EFFICIENCY ENHANCEMENTS FOR PRACTICAL TECHNIQUES FOR SEARCHES ON ENCRYPTED DATA

Pitigala Arachchillage Pansilu Madhura Bhashana Pitigalaarachchi
(179342K)

Degree of Master of Science

Department of Computer Science and Engineering
University of Moratuwa
Sri Lanka

February 2020

EFFICIENCY ENHANCEMENTS FOR PRACTICAL TECHNIQUES FOR SEARCHES ON ENCRYPTED DATA

Pitigala Arachchillage Pansilu Madhura Bhashana Pitigalaarachchi
(179342K)

Thesis submitted in partial fulfilment of the requirements for the degree
Master of Science in Computer Science

Department of Computer Science and Engineering
University of Moratuwa
Sri Lanka

February 2020

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another per-son except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works

....

Pansilu Pitigalaarachchi Date

I certify that the declaration above by the candidate is true to the best of my knowledge and he has carried out research for the Masters thesis under my supervision.

....

Dr. Chandana D. Gamage Date

ACKNOWLEDGMENTS

I would like to offer my sincere gratitude for my supervisor, Dr Chandana Gamage for his valuable inputs and support towards managing this research along with my personal and professional commitments. Without his continuous help and feedback this thesis would not have been a success. Also I would like to thank him for his guidance and directions in progressing with this research as well as the entire master's programme.

My sincere thanks goes to Dr. Shantha Fernando and the rest of the staff of the department of Computer Science and Engineering for the knowledge and support given to me in this master's programme.

I would like to express my sincere appreciation for my wife, parents and family for their continuous support for my work, studies and specially this research. Finally I like to thank all my friends and colleagues who have supported me in this endeavor.

ABSTRACT

Information security has become one of the major focus areas for any organization. More often, organizations see the need of outsourcing their data storages in meeting the operational and security objectives. This gives rise to a new problem of privacy protection of the data stored with a third party. As a solution the data is encrypted before storing with a third party data service provider. Thus when the users need to process the data, the safer option is to download the data into a secure user machine and perform the operations on the decrypted data. This creates an additional overhead of having to download a large amount of data and decrypt them even to perform a simple calculation on the data stored in the encrypted form. Therefore the possibility of secure data processing at the remote third party storage has become an interesting problem to solve. In order to preserve the privacy the data cannot be allowed to be decrypted at the third party storage. One form of the solution is to facilitate computations on the data stored in encrypted form. The users can make requests from the data service provider and if the service provider can perform operations on the encrypted data itself and provide the answer the above mentioned overhead can be avoided. This brings the focus of this research on to the studying of computing on encrypted data with specific focus on searchable encryption. As pert of the research, the current literature of computing on encrypted data is studied to identify a suitable searchable encryption scheme for practical use. Followed by the literature study, an existing symmetric searchable encryption scheme is selected for a detailed study. Here a complete implementation of the scheme is proposed and the test results are analyzed. Based on the results, a keyword extraction mechanism is proposed to improve the performance of the scheme. Finally significant performance improvements, 89.83% reduction in extra space usage due to searchable encryption and 92.11% improvement in single key word search time has been achieved. In addition to that, use cases in capital markets are studied to understand the possibilities of practical use and challenges.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Computing on Encrypted data

With the growing trend of providing many of the IT services through cloud based computing technology, it is necessary to re-look at the traditional model of providing information security. This need arises as storing of user data and user computations will now take place in remote servers which are not under the full control of owners of data. Therefore information security and secure processing of data are major requirements in current context. Computing on encrypted data is a computational challenge. Based on the recent research it is evident that this is a possibility and there is much to explore in this topic. This research is an effort to add more clarity for the area of computing on encrypted data.

With the growing needs of information management, many organizations face the challenge of effectively storing their data in a confidential manner. Some large organizations maintain their own internal data storages and their data is managed internally and are not exposed to external parties. But due to global expansions in organizations, high initial capital requirements of in-house storages, globally distributed user base and high maintenance expenditures, many organizations outsource their data storages in to commercial data service providers. With the recent boost in cloud computing, cloud storages have an increasing demand. Being able to access the data in the cloud from anywhere, data processing platforms and services provided by cloud service providers and low expenditure have made the cloud storages more popular.

When the data is stored with a third party, data is stored in encrypted form. Even though the confidentiality is ensured by proper encryption, it does not solve all underlying problems associated with external data storages such as cloud. One such problem is secure data extraction from the cloud. When the data is encrypted and stored at a cloud service provider, the users have a need of extracting only the required data for processing. The straightforward option is requesting the full data set from the storage and downloading in to a secure local machine where the data can be decrypted. Then the user is able to query and filter out the data under interest. The two major problems associated with this is having to download a large amount of data and having to decrypt them prior to the start of data processing. When a data requester requests for a specific set of data, the data service provider faces the challenge of performing search operations on encrypted data to facilitate the specific query from the user. Therefore performing search operations on data in encrypted form is one of the biggest challenges faced by the industry. The second requirement is to securely process the extracted data at the cloud it self to facilitate data processing requirements of users.

1.2 Research Objectives

In this context, one major challenge of computing on encrypted data is selecting suitable cryptographic protocols for encryption. These encryption algorithms must be selected in a way that efficient search operations and secure computations on encrypted data can be facilitated. Therefore there is a need for secure schemes that facilitate search operations on the data in encrypted form and computing on search results. Also it is important to evaluate possible implementations to identify the suitability of such schemes for practical use. Therefore this research aims to explore the domain of computing on encrypted data to achieve the following research objectives.

- Understanding the current topics of searching and processing on encrypted data.
- Studying and proposing an efficient implementation of a selected secure computing scheme.

- Studying the practical usability and proposing suitable efficiency enhancements.
- Identifying and studying practical industrial applications of the selected scheme.

Chapter 2

Encryption, Searchable and Homomorphic Encryption

Under this section, the related work and the background information pertaining to computing on encrypted data are discussed in detail.

2.1 Data Encryption, Symmetric and Asymmetric key Encryption

2.1.1 Data Encryption

Confidentiality is one of the key security objectives in any application scenario. When data is stored with a commercial data service provider, the confidentiality is ensured by encryption. In encryption the sensitive data is encoded in a way that only the authorized parties can access the data if they possess the secret key. With the advancements of modern encryption over the past century two main branches, symmetric key and asymmetric key encryption have been evolved to be used in existing commercial applications. Currently, based on the application various cryptographic protocols are being employed in commercial environments.

Symmetric key encryption is one of the major branches of cryptography. This is a class of operations where a single key is used for both encryption and decryption of data. i.e. The plain text is encrypted using a secret key and the same is used to decrypt the cipher text back in to plain text. In order to maintain the secrecy the secret

key must be kept a secret by parties involved with encryption and decryption. The other challenge is to generate a secret key and share it among the sender and receiver of the encrypted communication. With the ever growing security requirements in secret communication, new variants of symmetric key encryption have been proposed by scientists to suit the needs of the industry. Eg: The replacement of DES by AES-Advanced Encryption Scheme [1].

Asymmetric Key Encryption differs from the symmetric key encryption as there are two keys involved with the operation. This operation requires a public and a private key pair. Public key is known by everyone and the private key is a secret only known by the message receiving end. The sender has to encrypt the plain text with the public key and receiver can decrypt the cipher text using the secret key. This scheme does not require any pre agreement between the sending and receiving parties. This facilitates many to one secret communications as the public key can be used by anyone to encrypt a message and sent to the receiver where only this particular receiver is aware of the secret key in the public-private key pair.

2.1.2 Symmetric key Encryption

Symmetric key encryption has been widely used and it is being studied in different operating modes. The symmetric key encryption uses the same key for both encryption and decryption. As per the literature [2] the process is represented as follows.

$$Encryption : c = E(k, m) \tag{2.1}$$

$$Decryption : m = D(k, c) \tag{2.2}$$

where ‘m’ and ‘c’ refers to plain text and cipher text respectively.
The Encryption and Decryption algorithms E and D are publicly known and the secret key‘k’ needs to be available. The major challenge in symmetric key cryptography is to agree upon a common secret key by the two parties who need to have a secret communication. To achieve key exchange, various methods of public key cryptography

have been studied. More recently novel methods such as quantum key distribution [3] has also been studied.

2.1.3 Asymmetric Key Encryption

In 1976 Diffie and Hellman [4] proposed the concepts of asymmetric key cryptography commonly known as Public key cryptography [5]. This scheme proposed an unconventional method for key distribution, one of the fundamentals in modern cryptography which is now commonly known as Diffie- Hellman Key Exchange. Later on some significant contributions have been made to the field of public key cryptography. In 1978 Rivest et al. [6] proposed a public key scheme, which is widely known as RSA algorithm and in 1985 El Gamal scheme [7] was proposed based on the discrete logarithm problem, which is another form of the Diffie-Hellman problem.

2.1.4 Security Requirements

The popular problem in encryption is that one party has to secretly communicate with another party without letting an adversary extract significant information about the secret message. Encryption has been proposed as a solution for this fundamental problem. Therefore the security of encryption is widely discussed [8]–[11]. The security of a scheme depends on the algorithm as well as the usage of the scheme. Security of a symmetric key encryption is affected by how the secret key is generated, the length of the secret key, lifetime of a secret key and key distribution etc. When it comes to asymmetric key encryption, the security varies based on the size of the parameters in the scheme, key generation and management of public and private keys etc. Additionally the security behind the mathematical assumptions and various steps in the cryptographic algorithms are analyzed in detail under the literature on security requirements.

2.2 Background of Searchable Encryption(SE)

Searchable encryption [12]–[15] allows the users to encrypt secret data and store in a remote cloud server while allowing them to perform search operations on encrypted data. In order to facilitate this, special variants of encryption have to be carried out before the data is stored. Currently searchable encryption is being evolved under two main branches [16] known as SSE-Symmetric Searchable Encryption and PKES-Public Key Encryption With Keyword Search. In parallel to these, various forms of secure indexing [17] methods are also being developed to perform searches on encrypted data.

Searchable Symmetric Encryption [16], [18] is a form of Symmetric Key encryption which allows a user to encrypt data using a secret key while providing trapdoors to perform search operations on symmetrically encrypted data. Therefore searchable symmetric key encryption requires new variants of symmetric key encryption. The trapdoors are being used by the data service provider to search on the encrypted data when a data query is received with a specific keyword from a data requester. Handing variable size keywords and time complexity of the search operations are among the major challenges in this scheme.

Public key encryption facilitates anyone having the public key to encrypt the secret data using the public key. This scheme [19], [20] facilitates the server to use a special key provided by the data requester to perform a search operation on the encrypted data. However the server is unable to do so without the special key provided by the data requesters. The fundamental requirement of this special key provided by the data requester is that it must not be the secret key used for the public key encryption, but is derived by the data requester considering the keyword and the scheme specifics.

As widely discussed by Boneh et al [21] in 2004, the searchable encryption has also been studied under two main storage settings, private and public databases.

Private databases: A single user who is constrained by storage capacity may wish

to encrypt the secret data and store in a remote server. When required the secret key is used to derive retrieval parameters to be sent to the remote server. Work on this branch is discussed under searchable symmetric encryption.

Public databases: A remote public database holds data and multiple users may wish to search and retrieve some data without revealing what has been retrieved to the database. This is widely studied under private information retrieval (PIR).

As per the literature PIR has been first introduced in the published work of Chor et al [22]. Two main objectives of PIR are to facilitate the remote access of information stored in organized databases and facilitating private database access. In private access the users are able to retrieve information from the remote server without revealing the identity of the items being queried. One major problem of such a scheme is that the users need to know the physical address of the item being retrieved. One of the solutions is having a database which provides information from the applicable physical address corresponding to a keyword provided by the user. Similar studies [23] have been carried out to enhance the PIR afterwards. As analyzed in literature [24], [25], generally PIR demands strong information theoretic security bounds which makes it harder to propose practical PIR schemes. That means PIR facilitates searching on encrypted data with increased security and increased security comes with far less efficiency.

However if the data is not public, but provided by multiple sources a different mechanism is required to facilitate querying by the private user. Public key encryption based systems are proposed to facilitate private key based searches on public key encrypted data provided by many users.

Prior to the studies on private information retrieval, secure multi party computation based approaches have been evaluated [26], [27] to achieve secret computation objectives. The major objective of secure multiparty computation is to derive the value of a

public function with the use of private data held by several untrusted parties. As stated in literature [24] the major problem of multiparty computation is that it requires a high overhead.

Secure Indexes is another form of searchable encryption. Indexes are generally used to order and organize lists. In the case of encrypted documents, secure indexes can be used to order the documents in a searchable manner. In year 2004, Goh has formally defined the the properties of a secure index as follows.

1. Secure index is a data structure.
2. It allows a person who queries with a 'Trapdoor' for a word 'x' to test if the index contains 'x'.
3. The index reveals no information about its contents without valid trapdoors.
4. Trapdoors can only be generated with a secret key.
5. It provides semantic security against an Adaptive Chosen Keyword Attack.
   i.e. If a document 'D'" contains 'n' words, Suppose an adversary already knows m words and needs to gain information on the rest of the set S:'n-m' words, Even if the adversary has the access to the other index document pairs, and can adaptively obtain trapdoors for words except those in S, He can never deduce any information about any word in the set S from D's index.

