SUDAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

COLLEGE OF GRADUATE S TUDIES

Searching Encrypted Data by Using Symmetric Searchable Encryption

البحث في البيانات المشفرة بإستخدام الخوارزميات المتناظرة القابلة للبحث

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

Submitted by:

Salma Eisa

Supervised by:

Dr. Faisal Mohammed Abdalla

February 2016

DEDICATION

To my family

To my teachers

To my husband

To my best friends

I. Acknowledgment

I would first like to express my sincere thanks to my supervisor, Dr. Faisa Mohammed Abdulla for all his support throughout the course of the project.

I would also like to thank my parents and my husband, who have always been prepared to go out of their way to give their utmost support during my study.

Finally, I would like to thank my friends, in particular Afaf Yousif, for her friendship, help and support during my time knowing her.

II. Abstract

Cloud storage has become increasingly prevalent in recent years. It provides a convenient platform for users to store data that can be accessed from anywhere at anytime without the cost of maintaining a storage infrastructure. However, cloud storage is inherently insecure, hindering general acceptance of the paradigm shift. To make use of storage services provided by a cloud, users would need to place their trust, at least implicitly, in the provider. There have been a number of attempts to alleviate the need for this trust through cryptographic methods. An immediate approach would be to encrypt each file before uploading it to the cloud. This approach, calls for a new searching mechanism over encrypted data stored in the cloud.

This dissertation considers a solution to this problem using Symmetric Searchable Encryption (SSE) Scheme. The scheme allows users to offload search queries to the cloud. The cloud is then responsible for returning the encrypted files that match the search queries (also encrypted). Most previous work was focused on keyword search in the Honest-but-Curious (HBC) cloud model, while some more recent work has considered searching on phrases. Recently, a new cloud model was introduced that supersedes the HBC model. This new model, called Semi-Honest but Curious (SHBC), is less restrictive over the actions a cloud can take. In this dissertation, we present a system that are secure under this new SHBC model.

III. المستخلص

أصبحت الحوسبة السحابيه (data cloud) اكثر انتشارا في الفتره الاخيره. لأنها تيوفر منصة ملائمة للمستخدمين لتخزين البيانات مع امكانية الوصول إليها من أي مكان وفي أي وقت دون تكلفة الحفاظ على البنية التحتية للتخزين. ومع ذلك، فإن الحوسبه السحابية بطبيعتها غير آمنه، مما يعوق القبول العام لهذا التطورز للاستفادة من خدمات التخزين التي توفرها ، فإن المستخدمين بحاجة إلى وضع ثقتهم، ضمنيا على الأقل، في مزود الخدمه. هناك عدة محاولات لتخفيف الحاجة لهذه الثقة من خلال وسائل التشفير. ببساطه يمكن تشفير الملفات قبل ارسالها للخادم ولكن هذه الطريقه استدعت وجود آليه للبحث في هذه البيانات المشفرة.

هذه الأطروحة تطبيق لمقترح حل لهذه المشكلة باستخدام هيكله للبحث في البيانات المشفره. هذا نظام يسمح للمستخدمين لاجراء استعلامات البحث على الخادم دون الحوجه لفك التشفير ولا اعطاء الخادم اي معلومات .

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Introduction

As a result of information explosion now days, it becomes a nightmare for most businesses to handle the expenses of managing these information, trimming their IT expenditure become a constant need. Outsourcing coming up as a good solution.

Outsourcing comes up as a solution of managing information systems and increase capacity or adds capabilities on the fly without investing in new infrastructure, training new personnel, or licensing new software. One of the most efficient forms of outsourcing is data cloud. Cloud computing is a general term for anything that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flowcharts and diagrams [1,2].

With the development of the idea of outsourcing, more and more private and confidential information is being centralized into servers that not owned by the real owner of the information. Therefore, people are increasingly concerned about the security of their data. In a trusted server,

access control mechanism can be an effective way to protect your information. But in real word, not all servers can be trusted. So as alternative cryptographic techniques needed to protect this information.

Cryptography is the study of designing techniques for ensuring the secrecy and/or authenticity of information; it is probably the most important aspect to prevent against the increased risk of theft of proprietary information. Although these threats may require a variety of countermeasures, encryption is a primary method of protecting valuable electronic Information [3].

Encryption is an automated tool to convert the data into a form that con not be understood by unauthorized people. There are two forms of encryption in common use: the first one is the conventional, or symmetric, encryption where sender and recipient share a common key to encrypt and decrypt the data. The second one is the public-key, or asymmetric, encryption which uses two keys one for the sender to encrypt and the other for the recipient to decrypt. Encryption can be by dividing the data set to equal blocks or encrypt it each stream by its self. In order to apply the encryption we need what we call a mode of operation. A mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream [3].

To make use of the stored information, we must be able to perform a search easily, and with the sensitive situation of the encrypted data on the cloud the search should be performed without revealing private information. This concept called Privacy-Preserving Computation (PPC); it's a branch of cryptography that deals with the question of whether or not a function with input and output reveals much.

