جـــامـــعـــة الــــشـــرق الأوســـط
MIDDLE EAST UNIVERSITY
Amman - Jordan

Homomorphic Encryption of Text Documents

التشفير المتماثل للمستندات النصية

Prepared By

Omar Hanash

Supervisor

Dr. Mudhafar Al-Jarrah

Thesis Submitted in Partial Fullfillment of the Requirements for the Degree of Master in Cloud Computing

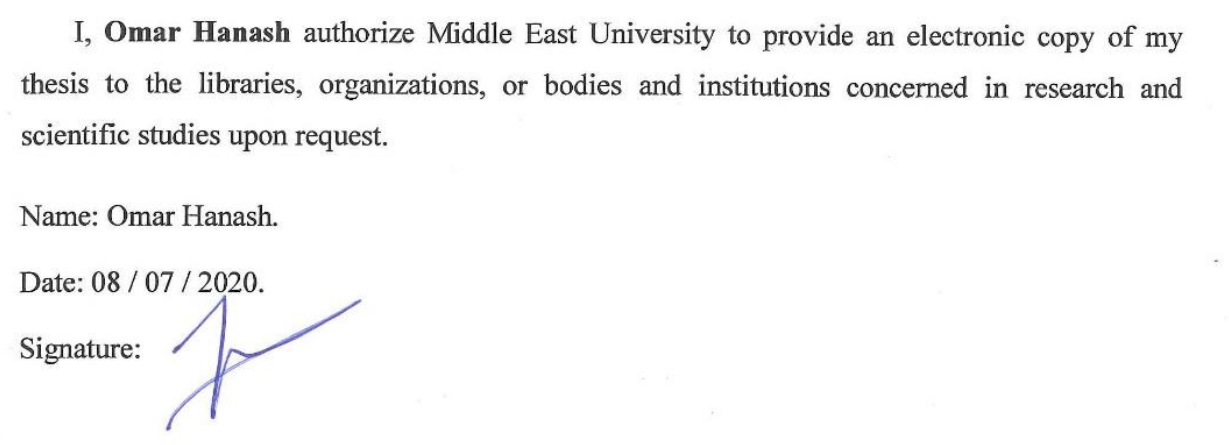
Department of Computer Science

Faculty of Information Technology

Middle East University

June, 2020

Authorization



I, **Omar Hanash** authorize Middle East University to provide an electronic copy of my thesis to the libraries, organizations, or bodies and institutions concerned in research and scientific studies upon request.

Name: Omar Hanash.

Date: 08 / 07 / 2020.

Signature: [illegible]

Examination Committee Decision

This is to certify that the thesis entitled "Homomorphic Encryption of Cloud-Based Text Document" was successfully defended and approved on 17-06-2020

Examination Committee Members Signature

(Supervisor)
Dr. Mudafar Al Jarrah
Associate Professor, Department of Computer Science
Middle East University

[illegible]

(Chairman of Examination and Committee Member)
Dr. Hesham Abu Saymeh
Associate Professor, Department of Computer Science
Middle East University

[illegible]

(External Committee Member)
Prof. Mohammad Ahmad Alia
Professor, Department of Computer Science
Al Zaytoona University

[illegible]

Acknowledgment

First, I give thanks, and praise to Allah for his mercy, and reconcile and for granting me knowledge, confidence, patience to pass this Master thesis successfully.

Also, I would like to express my gratitude to my thesis advisor. **Dr. Mudafar Al-Jarah** for the complete guidance throughout the thesis stages, and for the critical assistance in designing and preceding the methodology of my research.

Finally, I thank all those, who have helped me directly or indirectly in the successful completion of my research work.

Omar Hanash

The Researcher

Dedication

To the one who always kept me in her prayers, and did not save any effort to assist me throughout my life, **my Beloved Mother.**

To **My Father**, who has been always struggling to assure us a decent life, who raised me on the acts of mannerism, who kept admonishing me by the trust and honesty.

To my gorgeous **Brothers** who kept pushing me off the boundaries and assisted me in all the possible means.

To the woman who struggled with me through this journey and never stopped supporting and dedicating every possible means to support me through this journey, **My Wife**

To the light of my eyes, to my heart, **My Daughter**.

I dedicate my effort

Omar Hanash

Table of Content

List of Tables

List of Figures

List of Abbreviations

Abbreviations	Meaning
HE	Homomorphic Encryption
PHE	Partial Homomorphic Encryption
FHE	Fully Homomorphic Encryption
RSA	Rivest-Shamir-Adleman
Enc	Encryption
DataSet 1	Pride and Prejudice
DataSet 2	The Works of Edgar Allan Poe
DataSet 3	Alice's Adventures in Wonderland
DataSet 4	ION
DataSet 5	A Journal of the Plague Year
DataSet 6	The Adventures of Sherlock Holmes
DataSet 7	The Moby Dick; Or, The Whale
DataSet 8	The Yellow Wallpaper
DataSet 9	Frankenstein; Or, The Modern Prometheus
DataSet10	Importance of Being Earnest: A Trivial Comedy
VC++	Visual C++

Homomorphic Encryption of Text Documents

Prepared by:

Omar Hanash

Supervised by:

Dr. Mudhafar Al-Jarrah

Abstract

The big expansion of sensitive data stored on cloud computing platforms has focused the attention on the need for more secure data protection technologies. Homomorphic encryption has emerged as an important approach to protect cloud-based data by allowing operations on the encrypted data to be carried out without the need for decrypting the data while it is on the cloud. The work in this thesis extends the homomorphic encryption approach to deal with securing text documents that are stored on the cloud, by allowing the encrypted text documents to be searched using encrypted queries, thereby to make the query and its results ambiguous to a potential intruder. A homomorphic text encryption model is presented which provides text document encryption, query encryption, and ciphertext documents search using ciphertext queries based on ciphertext word index. The developed encryption algorithm performs half-byte swapping of character pairs directed by random numbers generated using a 16-decimal digit secret key that is used in the encryption and decryption processes. The proposed model has been implemented as a working encryption system using the C++ programming language. Experimental work was carried out using a corpora of 10 public text documents in the English language ranging in size from 26 KB to 1.2 MB. Performance analysis was carried out, the results showed that the average encryption time per KB was 288 milliseconds, while the average decryption time per KB was 160 milliseconds. Accuracy of the encryption system was evaluated by comparing the decrypted documents with the original plaintext documents, which yielded complete equivalence between every pair of documents. The thesis ends with conclusions and suggestions for future work.

Keywords: homomorphic, encryption, decryption, text document, ciphertext, plaintext, cloud computing

التشفير المتماثل للمستندات النصية

اعداد: عمر الحنش

اشراف: د. مظفر الجراح

الملخص

ركز التوسع الكبير في تخزين البيانات الحساسة على منصات الحوسبة السحابية الانتباه على الحاجة إلى تقنيات حماية البيانات الأكثر أمانًا. برز التشفير المتماثل كوسيلة مهمة لحماية البيانات المخزنة في السحب من خلال السماح بتنفيذ العمليات على البيانات المشفرة دون الحاجة إلى فك تشفير البيانات أثناء وجودها على السحب الحاسوبية. ان العمل في هذه الرسالة يساعد على توسيع التشفير المتماثل للتعامل مع تأمين المستندات النصية المخزنة على السحب الحاسوبية، من خلال السماح بالبحث في المستندات النصية المشفرة باستخدام الاستعلامات المشفرة ، وبالتالي جعل الاستعلام ونتائجه غامضة بالنسبة إلى متسلل محتمل. يتم تقديم نموذج تشفير النص المتماثل الذي يوفر تشفير المستند النصي وتشفير الاستعلام والبحث في مستندات النص المشفر باستخدام استعلامات النص المشفر بناءً على فهرس النص المشفر. تقوم خوارزمية التشفير المطورة بإجراء تبديل نصف بايت لأزواج الأحرف الموجهة بواسطة أرقام عشوائية تم إنشاؤها باستخدام مفتاح خاص مكون من 16 رقمًا عشريًا يتم استخدامه في عمليات التشفير وفك التشفير. تم تنفيذ النموذج المقترح كنظام تشفير يعمل باستخدام لغة البرمجة .++ C و تم تنفيذ العمل التجريبي باستخدام مجموعة من 10 مستندات نصية عامة باللغة الإنجليزية تتراوح في الحجم من 26 كيلوبايت إلى 1.2 ميجابايت. أظهرت النتائج أن متوسط وقت التشفير لكل كيلوبايت كان 288 مللي ثانية ، بينما كان متوسط وقت فك التشفير لكل كيلوبايت 160 مللي ثانية. تم تقييم دقة نظام التشفير من خلال مقارنة المستندات التي تم فك تشفيرها مع مستندات النص العادي الأصلية ، والتي أسفرت عن معادلة كاملة بين كل زوج من المستندات. الأطروحة تنتهي باستنتاج واقتراح للعمل في المستقبل.