2.3 Searchable Encription

2.3.1 SSE - Symmetric Searchable Encryption

Bennet et al [21] has been extensively studied censorship resistant publishing systems which has different approaches of secure content sharing. In this work they have studied some practical systems and models in detail. One such systems uses the hash of the content as the key. Another uses a distributed file sharing system which first breaks the original file into eight blocks. For the reconstruction any four of the eight pieces

is sufficient. In this scheme each blocks are hashed to generate a unique identity tag which can be used to retrieve the blocks. They have also studied more generalized approaches like splitting a file into n shares and using any k out of n shares to reconstruct the entire document.

In year 2000 Song et al [24] has presented their studies on practical techniques for searches on encrypted data. The general setting studied here is a scenario where a bandwidth constrained user storing documents on an untrusted server. They claim to have proposed an efficient secret key based method. In this method, if a user needs to retrieve all the documents containing a particular keyword the user has to provide a piece of information derived based on the keyword to the server.

Later Golle et al [26] has proposed an enhancement for this method by proposing a secure conjunctive keyword search over encrypted data. They claim their method to be provably secure and having a moderate storage cost.

2.3.2 System models for Symmetric Searchable Encryption

Introduction to the scheme

Over the time various models have been proposed and discussed in literature. The Symmetric Searchable Encryption model proposed by Song, Wagner and Perrig [24] is as follows. This discussion covers basic definitions applicable for the above scheme, a basic model and an enhanced model proposed by addressing some security issues in the basic scheme.

Definitions

Pseudorandom Generator *G*

The pseudorandom generator uses a stream cipher and generates a sequence of pseudorandom bit streams $S_1, S_2, ..., S_l$ based on a secret seed value. Each S_i is *n-m* bits long.

Pseudorandom Function *F*

The pseudorandom function takes a key k_i as an input and, maps a *n-m* bit long bit

string in to a *m* bit string.

Pseudorandom Permutation

The pseudorandom permutation is a block cipher operating in the Electronic Code Book(ECB) mode. It computes the cipher text X_i as

$$X_i = E_{k''}(W_i) \tag{2.3}$$

where k'' and W_i are the secret key and plain text respectively.

The Basic Scheme

Documents and Keywords

Any document *D* is considered to have a sequence of words $W_1, W_2, ..., W_l$ and the maximum word length is considered as *n* bits.

Building Blocks

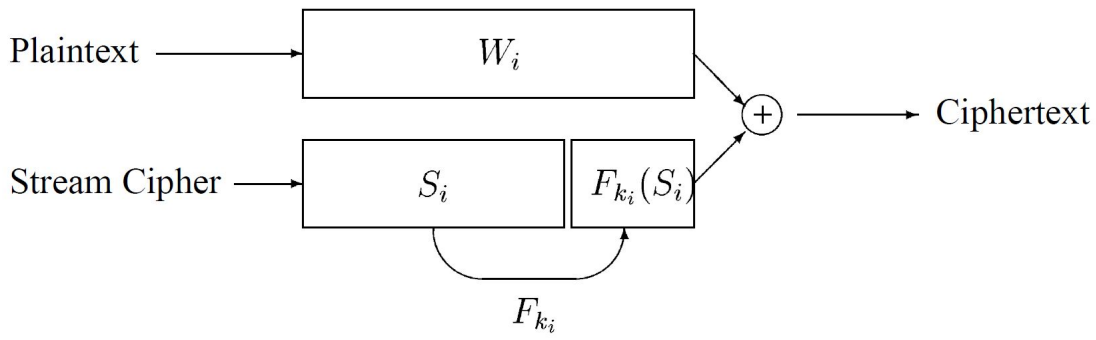

Figure 2-1: Basic protocol proposed by Song, Wagner and Perrig [24]

Encryption

Step1: A sequence of pseudorandom streams $S_1, S_2, ..., S_l$ are geenerated. Each pseudorandom value is *n-m* bits long.

Step2: In order to encrypt the word W_i in position *i*, T_i is calculated as

$$T_i = \langle S_i, F_{k_i}(S_i) \rangle \tag{2.4}$$

Each T_i value is *n* bits long.

Step3: Cipher text C_i is calculated as

$$C_i = W_i \oplus T_i \tag{2.5}$$

The value *k* is secret and only known by the party performing the encryption. It is possible to select k_i as $k_i = k$ for all *i* or select a different *k* for each word in the sequence. Therefore the pseudorandom stream $T_1.T_2, ..., T_i..., T_l$ can only be generated by the encrypting party. In order to decrypt the document the secret k_i has to be known. It is also correct to say that the scheme is secure if the Pseudorandom Function *F* and the Pseudorandom Generator *G* are secure.

Search Operation

If the encrypting party, Data Requester wants to search for the word *W*, following steps to be executed.

Step1: Data Requester has to provide the word W and k_i corresponding to to each location *i* where the word *W* may occur.

Step2: Search party has to compute the value $C_i \oplus W$.

Step3: Verify if $C_i \oplus W$ is in the form $\langle s, F_{k_i}(s) \rangle$ where *s* is the first *n-m* bits of $C_i \oplus W$.

Issues to be addressed

In this basic scheme the Data Requester has to reveal all k_i when requesting for a search over the entire document. If not the Data Requester needs to have knowledge on the exact locations where the word *W* may appear. To address above issues, a novel scheme with new properties has been proposed by Song, Wagner and Perrig [24].

An Enhanced Scheme

Building Blocks

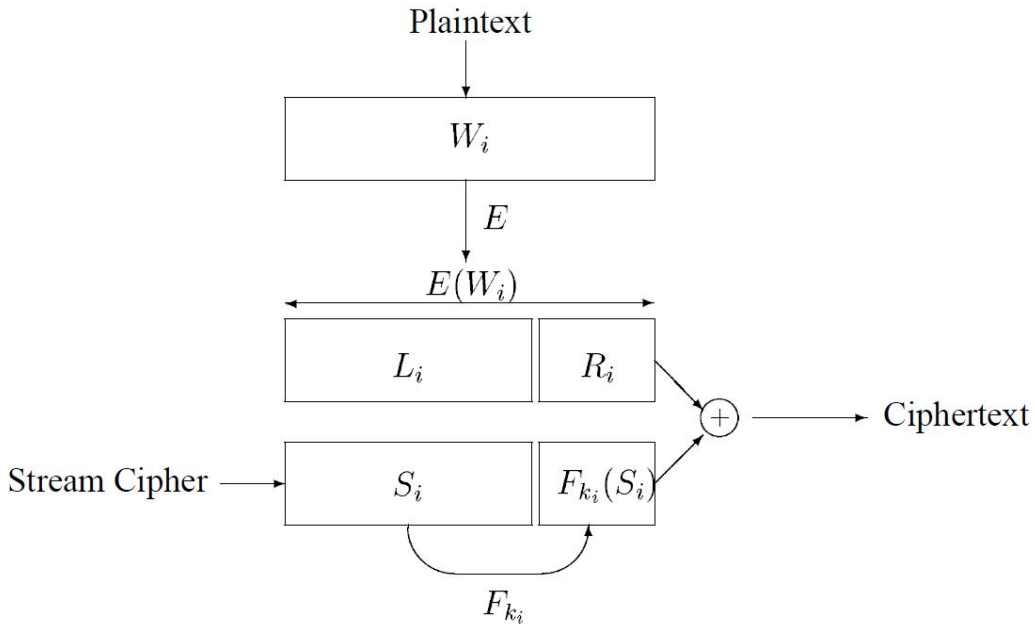

Figure 2-2: Final protocol proposed by Song, Wagner and Perrig [24]

Encryption

Step1: Pre-encrypt each word *W*. Let the pre-encrypted output be

$$X_i = E_{k''}(W_i). \tag{2.6}$$

After the pre-encryption the initial words $W_1, W_2, ..., W_l$ are converted in to a new sequence of pre-encrypted words $X_1, X_2, ..., X_l$.

Step2: Split the pre-encrypted word in to two parts as

$$X_i = \langle L_i, R_i \rangle \tag{2.7}$$

where L_i denotes the first *n-m* bits and R_i denoted the last *m* bits of X_i.

Step3: Compute k_i as

$$k_i = f_{k'}(L_i) \tag{2.8}$$

Step4: Compute T_i value as

$$T_i = \langle S_i, F_{k_i}(S_i) \rangle \tag{2.9}$$

Step5: Ciphertext C_i is calculated as

$$C_i = X_i \oplus T_i \tag{2.10}$$

Search Operation

If the encrypting party, Data Requester wants to search for the word *W*, following steps to be executed.

Step1: Data Requester has to pre-encrypt the word *W* using the symmetric encryption algorithm

$$X = E_{k''}(W) \tag{2.11}$$

Step2: Compute

$$k = f_{k'}(L) \tag{2.12}$$

Step3: Send $\langle X, k \rangle$ to the data service provider.

Step4: DSP computes the value of $C_i \oplus X$

Step5: Verify if $C_i \oplus X$ is in the form $\langle s, F_k(s) \rangle$ where *s* is the first *n-m* bits of $C_i \oplus X$.

Controlled searching

To address the problem of having to reveal k_i applicable for each position of words, the authors have proposed a k_i value derived based on the word being encrypted. Thus the data requester does not need to know the corresponding k_i applicable for every position of the word being searched may appear.

Hidden Searches

The pre encryption of plaintext will support for hidden queries. Since the data requester does not have to reveal the actual plaintext word being searched, the data service provider is not able to learn additional information on the data being searched.

Also the symmetric pre encryption operates in the ECB mode. Therefore the cipher text created by pre-encryption will not depend on the position i in the document where the word is found, It will only be dependent on the plaintext word W.

2.3.3 PEKS - Public Key encryption with keyword search

As per literature the first method on PEKS has been proposed by Boneh et al [19]. They have extensively discussed possible use cases of a public key encryption based keyword searching system.

Scenario1:

A sender encrypts an email with the receiver's public key. An email gateway wants to check if the email contains a specific keyword(Eg: the word "Urgent") and route the email accordingly. The method proposed by Boneh et al enables the sender to provide a key to the gateway which enables the gateway to test whether the specific word is a keyword in the email.

Scenario2:

An email server or a fileserver stores various files publicly encrypted by various users. Boneh et al also suggests a method for the secret key holder to query for files carrying a specific keyword by sending the server a key to be used for the search operation.

2.3.4 System models for Public key Encryption with Keyword Search

Introduction to the scheme

The symmetric key encryption model proposed by Boneh, Crescenzo, OStrovsky and Persiano is as follows. They have studied models to perform searches over public key encrypted data.

Definitions

KeyGen(s)

The algrithm takes a securiy parameter s and generates a public and private key pair

A_{pub}, A_{priv}

PEKS(A_{pub}, W)
The algorithm takes the public key and a keyword to produce a searchable encryption of *W*.

Trapdoor(A_{priv}, W)
The trapdoor algorithm takes the private key and a keyword to generate a trapdoor T_W for the search operation.

Text(A_{pub}, S, T_W)
Test algorithm tests if *W* = *W'* for a searchable encryption S and a trapdoor T_w where S and T_w are defined as follows.

$$S = PEKS(A_{pub}, W') \tag{2.13}$$

$$T_W = Trapdoor(A_{priv}, W) \tag{2.14}$$

Scheme and Operation

Encryption
The scheme is proposed to perform searches over public key encrypted messages. The key gen algorithm generates the public and private key pair. The party encrypting the message uses the public key to encrypt the message and the keywords accordingly. When *E* denotes the encryption, a typical message will be in the form of
$[E_{A_{pub}}[msg], PEKS(A_{pub}, W_1), ..., PEKS(A_{pub}, W_k)]$

Search

The party requesting for the search generates a trapdoor using the private key and that is used by the searching party to scan the messages for specific keywords. The data requester has to generate separate trapdoors for each keyword being requested to

be searched.

2.3.5 Applications

Applications of SSS

One of the major applications proposed for the Searchable Symmetric Encryption is to be used by commercial data service providers. This allows data service providers to provide services for different individual users. Each storage scarce user is able to encrypt their files using their own key and outsource the storage for a secure data service provider. The search operation can be instructed by the user who possesses the symmetric key. Therefore this type of an encryption only facilitates search operation for the party holding the secret key.

Applications of PEKS

One of the major applications of a publicly encrypted scheme is that the data may originate from any party holding the public key. Therefore this presents promising applications on data directed towards a single entity, but originated by different sources. One such use is an email server. The recipient may receive emails encrypted by the public key and the server may entertain searches based on the trapdoors derived based on the secret key. It can also be applied for any gateway processing encrypted messages originated by multiple sources. An email gateway may sort messages coming towards a specific individual based on specific keywords. The intended recipient holding the secret key may provide the gateway with trapdoors to perform searches without compromising the security.

2.4 Homomorphic Encryption

2.4.1 Preliminaries

Homomorphic encryption is a specific form of encryption which allows operations to be carried on the cipher text. If such a system is employed the two main problems discussed above can be solved. i.e. data processing can be carried out at the remote storage itself.

The homomorphic Encryption can be formally represented as follows. Let m_a, m_b be plain text messages and c_a, c_b be the corresponding cipher texts. Suppose some encryption scheme(Enc) encrypts the plain text messages as follows.

$$Enc\langle m_a, m_b\rangle \Rightarrow \langle c_a, c_b\rangle \tag{2.15}$$

The scheme is said to be homomorphic over some operation $\otimes$ if there exist an operation $\odot$ s.t. ,

$$Enc[\, m_a \otimes m_b] \, = \langle c_a \odot c_b\rangle \tag{2.16}$$

When homomorphic encryption is employed the data service provider can perform operations on the cipher text without decrypting the data. This preserves the confidentiality of data. Also this approach does not require a large amount of data to be decrypted after downloading in to a secure client machine. Once the data service provider sends the processed output in the encrypted from, the user can simply decrypt the output to obtain the expected result.