1.2 Problem statement

Performing a search over encrypted data using any mode of operation is not easy because the fact that mode of operations can use an encrypted block to encrypt the previous or next one. So decryption is needed before searching.

In case of outsourced encrypted data, searching need either the server - which we do not trust - to know the encryption key and that is obviously not recommended or other wise to download all the encrypted data set, decrypt it and search it client-side, this is not practical too due to the possibility of large amount of data in addition to consuming the bandwidth and other resources.

1.3 Objectives

The goal is to implement a technique to retrieve a search answers on encrypted data while not revealing any information beyond the presence

(or absence) of the keywords (of the query) in each document and without having to decrypt the data.

The implementation aimed to achieve searches in one communication round , with high efficiency in respect to time and space. Furthermore it should not leak any information beyond the access pattern.

1.4 Scope

Implementing a technique that allow the authorized users to search for keywords over a data that is encrypted using the symmetric encryption without revealing the private key to the server neither perform it in the client side. The server in this search does not take into account the trapdoors and search outcomes of previous searches.

1.5 Research methodology

The symmetric encryption will be used to encrypt the documents and upload it to a server. A technique that allows the server (if authorized) or the authorized users to search for keywords over the encrypted documents will be implemented. Then the efficiency and the performance of this tool will be evaluated .

1.6 Thesis organization

This thesis has five chapters, after the introductory chapter there is chapter two: Background and Literature review. This chapter provides a background about the cryptographic techniques used in the proposed implementation and a background about the search techniques. It also provides a review about the formal proposed searchable encryption schemes. Then there is chapter three: Methodology and Implementation. The chapter illustrates the structure of the system and the design decisions made in order to achieve the goals. Chapter four: Results and Snapshots. The results of the implementation will be shown and the success of the project will be evaluated in terms of time, space overheads and security.

Last chapter is Conclusion and Recommendation gives a summary and discuss the future work.The Appendix contains the code for the project.

Chapter 2. Background and Literature Review

2.1 Introduction

This chapter presents the key concepts of the searchable encryption problem, as well as background information about the proposed technique. Moreover a brief summary about the previous techniques of searchable encryption in historically order and their advantages and disadvantages.

2.2 Cryptographic Definitions

2.2.1 Block Cipher vs. Stream Cipher

Block Cipher is a symmetric encryption algorithm in which the plaintext is processed by dividing it into equal blocks (typically 64 or 128) then converts each block to ciphertext. Stream Cipher is also symmetric encryption algorithm in which ciphertext output is produced bit-by-bit or byte-by-byte from a stream of plaintext input [3].

2.2.2 AES

Advanced Encryption Standard is National Institute of Standards and Technology specification for encrypting sensitive (unclassified) American federal information. A cipher called Rijndael chosen to be the AES. It's an iterative symmetric block cipher algorithm developed by Belgian cryptographers Joan Daemen of Proton World International and Vincent Rijmen of Kathlieke Universiteit Leuven. Rijndael was designed based on

the following three criteria: Resistance against all known attacks; Speed and code compactness on a wide range of platforms and Design simplicity.

2.2.2.1 AES Structure

In AES the plain text is divided into blocks that consist of 4 columns each is column is 4 bytes and the key is expanded to array of 32-bit words w[i] using the s-box and Rcon. Each 4 words of the key used as a key in each round. Initially the first round key is added (XOR) to the block then the iterative rounds started. The rounds are 9, 11 or 13 depends on the key length. Each round has four different stages, one of permutation and three of substitution such that,

Substitute bytes: Uses an S-box to perform a byte-by-byte substitution of the block.

ShiftRows: permute the row bytes between the columns.

MixColumns: arithmetic substitution.

AddRoundKey: Bitwise XOR with the round key.

The last step is another round only without the mixing column stage [3].

2. 2. 3 Triple DES

After the original Data Encryption Standard (DES) defeated with relative ease. Triple was designed to replace it. At one time, it was the recommended standard and the most widely used symmetric algorithm in the industry[3].

Triple DES uses three individual keys with 56 bits each. The total key length adds up to 168 bits, but experts would argue that 112-bits in key strength is more like it. The encryption algorithm is:

$$ciphertext = Enc_{k_3}(Dec_{k_2}(Enc_{k_1}(plaintext)))$$

2.2.4 Mode of operations

A mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream [3].

2.2.5 Block Cipher Modes of operations

There are five modes of operation for use with symmetric block ciphers electronic codebook mode, cipher block chaining mode, cipher feedback mode, output feedback mode and counter mode [3].

2.2.6 Pseudo-Random Generator (PRG)

Pseudorandom generator is a deterministic algorithm that makes use of mathematical formulas to generate a sequence of numbers that is not statistically random but can pass a number of randomness tests. Those numbers are called pseudorandom numbers.

Any secure block cipher can be used as a secure pseudo-random number generator by running it in counter mode and encrypting serial of numbers using a random key [3].

2. 2. 7 Pseudo-Random Functions (PRF)

Informally, a pseudorandom function is a function which is indistinguishable from a random function. A collection of pseudorandom functions is called a pseudorandom functions family [3].