الكلمات المفتاحية: التشفير، فك التشفير، مستند نصي، نص مشفر، نص عادي، الحوسبة السحابية، التشفير التماثلي

Chapter One
Introduction

1.1 Research Topic:

This thesis investigates Homomorphic Encryption of text documents to develop a secure text-based search over the cloud.

Cloud computing is delivering the traditional computing services that can be found at any premises that include services like servers, storage, databases, networking, software, and in recent days AI over the Internet, on remote data centers. Individuals and companies whether they are large or small can benefit from the cloud services and they only pay for the cloud services they use only, which makes the clouds sometimes a cheaper solution.

Homomorphic encryption is a ciphering method that allows any data to remains encrypted while it is being processed and manipulated. Homomorphic Encryption allows data owners from individuals or company owners or sometimes the authorized third parties (such as cloud providers) to apply the functionality to encrypted data directly without the need to disclose data. The Homomorphic encryption system is similar to other forms of public encryption as it uses a public key for data encryption and only allows the individuals with the corresponding private key to access the unencrypted data. However, what distinguishes it from other forms of encryption is that it allows algebraic operations and allows users to perform a variety of mathematical operations on the encrypted data.

1.2 Problem statement

Cloud storage has the advantages of offering low-cost services for the cloud user, with high scalability options, and easy to manage platforms. Having premises or individuals confidential and sensitive data is saved to third parties cloud providers especially cheap ones, that we can say that it has changed how the individuals and premises save their data has raised some questions to some of the cloud users, which made some of the corporates have private clouds computing, but that also never solves the issue of the data security. Many companies have to stick with old fashioned and known storage model to overcome their concerns. One of these concerns was accessing the data by unauthorized personnel or data theft, so encrypted storage was adopted.

With the arrival of the cloud computing paradigm and the proliferation of online services, the Internet stores not only information for sharing, but also a large amount of personal data demanding restricted access and privacy protection. Secure management of personal data stored online is an increasingly important issue, which demands a balance between data confidentiality and availability. Technologies that can enable secure online data management are going to be critically important for cloud computing to reach its full potential.

Traditional privacy protection for online personal data focuses on access control and secure data transmission to ensure that the data can be securely transmitted to the server and unauthorized people cannot access the data. Once the data arrives at the server, the server decrypts the data and operates on plaintext in order to provide services to users, such as search and data summarization. This makes the user’s private information vulnerable to untrustworthy service providers and malicious intruders. For example, personal photo albums can potentially be viewed by a system administrator if stored online in plaintext. Encryption of the data stored on the server using traditional cryptographic ciphers directly makes it difficult for the server to process the data, and for the user to retrieve information from the encrypted database. Therefore, it is both desirable and necessary to develop technologies for information retrieval over encrypted databases that can protect users' privacy without sacrificing the usability and accessibility of the information.

1.3 Goals and objectives

Implementing a new method to encrypt sensitive and confidential text documents before uploading them to the cloud where they are being stored, so when using this approach, the cloud server employees and anyone in the middle such as hackers or unauthorized personnel will not be getting any useful information about the files, as the server will be hosting only the ciphertext, encrypted index, and the encrypted query. Therefore, using this approach will guarantee the confidentiality not only of the stored documents but also of the query and its results.

The following objectives are sought to be realized:

1. The design of a new algorithm that will achieve the homomorphic encryption of text documents, based on half-byte swapping guided by random numbers that are generated from a secret key.
2. Implementation of the designed algorithm.
3. Dataset selection for the evaluation phase.
4. Performance analysis of the implemented system using the selected dataset.

1.4 Research questions:

1. How will the cloud-based text documents be encrypted homomorphically and how will the exncrypted documents be searched without decrypting the documents or the query anywhere outside the local computer?
2. What will be the encryption key, what is its data type?
3. What will be the algorithm steps that will achieve homomorphic encryption and search on text document word
4. What are the content and structure of the word index and will it be stored in plaintext or ciphertext ?

5. Where will the encryption and decryption of the text documents be performed ?

1.5 Delimitations

The work will not involve numeric manipulation or algebraic processing on numeric data. Also, the text search will be based on single-word terms, which can be extended later on to phrases and multi-word terms.

Chapter Two

Background and Related Work

2.1 Cloud computing

Cloud computing's biggest advantages are that the cloud is available at any time the user wants to access it, network access, resource pooling, elasticity, and measured service. Availability means that cloud users can access and they can manage their computing resources at any time and from anywhere, as long as the clouds are connected to the internet and up. Pooled resources mean that cloud users can use from a pool of computing resources if they need more resources to be added to their current cloud if the current setup they have is not enough. Elasticity means that services can be scaled larger or smaller, depending on the cloud user requirements. Moreover, the cloud user will pay only for what is being used from the resources of the cloud. (Tharam Dillon et al. 2010)

2.2 Cloud Computing Services:

There are mainly three services that the cloud provides provide for their customers, the most popular services are (Junjie Peng et al. 2009)

Platform as a Service (PaaS): The cloud provider provides a platform for the creation of software that is delivered to customers over the web. PaaS allows users to create applications easily without the hassle of buying and maintaining the software or infrastructure.

Software as a Service (SaaS): The cloud provider provides the applications that are needed by the cloud users as a service. Applications are connected to customers' cloud via the Internet and applications are owned and operated by customers. Ex: Google Apps, Zoho, Kayako…etc

Infrastructure as a Service (IaaS): The cloud provider delivers the computing infrastructure such as storage or server space, servers, and network infrastructure as on-demand service. Instead of purchasing the computer hardware from vendors, cloud users can order IaaS based cloud and they pay per what they use.

Many large companies like Google, Amazon, Microsoft, IBM, Alibaba, and many more are developing cloud infrastructure to providing services to those who wish to host their work or rent a cloud server through the internet. Cloud computing has offered a new mean of utilization the computing resources by sharing the resources with several users, and each user can just pay for what is being used from resources without and data can be accessed at any time and anywhere as long there is an internet connection between the cloud users and the cloud providers.

2.3 Homomorphic Encryption

Homomorphic Encryption is considered different from the traditional and known encryption methods by permitting computations to be done to the ciphered data (encrypted data) directly without the need of decrypting the data or any access to the secret key used to encrypt the original data. The result of the computation is still in an encryption form after all the needed computation is done on the encrypted data the result is still encrypted until the user decrypts the data again to see the results. The outcome of the computation data that happens on the Encrypted data is the same result of the computation done on the plain data itself. (Amit Joshi et al. 2019)

2.3.1 History of Homomorphic Encryption

Ronald Rivest and Leonard Adleman suggested the concept of homomorphic encryption in 1978. However, for 30 years the progress is very slow. In 1982, Shafi Goldwasser and Silvio Micali proposed their encryption system that able to encrypt one bit in additive homomorphic encryption. Pascal Paillier 1999 suggested another additive homomorphic encryption. In 2005, Dan Boneh, Eu-Jin Goh, and Kobi invented a security system of encryption, which conducts only single multiplication but a large number of additions. In 2009, Craig Gentry constructs a fully homomorphic encryption-based system that able to conduct both addition and multiplication at the same time. (Jabbar 2009)

The data stored in the cloud will not be in an encrypted format. If it is stored in an encrypted way that can solve issues like Availability, Data security, and Third-party control. But the problem is the user will not be able to depend on the cloud service provider to carry out the computation of data. For this the data will be decrypted first then will be shipped to the user for computation. So the cloud provider has to decrypt the data first thus nullifying the issues of privacy and confidentiality, perform the computation, and then send the result to the user (Kanagavalli 2014). Suppose if the user could carry out any arbitrary computation on the hosted data, then without the cloud provider learning about the users' data, computation is done on the encrypted data without prior decryption. In this scenario, the promise of homomorphic encryption takes a call (Payal 2014).

Homomorphic encryption schemes are methods that allow the transformation of ciphertexts C(M) of message M , to ciphertexts C(f(M)) of a computation/function of message M, without disclosing the message. Generally, an encryption scheme contains three-step algorithms. They are

1. Key Generation - creates two keys i.e. the secret key sk

2. Encryption - encrypts the plaintext m with the secret key sk to yield ciphertext c.

3. Decryption - decrypts the ciphertext c with the secret key sk to retrieve the plaintextm.

In addition to the above stated three steps, homomorphic encryption schemes involve another 4 steps namely Storage, Request, Evaluation, and Response.

In the cloud-based environment, the key generation takes place at the client-side and encrypts the data with the encryption key and sends the data to the cloud server along with sk . The encrypted data is stored in the database along with the key. Whenever the client wants to operate it sends the request to the service provider. The service provider forwards the request to the processing server .the processing server operates as per request. The service provider then returns the processed result to the client in the response phase. The client finally decrypts the result returned by the service provider with the secret key sk. Among the homomorphic encryption schemes available depending on the operations performed on data, can be classified into three main categories namely: Partially Homorphic Encryption(PHE), Some What Homomorphic Encryption (SWHE) and Fully Homomorphic Encryption(FHE) .