The elegance of this scheme is that the decrypted answer matches the results when the same operation is performed on plain text. Rivest and Adleman laid the conceptual foundations [28] for privacy homomorphisms in 1978 which drew a significant attention towards the concepts of homomorphic encryption. Well known asymmetric key cryptographic algorithms such as RSA and ElGamal exhibits homomorphic properties for certain mathematical operations. The Fully-Homomorphic encryption theoretically allows any function to be evaluated on encrypted data. For years this was an open

problem until in 2009 Gentry's scheme [29], [30] laid the theoretical foundations for a novel approach on fully homomorphic encryption which allows the computations of arbitrary functions on encrypted data. However this method is not practically achievable due to the rapid increase of cipher text size and computation time with the security level. Later there have been contributions [31]–[34] for the domain of fully homomorphic encryption, some based on Gentry's scheme but these attempts have not brought the fully homomorphic encryption in to a truly practical level.

2.4.2 Homomorphic Re-Encryption

Even though the homomorphic encryption is capable of supporting computations on encrypted data it is a single user system, which has been identified as one of the major limitations of homomorphic encryption. i.e. only the holder of the secret key can access the results of computations on encrypted data. This causes a lot of practical problems when the data is collected and stored in the encrypted form without targeting a specific data user. Recently Ding et al. [35] proposed a novel scheme which can extend the single user system in to a multi user system. Suppose there is a data service provider (DSP) who stores the encrypted data and an access control server (ACS) facilitating data retrieval for data requesters (DR). Data is encrypted with a public key applicable for both the DSP and ACS and stored with the DSP. Once a data request is received, DSP processes the encrypted data in a way only the ACS can recover the result. At ACS the result is processed again to deliver the intended output for authorized data requesters. This way the ACS and DSP collaborates to re-encrypt the encrypted data stored at DSP to facilitate distinct data requesters (multiple users).

2.4.3 Public key schemes for Homomorphic Re-Encryption

The security of a given public key scheme has been a contributing factor for the popularity of the public crypto systems proposed after the Diffie-Hellman scheme. As discussed in the literature [36], it has been identified that the asymmetric key cryptography has employed two major classes of trapdoor techniques. The first is based on the RSA problem and the second is based on Diffie-Hellman problem. As pointed out by

Paillier [36] in 1999, a third class of a trapdoor technique has been evolved based on the high degree residuosity classes and then Paillier proposed a new trapdoor mechanism based on composite residuosity classes. Later Cramer and Shoup [37] proposed an enhanced public key system based on the decision composite residuosity assumption of Paillier's cryptosystem and another based on the quadratic residuosity assumption. Bressen et al. [38] proposed an improved version, inspired by the schemes proposed by Paillier and Cramer & Shoup.

Bressen et al. discusses an interesting scenario where the head of a group may want to be able to read any message sent to the members of the group. In order to solve this problem they propose a novel public-key crypto scheme, which allows for a double decryption mechanism based either on the factorization of the modulus, or on the knowledge of a discrete logarithm. In the second mechanism the knowledge of a discrete logarithm helps to decrypt cipher texts which have been encrypted with a specific key only. In contrast, the factorization of the modulus helps to decrypt any cipher text whatever the key is. In the upcoming sections we discuss the public key crypto scheme proposed by Bressen et al. in detail.

2.4.4 Homomorphic properties

When it comes to the homomorphic properties, various public crypto schemes exhibit hmomorphism over different mathematical operations. Among many discussed in literature, apart from RSA, ElGamal and Paillier, Goldwasser and Micali [39] and Naccache and Stern [40] can also be identified as some of the well-known schemes having homomorphic properties.

Let m_1 and m_2 be plain text messages.

RSA and ElGamal Schemes

These schemes are homomorphic over multiplication. The product of two cipher texts provides the encrypted answer for the product of two plain text messages.

i.e.

$$Enc[\, m_1.m_2] \; = Enc[\, m_1] \, .Enc[\, m_2] \tag{2.17}$$

Goldwasser and Micali scheme

This scheme is additively homomorphic over binary numbers.
i.e. for each $m_i \in \{0,1\}$

$$Enc[\, m_1 + m_2] \; = Enc[\, m_1] \, .Enc[\, m_2] \tag{2.18}$$

Paillier Cryptosystem

This schemes is an additively homomorphic scheme.
Let m be some plain text and k be an integer,Then

$$Enc[\, m_1 + m_2] \; = Enc[\, m_1] \, .Enc[\, m_2] \tag{2.19}$$

Also this scheme has some additional homomorphic properties such as homomorphism over scalar multiplication.
For a positive integer n,

$$(Enc[\, m] \,)^k (mod \; n^2) = Enc[\, km] \, (mod \; n) \tag{2.20}$$

2.4.5 System models for Homomorphic Encryption

Data Service Providers and Access Control Servers

In this discussion we would like to focus on two of the main stakeholders belonging to this research topic. Data Service Provider (DSP) is an entity capable of storing encrypted data and performing operations on the encrypted data. Any user can encrypt its own data and store them at the DSP. Conceptually an Access Control Server (ACS) controls the access to the data processing results performed by DSP. Once a Data Requester (DR) makes a processing request, the processing result is not directly provided to the DR by a DSP. Instead the requests are provided to DR through the ACS. This allows the DSP to take control of the data processing while ACS handles the access control for the requests made by multiple eligible data requesters.

Introduction to the scheme

A typical system may consist of a single Data Service Provider (DSP) and a single Access Control Server (ACS). The system proposed by Ding, Yang and Deng [35] supports a single DSP and multiple ACSs. The system modes proposed by them is as follows.

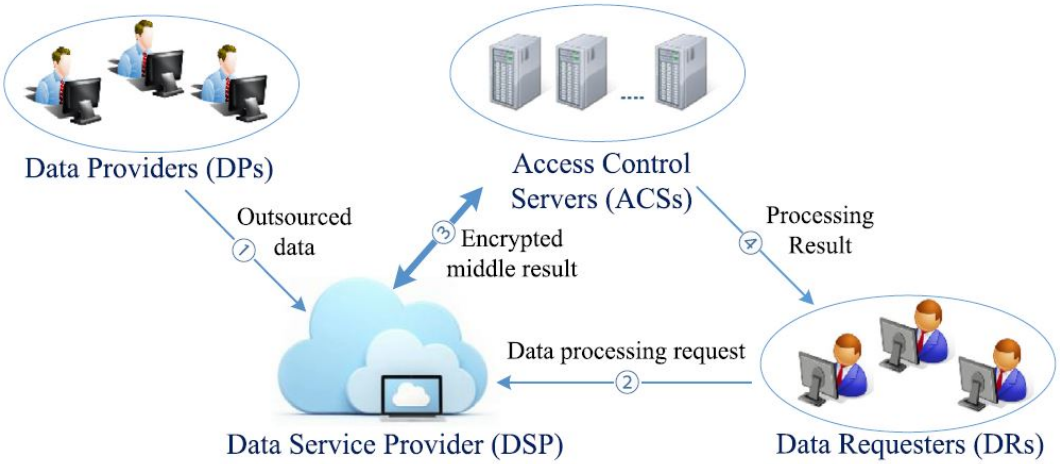

Figure 2-3: System model proposed by Ding, Yang and Deng [35]

The nature and responsibilities of each of the stakeholders has been formally defined as follows.

1. Data Service Provider (DSP) - Served by cloud service provider, which stores user data and provides some computation service.

2. Access Control Server (ACS) - In-charge of secure data computation with data access control for its users. There could be multiple ACSs that are operated by different organizations. Here a user can freely choose an ACS it trusts for service consumption.

3. Data Requesters (DR) - Consumes the data stored with the DSP. Since the data stored are in encrypted form, the data processing results obtained through ACSs are also in the encrypted form.

4. Data Providers (DP) - Mainly collects data and encrypts data and stores them with the DSP.

Protocol

The protocol used by this system is what has been proposed by Bresson et al. Given two large primes p and q, let

$$n = p * q \tag{2.21}$$

When *G* is the cyclic group of quadratic residues modulo n^2. Let h and g be two elements of maximal order in *G*. If h is computed as g^x mod n^2 where $x \in n_R$ [1,$\lambda(n^2)$] , then x is coprime with ord(G) with high probability, and thus h is of maximal order. Here λ(*) is the Euler function.

Key Generation

The public parameters are n, g and h is derived by randomly choosing a secret value x $\in$ [1, ord(*G*)] as follows.

$$h = g^x \; mod \; n^2 \tag{2.22}$$

Encryption

Given a message m $\in Z_n$, random number r is chosen in Z^*_n. The encryption is done

as follows.

$$(T, T') = [h^r(1 + mn), g^r] \ (mod\ n^2) \tag{2.23}$$

Decryption

When the secret, x is known m can be obtained as follows.

$$m = L(T/(T')^x \ mod\ n^2) \tag{2.24}$$

$$L(u) = (u - 1)/n \tag{2.25}$$

Working Examples of the existing concepts

Key Generation for small primes

Let p, q, p', q' be primes which satisfy the following properties.

$$p = 2p' + 1 \tag{2.26}$$

$$q = 2q' + 1 \tag{2.27}$$

i.e. p and q are safe primes. Let $p' = 2, q' = 3$ and $p = 5,\ q = 7$. Hence $n = p * q = 35$ and n^2 = 1225. Select the generator g as 891. The DSP and ACS respectively generate key pairs.

$sk_{DSP} = a$, pk_{DSP} = g^a

Let sk_{DSP} = 5, Hence pk_{DSP} = 891^5 mod 1225 = 151

sk_{ACS} = b, pk_{ACS} = g^b

Let sk_{ACS} = 19, Hence pk_{ACS} = 891^19 mod 1225 = 611

Then the Diffie-Hellman key

$$PK = (pk_{DSP})^{sk_{ACS}} = (pk_{ACS})^{sk_{DSP}} = g^{ab}\ mod\ n^2 \tag{2.28}$$

Therefore PK = 611^5 mod 1225 = 1026

Public system parameters: g, n, PK = 891, 35, 1026

Encryption with the public key of DSP and ACS

In order to facilitate a multi user system the plain text is encrypted using the public key of the two servers, PK. First a random number r $\in$ [1, n/4] is selected and the plain text m_i is encrypted as follows. For the exse of explanation, the cipher text of m_i is denoted as $[m_i]$.

And,

$$[m_i] = [m_i]_{PK} = (T_i, T_i') = [(1 + m_i * n) * PK^r, g^r] \ (mod\ n^2) \tag{2.29}$$

Let r = 7, m_i = 22 and Then,

T_i = (1+22*35)*1026^7 mod 1225 = 746

T_i' = 891^7 mod 1225 = 1206

$[m_i]_{PK}$ = [746, 1206]. The m_i encrypted with PK can only be decrypted under the cooperation of DSP and ACS.

Decryption Option 1 : Two level decryption with a fully trusted ACS

Partial Decryption at DSP : with SK^{DSP}

Once the cipher text is received by DSP, it is transferred in to another cipher text that can be decrypted by ACS as follows.

$$[mi]_{pk_{ACS}} = (T_i^{(1)}, T_i^{'(1)}) = (T_i, (T_i')^{sk_{DSP}}) \tag{2.30}$$

Therefore $[mi]_{pk_{ACS}}$ = [746, 851]

Partial Decryption at ACS : with SK^{ACS}

$T_i^{'(2)}$ = $(T_i^{'(1)})^{sk_{ACS}}$ = 851,

$$[m_i] = L(T_i^{(1)}/T_i^{'(2)}\ mod\ n^2) \tag{2.31}$$

$$L(u) = (u-1)/n \tag{2.32}$$

Therefore $(T_i'^{(2)})^{-1}$ (mod 1225) = 226. u = 746*226 mod 1225 = 771. Hence m_i = (771-1) /35 = 22.

Decryption Option 2 : Somewhat Re-Encryption for a single user

Different from the scheme above this is a new method which aims to transfer the encrypted data to the cipher text under the public key of an authorized requester. Let the private and public key pair of an authorized data requester (DR_j) be as follows,

$$(sk_j, pk_j) = (k_j, g^{k_j} \ mod \ n^2) \tag{2.33}$$

Based on the above, Let (sk_j, pk_j) = [10,751].

First Phase of Re-Encryption by DSP This step ensures that ACS can not decrypt the cipher text to obtain the plain text, message.

Step1: Select a public computation identifier CID = 117. Compute h_1 as,

$$h_1 = H((pk_j)sk^{DSP}||CID) \tag{2.34}$$

Then h_1 = 125 (mod 1225) (Using CRC32)

Step2: Cipher text is computed as,

$$[\hat{T}, \hat{T}'] = [T_i, (T_i'^{sk_{DSP}}) * g^{h_1}] \tag{2.35}$$

$[\hat{T}, \hat{T}']$ = [746, 51].

Second Phase of Re-Encryption by ACS

Step1: h_2 is computed as

$$h_2 = H((pk_j)^{sk_{ACS}}||CID) \tag{2.36}$$

h_2 = 767 (mod 1225) (Using CRC32)

Step2: Cipher text is computed as

$$[\bar{T}, \bar{T}'] = [\hat{T}, (\hat{T}')^{sk_{ACS}} * g^{h_2}] \tag{2.37}$$

[$\bar{T}$, $\bar{T}'$] = [746, 281].