2. 2. 8 Pseudo-Random Permutation (PRP)

A Pseudorandom permutation is a function which is indistinguishable from a random permutation. A collection of pseudorandom permutations is called a pseudorandom permutations family [3].

2. 2. 9 Blum Blum Shub Pseudo-Random Generator

Blum, Blum and Shub is a cryptographically secure pseudorandom number generator proposed by Lenore Blum, Manuel Blum and Michael Shub that is derived from Michael O. Rabin's oblivious transfer mapping.

Blum Blum Shub takes the form:

$$x_{n+1} = x_n^2 \bmod M$$

It gets its security from the difficulty of computing discrete logarithms[14].

2. 2. 10 Blum Micali Pseudo-Random Generator

Blum and Micali is a provably secure pseudo-random number generator and was created by Manuel Blum [14], who was also involved in the implementation of the Blum Blum Shub generator, and Silvio Micali.

Blum and Micali takes the form:

$$x_{n+1} = g^{x_n} \bmod P$$

It also gets its security from the difficulty of computing discrete logarithms[15].

2. 2. 11 Identity-Based Encryption

Identity-based encryption (IBE) is a public-key cryptographic scheme where the public key of the user is unique information about his identity such as the email. A key authority uses this ID information to generate a public key and its corresponding private key, the sender can use this public key with the ID information (e.g. email) to encrypt the message. The receiver can contact the key authority to get the private key. Compared with typical public-key cryptography, this greatly reduces the complexity of the encryption process since no need to a digital certification and no advance preparation needed.

2. 2. 12 Chosen-plaintext attacks

A chosen-plaintext attack (CPA) where the attacker is allowed to ask for encryptions of multiple messages. The goal of the attacker is to gain more information and reveal the secret key.

2. 2. 13 Semantic Security Against chosen Keyword Attacks

A security definition first introduced by Goh [5] also known as indistinguishability against chosen keyword attacks IND-CKA. It makes sure that the adversary cannot deduce the document's content from its index.

2. 2. 14 One way function

A one way function is a function that is easy to compute but computationally infeasible to inverse. Meaning it is easy to compute y=f(x) but impossible to compute $x=f^{-1}(y)$.

2. 2. 15 Trapdoor function

A trapdoor function is a one way function that can be computationally feasible to compute the inverse if secret information is given.

2. 3 Searching techniques

2. 3. 1 Exact-match vs. sub-match

In the Exact-match searching the user search in a document set for a string, the documents that contain at least one exact instance of the string is retrieved. While in the sub-match searching the documents retrieved are the ones that contain also a sub-string of the searching string.

There are subclasses of the sub-match, such as left-most match which matches the query against the left part of words. Another subclass is the complete sub-string match where the query could be found at any index within another word.

2. 3. 2 Linear Search vs. Pre-processed Index

In linear search each document in the document set is traversed linearly from the beginning to the end in order to match against a given

search query. This process can be very slow when used on large documents, as well as computationally expensive.

Pre-processed Index Search can greatly reduce search time in large documents. An index is created at the storage stage which contains each unique word exist in the document. When a search is performed, it will only retrieve the documents that their indexes contain the query. Obviously, this method increases the disk size required to store each document, as they will need to be stored side-by-side with their index. Also, initial processing time is added by the index creation algorithm. This technique is more suited to applications where the frequency of queries exceeds that of updates.

2.4 Searchable Encryption

Outsourcing data in encrypted form is recommended but we do not want to scarify the functionality for the security. The client wants to be able to easily search the encrypted data or allow others to search it without having access to plaintext or downloading everything then decrypt. Searchable Encryption enables the user to search encrypted keywords without compromising the security of the original data.

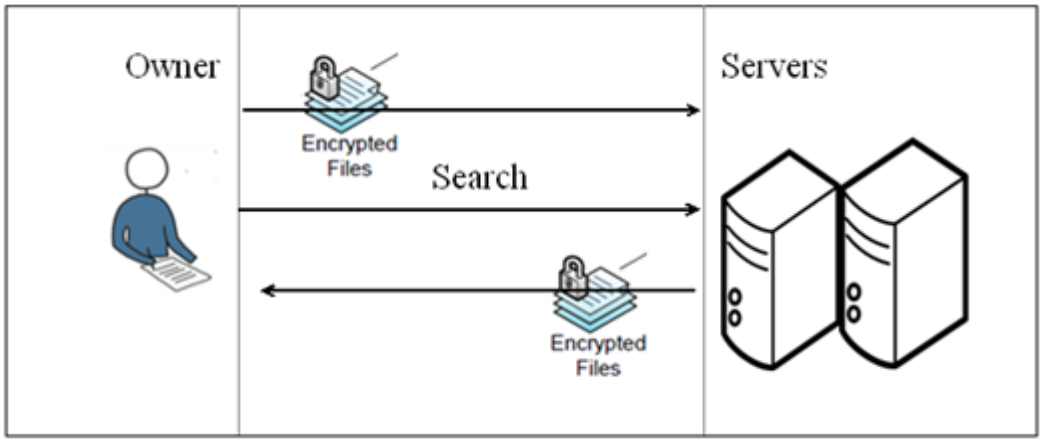


Figure 2.1: Single user setting

The typical participants of a secure search system in outsourced data involve the server, the data owner, and the data user.