2.4 Categories of Homomorphic Encryption

Homomorphic encryption can be classified into 2 main parts: Partially Homomorphic Encryption (PHE) and Fully Homomorphic Encryption (FHE). (Gentry 2009)

Homomorphic Encryption (PHE), such as the traditional and knows RSA, ElGamal, Paillier, and other known encryption methods enables executing algebraic operations to the ciphertext (Encrypted data) that can be either multiplication or addition operation.

Constructing a Partially Homomorphic Encryption that can brace both algebraic operations simultaneously was very hard, even though Boneh et al. came near. Gentry in the year 2009 had managed to make the two arithmetic operations with one another.

2.4.1 Partially Homomorphic Encryption

Partially Homomorphic Encryption PHE is an old and known method for years, as it enables us to operate on the ciphered data ((Encrypted data)) without the need to Encrypt it first, these operations are algebraic operation like additions and multiplication. Addition PHE like Paillier and multiplicative like ElGamal cryptosystems. (Moore C 2014)

Homomorphic Encryption is considered to be as multiplicative if there is a function to find the result of Enc(x * y) from Enc (x) and Enc (y) not knowing the original values of x and y. Such as RSA and ElGamal Algorithms.

Homomorphic Encryption is considered to be as multiplicative if there is a function to find the result of Enc(x + y) from Enc (x) and Enc (y) not knowing the original values of x and y. Such as Paillier algorithms.

2.4.2 Fully Homomorphic Encryption

As we have discussed earlier the Partially Homomorphic encryption permits the homomorphic computation on only one mathematical operation, so it can be an additional function or a multiplication function, on the ciphered data (Encrypted data).

The challenge to develop of a scheme or a function that permits unlimited additions or multiplications to be done on the ciphertext was a challenging issue until Craig Gentry in the year 2009 has solved this issue and proposed the 1st holy grail solution of Fully Homomorphic Encryption FHE.

What Gentry proposed and worked on was supporting both addition and multiplication on ciphertext at the same time, by performing AND $\wedge$ and XOR$\oplus$ on the ciphertext. In algebra, there are too many methods that can be used to turn the complex function into more simple ones. With this technique, a function can be transformed to use only a specific logical operation (e.g. $\wedge$ or $\oplus$).

For example, $\neg A$ can be expressed as $A \oplus 1$, or it is expressed as $A \vee B$, this can be converted into $(\neg A) \wedge (\neg B)$, then converted into $(A \oplus 1)$ ^ $(B \oplus 1)$. By utilizing such techniques, all functions can be converted into a series of ($\wedge$) and ($\oplus$) operations. And that was the basis of what Gentry has worked on and proposed.

Gentry has introduced the lattice-based cryptography. Gentry has proposed fully homomorphic encryption relying on the following scenario, starting from somewhat homomorphic encryption using ideal lattices (Gentry 2009) are limited to evaluating low-degree polynomials over encrypted data. It is limited because each ciphertext is noisy in some sense, and this noise grows as one adds and multiplies ciphertexts until ultimately the noise makes the resulting ciphertext indecipherable. Next, it squashes the decryption procedure so that it can be expressed as a low-degree polynomial, which is supported by the scheme. Finally, it applies a bootstrapping transformation, through a recursive self-embedding, to obtain a fully homomorphic scheme. (Gentry 2010)

In a mathematical perspective the Fully Homomorphic Encryption scheme is quadrant polynomial algorithms (Gen, Enc, Dec, $Eval$) where:

• (λ): The algorithm of key generation.

• (m, pk): The encryption algorithm, where it takes a plaintext m and a public key pk as inputs and outputs a ciphertext c.
• (c, sk): Is the decryption algorithm, takes as input a ciphertext c and a secret key sk and outputs a plaintext m.

• $Eval(C, c1, \ldots, cn)$: This is an evaluation algorithm, takes as input C and ciphertexts $c1, \ldots, cn$ and verifies $Dec(Eval(C, c1, \ldots, cn), sk) = C(m1, \ldots, mn)$.

2.5 Homomorphic Encryption Schemes in cloud storage challenges

The following are the few challenges that HE schemes provide which we have considered in our work: a) Efficiency b) Robustness c) Delay. PHE algorithms are very effective to ensure the security of applications and data in the model where data can be encrypted during the transfer phase. They are very useful in the case of a cloud service model of SaaS or PaaS but not as useful in the IaaS model as it requires that the secret key is transmitted at a given time usually while booting the VM (Mallaiah 2014). The Robustness of HE scheme depends on the size of the encryption key. But the use of large size key makes the system too slow. The large size of public keys affects the size of ciphertext, encryption time, decryption time, and data processing time. The parameters that have to be considered while using Homomorphic Encryption schemes are a)The size of the encryption key b)The effect encryption key size on the ciphertext c)Time taken for encryption d)The decryption time and e)Size of the secret key . The existing works (Mbarek et al. 2016) (Sushila et al. 2016) concentrates on providing security to the data where the level of noise grows linearly with the multiplicative depth of the data being evaluated. To overcome this issue a technique called modulus switching is adopted. Let c be a valid encryption of m under s modulo q and that s is a short vector. Suppose also that c’ is a simple scaling of c. i.e c’≡c mod 2 means that c’ is a valid encryption of m under s modulo p using usual decryption equation. This method allows the change of inner modulus in the decryption equation. Here the correctness of decryption under the same secret key. This technique is called as modulus switching technique. Formally the method can be defined as For integer vector x and integers q>p>m , x’ is defined as x’←Scale(x,q,pr,r) and x’ is the R –vector closest to(p/q).x that satisfies x’=x mod r (Kanagavalli 2015).

2.6 Full-text search

As computing and computing power evolve, regardless of data storage or transmission in the network, it is always advised that personal of corporate data must be encrypted. Data encryption can ensure its security, confidentiality, and integrity, and avoid theft and modification of data by unauthorized users when it is being stored or being transferred. The Homomorphic encryption algorithm was proposed by Rivest et al. In 1978, researchers later called it the "Holy Grail" (Genty 2009) in the field of cryptography. The feature of the algorithm is to perform arithmetic operations on the encryption text without knowing the key. After decrypting the ciphered data that has the arithmetic operations done on, the result should be equivalent to the corresponding operation that will be on the in plain text, F(Enc (m)) = Enc (F(m)). In the year of 2009 Gentry introduced the Fully Homomorphic Encryption scheme (all-homomorphic encryption scheme based on ideal lattice over polynomial ring). In 2011, Coron (Coron et al. 2011) has made some modification and improvements on what has Gentry proposed scheme and introduced a chart that adding public keys to encryption text in product form can reduce the size of public keys, which is the size of the public key o (λ7). (Lijuan Wang , 2019)

Access control is controlling the authorized users' access to protected network resources, and preventing unauthorized users from modifying or reviewing sensitive data, and preventing illegal users from illegally accessing data through certain permissions, to achieve the security and safety of network resources.

Retrieval of ciphertext from the cloud can be done via various ways and methods, but in this study, we will be achieving it by is recovery based on the security index, which is the indexing of keywords in the ciphertext and the search for keywords if they are in the index. Let one depend on retrieving the ciphertext clearing, the method is to find the keyword and each word matches the ciphertext and confirms whether the keyword exists. Encrypting text retrieval is a continuous improvement from a simple recovery of a single keyword to search for multiple keywords, from low accuracy to high accuracy. (Lijuan Wang , 2019)

2.7 Substitution cipher

In cryptography, a substitution cipher is a method of encrypting by which units of plaintext are replaced with ciphertext, according to a fixed system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing the inverse substitution.

Substitution ciphers can be compared with transposition ciphers. In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.

There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a simple substitution cipher; a cipher that operates on larger groups of letters is termed polygraphic. A monoalphabetic cipher uses fixed substitution over the entire message, whereas a polyalphabetic cipher uses a number of substitutions at different positions in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice versa.

