digit	0.87ms	211922818947	5027054591	211922818947

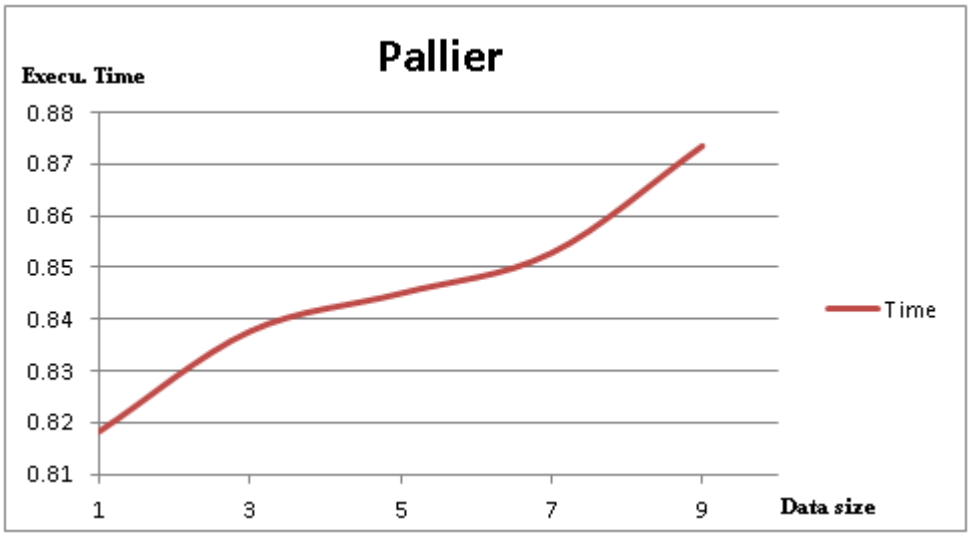


Figure (4-14): Timing Execute Additive PHE (Multi Data size)

1- Compute data flexibility for Add PHE (paillier)

From the Table (4-11), the researcher has implemented equation (4-1) that is used to compute the ratio of data flexibility. It has been used this equation for each Data size on fixed Key size through Paillier algorithm to find optimal outcomes. For example:-

(3/1)*100% =300%

2- Compute performance for Add PHE (paillier)

From the Table (4-11), the researcher has implemented equation (4-2) that is used to compute the ratio of performance loss. It has been used this equation for each Data size on fixed Key size through Paillier algorithm to find optimal outcomes. For example:-

1→ 3 = 83-81/83*100% = 2%

Table (4-12): Calculate Data Flexibility and Performance Ratio for Additive PHE (Multi Data size)

Data Size	Data Flexibility	Performance loss
1→ 3	300%	2%
1→ 5	500%	5%
1→ 7	700%	5%
1→ 9	900%	7%
3→ 5	166%	1%
3→ 7	233%	2%
3→ 9	300%	5%
5→ 7	140%	1%
5→ 9	180%	3%
7→ 9	128%	2%

In this experiment, the length (9 digits) of the data size has the maximum data flexibility, while the length (1 digit) of the data size has the minimum data flexibility. The increasing data flexibility depending on the increasing data size in each experiment, are illustrated in result Table (4-12) and as shown below:-

- The increasing data size (from 1 to 3) digits, the ratio resulted in the data flexibility is (300%).
- The increasing data size (from 1 to 5) digits, the ratio resulted in the data flexibility is (500%).
- The increasing data size (from 1 to 7) digits, the ratio resulted in the data flexibility is (700%).

- The increasing data size (from 1 to 9) digits, the ratio resulted in the data flexibility is (900%) which considered the maximum data flexibility.
- The increasing data size (from 3to 5) digits, the ratio resulted in the data flexibility is (166%).
- The increasing data size (from 3 to 7) digits, the ratio resulted in the data flexibility is (233%).
- The increasing data size (from 3 to 9) digits, the ratio resulted in the data flexibility is (300%).
- The increasing data size (from 5 to 7) digits, the ratio resulted in the data flexibility is (140%).
- The increasing data size (from 5 to 9) digits, the ratio resulted in the data flexibility is (180%).
- The increasing data size (from 7to 9) digits, the ratio resulted in the data flexibility is (128%) which considered the minimum data flexibility.

The data size used is affected directly on the performance through effects on the time. The length (9 digits) of the data size has the maximum Performance loss while the length (1 digit) of the data size has the minimum Performance loss. The increasing Performance loss depending on the increasing data size in each experiment, as shown below:-

- The increasing data size (from 1 to 3) digits and (from 3 to 7) digits and (from 7 to 9) digits, the ratio resulted in the performance loss is (2%).
- The increasing data size (from 1 to 5) digits and (from 1 to 7) digits and (from 3 to 9) digits, the ratio resulted in the performance loss is (5%).

- The increasing data size (from 1 to 9) digits, the ratio resulted in the performance loss is (7%) which considered the maximum performance loss.
- The increasing data size (from 3to 5) digits and (from 5 to 7) digits, the ratio resulted in the performance loss is (1%) which considered the minimum performance loss.
- The increasing data size (from 5 to 9) digits, the ratio resulted in the performance loss is (3%).

4.4.2.2 Execution Time of Multi-Data Size by Using RSA Formula

Figure (4-15) shows the use of RSA equation on different sizes of data (from one to nine digits) with selected fixed key size (128 bits).