The data owner outsources the encrypted dataset, where the data can be encrypted using any secure encryption technique. In a single user setting the owner may need to search over this data (Figure 2.1). In a multiuser setting (Fig. 2.2) the

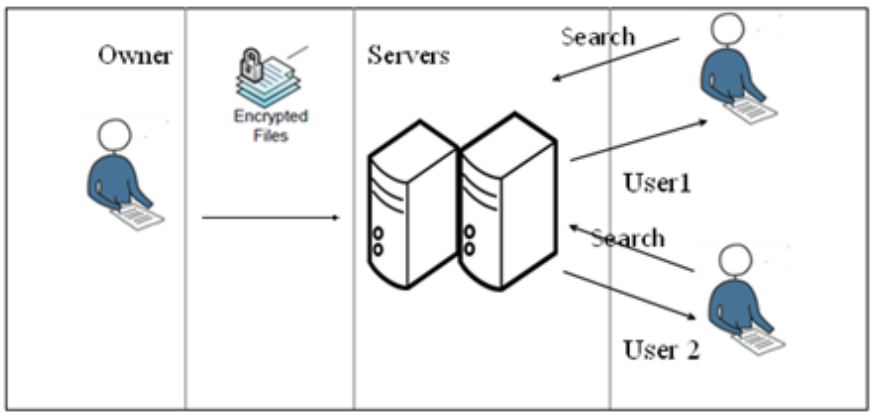


Figure 2.2: Multiuser setting

owner outsourced the encrypted dataset and another authorized user or users want to search over it. The user who wants to search first generates a trapdoor with the keyword of interest or requests such trapdoor by sending a set of intended keywords to the data owner. Then the data user submits the trapdoor to the server. The server will execute the search program with the trapdoor as the input and the results will send back to the user.

The data owner outsources the encrypted dataset, where data can be encrypted using any secure encryption technique. In a single user setting the owner its self may need to search over this data (Figure 2.1). In a multiuser setting (Figure 2.2) the owner outsourced the encrypted dataset and another authorized user or users want to search over it. The user who wants to search first generates a trapdoor with the keyword of interest or requests such trapdoor by sending a set of intended keywords to the data owner. Then the data user submits the trapdoor to the server. The server will execute the search program with the trapdoor as the input and the results will send back to the user.

2.4.1 Symmetric Key Based Search

The data owner encrypts his data using a symmetric encryption before storing it in the server. Searchable Symmetric encryption (SSE) provides the owner or a user with a private key the ability to search for keywords over the encrypted data.

2. 4. 2 Public key Based Search

Public key Based Search relies on public key encryption such that the user who encrypts the data and stores it in the server is not the owner of the decryption key but only the owner of the private key can perform a search for a keyword. Consider the scenario where Bob encrypt a message to Alice using Alice public key and send it to her, Alice want the server to redirect the messages which including specific words without giving the server her private key. Public key searchable encryption define the mechanism that enables Alice to provide a key to the server that enables the server to search for the specific word the message without learning anything else about the message.

2. 4. 3 Adaptive vs. Non adaptive

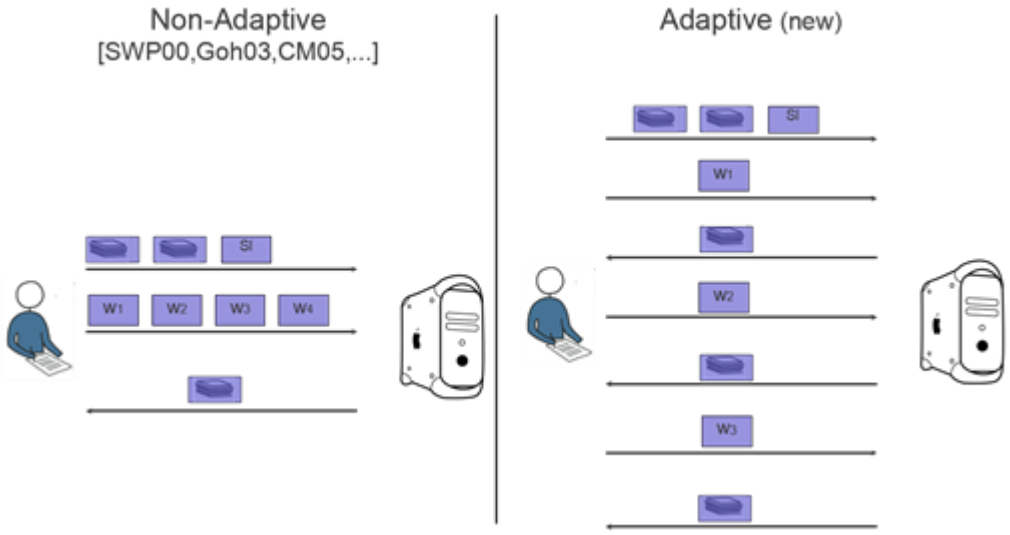


Figure 2.3: vs. non adaptive search [16]

The non-adaptive search (Figure 2.3), only considers adversaries that make their search queries without taking into account the trapdoors and

search outcomes of previous searches while the adaptive search considers adversaries that choose their queries as a function of previously obtained trapdoors and search outcomes.