2.8 Scrambling Text with Codes and Ciphers

There are many different ways to “scramble” text or hide its meaning in such a way that only authorized persons (at least in theory) are able to read it. This scrambled (encrypted) text is called cipher text. A method for encrypting text is called a cipher or a code. Technically, a code uses substitution at the word or phrase level, whereas a cipher works at the level of individual letters or digits. The two words are often used interchangeably, but computerized cryptographic techniques generally rely on ciphers that operate on the binary form of the data by applying an algorithm (a mathematical calculation). Some common cipher/code types are:

- Substitution
- Transposition
- Obscure languages

Substitution Ciphers Simple substitution is a method often used by children in their first experiments with secret code. A substitution cipher merely substitutes different letters, numbers, or other characters for each character in the original text. The most straightforward example is a simplistic substitution in which each letter of the alphabet is represented by a numerical digit, starting with 1 for A. The message goodbye then becomes 7-15-15-4-2-25-5. This code is obviously extremely easy to break. The Caesar

Cipher used a simple shifting method, in which each letter of the message is represented by the letter two places to the right in the alphabet (A becomes C, B becomes D, and so on). Other substitution methods can be much more difficult to crack. For example, if two parties exchanging communications have an identical copy of a particular book, they might create a message by referencing page, line, and word numbers (for example, 73-12-6 tells you that the word in the message is the same as the sixth word in the twelfth line on page 72 of the code book). In this case, anyone who doesn't have a copy of the book (and to cite the correct pages, it must be the exact same edition and print run) will not be able to decipher the message. Some types of substitution ciphers are:

- Monoalphabetic substitution: Each letter is represented by another letter or character in a one-to-one relationship.
- Polyalphabetic substitution: Different cipher-text characters can represent the same plain-text letter, making it more difficult to decrypt messages using the frequency analysis technique. Renaissance architect and art theorist Leon Battista Alberti is credited with developing this technique, earning him recognition as the "father of Western cryptography."
- Polygraphic (block) cipher: Several letters (or digits when we're dealing with binary data) are encrypted at the same time, using a system that can handle all the possible combinations of a set number of characters.
- Fractionation: Multiple symbols are substituted for each plain-text letter, and then the letters or digits are transposed.

Chapter Three Methodology and Proposed work

3.1 Overview

This chapter presents the proposed work, which is developing a new Encryption and Decryption techniques of text documents based on the concept of Homomorphic Encryption (HE), to store the ciphertext documents on the cloud and to search these documents using ciphertext queries. Using this approach, neither the search queries nor the search results will be in plaintext format, hence an intruder who might intercept the query or its result will not comprehend what is being searched. The study will extend the Homomorphic approach into text search through algorithm design to be followed by experimental work. This chapter consists of the following sections: Section 3.2 introduces the chapter topic and gives the motivations behind building the new algorithms. Section 3.3 presents the objectives of the proposed model. Section 3.4 describes the proposed encryption model. Section 3.5 presents the encryption / decryption algorithms. Section 3.6 presents the search process. Section 3.7 gives a summary of this chapter.

3.2 Introduction

Homomorphic Encryption is an encryption scheme that allows users to perform arbitrary operations on ciphered data without the need to decrypt it, ensuring the same results when performing these operations on the same plaintexts. As discussed in Chapter Two, there are three types of Homomorphic Encryptions; Fully Homomorphic Encryption, Partial Homomorphic Encryption, and Somewhat Homomorphic Encryption. In this study, we will be using the Fully Homomorphic Encryption as it allows a large number of evaluations to be processed on the ciphertext and most of the studies in the field use the arbitrary methods.

3.3 Objectives of the Proposed Model

The proposed model is designed to fulfill the following objectives:

1. Encrypt words content of the plaintext document using a secret key.
2. Create an index of the words of the ciphertext document.
3. Encrypt a search query using the same secret key.
4. Search the ciphertext index using the ciphertext query to locate matching documents.
5. Decrypt the ciphertext document using the same secret key.

3.4 Description of the Encryption Model

The proposed encryption model is based on the assumption that the plain text documents are encrypted and the resulting ciphertext documents are uploaded to the cloud. The encryption process will create an index of the ciphertext words. Subsequently, an authorized user who has the encryption secret key can encrypt his query words, search the cipher words index to locate the ciphertext documents that match the query, and allow the user to download the found cipher text documents. The downloaded ciphertext document will be decrypted using the proposed decryption algorithm.

Using this approach, the cloud server personnel and anyone in the middle as hackers or unauthorized personnel will not be getting any useful information from the uploaded files, as the server will be hosting only the ciphertext, encrypted index, and the encrypted inquiry. Therefore, using this approach will guarantee the transparently querying to the cloud server.

3.5 The Encryption / Decryption Algorithms

Character Pair Halves Swapping is the principal method that will be used in the encryption algorithm, where the text document is read via the encryption algorithm and will produce the ciphertext. The proposed algorithm is built to go through each word of the text document and do character pairs halves swapping within each word, guided by random numbers. Each character (byte) within a word is split into 4-bit halves, Left Half-Byte (LHB), and Right Half-Byte (RHB). Swapping of half-bytes between every two consecutive characters is carried out, taking into account the swapping between characters does not generate special or control characters. In the case of words that contain an odd number of characters, the last character will be considered a space character. Also, Character Pair Halves Swapping algorithm will have to ensure that the order of the characters in their word is preserved when applying the swapping steps.

3.5.1 The LHB/UHB Swap Encryption Algorithm

In this algorithm, the encryption will be done for the words and numbers but when coming to special characters it will be left as-is so that the text search work on words without consideration for any special characters as most text search applications deal with special characters as having special functions. A word can contain letters, digits, or "-".

The lower half-bytes (LHB) or the upper half-bytes (UHB) are swapped depending on a random number of 0 or 1 that applies to a character pair within a word. For example, if we have a random sequence of 0 0 1 1 then the first and second pairs have LHB swapping while the third and fourth pairs have UHB swapping. If the word contains an odd number of characters, the LHB of the end character is replaced with a reversible special value.

If a pair contains the same character ("AA") swapping will not affect so instead the LHBs of the two characters are replaced with a reversible value as in the case of the odd end-character.

3.5.2 The LHB/UHB Swap Encryption Algorithm

The Algorithms Structure for the LHB/UHB Swap is constructed as follows.

```
1. Read a 16 decimal digit integer to be used as the encryption/decryption as a secret
key (K)
2. Call the Seed function using K
3. Call the Random function 30 times where the random values should be 0 or 1 and
store the random numbers in an array called R
4. Process the plain-text document as follows:
   While Not EOF (plain-text file)
   Read text line in LINE
   Repeat for words in LINE      /* Repeat 1
       Get next word and store in WORD (a word is terminated by, ; : space ! @ #,
       etc), it can contain letters, digits, -
       WL = Word length
       NP = Number of pairs in WORD (not including the last digit if WL is odd)
       PairPos = 0    /* Pair Sequence in the word

       Repeat            /* Repeat 2
              PairPos +=1
              Extract next char pair from WORD and store in C1 and C2
              If C1 = C2
                     Replace LHB of C1 and C2 with Hex F – LHB
              Else
              If R(PairPos) = 0
                     Swap LHB of C1 with lower HB of C2
              Else
                     Swap UHB of C1 with UHB of C2
```

```
            Return C1 C2 to WORD

        Until PairPos = NP   /* End of Repeat 2
            If WL is odd
                C-End = WORD(WL)
                Replace LHB of C-End with Hex F – LHB
                Return C-End to the last the end character in WORD.
            Return WORD to LINE

    Until end of line   /* Repeat 1
    Save LINE to encrypted file

    End of While
```

3.5.3 The LHB/UHB Swap Decryption Algorithm

This algorithm reverses the LHB-UHB swap steps starting with the same secret key.

1. Read the secret key integer that is used in encryption (K)

2. Call the Seed function using K

3. Call the Random function 30 times where the random values should be 0 or 1 and store the random numbers in an array called R

4. Process the cipher-text document as follows:

```
  While Not EOF (cipher-text file)

  Read text line in LINE

  Repeat for words in LINE      /* Repeat 1
```

Get next word and store in WORD (a word is terminated by , ; : space ! @ #, etc), it can contain letters, digits, -

```
      WL = Word length

      NP = Number of pairs in WORD (not including the last digit if WL is odd)
```

```
PairPos = 0    /* Pair Sequence in the word
Repeat              /* Repeat 2
        PairPos +=1
Extract next char pair from WORD and store in C1 and C2
If C1 = C2
        Replace LHB of C1 and C2 with Hex F – LHB
Else
If R(PairPos) = 0
        Swap LHB of C1 with lower HB of C2
Else
        Swap UHB of C1 with UHB of C2
Return C1 C2 to WORD
Until PairPos = NP   /* End of Repeat 2
If WL is odd
        C-End = WORD(WL)
        Replace LHB of C-End with Hex F – LHB
        Return C-End to the last the end character in WORD.
Return WORD to LINE
Until end of line   /* Repeat 1
Save LINE to decrypted-text file
End of While
```

3.5.4 Indexing the ciphertext words

To facilitate the search process of the ciphertext document, each cipher word generated by the encryption algorithm is added to an index file, with the elimination of duplicate words. At the end of the encryption process, an index file is written that will be used in the search process.

3.5.5 Creating a Key

An integer of a maximum of 16 decimal digits will be used as the secret key. To increase the robustness of the encryption model, it is possible to use a larger key size, multiple of 16 decimal digits, and to compress the large key into 16 digits using perfect hashing. The process of using the secret key to generate the random numbers for encrypting the words will involve choosing a random number of 1 or 0 for each character pair swapping, as shown in section 3.4.3. The same method will be used to encrypt the search query and to decrypt the ciphertext document which matches the search query.