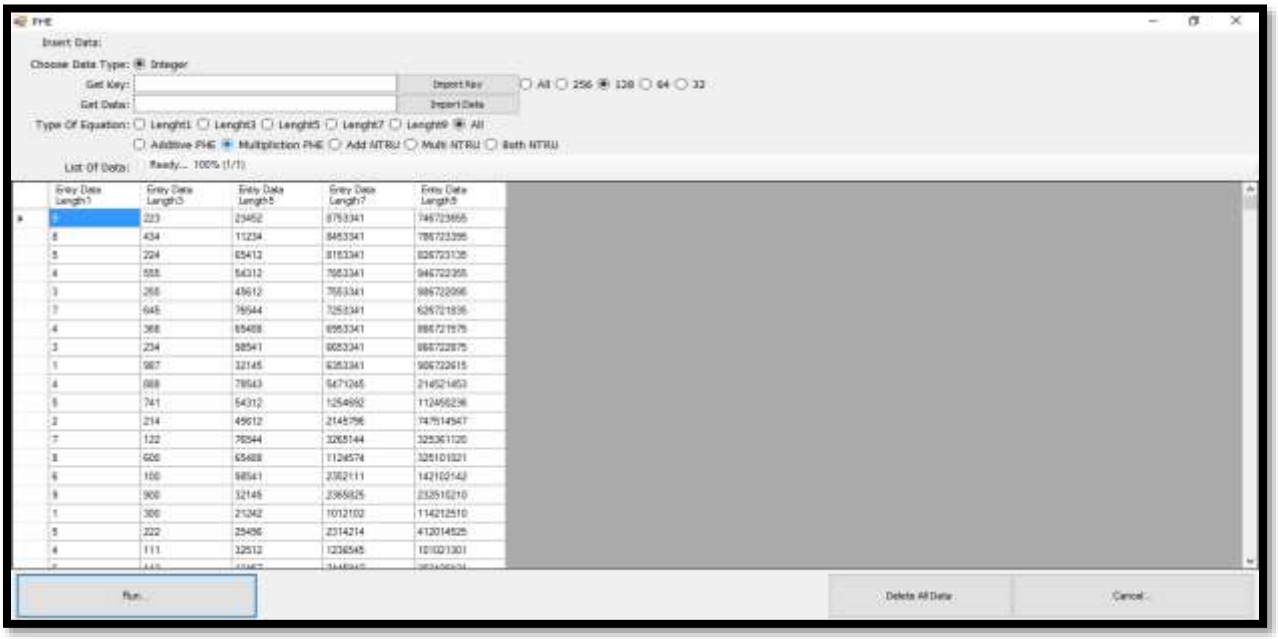


Figure (4-15): Execute Multiplication PHE (Multi Data size)

The result of previous experiment has shown in Table (4-13) and Figure (4-16). The aim of this experiment is to find the optimal data size on the key size (128 bits) through RSA equation, when importing (500) records as the number of input data. The difference between each time of each data is determined the best data size to gives data flexibility with minimum Performance loss.

Table (4-13): Result Execute Multiplication PHE (Multi Data size)

Key Size	Data Size	Execution Time	Total Numbers	Total Enc	Total Dec
128 bit	1 digit	0.88ms	∞	∞	∞
128 bit	3 digit	0.89ms	∞	∞	∞
128 bit	5 digit	0.92ms	∞	∞	∞
128 bit	7 digit	0.94ms	∞	∞	∞
128 bit	9 digit	0.99ms	∞	∞	∞

Note: The (∞) is representing the large number in Tables

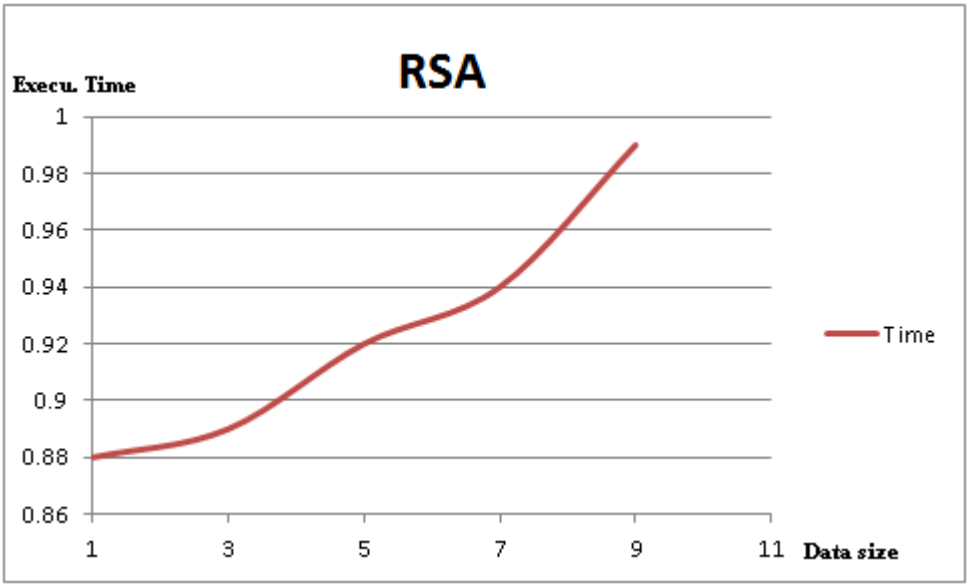


Figure (4-16): Timing Execute Multiplication PHE (Multi Data size)

1- Compute data flexibility Multiplication PHE (RSA)

From the Table (4-13), the researcher has implemented equation (4-1) that is used to compute the ratio of data flexibility. It has been used this equation for each Data size on fixed Key size through RSA algorithm to find optimal outcomes. For example:-

$$(3/1)*100\% = 300\%$$

2- Compute performance Multiplication PHE (RSA)

From the Table (4-13), the researcher has implemented equation (4-2) that is used to compute the ratio of performance loss. It has been used this equation for each Data size on fixed Key size through RSA algorithm to find optimal outcomes. For example:-

$$1 \rightarrow 3 = 89\text{-}88/89*100\% = 1\%$$

Table (4-14): Calculate Data Flexibility and Performance Ratio for Multiplication PHE (Multi Data size)

Data Size	Data Flexibility	Performance loss
1→ 3	300	1%
1→ 5	500	4%
1→ 7	700	6%
1→ 9	900	11%
3→ 5	166	3%
3→ 7	233	5%
3→ 9	300	10%
5→ 7	140	2%
5→ 9	180	7%
7→ 9	128	5%

In this experiment, the length (9 digits) of the data size has the maximum data flexibility, while the length (1 digit) of the data size has the minimum data flexibility. The increasing data flexibility depending on the increasing data size in each experiment, are illustrated in result Table (4-14) and as shown below:-

- The increasing data size (from 1 to 3) digits, the ratio resulted in the data flexibility is (300%).
- The increasing data size (from 1 to 5) digits, the ratio resulted in the data flexibility is (500%).
- The increasing data size (from 1 to 7) digits, the ratio resulted in the data flexibility is (700%).
- The increasing data size (from 1 to 9) digits, the ratio resulted in the data flexibility is (900%) which considered the maximum data flexibility.
- The increasing data size (from 3to 5) digits, the ratio resulted in the data flexibility is (166%).
- The increasing data size (from 3 to 7) digits, the ratio resulted in the data flexibility is (233%).
- The increasing data size (from 3 to 9) digits, the ratio resulted in the data flexibility is (300%).
- The increasing data size (from 5 to 7) digits, the ratio resulted in the data flexibility is (140%).
- The increasing data size (from 5 to 9) digits, the ratio resulted in the data flexibility is (180%).

- The increasing data size (from 7to 9) digits, the ratio resulted in the data flexibility is (128%) which considered the minimum data flexibility.

The data size used is affected directly on the performance through effects on the time. The length (9 digits) of the data size has the maximum Performance loss while the length (1 digit) of the data size has the minimum Performance loss. The increasing Performance loss depending on the increasing data size in each experiment, as shown below:-

- The increasing data size (from 1 to 3) digits, the ratio resulted in the performance loss is (1%) which considered the minimum performance loss.
- The increasing data size (from 1 to 5) digits, the ratio resulted in the performance loss is (4%).
- The increasing data size (from 1 to 7) digits, the ratio resulted in the performance loss is (6%).
- The increasing data size (from 1 to 9) digits, the ratio resulted in the performance loss is (11%) which considered the maximum performance loss.
- The increasing data size (from 3to 5) digits, the ratio resulted in the performance loss is (3%).
- The increasing data size (from 3 to 7) digits and (from 7 to 9) digits, the ratio resulted in the performance loss is (5%).
- The increasing data size (from 3 to 9) digits, the ratio resulted in the performance loss is (10%).
- The increasing data size (from 5 to 7) digits, the ratio resulted in the performance loss is (2%).