2.4.4 Searchable Encryption Design Goals

When constructing a searchable encryption scheme, we should consider that the scheme should be practical, the communication overhead should be as less as possible and the computation on both server and client should also be minimized. Multi-user setting is always an advance.

2.4.5 Searchable Encryption Security Requirement

Many security requirements are defined for searching over encrypted data. Since the server is not trusted, it should not be able to distinguish between documents from coded query, determine document contents, see search keyword or learn anything more than result. It certainly should not be able generate coded query. Even simple information such as the number of documents containing the keyword or the occurrence count of a keyword in a document can be used by the attacker to reverse-engineer the keyword in a trapdoor. Also the trapdoor although it generated using cryptographic technique to protect the keyword, the server can use other side channel attacks such as frequency analysis attack to identify the searched keyword.

The trapdoor should be generated in a random manner so that the attacker cannot know whether they contain the same set of keywords. This can further compromise the keyword privacy in that it allows the server to

accumulate frequencies of different search requests with respect to different keyword(s).

2. 4. 6 Secure Index

The secure index is a data structure that points the documents which contain the keyword in a search operation only if the user possess the trapdoor of the keyword which can only be generated using the secret key. Otherwise the index leaks no information about its content.

2.4.6.1 Notation and Preliminaries

Document collections: Let $D = (d_1, d_2, d_3, \ldots, d_n)$ be the set of all documents, $\Delta = (w_1, w_2, w_3, \ldots, w_d)$ be a dictionary of d words with 2^{Δ} be the set of all possible documents with words in Δ. We denote by $id(D)$ the identifier of document D, where the identifier can be any string that uniquely identifies a document such as a memory location. ***D(w)*** denoted for the lexicographically ordered list consisting of the identifiers of all documents in ***D*** that contain the word w.

The distinct keywords: $\delta(D) \subset \Delta$ is the set of distinct keywords in the document collection $D \in$ ***D.***

Symmetric encryption: A symmetric encryption scheme is a set of three polynomial-time algorithms $SKE = (Gen, Enc, Dec)$ such that Gen takes a security parameter k and returns a secret key K. Enc takes a key K

and a message m and returns a ciphertext c; Dec takes a key K and a ciphertext c and returns m if K was the key used to cipher c.

Broadcast encryption: $BE = (Gen, Enc, Add, Dec)$ such that Gen is a probabilistic algorithm that takes as input a security parameter k and outputs a master key mk. Let U be the set of all possible user identifiers for BE, Enc is a probabilistic algorithm that takes as input a master key mk, a set of users $G \in U$ and a message m, and outputs a ciphertext c. Add is a probabilistic algorithm that takes as input a master key mk and a user identifier $u \in U$ and outputs a user key uk_u. Dec is a deterministic algorithm that takes as input a user key uk_u and a ciphertext c and outputs either a message m or the failure symbol $\perp$. The broadcast encryption scheme is secure if its ciphertexts leak no useful information about the message to any user not in G.

Searchable Symmetric Encryption: An index-based SSE scheme over a dictionary Δ is a collection of five polynomial-time algorithms $SSE = (Gen, Enc, Trpdr, search, Dec)$ such that,

$K \leftarrow \mathsf{Gen}(1^k)$: *is a probabilistic key generation algorithm that is run by the user to setup the scheme. It takes as input a security parameter* k*, and outputs a secret key* K*.*

$(I, \boldsymbol{c}) \leftarrow \mathsf{Enc}(K, \mathbf{D})$: *is a probabilistic algorithm run by the user to encrypt the document collection. It takes as input a secret key* K *and a document collection* $\mathbf{D} = (D_1, \ldots, D_n)$*, and outputs a secure index* I *and a sequence of ciphertexts* $\boldsymbol{c} = (c_1, \ldots, c_n)$*. We sometimes write this as* $(I, \boldsymbol{c}) \leftarrow \mathsf{Enc}_K(\mathbf{D})$*.*

$t \leftarrow \mathsf{Trpdr}(K, w)$: *is a deterministic algorithm run by the user to generate a trapdoor for a given keyword. It takes as input a secret key* K *and a keyword* w*, and outputs a trapdoor* t*. We sometimes write this as* $t \leftarrow \mathsf{Trpdr}_K(w)$*.*

$X \leftarrow \mathsf{Search}(I, t)$: *is a deterministic algorithm run by the server to search for the documents in* $\mathbf{D}$ *that contain a keyword* w*. It takes as input an encrypted index* I *for a data collection* $\mathbf{D}$ *and a trapdoor* t *and outputs a set* X *of (lexicographically-ordered) document identifiers.*