Decryption on Re-Encrypted Data by DR

Step1: Based on the CID DR_j computes the hash values as follows.

$$h_1^{'} = H((pk^{DSP})sk_j||CID) \tag{2.38}$$

$h_1{}^{'}$ = 125, Hence equal to h_1

$$h_2{}' = H((pk^{ACS})sk_j||CID) \tag{2.39}$$

$h_2{}'$ = 767, Hence equal to h_2

Step2: Compute plain text as,

$$m_i = L(\bar{T} * pk_{ACS}^{h_1^{'}} * g^{h_2^{'}}/\bar{T}'\ mod\ n^2) \tag{2.40}$$

$(\bar{T}')^{-1}(mod\ n^2)$ = 946. $m_i = L$(1051*946 mod 1225) = (771-1) /35 = 22.

Working examples on Homomorphic Properties

As explained in the above sections, first the key generation algorithm needs to be run to negotiate respective Diffie- Hellman keys. In this section we assume the presence of a single ACS and a DSP. Following examples have been prepared based on the key generation explained above.

1:Addition

This scheme is additively homomorphic. Therefore the product of two cipher text will

provide the encrypted result of the product of two plain text messages.

$$[\sum_{i=1}^{N} m_i] = \prod_{i=1}^{N} [m_i] \tag{2.41}$$

However there is one condition to be satisfied. i.e. in order to obtain the sum of N pieces of data, the condition m_i < n/N needs to be satisfied by the plain text messages. Let m_1 = 4 and m_2 = 5 , be two plain text messages. Then, $[m_1]_{PK}$ = [1166, 1206] and $[m_2]_{PK}$ = [326,1206].

Enc[m_1 + m_2] = $[m_1]_{PK}$* $[m_2]_{PK}$ = ([1166, 1206])*([326, 1206]) = [366,361].

Therefore the answer in the form of cipher text = [366,361].

Partial Decryption at DSP : [366,$361^{SK_{DSP}}$] = [366,226]

Partial Decryption at ACS : [226^{SKDSP}]$^{-1}$ (mod1225) = 851.

Hence [m_1 + m_2] = (366*851 -1)/35(mod1225) = 9 = [4 + 5].

2:Subtraction

To obtain the subtraction result, first the [-m_i] needs to be computed at DSP as

$$[-m_i] = [m_i]^{n-1} \tag{2.42}$$

The subsequent operation is similar to addition.

Let m_1 = 9 and m_2 = 5 , be two plain text messages. Then, $[m_1]_{PK}$ = [641, 1206] and $[m_2]_{PK}$ = [326,1206]. And [-m_2] = $(326)^{35-1}$ (mod 1225) = 151. Therefore the answer in the form of cipher text = [16,361].

Partial Decryption at DSP : [16,$361^{SK_{DSP}}$] = [16,226].

Partial Decryption at ACS : [$226^{SK_{DSP}}$]$^{-1}$ (mod1225) = 851. Hence [m_1 + m_2] = (16*851 -1)/35(mod1225) = 4 = [9 - 5].

Chapter 3

Implementation and Results

3.1 Introduction

Considering the literature, the topic of symetric searchable encryption has been selected as the main topic of study. Therefore the symmetric searchable encryption scheme proposed by Song, Wagner and Perrig [24] was selected for implementation and performance study to identify the suitability for practical applications. This section covers the implementation specifics of the symmetric searchable encryption scheme discussed in section 2.3.2. It covers implementation specifics of the scheme, system model for a practical implementation, test results and a detailed analysis of the results.

3.2 System model

This section elaborates different building blocks and attributes of the proposed implementation.

Word Size

The word size determines the size of an encrypted keyword. For the implementation depicted in figure 3-1, a fixed word length of 128 Bits has been used and padding has been used when the length of the plaintext word W_i is less than 16 Bytes. To facilitate decryption, the last Byte of the padded word has been used to contain the length of the padding. Suppose a plaintext word "conspiracy" needs to be encrypted and the word is

padded with ‘0’ to bring the plaintext word length to the word size of the scheme. The padded word is represented as ‘conspiracy000005’. For simplicity, it is assumed that the scheme is used with plaintext words with length < 16 bytes.

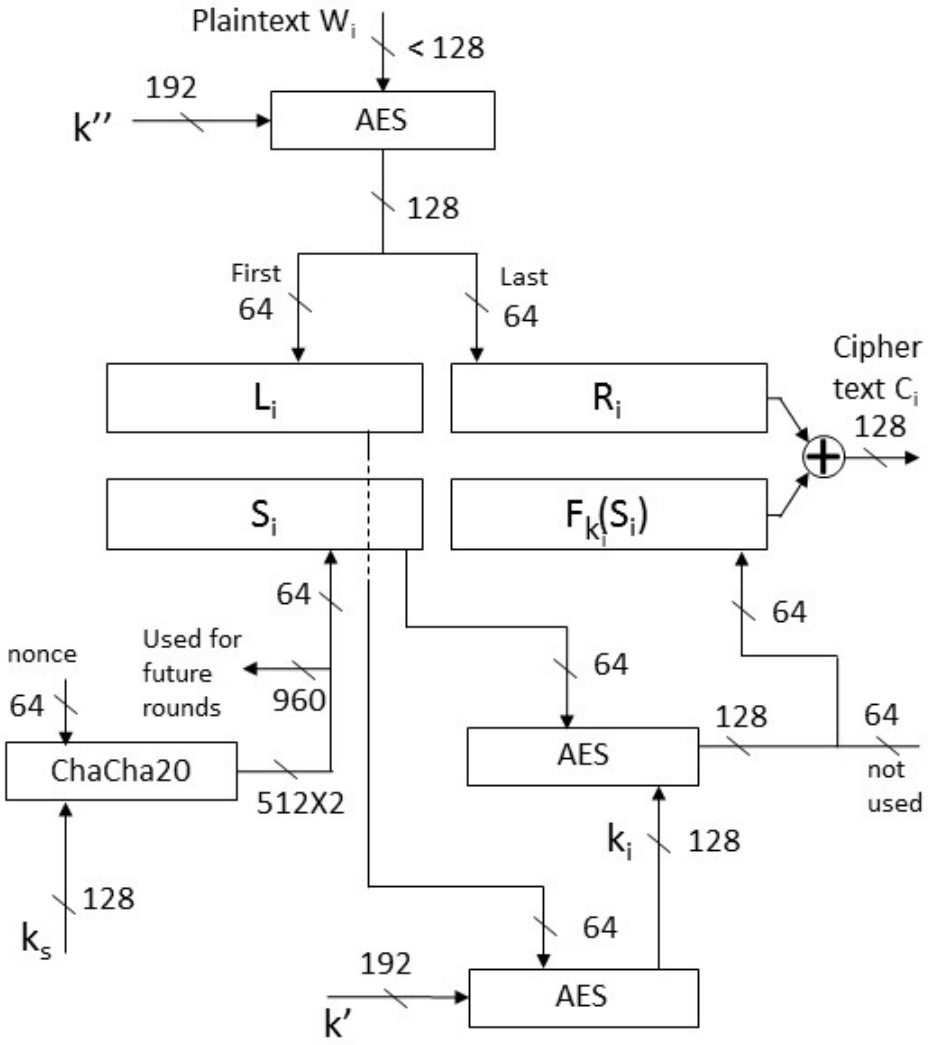

Figure 3-1: Implementation specifics for the scheme proposed by Song, Wagner and Perig [24]

Pseudorandom Permutation

A pseudorandom permutation is used to convert a plaintext word in to a pseudorandom Bit stream of 128 Bits. Advanced encryption scheme (AES) with a 192 Bit key size was used as the pseudorandom permutation. A padded plaintext keyword is AES

encrypted as part of the searchable encryotion. Due to this, the real plaintext keyword is hidden from the entire operation.

Pseudorandom Function

As depicted in Figure 3-1, this scheme requires two pseudorandom functions and AES was selected as the candidate for both, but with two different key lengths.

Pseudorandom Generator

ChaCha20 [41] scheme was used as a pseudorandom generator. This generates an output string of 1024 Bits and can be used to encrypt multiple plaintext words.

Operation

Figure 3-1 contains the block diagram of implementation. The three secret keys k_s, k' and k'' are of lengths 128, 192 and 192 bits respectively. A 64 Bit string is extracted from the output of ChaCha20 for the purpose of searchable encryption of a given plaintext word. During encryption, L_i is padded and encrypted with k' to generate k_i having a length of 128 Bits. Since S_i is with 64 Bits, it is padded and encrypted with key k_i to generate an output of 128 Bits and the first 64 bits are taken as $F_{k_i}(S_i)$. Finally as shown under equation 2.5, the ciphertext C_i of 128 bits is computed.

3.3 Implementation

An implementation with above specifics was done using 'C' language and the final module consisted of the following sub modules as depicted under figure 3-2. The local site component contains separate sub modules for encryption, decryption and search requests. Apart from that there is a septate sub module to manage the local plaintext file storage. The remote site module contains sub modules for file handling, searching and ciphertext storage.

The encryption module encrypts plaintext files and places the ciphertext files at the remote file handler. Remote file handler directs the ciphertext files in to the ciphertxt

storage sub module. When a user initiates a search query, the search requesting sub module at local site prepares the necessary queries based on the cryptographic keys. When the search query is received by the searching sub module at remote site, search operations are performed across the ciphertext file storage. Files containing the desired keywords are extracted by the file handler at the remote site and they are directed to the decryption sub module at the local site. Finally the decryption sub module decrypts the files and places at the local storage for the consumption of the user.

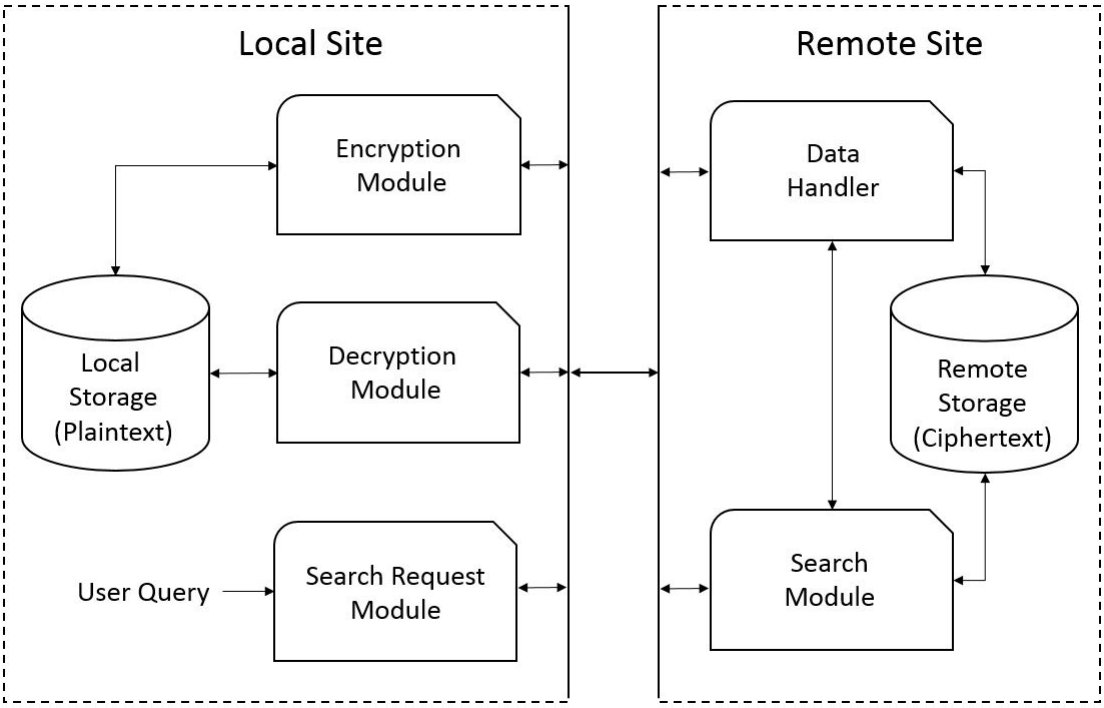

Figure 3-2: A representation of the submodules in the implementation

3.4 Evaluation and Results

3.4.1 Data Set

It was decided to choose one language and carry out the evaluations. Therefore English was selected as the preferred language for the evaluations and all tests were carried out on real English text files. It was observed that most real life text files are in the range few KBs to few MBs. For testing purposes, 159 text files were used and they were

in the range of 75 KB to 8 MB. The total size of the data was 99.425 MB. Table 3.1 contains the distribution of file sizes. The tests were carried out on an Intel Core i5 2.3GHz CPU with 8 GB memory.

Range	**No of Files**	**Sum(File Size) in MBs**
0-99 KB	26	2.29
100-199 KB	47	5.99
200-299 KB	9	2.08
300-399 KB	11	3.74
400-499 KB	9	3.89
500-599 KB	12	6.43
600-699 KB	2	1.26
700-799 KB	1	0.76
800-899 KB	3	2.43
900-1023 KB	3	2.28
1-1.99 MB	28	40.79
2-2.99 MB	5	11.63
3-3.99 MB	1	3.29
4-4.99 MB	1	4.13
7-7.99 MB	1	7.91

Table 3.1: Distribution of plaintext file sizes

3.4.2 Encryption

Files were sequentially encrypted and uploaded to the remote site component. Multiple rounds of encryption has been carried out and the average time was recorded. The different times taken for the encryption of each file is available in figure 3-3. The total average time taken to encrypt the file set was 74.192 s.