3.6 Searching the Ciphertext Documents

Storing data in the cloud in an encrypted form is a very common procedure most companies do these days, as encrypting the stored data on the cloud will keep the sensitive data away from hackers or be accessed by unauthorized users. Searching encrypted data using an encrypted query is one solution to this issue, as no one can know the ciphered text or the ciphered query or the result of the search. Fully homomorphic encryption allows us to query on the encrypted data with an encrypted query.

To search the ciphertext document, the search words are encrypted using the LHB/UHB Swap Encryption algorithm. The generated search cipher words are used to search the index file in order to find the ciphertext document. After the search on the ciphertext, an index is done a message will be appearing to the user with the search result and where the query word is located. The result will contain the place of where the inquired word is located at (at which indexed document), then the user can download the ciphertext documents and decrypt them on his work station to generate the original plain text documents as they were before they were encrypted with one of the encryption algorithms.

The search will not involve multi-term information retrieval type of queries as this is outside the scope of this research.

3.7 Summary

In this chapter, a new way of encrypting data is proposed as a solution to the homomorphic encryption of documents that will be stored on the cloud. This is achieved by converting the words in each document to its 8 bits binary origin then a swapping will be applied to these characters ensuring that the resulted words don’t contain control or special characters. Then a search done on the index files that have been generated by the Encryption process and the result will be decrypted to a plain text format and given to the user.

Chapter Four

Implementation and Experimental Results

4.1 Overview

This chapter presents the implementation of the proposed model and the experimental results of the proposed algorithm of homomorphic encryption. This chapter organized as the following: section 4.2 presents a description of the implementation of the proposed algorithm as a working system; section 4.3 provides an introduction to the conducted experiments; section 4.4 presents the datasets which are used in the implementation, and section 4.5 discusses the parameter settings for the proposed Algorithms. Section 4.6 discusses the measurements that are used to evaluate the proposed Algorithm. Section 4.7 presents the results of the implementation and shows the performance of the proposed algorithms. Lastly, Section 4.8 provides a summary of the chapter.

4.2 Introduction

The proposed Algorithm which has been elaborated and discussed in Chapter Three is coded using the VC++ programming language. Implementation stages, to get and compare the results, are elaborated in this chapter. The performance of the proposed algorithms will be explained in the ensuing sections. Accuracy is measured by comparing the decrypted the ciphertexts files with the original plain text files.

4.3 Datasets

This section describes the properties and lists some statistics about the utilized datasets.

For the purpose of this research, the data were downloaded from Project Gutenberg (www.gutenberg.org), which provides free eBooks and they offer it on their website for all educational purposes. 10 samples were downloaded with various sizes that will be used in the process of encrypting and decrypting and searching the inquiry among them so that the encryption and decryption and search inquiry will be done on the samples. 1st sample is the “Pride and Prejudice” by Jane Austen with the size of 781 KB, the 2nd is The Works of Edgar Allan Poe with the size of 26 KB 3rd is Alice's Adventures in Wonderland with the size of 170 KB 4th is the Ion sized 55 KB, 5th is A Journal of the Plague Year size 588 6th is The Adventures of Sherlock Holmes size 594 KB, 7th is the Moby Dick; Or, The Whale sized 1.2MB 8th is The Yellow Wallpaper with the size of 50 KB 9th is Frankenstein; Or, The Modern Prometheus size of 440 KB and the last one is the Importance of Being Earnest: A Trivial Comedy for Serious People size 139 KB.

4.4 Implementation

The proposed Encryption, Decryption, and Search methods, which are used in this study and the purpose of generating ciphertexts to maintain the security and integrity of the data that will be stored on the cloud storage, are implemented using the Visual C++ programming language. The chart in Fig. 4.1 shows the implementation’s system diagram.

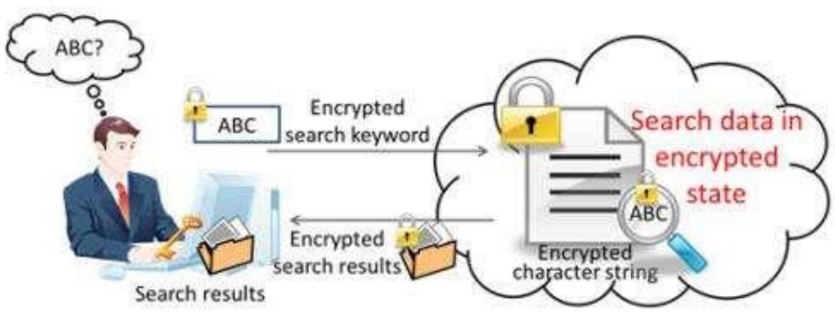


Figure 4.1: CloudCrypto System Diagram

4.4.1 Application modules

The proposed CloudCrypto solution consists of the following modules:

- **Encryption Module:** in this function and as discussed in Chapter Three, the encryption process of the plain text document of the cloud user is performed, to generate the ciphertext that will be stored in the cloud as the data of the organization that needs to be protected.

- **Indexing Module:** Indexing is done while encrypting the plain text, all encrypted documents generate an index file containing all the words in the encrypted document. Index file word entries are the result of the encryption module and they are stored in ciphertext format. The index file that is generated to provide for fast searching of inquiries using the Search function. Indexing function is embedded in the encryption module for faster indexing, rather than using a standalone indexing function. The indexing happens by using arrays for internal storage in the solutions, and when the encryption of the plain text finishes the array is sorted and duplicate words are removed. The index is generated as an array, and then it is written to a text file.

- **Search Module:** this function is used to search the index files for the query word submitted by the user which is in ciphertext format. The index file will be downloaded to the user's workstation and searched locally, and when a matched document is found, it will be downloaded in ciphertext format for decryption on the user's workstation.

- **Decryption Module:** in this module the ciphertext will be returned into its original plain text when the user searches for a document, using a ciphertext query, download the ciphertext document from the cloud storage, and performs the decryption process using the user's workstation. In this way, the stored document and the query will be in ciphertext mode, which prevents outsiders and unauthorized data access to the stored document and the query.

4.5 Experimental results

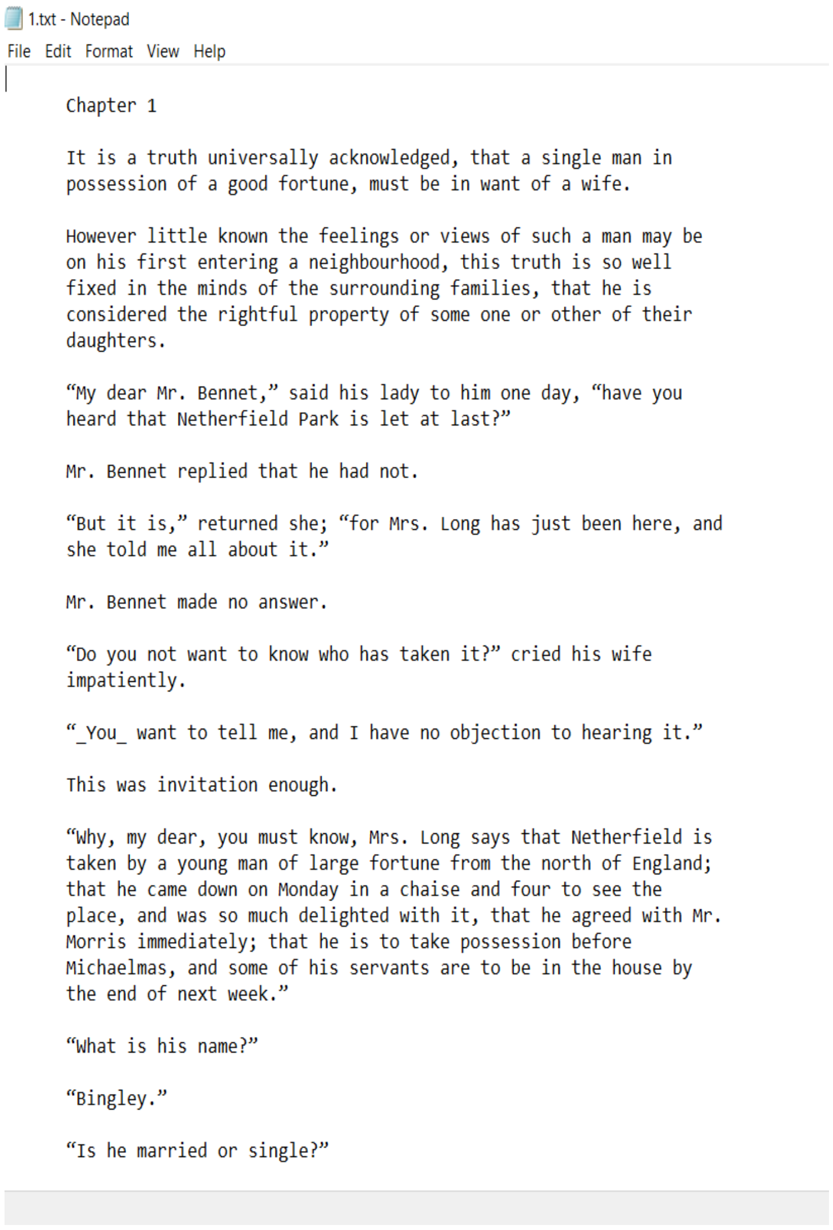
The elapsed time in milliseconds taken by the Encryption, Decryption, and Inquiry Search functions to process the 10 selected datasets collected from the Gutenberg project (www.gutenberg.org) are shown in Table 4.1.