$D_i \leftarrow \mathsf{Dec}(K, c_i)$: *is a deterministic algorithm run by the client to recover a document. It takes as input a secret key* K *and a ciphertext* c_i*, and outputs a document* D_i*. We sometimes write this as* $D_i \leftarrow \mathsf{Dec}_K(c_i)$*.*

[16]

2.4.6.2 Index-based SSE scheme

Given the encrypted document collection, an index-based SSE scheme to search over consists of five polynomial-time algorithm [16] such that,

K←Gen(1^k): is a probabilistic key generation algorithm run by the user generate the secret key K *from the security parameter k.*

*(I,c)←Enc(*K*,D): is a probabilistic algorithm run by the user that takes the secret key* K *and the document as input, and output ciphertext and secure index I.*

*t←Trpdr(*K*,w): is a deterministic algorithm run by the user to generate a trapdoor for a given keyword. It takes as input a secret key* K *and a keyword w, and outputs a trapdoor t.*

X*←Search(I,t): is a deterministic algorithm run by the server to search for the documents in that contain a keyword w. It takes as input an encrypted index* I *for a data collection D and a trapdoor t and outputs a set* X *of document identifiers.*

Di←Dec(K; ci): is a deterministic algorithm run by the client to recover a document. It takes as input a secret key K and a ciphertext ci, and outputs a document Di.

2.5 Related work

There has been several works to solve the problem of searching over encrypted data whether it's public key or private encryption. Ostrovsky and Goldreich work [7,8] on oblivious RAMs any type of queries can be performed with the strongest levels of security, namely the server only learns the size of the document collection. But it's less efficient in practice due to a big overhead in terms of bandwidth.

In an effort to reduce the overhead in the oblivious RAMs, Song et al. publish a paper titled Practical Techniques for Searches on Encrypted Data [4]. They developed a set of algorithms that allow searches over encrypted data and proof their model's security. Their model has complexity of O (n) for each document and relatively little space overhead.

Since Song et al.'s seminal work [4], searchable encryption has drawn a lot of attention. In 2004, Eu–Jin Goh et al. [5] and Chang and Mitzenmacher [6] address that the previous model not only can cause over head in large set of data, but also the underlying plaintext distributions is vulnerable to statistical attacks. Goh et al. [5] introduce a secure search scheme where queries can be executed over secure indexes rather than encrypted data themselves. Their scheme requires linear search time but can results in false positives. Later several papers published with the same concept. Goh et al. [5] introduce a security definition for SSE called in-distinguishability against chosen-keyword attacks (IND2-CKA).

Chang and Mitzenmacher [6] proposed a construction with linear search time too and without false positives, their solution also is independent of the encryption method chosen for the remote files and achieves forward privacy. They also introduce a security definition for SSE.

Regarding searchable encryption in public key scheme Boneh et al. proposed Public Key Encryption with Keyword Search (PEKS) scheme [9] in . Their scheme built based on a variant of the Computational Diffie-Hellman problem. It requires a secure channel between the receiver and the server so that the trapdoor not been send expose. Since constructing a secure channel is costly this solution may be not efficient in some cases.

Baek et al. proposed a secure channel free public key encryption with keyword search scheme [10] (SCF-PEKS). Their construction is based on a mathematical concept called bilinear pairing .In the opinion of Rhee, Park, Susilo and Lee Baek et al. scheme might be attacked by using a keyword-guessing attack if the attacker captures the trapdoor Therefore, Rhee et al.[11] enhances the model of Baek et al. to prevent such attacks and defines the "trapdoor in distinguishability".

In tern of security definitions, Goh [5] introduced IND1-CKA definition where a secure scheme generates indexes that appear to contain the same number of words for equal size documents. This means that given two encrypted documents of equal size and an index, Adversary cannot decide which document is encoded in the index. The trapdoors to be secure, since it is not required by all applications of secure indexes

Chang and Mitzenmacher [6] introduced a better IND-CKA in the sense that an adversary cannot even distinguish indexes from two unequal size documents In addition, Chang and Mitzenmacher tried to protect the trapdoors with their security definition. Unfortunately, their formalization of the security notion can be satisfied by an insecure SSE scheme.

Later, Goh introduced the IND2-CKA security definition, Given access to an index, the adversary (i.e., the server) is not able to learn any partial information about the underlying documents that he cannot learn from using a trapdoor that was given to him by the client, and this holds even against adversaries that can convince the client to generate indexes and trapdoors for documents and keywords chosen by the adversary (i.e., chosen-keyword attacks).

IND2-CKA does not explicitly require that trapdoors to be secure since this is not a requirement for all applications of secure indexes. One of which is searchable encryption. Important to note that different keyword requests may lead to the same search outcome.