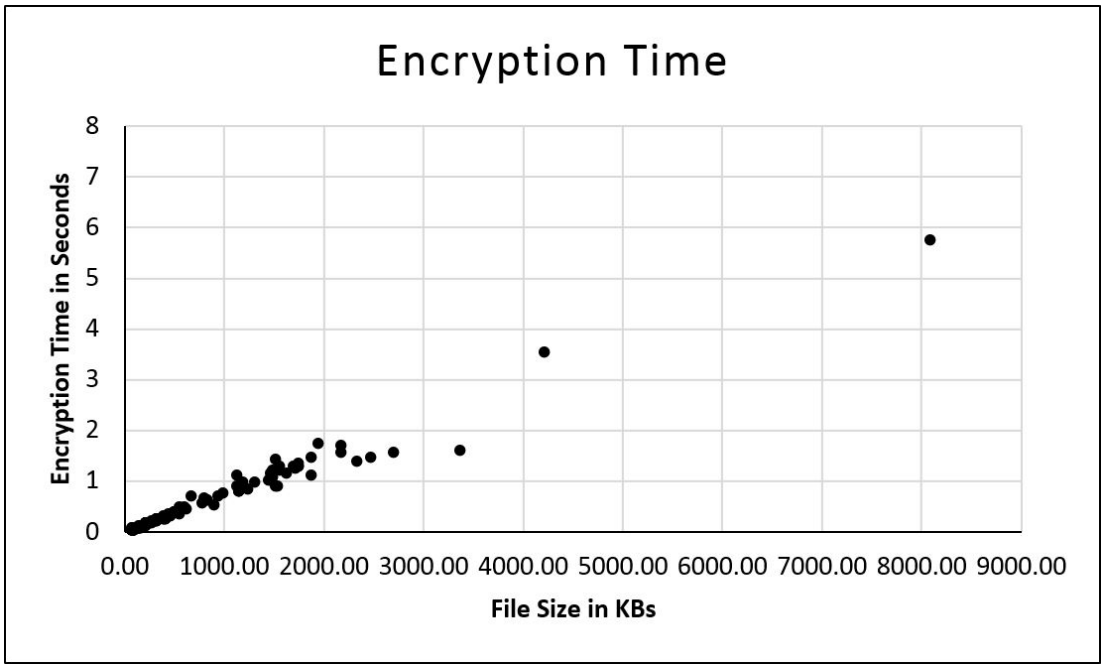

Figure 3-3: Time taken for searchable encryption

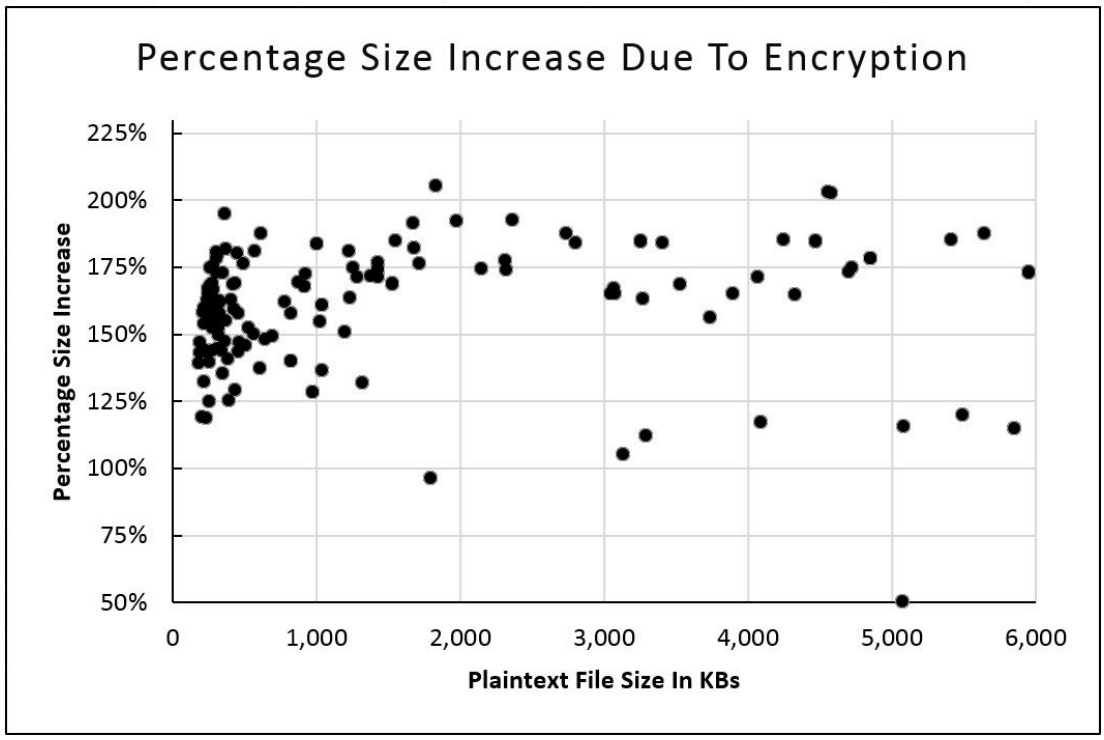

Figure 3-4: Percentage size increase due to searchable encryption

As per the results total encrypted file size equaled 259.59 MB. Since the encryp-

tion increased the file size, the percentage size increase due to encryption was also measured and recorded. Figure 3-4 shows a graph of percentage size increase due to encryption against the size of each plaintext file. As per the implementation results a 160.49% increase in the file database size due to symmetric searchable encryption is observed. Such an increase in the file size will consume additional storage for large file sets. Therefore it was evident that the size increase due to encryption needs to be controlled if such a scheme is to be employed in an industrial application.

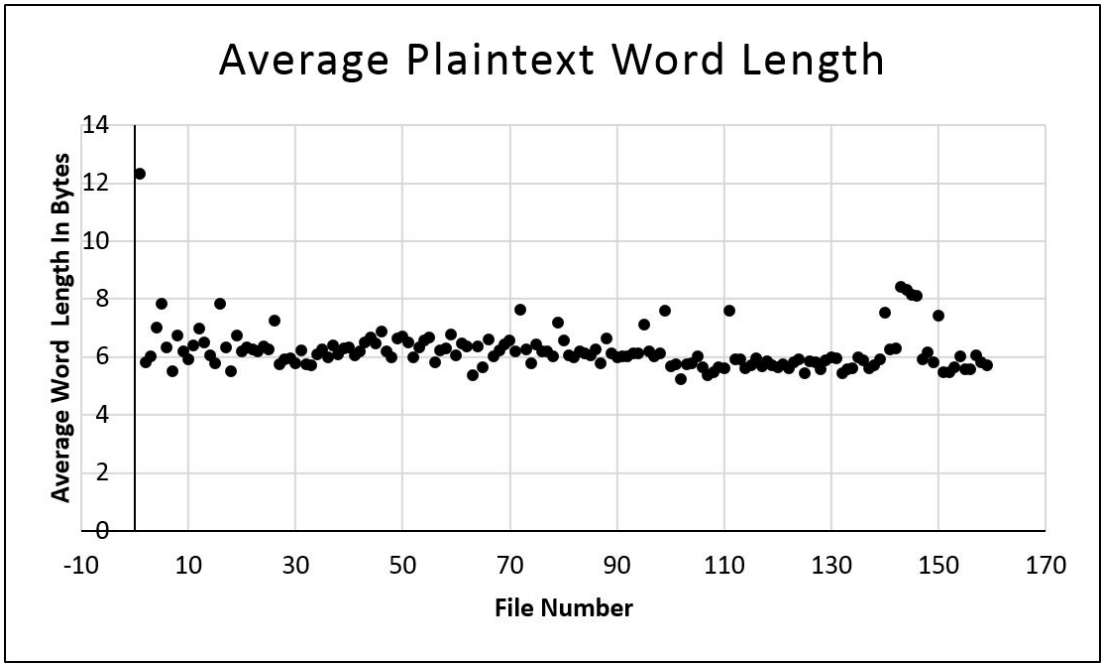

Figure 3-5: Average plaintext word length

The scheme treats each word as a keyword and encrypts each plaintext word separately. The combination of separately encrypted plaintext words produces the ciphertext file. Actual and real life files contain different characters along with actual words. This results in a lot of unnecessary keywords captured for searchable encryption under the generic scheme. Also the plaintext words of the files used for the evaluation are of different lengths. Please refer the figure containing the distribution of average plaintext word length per file.

Due to the initial block encryption, padding is inherent to the scheme. Therefore, the possible option is to attempt on reducing the number of keywords to achieve a

lower storage consumption by the ciphertext files.

3.4.3 Search

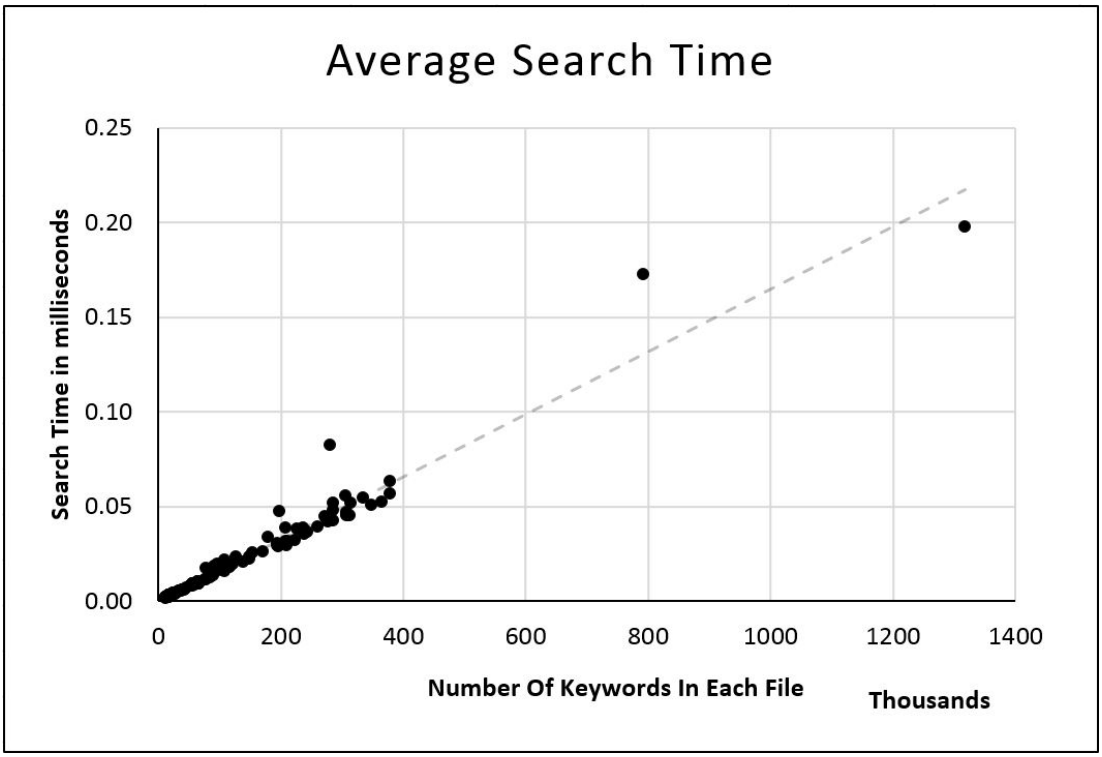

Figure 3-6: Average search time vs number of keywords in file

Followed by encryption multiple search operations were performed on different keywords and the average search time has been recorded. Total average time taken for a keyword search was 2.739 ms. Figure 3-6 shows how the average time taken for a single keyword search varies against the number of keywords in each file.

3.5 Analysis

Since the encryption is not a frequent activity, the 74.192 seconds taken for 99.425 MB is acceptable. However, for a database of 100 GB it can be expected to take 21.22 hours. In an industrial stetting, the time has to be reduced by introducing higher computing power.