- The increasing data size (from 5 to 9) digits, the ratio resulted in the performance loss is (7%).

4.4.2.3 Execution Time of Multi-Data Size by Using Additive NTRU Formula

Figure (4-17) shows the use of Additive NTRU equation on different sizes of data (from one to nine digits) with selected fixed key size (128bits).

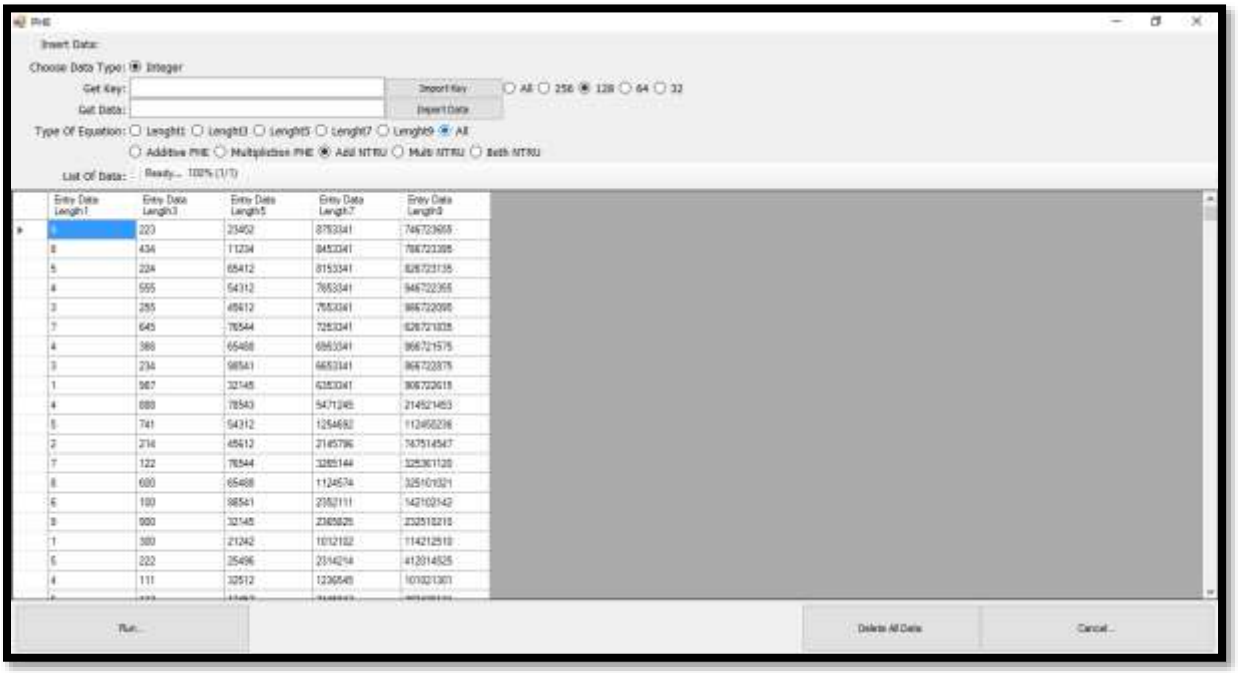


Figure (4-17): Execute Additive NTRU (Multi Data size)

The result of previous experiment has shown in Table (4-15) and Figure (4-18). The aim of this experiment is to find the optimal data size on the key size (128 bits) through Additive NTRU equation, when importing (500) records as the number of input data. The difference between each time of each data is determined the best data size to gives data flexibility with minimum Performance loss.

Table (4-15): Result Execute Additive NTRU (Multi Data size)

Key Size	Data Size	Execution Time	Total Numbers	Total Enc	Total Dec
128 bit	1 digit	1.20ms	2585	1770680748	2585
128 bit	3 digit	1.01ms	193658	504252597	193658
128 bit	5 digit	1.98ms	21477719	1403795769	21477719
128 bit	7 digit	1.90ms	2038962090	1494163018	2038962090
128 bit	9 digit	1.95ms	211922818947	2341332154	211922818947

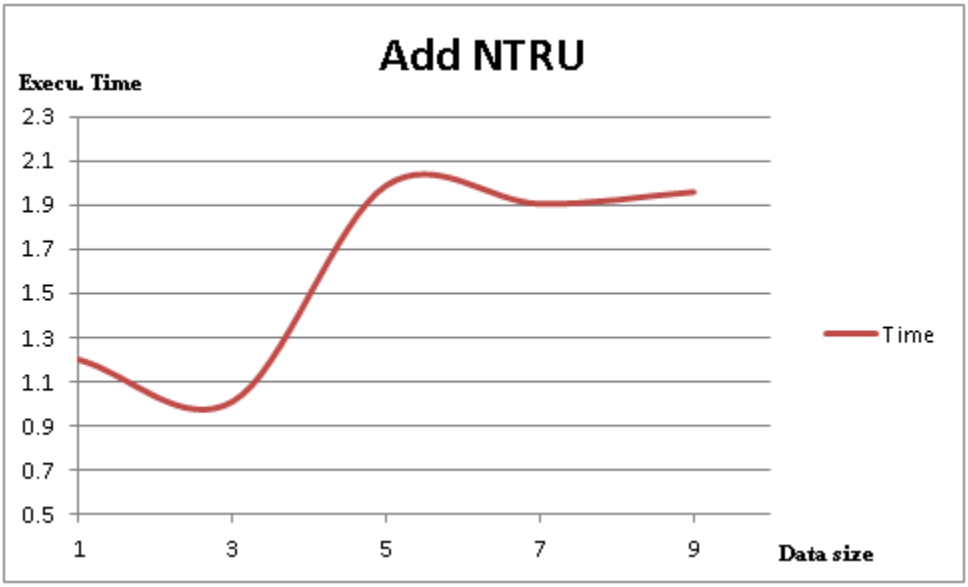


Figure (4-18): Timing Execute Additive NTRU (Multi Data size)

1- Compute data flexibility for add (NTRU)

From the Table (4-15), the researcher has implemented equation (4-1) that is used to compute the ratio of data flexibility. It has been used this equation for each Data size on fixed Key size through Additive NTRU algorithm to find optimal outcomes. For example:-

(3/1)*100% =300%

2- Compute performance for add (NTRU)