As mentioned before several schemes were introduced in this topic. Oblivious RAMs can solve the problem with its entire requirement but with high complexity and Song et al. decreased the complexity but the security model was weaker. Goh et al. proposed a better security in their scheme by introducing the IND2-CKA as a security definition but this definition does not require that the trapdoor be secure. Even so if the trapdoor is secure, that does not imply that the adversary cannot recover the word being queried. Chang and Mitzenmacher also introduced a security definition for

SSE that require a secure trapdoor but can be trivially satisfied by a scheme that in secure. So there is still a need for a practical and secure scheme without false positives or communication overhead.

Chapter 3. Methodology

3.1 Introduction

In this chapter, searches in one communication round were trying to be achieved, with high efficiency in respect to time and space. Furthermore the security for the indexes and the trapdoor need to be adequate. The chapter will also discuss the technical information about the system, including the system and software design decisions taken. As well as the structure of the system, showing the various modules and database organization.

3.2 Proposed method

An implementation of an adversarial models for SSE will be introduced which referred to as non-adaptive (SSE-1), only considers adversaries that make their search queries without taking into account the trapdoors and search outcomes of previous searches.

3.2.1 The model

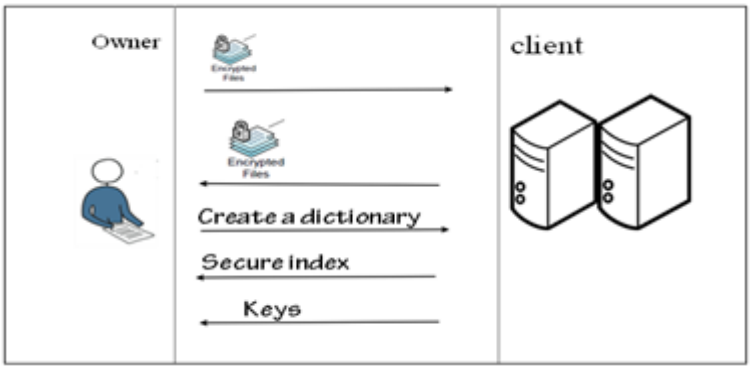


Figure 3.1: The client side setting

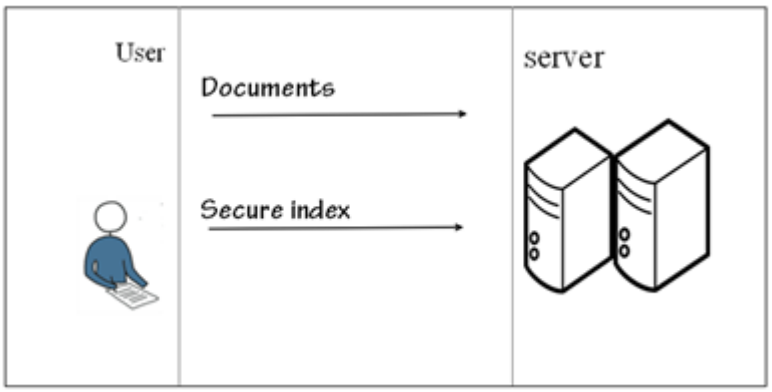


Figure 3.2: The server side setting

The model starts with the user (Figure 3.1), the user should first encrypt the documents, the system then construct an index for these documents and build a dictionary and a secure index. In the server (Adversary) (Figure 3.2) the encrypted document will be uploaded and so the secure index. To search (Figure 3.3) the user should create a trapdoor which will be used to find the list of the documents that contain the word.

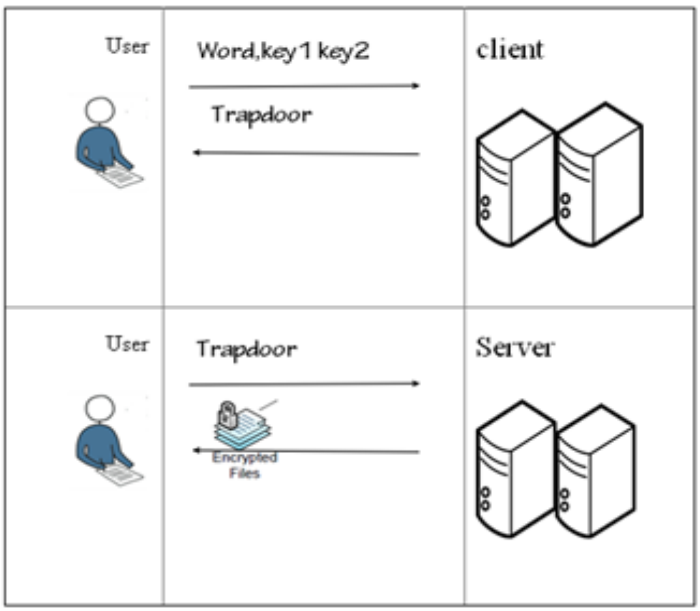


Figure 3.3: Search setting

A number of data structures will be used, including arrays, linked lists and lookup tables. Given an array A, if A[i] = x, then $addr_A(x) = i$. In addition, a linked list L of *n* nodes that is stored in an array A is a sequence of nodes $N_i = \langle v_i, addr_A(N_{i+1}) \rangle$, where $1 \le i \le n$, and where v_i is an arbitrary string and $addr_A(N_{i+1})$ is the memory address of the next node in the list. We denote by #L the number of nodes in the list L.