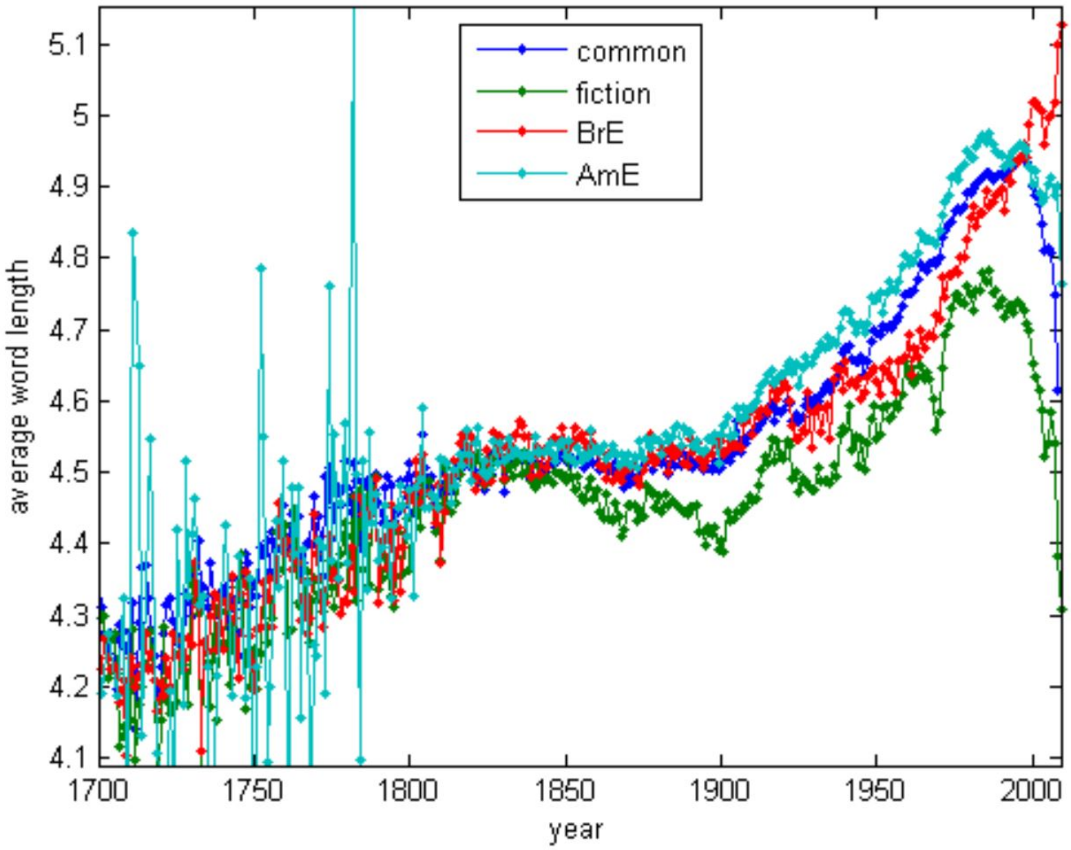

Figure 3-7: Variation of average English word length over the years as presented by Bochkarev, Shevlyakova and Solovyev [42]

Due to the nature of the scheme, it is required for the individual words to be encrypted separately for them to be searched later. Since each keyword is block encrypted as per the initial steps of the searchable encryption, padding is introduced to bring each key word to the block encryption length. This introduces an additional overhead. Studies have been carried out to analyze the English language and its prop-

erties such as word lengths, total number of words in the language, use of grammar words etc. As discussed by Bochkarev et al [42], the average English word length varies based on different time periods. For instance, it has been between 4.5 to 5 letters per word in the past two decades. Figure 3-7 contains how the average Eglish word length has varied over the time as presented by Bochkarev et al. The items in the legend, $common, fiction, BrE, AmE$ indicate the results for the common bases, fiction bases, British bases and American bases respectively.

Considering the average English word length, it is possible to evaluate the impact of padding. Since the word length used for the implementation is 16 bytes, the padding consumes on average 11 to 11.5 letters per encrypted keyword. Therefore theoretically on average 70 % of each keyword needs to padded for English documents originating from the last two decades.

With the implementation results we can expect the search time for a single keyword in a database of 100 GB to be 2.82 seconds. Also the search operation is having a time complexity of $O[n]$ where n is the number of words in the encrypted document. Therefore, an effective way of reducing n could provide significant gains in search time.

Based on the implementation of the original scheme, two potential improvements can be identified as follows which are also to be considered as practical objectives of the research.

- Reducing the size of encrypted files stored in remote server.
- Reducing the search time.

It is theoretically possible to use block encryption and achieve a reduced overhead in encrypted files compared to the original implementation. However to facilitate search operations, key words must be extracted and separately encrypted under the symmetric searchable encryption. Since a reduction in the number of keywords will provide direct gains in search time, there is a need for an efficient pre-processing mechanism for keyword extraction. Next chapter contains the proposed modified approach with the aim of achieving the above mentioned practical objectives.

Chapter 4

Keyword Separation, Proposed Scheme and Results

4.1 Refined Scheme

4.1.1 System Model

As per the original scheme all words in a file are treated as keywords. The new approach is to separate the keywords from the original plaintext file. Following refined scheme is proposed with a mechanism for efficient keyword separation. Table 4.1 contains the specifics of the scheme.

Original representation of a file	$[W1, W2, .., Wn]$ where W and n denotes a "word" and the number of words in the file respectively.
New representation of a file	$\langle [n'Keywords], [File] \rangle$ where n' is the reduced number of keywords after pre-processing and $n' << n$.
Cipher text file	$\langle [SE(W_{1'}), SE(W_{2'}), .., SE(W_{n'})] , [BE(File)] \rangle$ Where SE and BE are symmetric searchable encryption and block encryption respectively.

Table 4.1: Refined scheme with keyword separation

As per the new representation of a plaintext file, it is expected to extract n' number of keywords from the file where n' is significantly smaller than n, the original number of keywords in the plaintext file.

The modified scheme uses two encryption schemes. In order to facilitate search operations, the extracted keywords are encrypted using the Searchable Encryption pro-

posed in the original scheme while the data file is separately encrypted using a known block cipher. This approach has following advantages.

- Searching is only required to be performed on the reduced set of keywords which are encrypted using searchable encryption.
- A standard encryption scheme can be used to efficiently encrypt the plaintext database.
- A standard encryption scheme will result in lower space requirement for the encrypted data.

4.1.2 Keyword Separation

It is possible to derive a keyword extraction mechanism by studying the properties of the language of the plaintext. Since the research employs a set of English plaintext files, the properties of English language are considered. It is possible to perform such studies for any language with the aim of deriving efficient ways of keyworseparationon. When the nature of English language words is considered, following observations can be made.

- Not all words in a text paragraph are significant to be a key word.
- English grammar words such as pronouns, prepositions, conjunctions, determiners, exclamations take a significant portion of an English paragraph.
  Eg she, he, that, something, in, on, after, or etc.
- There are duplicate words.
- There are non-alphabetic characters/words.

Therefore a mechanism with the capability to identify and exclude the insignificant keywords while extracting the significant keywords is required. With above observations, it is possible to prepare a list of such insignificant, frequently occurring words that can be used as a baseline for keyword extraction. As part of the study an example list of 244 grammar words was prepared. Following steps are proposed for keyword

extraction.

Step 1: Word extraction by removing non alphabetic characters and words.

Step 2: Comparison with the "List of grammar words" and Remove grammar words.

Step 3: Remove strings that are single or two characters in length as we treat them to be insignificant as key words.

Step 4: Remove if string length is greater than or equal to the word size. As proposed earlier the selected word size is 16 Bytes. Last character of the word is to contain the pad length.

Step 5: Remove duplicate words.

4.2 Results

4.2.1 Pre-processing

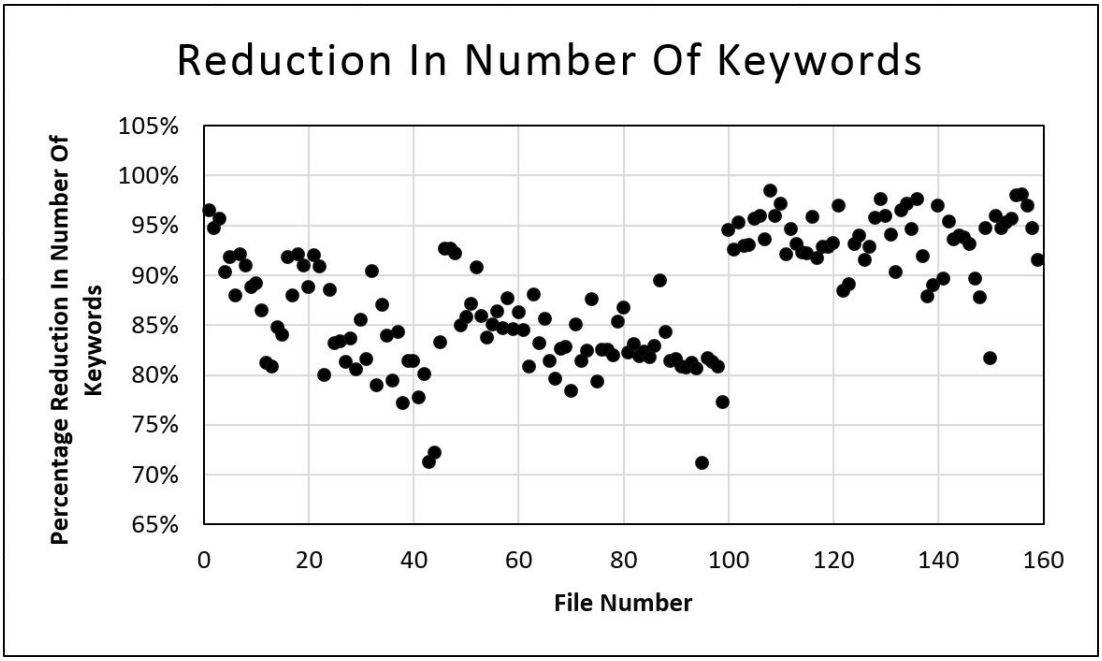

Figure 4-1: Percentage reduction in number of keywords due to pre-processing

Based on the new pre-processing mechanism proposed, all plaintext files at the local site were pre-processed. Total time taken to pre-process all the files of size 99.425 MB

was 9.279 s. Without pre-processing the number of keywords in a file was in the range of 11,000 to 11.9 million. Due to pre-processing it is reduced in to the range of 1,800 to 61,000 words.

Figure 4-1 shows the reduction in number of keywords per file as a percentage due to pre-processing. Average reduction in number of keywords due to pre-processing was 93.34%. Therefore, considering an $O(n)$ time complexity, we anticipate more than 90% improvement in search time due to this reduction.

4.2.2 Encryption

Keywords and text files were separately encrypted and encrypted file sizes and time taken has been recorded. Encrypted keyword size was 17 MB and block encrypted file size was 99.42 MB in the total of encrypted file sizes equaling 116.42 MB. The total time taken for encryption was 27.31 s. Therefore we can anticipate an encryption time of 7.76 hours for 100 GB and this can be brought to the applicable range of an industrial application by introducing industrial scale hardware for the processing.

4.2.3 Search

Once the pre-processing is completed, files were sequentially encrypted and made available for the remote site. Multiple search operations were performed on different keywords to understand the effect of the new scheme.

Tests were carried out on an Intel Core i5 2.3GHz CPU with 8 GB memory. As per the results the average search time taken for a single search operation is 0.216 ms. Figure 4-2 compares the time taken for a single keyword search under the two schemes and showcases the significant reduction in search time due to modified approach.

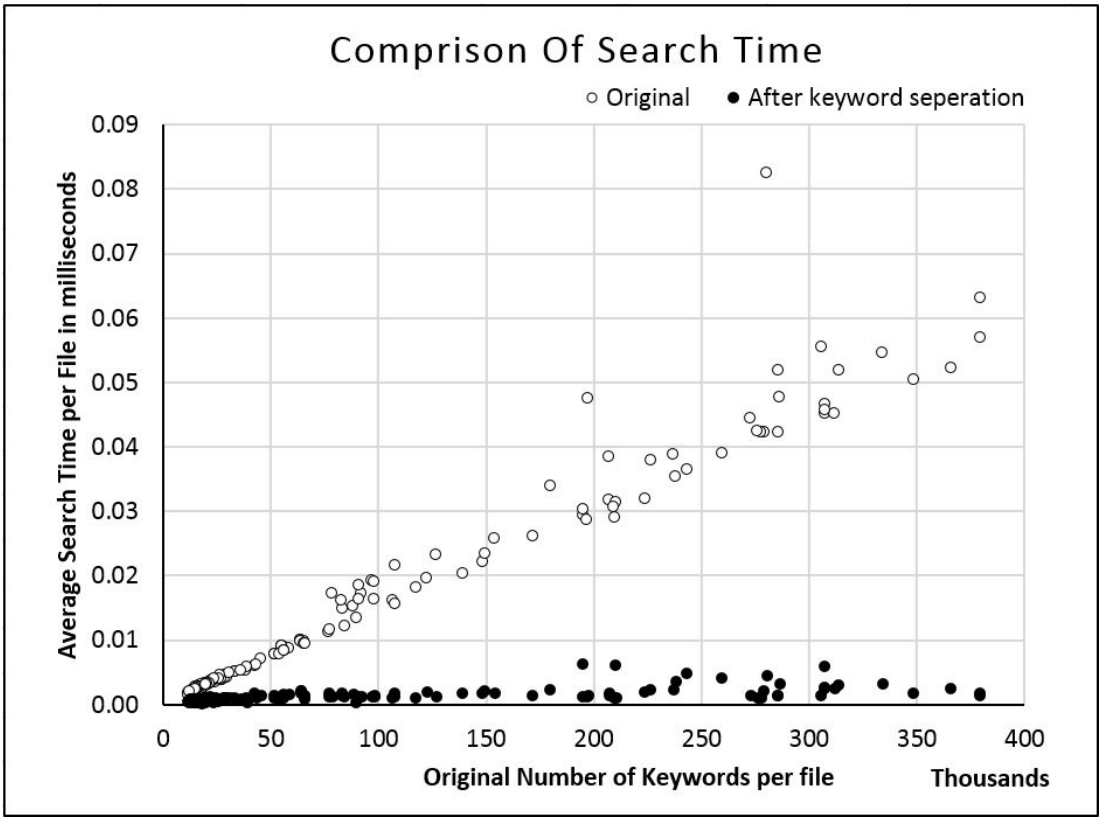

Figure 4-2: A comparison of search time under two approaches

4.3 Security Analysis

During implementation, it is possible to treat a key word as a fixed length string or a variable length string. The choice of ciphers are for a fixed length string due to the security requirements. On average an English word can be represented by few bytes and a variable length keyword based on the actual word length may lead to shorter ciphertext, hence susceptible to more collisions and brute force attacks. There are theoretical advantages in shorter word lengths such as the ability to use a shorter padding, generating shorter cipher text etc in achieving higher space efficiencies. It is possible to justify the choice of fixed word length, 128 Bits with the original ideas proposed as part of Song et al model and birthday paradox [43]. As part of the implementation of searchable encryption, a plaintext word is first encrypted using the advanced encryption scheme. The output is split in to two streams of length m and $n - m$ respectively

and used separately. In such a scheme, the probability that at least one collision happens after encrypting k words is $k(k-1)/2^{n-m+1}$. For thousand keywords the different collision probabilities are available in table 4.3.