Dataset Name	Dataset Size	Functions and Processing Time		
		Encryption Time	Decryption Time	Indexing Time
DataSet 1	781 KB	163,094 MS	147,634 MS	36,221 MS
DataSet 2	26 KB	446,63 MS	15,427 MS	38,641 MS
DataSet 3	170 KB	62,272 MS	39,447 MS	23,246 MS
DataSet 4	55 KB	42,020 MS	26,858 MS	20,487 MS
DataSet 5	588 KB	116,069 MS	87,067 MS	19,553 MS
Data Set 6	594 KB	98,270 MS	80,102 MS	21,466 MS
DataSet 7	1.2 MB	191,919 MS	103,678 MS	31,051 MS
DataSet 8	50 KB	41,209 MS	16,991 MS	25,801 MS
DataSet 9	440 KB	69,467 MS	42,287 MS	31,779 MS
DataSet 10	139 KB	58,358 MS	61,839 MS	30,218 MS

Table 4.1 Experimental Result

From the previous table, we can notice that as much as the file is getting larger the time for encryption and Indexing is getting bigger. The time the datasets took for being encrypted is measured by the difference between the function starting and ending, resulting the time that is stated in the table. The Indexing time is the time that the solution took to place the ciphers words of the document in an array then checking the duplicated words not to repeat the entries of the index file and then storing the words into the index file, each time a new file is encrypted a new index containing all encrypted words is created for that file.

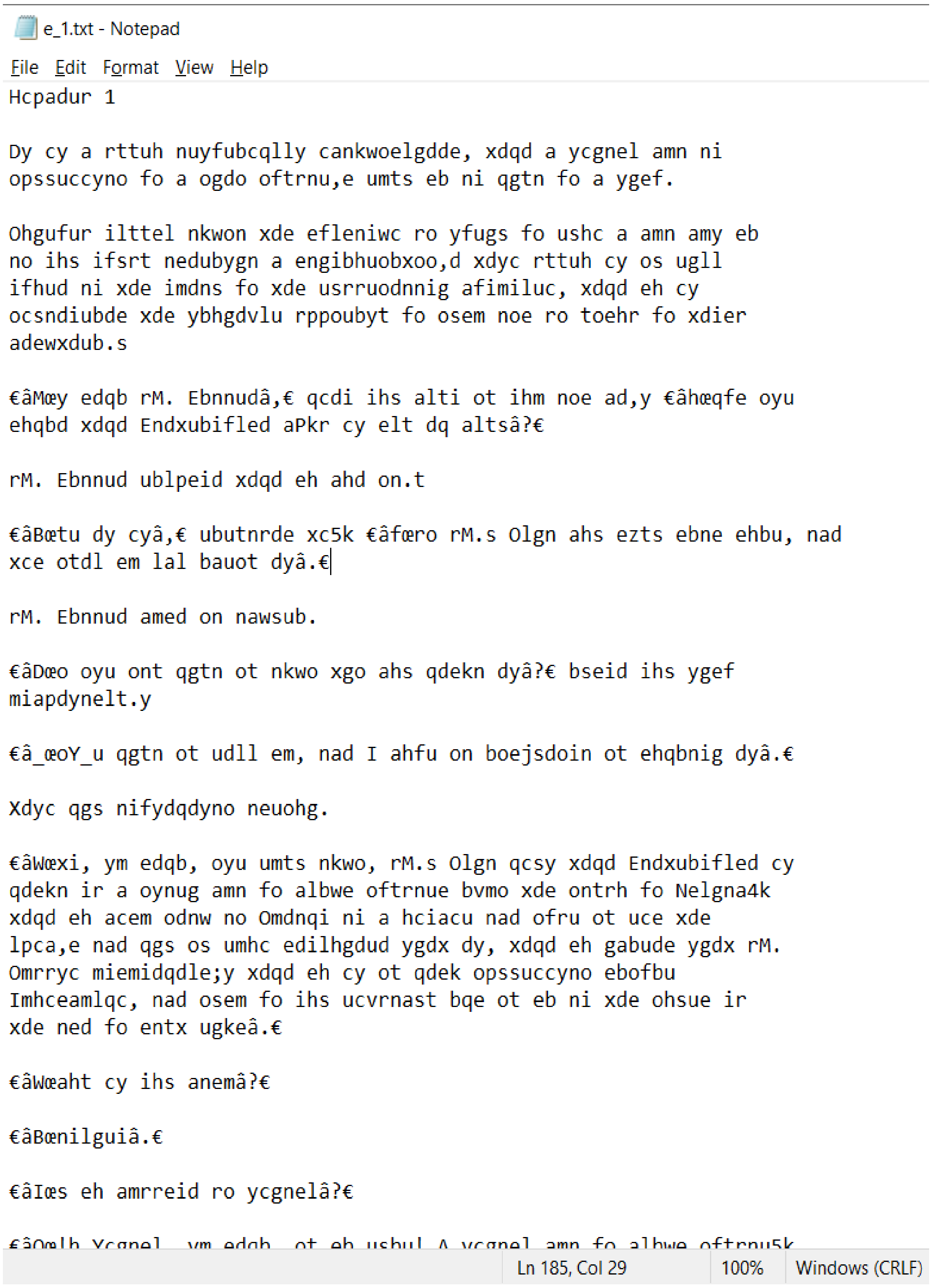
The following image is a sample of plain text from DataSet 1 downloaded from Project Gutenberg named Pride and Prejudice



```
Chapter 1

It is a truth universally acknowledged, that a single man in
possession of a good fortune, must be in want of a wife.

However little known the feelings or views of such a man may be
on his first entering a neighbourhood, this truth is so well
fixed in the minds of the surrounding families, that he is
considered the rightful property of some one or other of their
daughters.

“My dear Mr. Bennet,” said his lady to him one day, “have you
heard that Netherfield Park is let at last?”

Mr. Bennet replied that he had not.

“But it is,” returned she; “for Mrs. Long has just been here, and
she told me all about it.”

Mr. Bennet made no answer.

“Do you not want to know who has taken it?” cried his wife
impatiently.

“_You_ want to tell me, and I have no objection to hearing it.”

This was invitation enough.

“Why, my dear, you must know, Mrs. Long says that Netherfield is
taken by a young man of large fortune from the north of England;
that he came down on Monday in a chaise and four to see the
place, and was so much delighted with it, that he agreed with Mr.
Morris immediately; that he is to take possession before
Michaelmas, and some of his servants are to be in the house by
the end of next week.”

“What is his name?”

“Bingley.”

“Is he married or single?”
```

Figure 4.2: Plaintext

After decrypting the above plain text with the proposed solution we will be getting a new file that has all the words encrypted as in the following image



```
Hcpadur 1

Dy cy a rttuh nuyfubcqlly cankwoelgdde, xdqd a ycgnel amn ni
opssuccyno fo a ogdo oftrnu,e umts eb ni qgtn fo a ygef.

Ohgufur ilttel nkwon xde efleniwc ro yfugs fo ushc a amn amy eb
no ihs ifsrt nedubygn a engibhuobxoo,d xdyc rttuh cy os ugll
ifhud ni xde imdns fo xde usrruodnnig afimiluc, xdqd eh cy
ocsndiubde xde ybhgdvlu rppoubyt fo osem noe ro toehr fo xdier
adewxdub.s

€âMœy edqb rM. Ebnnudâ,€ qcdi ihs alti ot ihm noe ad,y €âhœqfe oyu
ehqbd xdqd Endxubifled aPkr cy elt dq altsâ?€

rM. Ebnnud ublpeid xdqd eh ahd on.t

€âBœtu dy cyâ,€ ubutnrde xc5k €âfœro rM.s Olgn ahs ezts ebne ehbu, nad
xce otdl em lal bauot dyâ.€

rM. Ebnnud amed on nawsub.

€âDœo oyu ont qgtn ot nkwo xgo ahs qdekn dyâ?€ bseid ihs ygef
miapdynelt.y

€â_œoY_u qgtn ot udll em, nad I ahfu on boejsdoin ot ehqbnig dyâ.€

Xdyc qgs nifydqdyno neuohg.

€âWœxi, ym edqb, oyu umts nkwo, rM.s Olgn qcsy xdqd Endxubifled cy
qdekn ir a oynug amn fo albwe oftrnue bvmo xde ontrh fo Nelgna4k
xdqd eh acem odnw no Omdnqi ni a hciacu nad ofru ot uce xde
lpca,e nad qgs os umhc edilhgdud ygdx dy, xdqd eh gabude ygdx rM.
Omrryc miemidqdle;y xdqd eh cy ot qdek opssuccyno ebofbu
Imhceamlqc, nad osem fo ihs ucvrnast bqe ot eb ni xde ohsue ir
xde ned fo entx ugkeâ.€

€âWœaht cy ihs anemâ?€

€âBœnilguiâ.€

€âIœs eh amrreid ro ycgnelâ?€

€âOœlh Ycgnel, ym edqb, ot eh ushul A ycgnel amn fo albwe oftrnu5k
```

Figure 4.3: Ciphered text Diagram

As noticed in the above image all the words have been encrypted generating a new file that contains a ciphertext that will be used by the organization to be uploaded to the clouds in order to preserve the security of the data.