From the Table (4-15), the researcher has implemented equation (4-2) that is used to compute the ratio of performance loss. It has been used this equation for each Data size on fixed Key size through Additive NTRU algorithm to find optimal outcomes. For example:-

$$1\rightarrow 3 = 120\text{-}101/120*100\% = 15\%$$

Table (4-16): Calculate Data Flexibility and Performance Ratio for Additive NTRU (Multi Data size)

Data Size	Data Flexibility	Performance loss
1→ 3	300%	15%
1→ 5	500%	39%
1→ 7	700%	37%
1→ 9	900%	38%
3→ 5	166%	49%
3→ 7	233%	47%
3→ 9	300%	48%
5→ 7	140%	4%
5→ 9	180%	2%
7→ 9	128%	3%

In this experiment, the length (9 digits) of the data size has the maximum data flexibility, while the length (1 digit) of the data size has the minimum data flexibility. The increasing data flexibility depending on the increasing data size in each experiment, are illustrated in result Table (4-16) and as shown below:-

- The increasing data size (from 1 to 3) digits, the ratio resulted in the data flexibility is (300%).
- The increasing data size (from 1 to 5) digits, the ratio resulted in the data flexibility is (500%).
- The increasing data size (from 1 to 7) digits, the ratio resulted in the data flexibility is (700%).
- The increasing data size (from 1 to 9) digits, the ratio resulted in the data flexibility is (900%) which considered the maximum data flexibility.
- The increasing data size (from 3to 5) digits, the ratio resulted in the data flexibility is (166%).
- The increasing data size (from 3 to 7) digits, the ratio resulted in the data flexibility is (233%).
- The increasing data size (from 3 to 9) digits, the ratio resulted in the data flexibility is (300%).
- The increasing data size (from 5 to 7) digits, the ratio resulted in the data flexibility is (140%).
- The increasing data size (from 5 to 9) digits, the ratio resulted in the data flexibility is (180%).
- The increasing data size (from 7to 9) digits, the ratio resulted in the data flexibility is (128%) which considered the minimum data flexibility.

The data size used is affected directly on the performance through effects on the time. The length (9 digits) of the data size has the maximum Performance loss while the length (1 digit) of the data

size has the minimum Performance loss. The increasing Performance loss depending on the increasing data size in each experiment, as shown below:-

- The increasing data size (from 1 to 3) digits, the ratio resulted in the performance loss is (15%).
- The increasing data size (from 1 to 5) digits, the ratio resulted in the performance loss is (39%).
- The increasing data size (from 1 to 7) digits, the ratio resulted in the performance loss is (37%).
- The increasing data size (from 1 to 9) digits, the ratio resulted in the performance loss is (38%).
- The increasing data size (from 3 to 5) digits, the ratio resulted in the performance loss is (49%) which considered the maximum performance loss.
- The increasing data size (from 3 to 7) digits, the ratio resulted in the performance loss is (47%).
- The increasing data size (from 3 to 9) digits, the ratio resulted in the performance loss is (48%).
- The increasing data size (from 5 to 7) digits, the ratio resulted in the performance loss is (4%).
- The increasing data size (from 5 to 9) digits, the ratio resulted in the performance loss is (2%) which considered the minimum performance loss.
- The increasing data size (from 7 to 9) digits, the ratio resulted in the performance loss is (3%).

4.4.2.4 Execution Time of Multi-Data Size by Using Multiplication NTRU Formula

Figure (4-19) shows the use of Multiplication NTRU equation on different sizes of data (from one to nine digits) with selected fixed key size (128 bits).

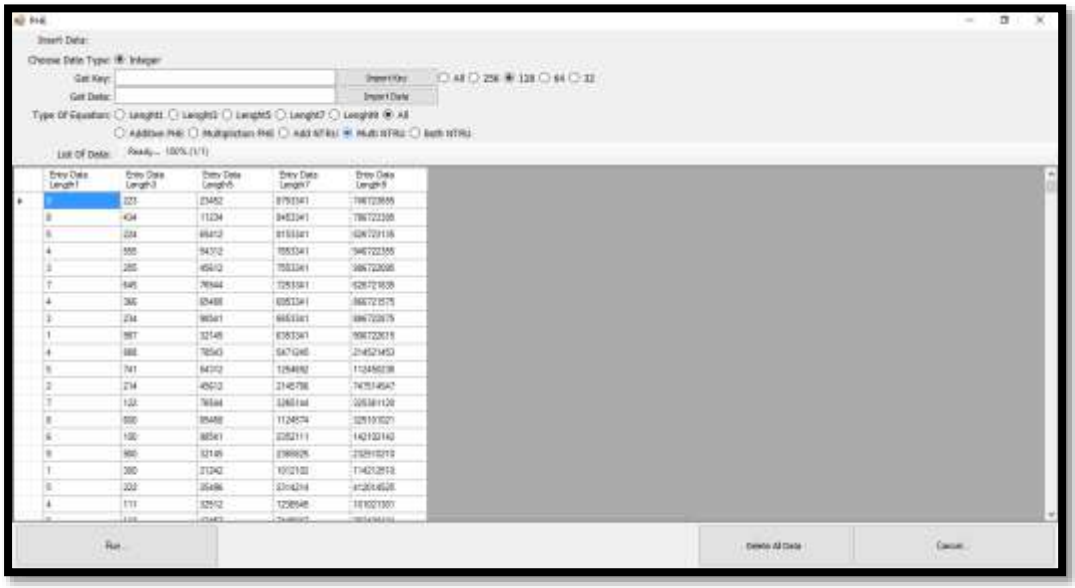


Figure (4-19): Execute Multiplication NTRU (Multi Data size)

The result of previous experiment has shown in Table (4-17) and Figure (4-20). The aim of this experiment is to find the optimal data size on the key size (128 bits) through Multiplication NTRU equation, when importing (500) records as the number of input data. The difference between each time of each data is determined the best data size to gives data flexibility with minimum Performance loss.