Parameters	**Probability of a collision in L_i**
n = 64, m = 32	1.16×10^{-4}
n = 128, m = 64	2.26×10^{-14}
n = 128, m = 32	1.16×10^{-4}

Table 4.2: Probability of a collision in L_i

For the implementation in this research, $n = 128$ and $m = 64$ have been selected. The choice of the stream cipher is ChaCha20 for which the security is established [44], [45] and serves practical implementation requirements.

As depicted in figure 3-1, $F_k(S)$ is derived based on the keyword being encrypted. Therefore, same keyword in multiple files will result in a different cipher text. However due to keyword separation, the number of key words is reduced and if the same keyword is present at multiple files at the same location, could result in same cipher text. This can be addressed by replacing the nonce used for the stream cipher with the XOR output of the file name and the nonce being used for the stream cipher. This additional processing step will not introduce any overhead for search.

When a data requester requests for a keyword search, part of the search trapdoor contains the search keyword pre-encrypted with the advanced encryption scheme using the same key, k'' used for encryption. Therefore, When the plaintext file is block encrypted as part of the modified scheme, it is important to use a different cipher or the same advanced encryption scheme with a different key. If the same cipher and the key are used, it must not be implemented in electronic code book mode. This is to avoid potential frequency analysis based attacks on the block encrypted file with the use of trapdoors provided by data requester.

4.4 Future Directions

The research has primarily focused on efficiency enhancements for single keyword queries which is acceptable for many practical applications. Current research suggests many theoretical models and schemes for conjunctive keyword searches and range queries on encrypted data. Addressing practical limitations, proposing efficiency enhancements and introducing practical implementation models for such schemes are interesting problems to solve in future.

As widely discussed in previous chapters, there is a need for secure processing of search results obtained by searching on ciphertext. Performing computations on search results while the data is in encrypted for is another challenge. As per the literature study, the concepts of homomorphic encryption can be used as a solution for this challenge. Therefore secure implementations and efficiency enhancements on homomorphic encryption models are also remaining problems to be addressed in future research.

Chapter 5

Example Use Cases

5.1 Introduction

Capital markets are comprised of Stock Exchanges, Securities Depositories, Clearing Houses and industry participants such as Brokers, Banks and Custodians etc. Brokers are able to submit trade orders. When trades orders are matched and executed in Stock Exchange, they are cleared through the Clearing House. The Securities Depository holds securities on behalf of the investors and provides services to industry participants such as Custodians and Banks. The post trade world is comprised with Clearing Houses, Securities Depositories and industry participants.

When it comes to post trade services, moving out of the traditional service model and transforming in to a cloud based service model is an interesting topic. When everybody is moving towards cloud, companies need a way of differentiating the services provided and gain the competitive edge. Therefore, finding novel and attractive ways of redefining the cloud based solutions is a requirement. Most post trade service providers keep a significant amount of private data of clients such as securities account details, cash account details, securities balance positions, investor identification numbers etc. Therefore secure storage of private data is a major problem. The consumers and the market are particularly concerned about cloud services and questions are already being raised from the market regarding the security of the data held by service providers.

With the growing needs of information management, many capital market infrastructures such as Sock Exchanges, Securities Depositories etc face the challenge of ef-

fectively storing their data in a confidential manner. Almost all of major institutions maintain their own internal data storages. Their data is managed internally and is not directly exposed to external parties. In this context, coming up with a secure data storage will draw the attention of the market and eventually help service providers in migrating their services and operations in to Cloud.

5.2 System Models for Capital Markets and Post Trade Processing

5.2.1 Stakeholders

There are few common stakeholders in a cloud based environment.

Cloud Service Provider (CSP):

An entity capable of storing encrypted data and performing operations on the encrypted data. This could be a commercial data service provider. A Securities Depository or a Clearing House can use the services provided by such a service provider by encrypting its own data and storing them at the CSP.

Access Control Server (ACS):

Conceptually an Access Control Server (ACS) controls the access to the data processing results performed by CSP. In a private cloud setting, the Securities Depository or the Clearing House may play the role of the ACS for the industry participants such as Brokers and Banks. It is also possible for this role to be borne by an independent regulatory body such as the Central Bank, Securities and Exchange Commission, Monitory Authority etc.

Data Requester (DR):

Industry participants can makes a processing request to the CSP via ACS. i.e.The processing result is not directly provided to the DR by a CSP. Instead the requests are provided to DR through the ACS. This allows the CSP to take control of the data processing while ACS handles the access control for the requests made by multiple eligible data requesters such as industry participants.

5.2.2 Models

Model I

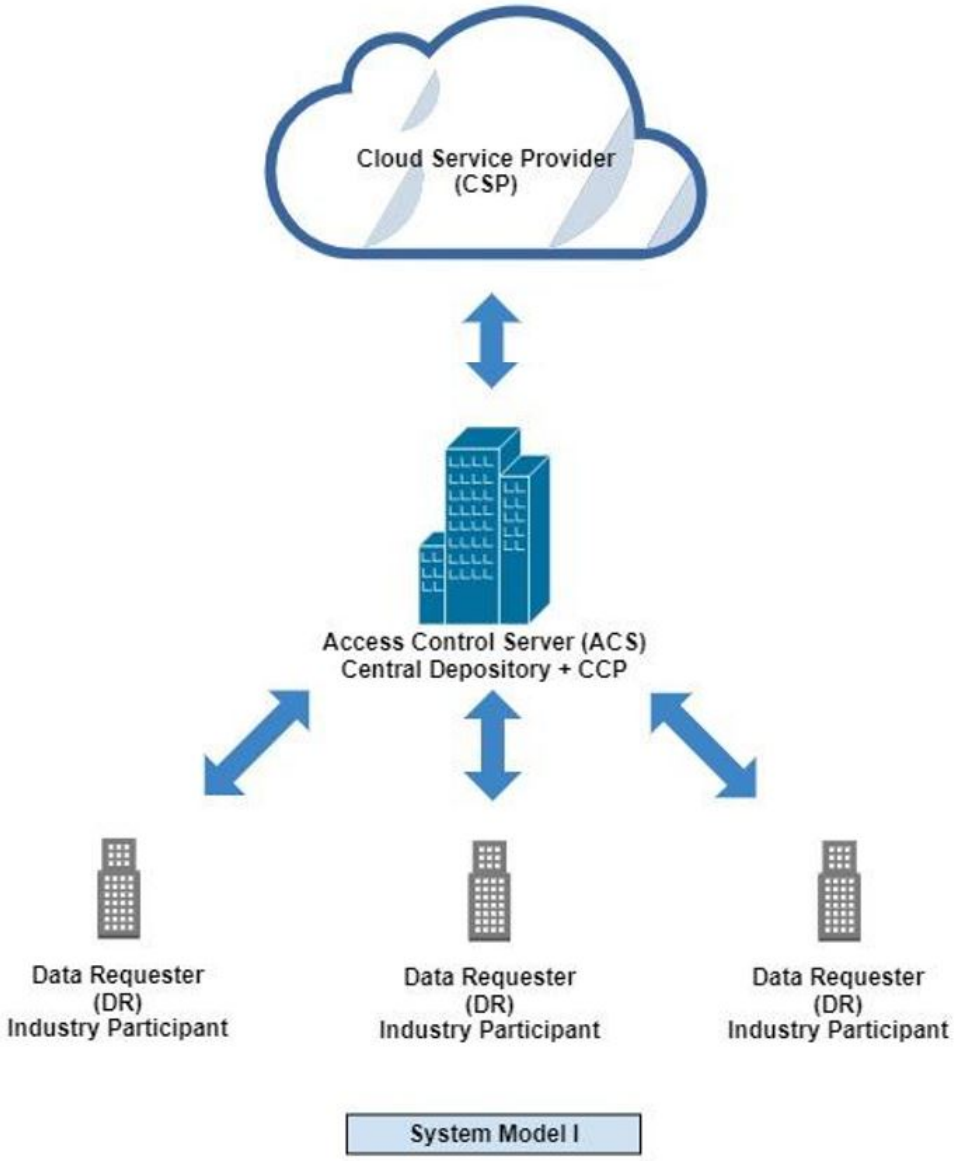

Figure 5-1: System Model I for encrypted clouds

The Securities Depository and the Clearing House may act as the Access Control Server to fulfill the data requests of following nature.

- Fulfilling their own data and processing needs. Here the Depository and the Clearing House acts as a data requester to facilitate in house data needs.

- Fulfilling data requests coming from industry participants such as brokers and banks. Here the Depository and the Clearing House provides a service for the rest of the data requesters.

Model II

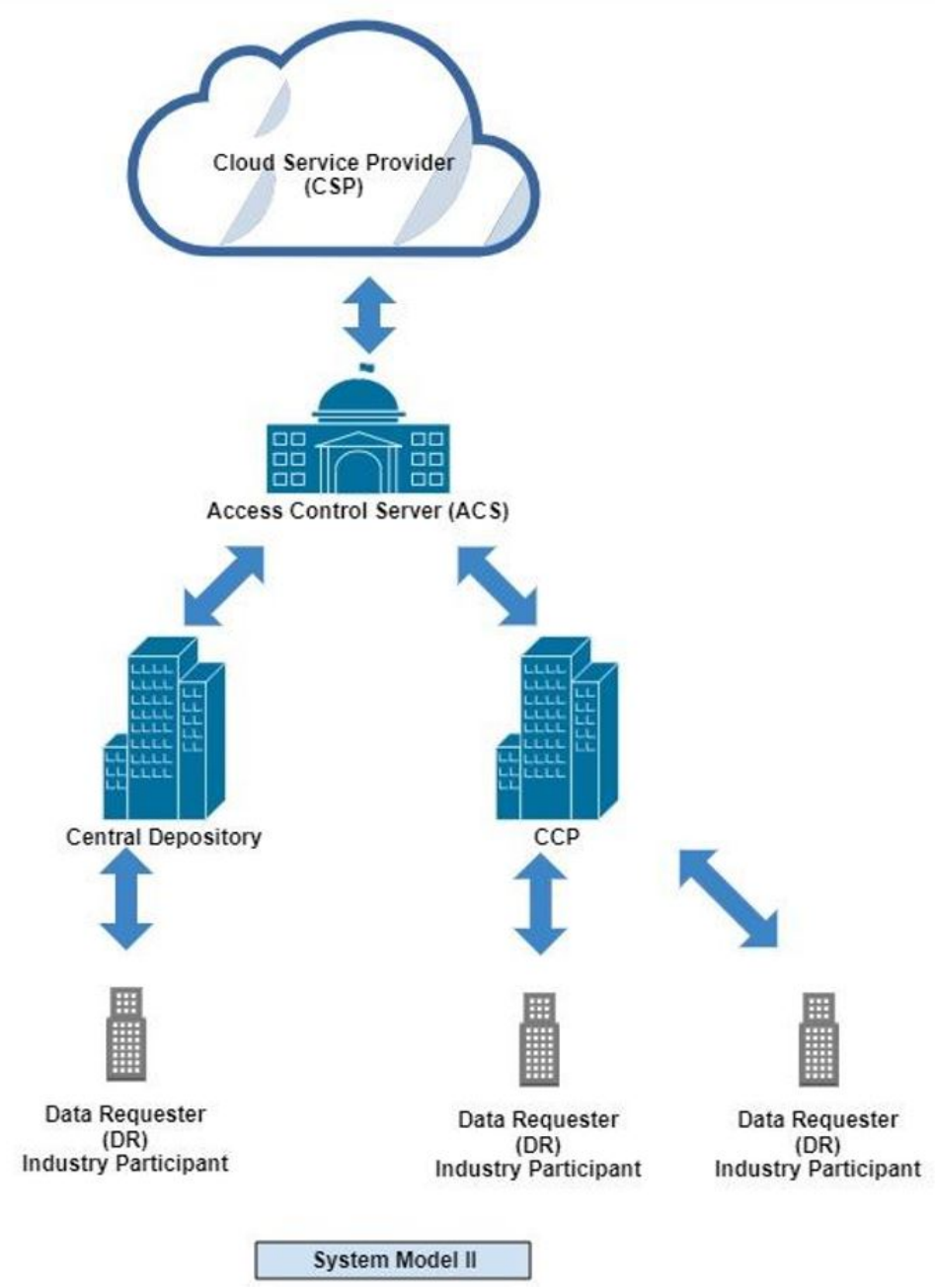

Figure 5-2: System Model II for encrypted clouds

A regulatory body such as the Exchange Commission, Central Bank or an Independent Entity may play the role of the Access Control Server to fulfill the data and processing needs of Securities Depository or Clearing House. The Securities Depository or the Clearing House may initiate data requests for themselves or on behalf of the industry participants connecting to them.

5.3 Operations on encrypted cloud stored data for capital market operations

If the primary data storage of a Securities Depository is maintained in a Cloud, there will be complex data processing requirements to facilitate the daily operations of the depository. One of the simplest form of performing mathematical operations on the encrypted data in the cloud would be as follows.

Use Case: Dividend(entitlement) Computation as part of corporate actions. During a dividend computation, each share holder may receive an incentive proportional the number of shares owned by the share holder.