4.5.1 Evaluation of the Encryption function

In order to evaluate the proposed solution, the ratio of the encryption to the decryption is calculated, to check the efficiency of the proposed solution for the Encryption and placed in Table 4.2 for the selected datasets.

Dataset Name	Dataset Size	Functions and Processing Time		
		Encryption Time	Decryption Time	Ratio of Encryption / Decryption Times
DataSet 1	781 KB	163,094 MS	147,634 MS	1.104
DataSet 2	26 KB	20,772 MS	208,867 MS	2.895
DataSet 3	170 KB	62,272 MS	39,447 MS	1.578
DataSet 4	55 KB	42,020 MS	26,858 MS	0.156
DataSet 5	588 KB	116,069 MS	87,067 MS	1.333
DataSet 6	594 KB	98,270 MS	80,102 MS	1.226
DataSet 7	1.2 MB	191,919 MS	103,678 MS	1.851
DataSet 8	50 KB	41,209 MS	16,991 MS	1.770
DataSet 9	440 KB	69,467 MS	42,287 MS	1.642
DataSet 10	139 KB	58,358 MS	61,839 MS	0.943
Average	404.3	115,221.9	62,400	1.7707

Table 4.2 Evaluation Of Encrpytion Function

From the above table, we can see that the ratio of encryption to decryption is reasonable for the file sizes that have been used in the study.

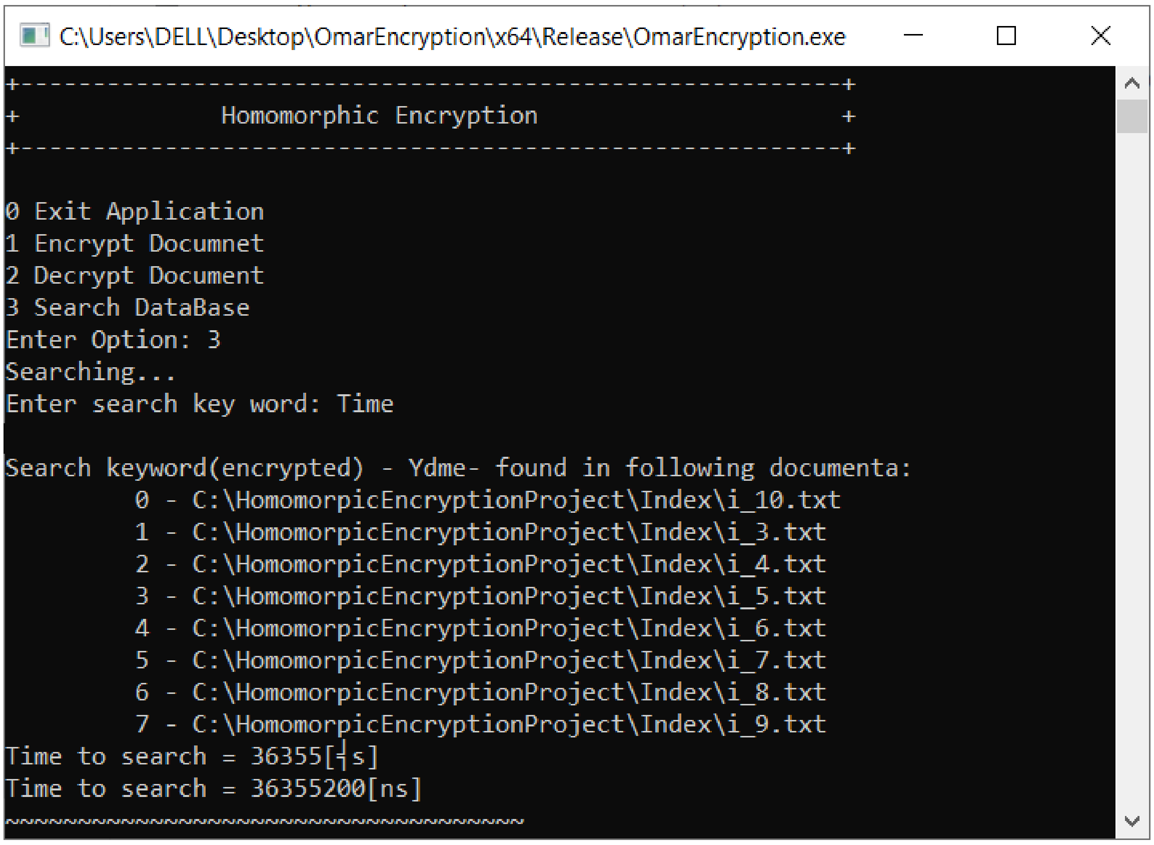
Table 4.3 shows the time to process a Kilo-Byte (KB) of data for each of the datasets, in Encryption and Decryption, for the 10 datasets. The average encryption time per KB for the 10 datasets is 288.991 MS. These results can be used to estimate the expected times of Encryption and Decryption for other sizes of data, for example, the estimated encryption time for a 10 MB text document will be around 48 minutes on a standard i5 laptop which was used in this experiment, and much less on a more powerful workstation.

Dataset Name	Dataset Size	Functions and Processing Time			
		Encryption Time (MS)	Encryption / KB (MS)	Decryption Time (MS)	Decryption/KB (MS)
DataSet 1	781 KB	163,094 MS	208.827	147,634 MS	189.032
DataSet 2	26 KB	44,663 MS	1,717.80	15,427 MS	593.346
DataSet 3	170 KB	62,272 MS	366.305	39,447 MS	232.041
DataSet 4	55 KB	42,020 MS	764	26,858 MS	488.327
DataSet 5	588 KB	116,069 MS	197.396	87,067 MS	148.073
DataSet 6	594 KB	98,270 MS	165.437	80,102 MS	134.851
DataSet 7	1.2 MB	191,919 MS	159.932	103,678 MS	86.398
DataSet 8	50 KB	41,209 MS	824.18	16,991 MS	339.82
DataSet 9	440 KB	69,467 MS	157.879	42,287 MS	96.106
DataSet 10	139 KB	58,358 MS	419.841	61,839 MS	444.884
Total	4043 KB	1,152,219 MS	284.991	621,330	153.680