Table (4-17): Result Execute Multiplication NTRU (Multi Data)

Key Size	Data Size	Execution Time	Total Numbers	Total Enc	Total Dec
128 bit	1 digit	1.80ms	2586	1165628124	2585
128 bit	3 digit	1.78ms	193659	37349786	193659
128 bit	5 digit	1.93ms	21477720	57394769	21477720
128 bit	7 digit	2.00ms	2038962091	1900192984	2038962091
128 bit	9 digit	2.66ms	∞	1.03986064	∞

Note: The (∞) is representing the large number in Tables

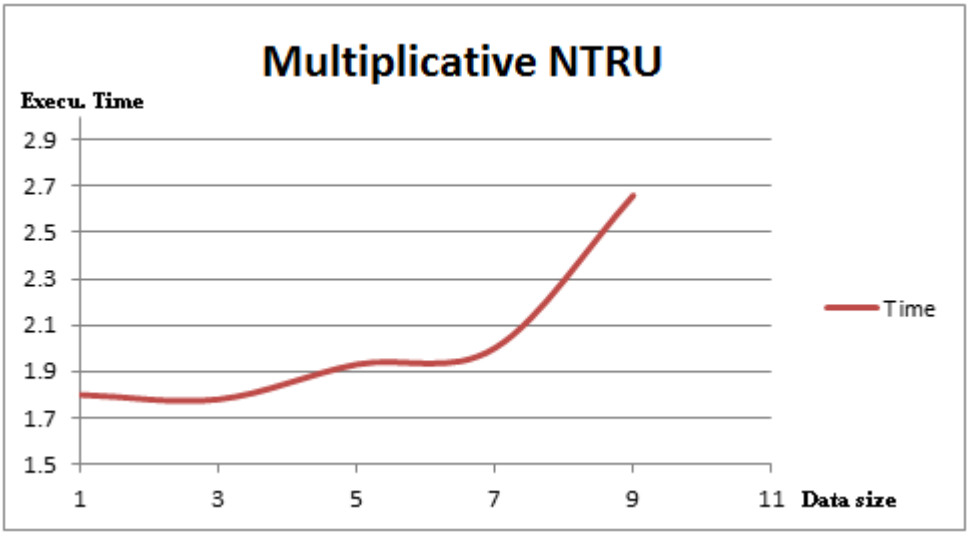


Figure (4-20): Timing Execute Multiplication NTRU (Multi Data size)

1- Compute data flexibility for multiplication (NTRU)

From the Table (4-17), the researcher has implemented equation (4-1) that is used to compute the ratio of data flexibility. It has been used this equation for each Data size on fixed Key size through Multiplication NTRU algorithm to find optimal outcomes. For example:-

(3/1)*100% =300%

2- Compute performance for Multiplication (NTRU)

From the Table (4-17), the researcher has implemented equation (4-2) that is used to compute the ratio of performance loss. It has been used this equation for each Data size on fixed Key size through Multiplication NTRU algorithm to find optimal outcomes. For example:-

$$1\rightarrow 3 = 180\text{-}178/180*100\% = \quad 1\%$$

Table (4-18): Calculate Data Flexibility and Performance Ratio for Multiplication NTRU (Multi Data size)

Data Size	Data Flexibility	Performance loss
1→ 3	300%	1%
1→ 5	500%	7%
1→ 7	700%	10%
1→ 9	900%	32%
3→ 5	166%	8%
3→ 7	233%	11%
3→ 9	300%	38%
5→ 7	140%	4%
5→ 9	180%	27%
7→ 9	128%	25%

In this experiment, the length (9 digits) of the data size has the maximum data flexibility, while the length (1 digit) of the data size has the minimum data flexibility. The increasing data flexibility depending on the increasing data size in each experiment, are illustrated in result Table (4-18) and as shown below:-

- The increasing data size (from 1 to 3) digits, the ratio resulted in data flexibility is (300%).
- The increasing data size (from 1 to 5) digits, the ratio resulted in the data flexibility is (500%).
- The increasing data size (from 1 to 7) digits, the ratio resulted in the data flexibility is (700%).
- The increasing data size (from 1 to 9) digits, the ratio resulted in the data flexibility is (900%) which considered the maximum data flexibility.
- The increasing data size (from 3to 5) digits, the ratio resulted in the data flexibility is (166%).
- The increasing data size (from 3 to 7) digits, the ratio resulted in the data flexibility is (233%).
- The increasing data size (from 3 to 9) digits, the ratio resulted in the data flexibility is (300%).
- The increasing data size (from 5 to 7) digits, the ratio resulted in the data flexibility is (140%).
- The increasing data size (from 5 to 9) digits, the ratio resulted in the data flexibility is (180%).
- The increasing data size (from 7to 9) digits, the ratio resulted in the data flexibility is (128%) which considered the minimum data flexibility.

The data size used is affected directly on the performance through effects on the time. The length (9 digits) of the data size has the maximum Performance loss while the length (1 digit) of the data size has the minimum Performance loss. The increasing Performance loss depending on the increasing data size in each experiment, as shown below:-

- The increasing data size (from 1 to 3) digits, the ratio resulted in the performance loss is (1%) which considered the minimum performance loss.
- The increasing data size (from 1 to 5) digits, the ratio resulted in the performance loss is (7%).
- The increasing data size (from 1 to 7) digits, the ratio resulted in the performance loss is (10%).
- The increasing data size (from 1 to 9) digits, the ratio resulted in the performance loss is (32%).
- The increasing data size (from 3to 5) digits, the ratio resulted in the performance loss is (8%).
- The increasing data size (from 3 to 7) digits, the ratio resulted in the performance loss is (11%).
- The increasing data size (from 3 to 9) digits, the ratio resulted in the performance loss is (38%) which considered the maximum performance loss.
- The increasing data size (from 5 to 7) digits, the ratio resulted in the performance loss is (4%).
- The increasing data size (from 5 to 9) digits, the ratio resulted in the performance loss is (27%).

- The increasing data size (from 7to 9) digits, the ratio resulted in the performance loss is (25%).

4.4.2.5 Execution Time of Multi-Data Size by Using Both NTRU Formula

Figure (4-21) shows the use of Both NTRU (Additive & Multiplication) equation on different sizes of data (from one to nine digits) with selected fixed key size (128 bits).