Step I: Key Data Separation
Suppose each securities balance position is treated as a distinct transaction. Securities Account Id and Balance can be extracted as key data from each balance position of the investors. A key data separated transaction in plaintext form can be represented as
$< SecuritiesAccountID >< Balance >< BalancePosition >$

Step II: Encryption
Since the entitlement computation requires scalar multiplication, the Balance can be encrypted using an encryption scheme homomorphic over scalar multiplication. The raw position may be block encrypted.
Eg: ElGamal encryption can be used to encrypt the securities Balance. Suppose the operator E denotes ElGamal encryption and C denotes the cipher text, then the en-

cryption of extracted balances is represented as follows.

$$C_{balance} = E(balance) \tag{5.1}$$

Step III: Dividend Computation

During dividend computation the balance is multiplied by the the entitlement multiplier. For the computation to take place at the cloud, the cloud application should be provided with the encrypted entitlement multiplier. Therefore the entitlement multiplier is encrypted by the ElGamal encryption as shown by equation 5.2 and sent to the cloud and scaler multiplication results are requested from the cloud.

Eg: Suppose the share holders are to receive 2 LKR for each share own by them, then the entitlement multiplier is 2. The scalar value 2 is ElGamal encrypted and sent to the cloud.

$$C_{entitlement\_multiplier} = E(entitlement\_multiplier) \tag{5.2}$$

Cloud server will compute the entitlement for each balance position by computing the specific homomorphic operation and provide the result in encrypted form. Eg: In the case of ElGamal encryption, the product of encrypted 'Balance' and encrypted 'Entitlement Multiplier' will provide the result of scalar multiplication in encrypted form as shown in equation 5.3.

$$C_{dividend} = C_{balance}.C_{entitlement_multiplier} \tag{5.3}$$

The Securities Depository is now able to decrypt the result received from the Cloud to obtain the dividend for each Securities Account. This facilitates the dividend computation at cloud without the securities balances being disclosed to the cloud server. Suppose the operator D denotes the decryption, then the dividend is obtained as follows.

$$dividend = D(C_{dividend}) \tag{5.4}$$

5.4 Cloud based reporting for capital market operations

It is important to look at the potential data querying use cases with an encrypted cloud. As a Post Trade service provider, a Securities Depository or a Clearing House may have to provide a reporting mechanism to aid the operations of the participant users. The traditional method of report generation is associated with querying from the local storage to generate a predefined set of reports for participants. For instance,

- Monthly Trade Report: Contains all the Trades reported within the month.
- Monthly Balance Statement: Contains the balance position updates for the month

One of the simplest form of performing searches on encrypted data for report generation would be as follows.

Use Case: Monthly trade report generation. Suppose each Trade is treated as a transaction containing multiple keywords.

Step I: Keyword Separation
Separates Keywords under interest. Suppose the reports are to be generated for each participant and Trading Date. Then the Trade Date, Trading Member are to be extracted as key words.
Eg: Trade Date = 2019-10-02, Trading Member = 202

Step II: Keyword Enrichment
Creates new keywords based on the report criteria. Suppose the report should include trades from the previous month. Then, the month of trade needs to derived as a new keyword based on the extracted trade date.
Eg: If Trade Date = 2019-10-02 then Month = 'October' is the derived keyword

Step III: Encryption

With step I and II, a raw transaction will have two parts.
i.e. $Transaction :< Keywords >< TransactionDetails >$. Keywords are encrypted based on a searchable encryption algorithm. This facilitates search operations on encrypted transactions. The 'Transaction Details' are to be encrypted using a standard symmetric algorithm. This achieves the space efficiency.

Step IV: Search
Suppose in the month of November a trade report is generated for each trading member for the trades reported in the previous month. The Securities Depository may request for all trades with keyword 'October' and the properties of the selected searchable encryption algorithm will facilitate the Cloud Service Provider in performing a keyword search on the encrypted trades without decrypting the data and provide the query result in encrypted form. The Securities Depository is able to decrypt the encrypted search result to obtain all the trades reported in October and group by the Trading Member to generate the individual reports.

5.5 Challenges

Selection of a Scheme
One of the major challenges of computing on encrypted data is selecting a suitable cryptographic protocol for encryption. There are no specific guidelines on selecting different cryptographic schemes for different applications. But what to select for a capital market environment is something to be identified based on the specific application. The selection of a suitable protocol determines what types of homomorphic properties are available for the application and which operations are possible on top of the ciphertext.

Cost vs Usability
Since various encryption schemes are homomorphic over different mathematical operations, an end to end implementation of a complete use case will require the adoption

of multiple encryption schemes.

Eg: Higher number of operations on encrypted data will require multiple copies of the same plaintext to be encrypted using different schemes applicable for each operation. This leads to overheads in the space usage in cloud.

Complexity

With the involvement of homomorphic and searchable encryption, a cloud based implementation will carry a high complexity. This demands specialized personal for design and development, specific hardware resources, longer implementation periods and difficulties in troubleshooting etc.

Performance

Time complexity of the operations is one of the major concerns for capital market technology solutions. Such implementations with heavy encryption operations and computations will create performance considerations. The effort required for the performance optimizations to meet the service level agreements will be significantly higher than a typical implementation.

Legal Issues

The judicial system in the country will control the use of personal and financial data. These laws will limit the types of data that can be outsourced in to cloud storages. However it is possible that the data in encrypted form may have exceptions. It's true that when the sensitive data is encrypted, the ciphertext is not readable. Hence when you outsource such data, technically you are not taking the data out of the country while they are in the sensitive plaintext form. Therefore it is clear that whether encrypted data can be outsourced may be subjected to legal interpretation.

Chapter 6

Conclusion

In this research the current state of the topic of computing of encrypted data was studied. It was identified that the research on computing on encrypted data can be categorized under two main branches, searchable encryption and homomorphic encryption. Different searchable and homomorphic encryption schemes were studied in detail. Later the symmetric searchable encryption scheme proposed by Song, Wagner and Perrig was extensively studied and the scheme was implemented to study the suitability of the scheme for practical use and its performance. Based on the performance results, the time taken for search operations and the file size increase due to encryption was identified as major bottlenecks if the scheme to be employed for industrial applications.

To address these issues a pre-processing mechanism for efficient extraction of significant keywords was presented. The key idea is to bring down the number of keywords in to a practically manageable range. The total time taken for the pre-processing of a file set of 100 MB took less than 10 seconds which is within practical limits. With these results, about 170.66 minutes of pre-processing time for a plaintext database of 100 GB can be expected.

In the modified scheme, the block encrypted plain text file component consumes no extra space. The only extra space usage is due to the separately encrypted keywords. An increase of 17% in the final encrypted file size was observed and that is due to the keyword files introduced by the modified scheme. It is also possible to introduce compression prior to the encryption and further reduce the size of block encrypted file

component. Compared to the original scheme which produces an additional 160.17 MB of data due to encryption, this modified scheme only produces additional 17 mBs due to encryption. This is a 89.83% reduction in extra space usage.

Time taken for a single keyword search has drastically dropped due to the modifications proposed. As per the results, a 92.11% improvement in search time has been achieved. For an encrypted remote DB of 100 GB, the anticipated a search time is 222 milliseconds per single keyword search under an Intel Core i5 @2.3GHz CPU and 8 GB memory. However modern data storages have much more processing power and an actual service provider with industrial scale computing power will achieve far lower search time than above for a database of 100 GB. Therefore with the above results the modified scheme can be selected as a practical scheme that is feasible for considerably larger databases with industrial scale hardware.

In addition to these, a study was also done to evaluate the practical applications of above techniques in capital markets domain. It was observed that there are challenges to overcome when the concepts on computing on encrypted data are to be applied for practical use. Further research is required to address and overcome these challenges in specific application scenarios. However it can be concluded that the computing on encrypted data has a wide array of practical applications and the concepts of searchable encryption can be implemented for practical use.

Bibliography

[1] J. Daemen and V. Rijmen, *The design of Rijndael: The wide trail strategy explained*. Springer, 2001.

[2] H. Delfs and H. Knebl, "Symmetric-key encryption", in *Introduction to Cryptography*, Springer, 2007, pp. 11–31.

[3] H. Bennett Ch and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing int", in *Conf. on Computers, Systems and Signal Processing (Bangalore, India, Dec. 1984)*, 1984, pp. 175–9.

[4] W. Diffie and M. Hellman, "New directions in cryptography", *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[5] M. E. Hellman, "An overview of public key cryptography", *IEEE Communications Magazine*, vol. 40, no. 5, pp. 42–49, 2002.

[6] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[7] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[8] A. K. Lenstra, "Unbelievable security matching aes security using public key systems", in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2001, pp. 67–86.

[9] A. Shamir, "On the security of des", in *Advances in Cryptology*, Springer-Verlag, 1985, pp. 280–281.

[10] J. Jonsson and B. S. Kaliski, “On the security of rsa encryption in tls”, in *Annual International Cryptology Conference*, Springer, 2002, pp. 127–142.

[11] Y. Tsiounis and M. Yung, “On the security of elgamal based encryption”, in *International Workshop on Public Key Cryptography*, Springer, 1998, pp. 117–134.

[12] Y.-C. Chang and M. Mitzenmacher, “Privacy preserving keyword searches on remote encrypted data”, in *International Conference on Applied Cryptography and Network Security*, Springer, 2005, pp. 442–455.

[13] D. Boneh and B. Waters, “Conjunctive, subset, and range queries on encrypted data”, in *Theory of Cryptography Conference*, Springer, 2007, pp. 535–554.

[14] L. Wu, B. Chen, K.-K. R. Choo, and D. He, “Efficient and secure searchable encryption protocol for cloud-based internet of things”, *Journal of Parallel and Distributed Computing*, vol. 111, pp. 152–161, 2018.

[15] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, “Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions”, in *Annual International Cryptology Conference*, Springer, 2005, pp. 205–222.

[16] Y. Wang, J. Wang, and X. Chen, “Secure searchable encryption: A survey”, *Journal of Communications and Information Networks*, vol. 1, no. 4, pp. 52–65, 2016.

[17] E.-J. Goh *et al.*, “Secure indexes.”, *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.

[18] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption: Improved definitions and efficient constructions”, *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[19] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search”, in *International conference on the theory and applications of cryptographic techniques*, Springer, 2004, pp. 506–522.

[20] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions", *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[21] K. Bennett, C. Grothoff, T. Horozov, and I. Patrascu, "Efficient sharing of encrypted data", in *Australasian Conference on Information Security and Privacy*, Springer, 2002, pp. 107–120.

[22] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval", in *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, IEEE, 1995, pp. 41–50.

[23] C. Cachin, S. Micali, and M. Stadler, "Computationally private information retrieval with polylogarithmic communication", in *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 1999, pp. 402–414.

[24] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data", in *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, IEEE, 2000, pp. 44–55.

[25] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data", in *International Conference on Applied Cryptography and Network Security*, Springer, 2004, pp. 31–45.

[26] R. Canetti, U. Feige, O. Goldreich, and M. Naor, "Adaptively secure multi-party computation", in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, ACM, 1996, pp. 639–648.

[27] O. Goldreich, "Secure multi-party computation", *Manuscript. Preliminary version*, vol. 78, 1998.

[28] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms", *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[29] C. Gentry, “Fully homomorphic encryption using ideal lattices. proceedings of the 41st annual acm symposium on symposium on theory of computing-stoc’09. vol. 9”, 2009.

[30] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*, 09. Stanford University Stanford, 2009, vol. 20.

[31] C. Gentry, “Computing arbitrary functions of encrypted data”, *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[32] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, “Fully homomorphic encryption over the integers”, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2010, pp. 24–43.

[33] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, “Fully homomorphic encryption over the integers with shorter public keys”, in *Annual Cryptology Conference*, Springer, 2011, pp. 487–504.

[34] C. Gentry and S. Halevi, “Implementing gentry’s fully-homomorphic encryption scheme”, in *Annual international conference on the theory and applications of cryptographic techniques*, Springer, 2011, pp. 129–148.

[35] W. Ding, Z. Yan, and R. H. Deng, “Encrypted data processing with homomorphic re-encryption”, *Information Sciences*, vol. 409, pp. 35–55, 2017.

[36] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes”, in *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 1999, pp. 223–238.

[37] R. Cramer and V. Shoup, “Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption”, in *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2002, pp. 45–64.

[38] E. Bresson, D. Catalano, and D. Pointcheval, “A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications”, in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2003, pp. 37–54.

[39] G. Shafi and S. Micali, “Probabilistic encryption”, *Journal of computer and system sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[40] D. Naccache and J. Stern, “A new public key cryptosystem based on higher residues”, in *Proceedings of the 5th ACM conference on Computer and communications security*, ACM, 1998, pp. 59–66.

[41] D. J. Bernstein, “Chacha, a variant of salsa20”, in *Workshop Record of SASC*, vol. 8, 2008, pp. 3–5.

[42] V. V. Bochkarev, A. V. Shevlyakova, and V. D. Solovyev, “The average word length dynamics as an indicator of cultural changes in society”, *Social Evolution & History*, vol. 14, no. 2, pp. 153–175, 2015.

[43] S. Goldwasser and M. Bellare, “Lecture notes on cryptography”, *Summer course Cryptography and computer security at MIT*, vol. 1999, p. 1999, 1996.

[44] G. Procter, “A security analysis of the composition of chacha20 and poly1305.”, *IACR Cryptology ePrint Archive*, vol. 2014, p. 613, 2014.

[45] A. Langley and Y. Nir, “Chacha20 and poly1305 for ietf protocols”, 2018.