Table 4.3: Time per KB

4.6 Search results

When searching for a string the application will encrypt the search string with the same encryption method that is used to encrypt the text files previously, then it will search among the index files that contain all the words pool within them, then it will return all the index files that contain the results of the search inquiry that has been done by the inquirer, as shown in the following image



```
C:\Users\DELL\Desktop\OmarEncryption\x64\Release\OmarEncryption.exe
+-----------------------------------------------------------+
+              Homomorphic Encryption                       +
+-----------------------------------------------------------+

0 Exit Application
1 Encrypt Documnet
2 Decrypt Document
3 Search DataBase
Enter Option: 3
Searching...
Enter search key word: Time

Search keyword(encrypted) - Ydme- found in following documenta:
         0 - C:\HomomorpicEncryptionProject\Index\i_10.txt
         1 - C:\HomomorpicEncryptionProject\Index\i_3.txt
         2 - C:\HomomorpicEncryptionProject\Index\i_4.txt
         3 - C:\HomomorpicEncryptionProject\Index\i_5.txt
         4 - C:\HomomorpicEncryptionProject\Index\i_6.txt
         5 - C:\HomomorpicEncryptionProject\Index\i_7.txt
         6 - C:\HomomorpicEncryptionProject\Index\i_8.txt
         7 - C:\HomomorpicEncryptionProject\Index\i_9.txt
Time to search = 36355[┤s]
Time to search = 36355200[ns]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 4.4: Search Result

As shown in figure 4.4 above the search inquiry was the word “Time”, and the application has encrypted it to the word “Ydme” using the encryption method, then it returned all the index files that contain the same word “YDME”, so that if there was a security breach and there was any information leakage or hacking the hackers or the sniffers will see the word “Ydme” that has no sense or meaning for them and will have a piece of incomplete information, but for the cloud users when the results come with the search results they will have the complete information about the place of where they can get the file that have the searched word inquiry. After the search is completed and the result is displayed to the user, the application will prompt an option that will enable the user to download the resulted documents after decrypting them.

4.7 Summary

This chapter presented the implementation of the proposed homomorphic encryption using visual C++ programming Language over several datasets to evaluate the encryption and decryption along with the search methods. The testing results show that whenever the text files are getting bigger in size the time that takes for encryption and decryption gets slightly bigger also.

The search takes time to get the results with respect to the number of the index files, the much the index files the more time that needed to search through the text files.

Encryption process summary step by step

Encryption time calculation starts Then it follows these actions

1. Then Open encryption key file, read the key, close the file, create and then the random array

2. Open text document for reading, then create a blank encrypted(output) document for writing ciphertext on it.

3. Read 1st line from the original document

4. Divide the line into words and loop through each word, extract character pairs from each word, and encrypt each word. Here if character pair has a special character or if a word on encryption generates an encrypted word containing special characters, then some other function has to be called for managing these situations. In effect encrypting, two characters have different execution loads depending on the characters in each pair. so all character pairs are not encrypted at the same speed.

5 Store each word in an array in encrypted form.

6. After encrypting all words in 1st line, write that line back to the output text file.

7. Repeat steps 2 to 5 until the end of the file.

8. Close the both original and output file

9. Time to complete steps 1 to 8 is calculated, this is the encryption time.

10. Sort the array contain words in the file.

11. Delete duplicate words from the array.

12 Write the index array to a file.

13 Calculate time between steps 10 to 12 to get the indexing time

Steps 1, 2, and 8 take the same time irrespective of the document size. So encryption time has two components, a fixed time component and variable time component depending on file size.

While the ratio is higher for smaller files and smaller for larger files. So the ratio tends to reach a constant value as file size increases.

Chapter Five

Conclusion and Future Work

5.1 Conclusion

In this thesis, a new encryption method for text Homomorphic Encryption is proposed, which is aimed to preserve the users of the cloud data integrity and elements the unauthorized usage of the data from unauthorized personals either from the same organization or the cloud providers, also the hackers who might have access to the cloud and teal the data or might get access to the data through any means like the connection to the cloud servers. Most of the previous studies and works that have been all have included arbitrary and especially addition or multiplication deduction on the ciphertext that is stored on the cloud storage or at the local PCs of the user. Also introducing the search method for words without the need to pull that from the clouds server and search on it and then upload it back to the clouds again which is very time consuming and cost consuming also. The contributions of this thesis are as follows:

1. Introducing a new method of Encrypting data that ensures data integrity and secrecy.
2. The proposed Encryption methods use a secret key of 64 bits and unique methods that have been used for this issue.
3. Indexing the ciphertexts into index files for making the search easier.
4. Information retrieval method for the data without compromising the data for unauthorized data access.

5.2 Future Work

Based on the present research on Homomorphic Encryption for text docum9fhxbxcents, the following ideas are suggested for future work:

- The proposed Homomprhic Encryption model for text documents can be extended to deal with multi-word and multi-term queries.
- The proposed model can be implemented within a standard textual database system.

- Investigating the enhancement of time efficiency of the proposed model for multi gigabytes corpora
- Investigate the integration of the proposed model with the traditional homomorphic encryption of arbitrary functions.

References :

Tharam Dillon & Chen Wu and Elizabeth Chang (2010), Cloud Computing: Issues and Challenges, IEEE International Conference on Advanced Information Networking and Applications DOI: 10.1109/AINA.2010.187

Junjie Peng ; Xuejun Zhang ; Zhou Lei ; Bofeng Zhang ; Wu Zhang ; Qing Li (2009) Comparison of Several Cloud Computing Platforms DOI: 10.1109/ISISE.2009.94

Durgesh Kumar Mishra, Nilanjan Dey, Bharat Singh Deora, Amit Joshi (2019), ICT for Competitive Strategies

Ihsan Jabbar, Saad Najim (2016) Using Fully Homomorphic Encryption to Secure Cloud Computing

Moore, C., O'Neill, M., O'Sullivan, E., Doröz, Y., & Sunar, B. (2014). Practical homomorphic encryption: A survey IEEE International Symposium on (pp. 2792-2795). IEEE Computer Society. DOI: 2014.6865753

Lijuan Wang , Lina Ge , Bo Geng , Qiuyue Wang 2019, Encryption Cipher Text Retrieval Scheme Based on Fully Homomorphic Encryption Enterprise Cloud Storage.

Ren Xunyi, a , Yan Shiyang (2016) Keyword-based Ciphertext Search Algorithm under Cloud Storage.

Chen Zhi-gang, Wang Jian, Chen Liqun and Song Xin Review of How to Construct a Fully Homomorphic.

Encryption Scheme (2014)

Migrating to the cloud: Oracle client/server modernization

Yi, X., Paulet, R., & Bertino, E. (2014). Fully Homomorphic Encryption. SpringerBriefs in Computer Science.

A. Akavia, D. Feldman, and H. Shaul. 2018. Secure database queries in the cloud: Homomorphic encryption meets corsets, Adi Akavia, Dan Feldman, and Hayim Shaul 2019, Secure Search via Multi-Ring Fully Homomorphic EncryptionCraig Gentry Shai Halevi 2009, Implementing Gentry's Fully-Homomorphic Encryption Scheme

Craig Gentry 2009 A FULLY HOMOMORPHIC ENCRYPTION SCHEME.

RALUCA ADA POPA (2014) BUILDING PRACTICAL SYSTEMS THAT COMPUTE ON ENCRYPTED DATA

Shundong LI , Sufang ZHOU , Jiawei DOU2* & Wenli WANG1 (2019) Polynomial AND homomorphic cryptosystem and applications

S. Terada, H. Nakano, S. Okumura, and A. Miyaji, 2018 On the security of Ring-LWE with homomorphic encryption.

M. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers," Proc. 29th Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '10), pp. 24-43, 2010.

S. Even, O. Goldreich, and A. Lempel, "A Randomized Protocol for Signing Contracts," Comm. ACM, vol. 28, no. 6, pp. 637-647, 1985.

C. Gentry and Z. Ramzan, "Single Database Private Information Retrieval with Constant Communication Rate," Proc. 32nd Int'l Colloquium on Automata, Languages and Programming (ICALP '05), pp. 803-815, 2005.

C. Gentry, "Computing Arbitrary Functions of Encrypted Data," Comm. ACM, vol. 53, no. 3, pp. 97-105, 2010.

Xun Yi, Mohammed Golam Kaosar, Russell Paulet, and Elisa Bertino, 2013 Single-Database Private Information Retrieval from Fully Homomorphic Encryption

Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M. Shorter public keys Fully homomorphic encryption over the integers with shorter public keys.

R.Kanagavalli and Dr.Vagdevi S,"A Survey of Homomorphic Encryption Schemes in Cloud Data Storage",International Journal of Recent Development in Engineering and Technology,Vol.3,Issue 1,2014,pp.71-75. www.gutenberg.org https://homomorphicencryption.org/

Payal V.Parmar,et.al ,"Survey of Various Homomorphic Encryption algorithms and Schemes",Interational Journal of Computer Applications(0975-8887), Vol.91,No.8, April 2014,pp.26-32.

K.Mallaiah,S.Ramachandram,"Applicability of Homomorphic Encryption and CryptDB in Social and Business Applications :Securing Data Stored on the Third Party Servers while Processing through Applications ",International Journal of Computer Applications (0975-8887),Vol.100,No.1,2014.

Zvika Brakerski,Craig Gentry and Vinod Vaikunthanathan,"(Leveled) Fully Homomomorphic Encryption without Bootstrapping",ACM transactions on Computation Theory,2014.

Rachana Jain, Sushila Madan,Bindu Garg,"Homomorphic Framework to Ensure Data Security in Cloud Environment",ICICCS 2016, pp.177-181. D:O:I 978-1-5090-2084- 3/16

Mbarek Marwan,Ali Kartit and Hassan Ouahmane,"Towards a Secure Cloud Database using Paillier's Homomorphic Cryptosystem" , IEEE Proceedings ,2016

R.Kanagavalli and Dr.Vagdevi S,"A Mixed Homomorphic Encryption Scheme for Secure Data Storage in Cloud", IEEE Intemational Advanced Computing Conference IACC2015,2015,D.O.I:10.1109/IADCC.2015.7154867.

Debra Littlejohn Shinder Michael Cross. "Scene of the Cybercrime" 2nd Edition Paperback ISBN: 9781597492768 eBook ISBN: 9780080486994