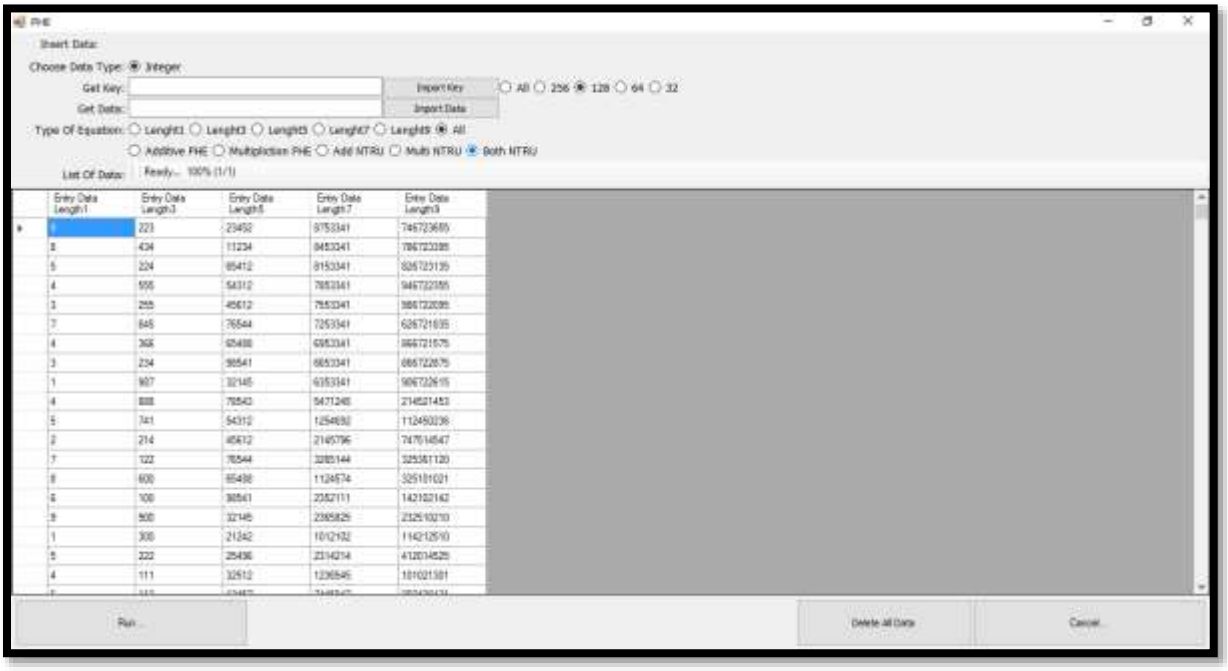


Figure (4-21): Execute Both NTRU (Multi Data size)

The result of previous experiment has shown in Table (4-19) and Figure (4-22). The aim of this experiment is to find the optimal data size on the key size (128 bits) through Both NTRU equation, when importing (500) records as the number of input data. The difference between each time of each data is determined the best data size to gives data flexibility with minimum Performance loss.

Table (4-19): Result Execute Both NTRU (Multi Data size)

Key Size	Data Size	Execution Time	Total Numbers	Total Enc	Total Dec
128 bit	1 digit	1.94ms	5171	∞	5171
128 bit	3 digit	3.00ms	387317	382289047	387317
128 bit	5 digit	3.87ms	54647538686	1038892822	54647538686
128 bit	7 digit	3.99ms	54647538686	∞	54647538686
128 bit	9 digit	5.22ms	7632123445	∞	7632123445

Note: The (∞) is representing the large number in Tables

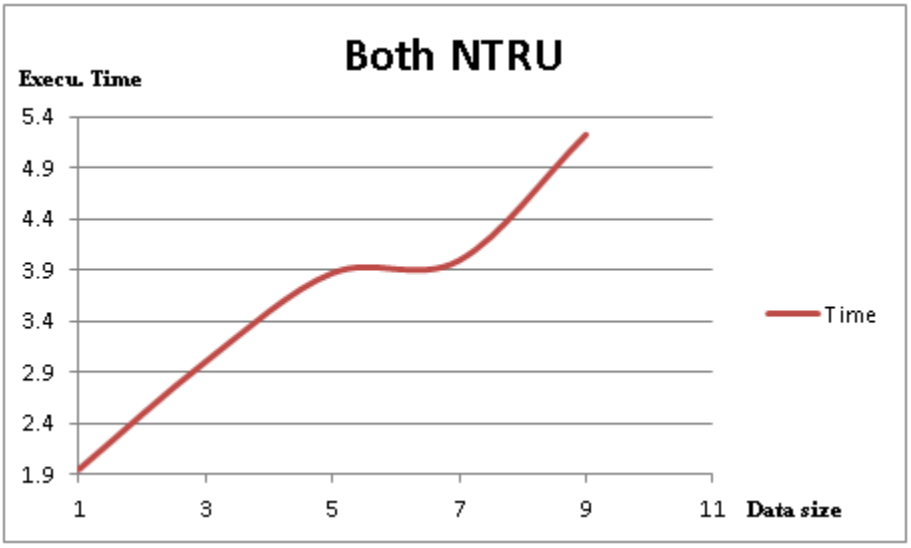


Figure (4-22): Timing Execute Both NTRU (Multi Data size)

1- Compute data flexibility for Both (NTRU)

From the Table (4-19), the researcher has implemented equation (4-1) that is used to compute the ratio of data flexibility. It has been used this equation for each Data size on fixed Key size through Both NTRU algorithm to find optimal outcomes. For example:-

(3/1)*100% =300%

2- Compute performance for Both (NTRU)

From the Table (4-19), the researcher has implemented equation (4-2) that is used to compute the ratio of performance loss. It has been used this equation for each Data size on fixed Key size through Both NTRU algorithm to find optimal outcomes. For example:-

$$1\rightarrow 3 = 300\text{-}194/300*100\% = 35\%$$

Table (4-20): Calculate Data Flexibility and Performance Ratio for Both NTRU (Multi Data)

Data Size	Data Flexibility	Performance loss
1→ 3	300%	35%
1→ 5	500%	50%
1→ 7	700%	51%
1→ 9	900%	63%
3→ 5	166%	22%
3→ 7	233%	25%
3→ 9	300%	43%
5→ 7	140%	3%
5→ 9	180%	26%
7→ 9	128%	23%

In this experiment, the length (9 digits) of the data size has the maximum data flexibility, while the length (1 digit) of the data size has the minimum data flexibility. The increasing data flexibility depending on the increasing data size in each experiment, are illustrated in result Table (4-20) and as shown below:-

- The increasing data size (from 1 to 3) digits, the ratio resulted in the data flexibility is (300%).
- The increasing data size (from 1 to 5) digits, the ratio resulted in the data flexibility is (500%).
- The increasing data size (from 1 to 7) digits, the ratio resulted in the data flexibility is (700%).
- The increasing data size (from 1 to 9) digits, the ratio resulted in the data flexibility is (900%) which considered the maximum data flexibility.
- The increasing data size (from 3to 5) digits, the ratio resulted in the data flexibility is (166%).
- The increasing data size (from 3 to 7) digits, the ratio resulted in the data flexibility is (233%).
- The increasing data size (from 3 to 9) digits, the ratio resulted in the data flexibility is (300%).
- The increasing data size (from 5 to 7) digits, the ratio resulted in the data flexibility is (140%).
- The increasing data size (from 5 to 9) digits, the ratio resulted in the data flexibility is (180%).
- The increasing data size (from 7to 9) digits, the ratio resulted in the data flexibility is (128%) which considered the minimum data flexibility.

4.5 Summary:-

This thesis has been implemented two algorithms (Partially Homomorphic and NTRU) to obtain the results shown in the following Tables (4-21, 4-22, 4-23, and 4-24). These tables identify the time difference between two similar experiments depending on different parameters (Key Size and Data Size). It means RSA algorithm with Multiplication NTRU and paillier algorithm with Additive NTRU:

1- Computed Time of Fixed Data Size with Multi-Key Size:-

- Paillier Algorithm with Additive NTRU

Table (4-21): Compare Time between Paillier and Additive NTRU with Different Key Size

Key size (bits)	Time of paillier	Time of Additive NTRU
32	0.80	0.80
64	0.81	1.55
128	0.83	2.39
256	0.95	3.20
Average time	0.85	1.98

The Table (4-21) will show the result obtained from the simulated equation for paillier with Additive NTRU in different Key size parameter. This result shown the encryption time of Additive NTRU is more than the encryption time of paillier almost in all cases and the average time in Additive NTRU is more than the average time of Paillier.

- RSA Algorithm with Multiplication NTRU

Table (4-22): Compare Time between RSA and Multiplication NTRU with Different Key Size

Key size (bits)	Time of RSA	Time of Multiplication NTRU
32	1.65	1.48
64	1.88	2.51
128	1.90	3.64
256	2.64	5.04
Average time	2.02	3.17

The Table (4-22) will show the result obtained from the simulated equation for RSA with Multiplication NTRU in different Key size parameter. This result shown the encryption time of Multiplication NTRU is more than the encryption time of RSA almost in all cases and the average time in Multiplication NTRU is more than the average time of RSA.

- Both PHE Algorithm with Both NTRU

Table (4-23): Compare Time between Both PHE and Both NTRU with Different Key Size

key Size (bits)	Time of Paillier+RSA	Time of Both NTRU
32	2.45	2.55
64	2.69	2.71
128	2.73	3.01
256	3.59	4.74
Average time	2.86	3.25

The Table (4-23) will show the result obtained from the simulated equation for Both PHE with Both NTRU in different Key size parameter. This result shown the encryption time of Both NTRU is more than the encryption time of Both PHE almost in all cases and the average time in Multiplication NTRU is more than the average time of Paillier & RSA.

2- Computed Time of Fixed Key size with Multi-Data Size

- Paillier Algorithm with Additive NTRU

Table (4-24): Compare Time between Paillier and Additive NTRU with Different Data Size

Data Size (digits)	Time of paillier	Time of Additive NTRU
1	0.81	1.20
3	0.83	1.01
5	0.84	1.98
7	0.85	1.90
9	0.87	1.95
Average time	0.84	1.61

The Table (4-24) will show the result obtained from the simulated equation for paillier with Additive NTRU in different Data size parameter. This result shown the encryption time of Additive NTRU is more than the encryption time of paillier in all cases and the average time in Additive NTRU is more than the average time of Paillier.

- RSA Algorithm with Multiplication NTRU

Table (4-25): Compare Time between RSA and Multiplication NTRU with Different Data Size

Data Size (digits)	Time of RSA	Time of Multiplication NTRU
1	0.88	1.80
3	0.89	1.87
5	0.92	1.93
7	0.94	2.00
9	0.99	2.66
Average time	0.92	2.03

The Table (4-25) will show the result obtained from the simulated equation for RSA with multiplication NTRU in different Data size parameter. This result shown the encryption time of multiplication NTRU is more than the encryption time of RSA in all cases and the average time in multiplication NTRU is more than the average time of RSA.

- Both PHE Algorithm with Both NTRU

Table (4-26): Compare Time between Both PHE and Both NTRU with Different Data Size

Data Size (digits)	Time of Paillier+RSA	Time of Both NTRU
1	1.69	1.94
3	1.72	3.00
5	1.76	3.87
7	1.79	3.99
9	1.86	5.22
Average time	1.76	2.83

The Table (4-26) will show the result obtained from the simulated equation for Both PHE with Both NTRU in different Data size parameter. This result shown the encryption time of Both NTRU is more than the encryption time of Both PHE almost in all cases and the average time in multiplication NTRU is more than the average time of RSA.

Chapter Five

Conclusion & Future Work

5.1 Conclusion

Cloud computing is a new approach for distributed network to share resources between users. Thus, many researchers focused on the protection of data in the cloud, and presented different methods for providing appropriate security method to this data.

The encryption techniques are the most popular method to provide the data secure. This thesis focused on the Homomorphic and NTRU encryption methods. It implemented two types of Homomorphic techniques, RSA which represents the multiplication operation and Paillier which represents the additive operation. Also, it implemented two types of operations over NTRU, additive NTRU and multiplication NTRU, and both NTRU.

The researcher implemented each experiment on two types of parameters (key size and data size). It concludes the results from these experiments to find the optimal parameters that affect the performance and security of Homomorphic and NTRU. Noted the Tables (4-21, 4-22, 4-23, 4-24, 4-25 and 4-26) presented the affected key size and data size on the performance very obvious, where the time of NTRU encryption increased 50% - 60% from the PHE depend on increased key size and data size.

It has been built many experiments depending on the analysis performance between PHE and NTRU to investigate the main goal of this research. This thesis accomplished many experiments to study the effect of several parameters on the performance and security. It was found the optimal point as a trade-off between security level and performance. The thesis has been computed the gain of security with the performance and data flexibility with performance as a percentage then compared between them. The results of the experiments have answered the questions of the thesis (as listed in section… 1.3). These results will be presented as follows:

- The effect of using different key size

1- Paillier-PHE: For Paillier, this thesis has found that increasing the size of key from size equal 32 bits to size equal 128 bits will gaining 400% security level with 4% losing performance. From Table (4-1), it is clear that the optimal key size for paillier (PHE) is 128 bits.
2- RSA-PHE: For RSA, This thesis has found that increasing the size of key from size equal 32 bits to size equal 256 bits will gaining 800% security level with 1% losing performance. From Table (4-3), it is clear that the optimal key size for RSA (PHE) is 256 bits.
3- Add-NTRU: For Add NTRU ,This thesis has found that increasing the size of key from size equal 64 bits to size equal 256 bits will gaining 400% security level with 52% losing performance. From Table (4-5), it is clear the optimal key size for Add (NTRU) is 256 bits.
4- multiplication -NTRU: For multiplication NTRU, This thesis has found that increasing the size of key from size equal 64 bits to size equal 256 bits will gaining 400% security level with 50% losing performance. From Table (4-7), it is clear the optimal key size for multiplication (NTRU) is 256 bits.
5- Both-NTRU: This thesis has found that increasing the size of key from size equal 32 bits to size equal 128 bits will gaining 400% security level with 15% losing performance. From Table (4-9), it is clear the optimal key size for both (NTRU) is 128 bits.

-The effect of using different data size

1- Paillier PHE: This thesis has found that increasing the data size from 3 digits to7 digits will be gaining 233% data flexibility with 2% losing performance. For Table (4-11), it is clear the optimal data size for Paillier (PHE) is 7 digits.
2- RSA PHE: This thesis has found that increasing the data size from 1 digits to5 digits will be gaining 500% data flexibility with 4% losing performance. For Table (4-13), it is clear the optimal data size for RSA (PHE) is 5 digits.
3- Add NTRU: This thesis has found that increasing the data size from 5 digits to 9 digits will be gaining 180% data flexibility with 2% losing performance. For Table (4-15), it is clear the optima data size for Add (NTRU) is 9 digits.
4- multiplication NTRU: This thesis has found that increasing the data size from 1digits to 5 digits will be gaining 500% data flexibility with 7% losing performance. For Table (4-17), it is clear the optimal data size for multiplication (NTRU) is 5 digits.
5- Both NTRU: This thesis has found that increasing the data size from 5digits to 7 digits will be gaining 140% data flexibility with 3% losing performance. For Table (4-19), it is clear the optimal data size for Both (NTRU) is 7 digits.

The researcher finds out that the optimal key size is 256 bits for RSA and NTRU (additive, multiplication) which give high security level and minimal Performance loss. Also, the thesis finds out that the optimal key size is 128 bits for Paillier and NTRU (Both) which give high security level and minimal Performance loss. For the optimal data size, the optimal data size is for RSA with 5 digits with minimal Performance loss (4%).

Based on the execution time, the researcher finds out that the equation of Paillier is the better than Additive NTRU, the equation of RSA is the better than multiplication NTRU, and finally, Both the PHE is the better than Both NTRU. At the end, worked to investigate accepted results through what was implemented in this thesis and took into consideration the nature of using the security from the key size and its impact on the performance also with the data size and its impact on performance.

5.2 Future Work

At the end of this thesis, the researcher offers some suggestion for future work to provide more security with the performance at the same time. These suggestions are:-

- It is possible to implement ElGamal algorithm instead of RSA of multiplication Homomorphic

- It is possible to implement Benaloh algorithm instead of Paillier of Additive Homomorphic

- It is possible to implement Alt-L´opez, Tromer and Vaikuntanathan (LTV) instead of NTRU

Algorthim.

Reference

- Abu Sharkh, M., Jammal, M., Shami, A., & Ouda, A. (2013). **Resource allocation in a network-based cloud computing environment: design challenges**. Communications Magazine, IEEE, 51(11), 46-52.
- Ali, M., Khan, S. U., & Vasilakos, A. V. (2015). **Security in cloud computing: Opportunities and challenges**. Information Sciences, 305, 357-383.
- Barker, E., Barker, W., Burr, W., Polk, W., & Smid, M. (2012) **Recommendation for Key Management-Part 1: General (Revision 3)**. NIST Special Publication, 800-57.
- Çetin, G. S., Doröz, Y., Sunar, B., & Savas, E. (2015). **Low Depth Circuits for Efficient Homomorphic Sorting**. IACR Cryptology ePrint Archive, 2015, 274.
- Chang, V., Walters, R. J., & Wills, G. (2013). **The development that leads to the Cloud Computing Business Framework**. International Journal of Information Management, 33(3), 524-538.
- Chen, D., & Zhao, H. (2012). **Data security and privacy protection issues in cloud computing**. In Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on (Vol. 1, pp. 647-651). IEEE.
- Dahab, R., Galbraith, S., & Morais, E. (2015). **Adaptive key recovery attacks on NTRU-based somewhat homomorphic encryption schemes**. In *Information Theoretic Security* (pp. 283-296). Springer International Publishing.
- 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). Ieee.

- Doröz, Y., Hu, Y., & Sunar, B. (2014). **Homomorphic AES evaluation using NTRU**. *IACR Cryptology Print Archive*, 2014, (39).
- Hu, Yin. (2013). **Improving the efficiency of homomorphic encryption schemes** (Doctoral dissertation, Worcester Polytechnic Institute).
- Huang, Q. L., Yang, Y. X., FU, J. Y., & NIU, X. X. (2013). **Secure and privacy-preserving DRM scheme using homomorphic encryption in cloud computing**. *The Journal of China Universities of Posts and Telecommunications,* 20(6), 88-95.
- Li, Y., Zhou, J., & Au, O. C. (2015). **Reducing the ciphertext expansion in image homomorphic encryption via linear interpolation technique**. In 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP) (pp. 800-804). IEEE.
- Liu, B. (2015). **Efficient Architecture and Implementation for NTRU Based Systems** (Master dissertation, University of Windsor).
- Liu, Z., Chen, X., Yang, J., Jia, C., & You, I. (2014). **New order preserving encryption model for outsourced databases in cloud environments**. Journal of Network and Computer Applications.
- Ma, Q. (2013). **The LTV Homomorphic Encryption Scheme and Implementation in Sage (*Doctoral dissertation, Worcester Polytechnic Institute*)**.
- Majithia, S., Singh, S., (2013). **Performance Evaluation of NTRU Algorithm on Cloud Network on an Android Platform**. International Journal on Recent and Innovation Trends in Computing and Communication, Vol. (1), Issue (11), PP(825 – 829).
- Majithia, S., Singh, S., (2013). **Implementation of NTRU on Cloud Network in an Android Platform and Comparison with DES and RSA**. International Journal of Advanced Research in Computer Science and Software Engineering, 3(11), 100-105.

- Mol, P., & Yung, M. (2008). **Recovering NTRU secret key from inversion oracles**. In Public Key Cryptography–PKC 2008 (pp. 18-36). Springer Berlin Heidelberg.
- Paillier, P. (1999). **Public-key cryptosystems based on composite degree residuosity classes**. In Advances in cryptology—EUROCRYPT'99 (pp. 223-238). Springer Berlin Heidelberg.
- Rahman, M. M., Saha, T. K., & Bhuiyan, M. A. A. (2012). **Implementation of RSA Algorithm for Speech Data Encryption and Decryption**. International Journal of Computer Science and Network Security (IJCSNS), 12(3), 74.
- 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.
- Rohloff, K., & Cousins, D. B. (2014). **A scalable implementation of fully homomorphic encryption built on NTRU**. *In Financial Cryptography and Data Security* (221-234). Springer Berlin Heidelberg.
- Samanthula, B. K., Howser, G., Elmehdwi, Y., & Madria, S. (2012). **An efficient and secure data sharing framework using homomorphic encryption in the cloud.** *In Proceedings of the 1st International Workshop on Cloud Intelligence* (8). ACM.
- Sarwar, A., & Khan, M. N. (2013). **A Review of Trust Aspects in Cloud Computing Security**. *International Journal of Cloud Computing and Services Science*, *2*(2), 116.
- 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**. In Advances in Cryptology-ASIACRYPT 2010 (377-394). Springer Berlin Heidelberg.

- Steinfeld, R. (2014). **NTRU cryptosystem: Recent developments and emerging mathematical problems in finite polynomial rings**. Algebraic Curves and Finite Fields: Cryptography and Other Applications, 16, 179.
- Tebaa, M., El Hajji, S., & El Ghazi, A. (2012). **Homomorphic encryption applied to the cloud computing security**. *In Proceedings of the World Congress on Engineering* (1), (4-6).
- Youssef, A. E. (2012). **Exploring cloud computing services and applications**. Journal of Emerging Trends in Computing and Information Sciences, 3(6), 838-847.
- Zissis, D., & Lekkas, D. (2012). **Addressing cloud computing security issues**. *Future Generation computer systems*, 28(3), (583-592).