The construction of the mode consists of a client side and a server side. The client side in which the client should first encrypt the data using AES structure . The database for the encrypted documents will be created and each document given an id $id(D)$. Then a table will be created as a dictionary Δ for these documents as well as a table that consists of word column and other columns for the corresponding documents that contained the word. This table helps to create the secure index which constructed next. It consists of two data structures:

A: an array in which, for all $w \in \delta(D)$, we store an encryption of the set $D(w)$.

T: a look-up table in which, for all $w \in \delta(D)$, we store information that enables one to locate and decrypt the appropriate element from **A**.

For each $w_i \in \delta(D)$ I created a linked list L_i. The nodes in L_i is the identifiers of the documents that contain the word w_i, $id(D(w_i))$. These nodes stored in the array ***A*** and permuted randomly by Blum Blum Shub Pseudo Random Generator denoted by φ. Each node contains - beside the identifier -a pointer to the next node in L_i in respect to A and the key

used to encrypt it. This node will be encrypted (all these keys are randomly generated). Array A then padded. The set of remaining entries is set to random values of the same size as the existing nodes in A. (Figure 3.4).

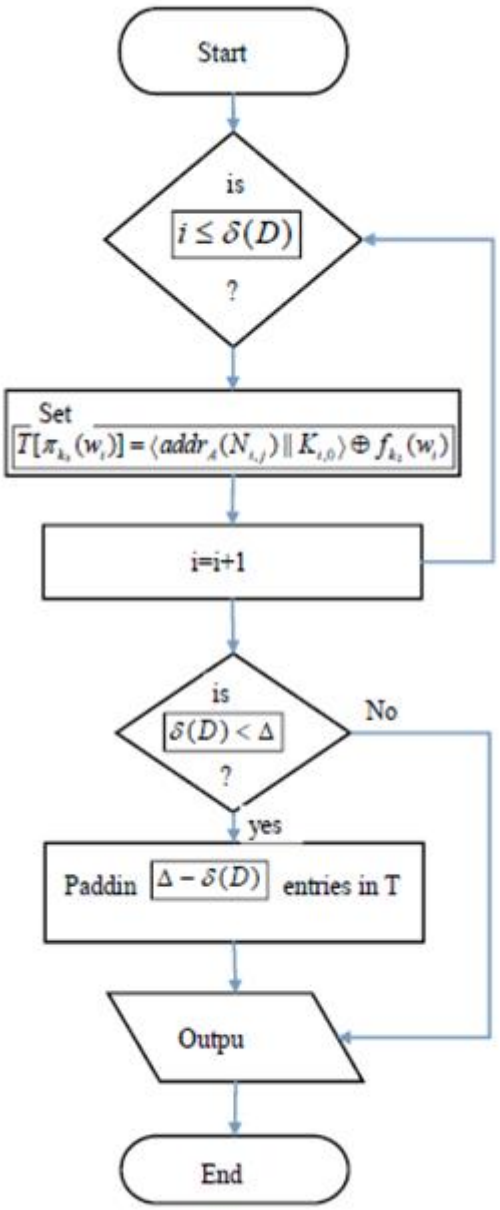
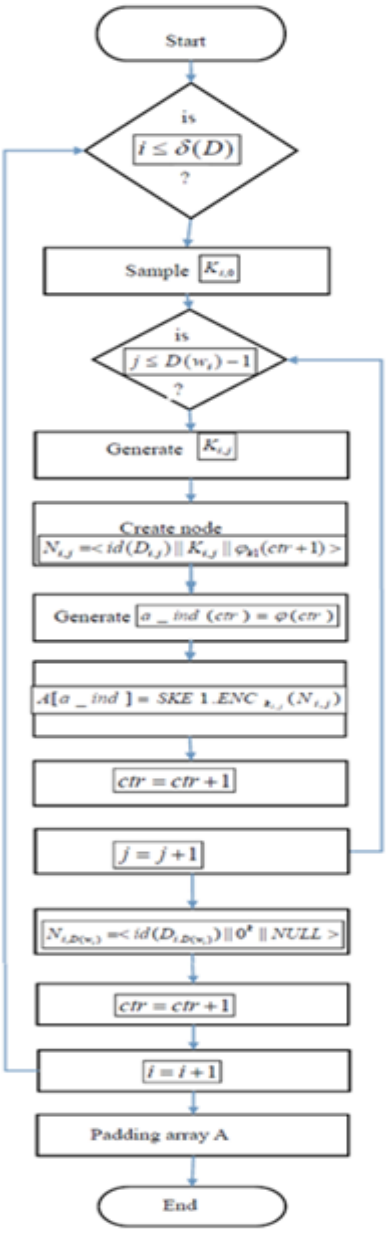


Figure 3.4: Building array A

The lookup table ***T*** (Figure 3.5)**,** has entries such that for each $w_i \in \delta(D)$ there is an entry consist of *<address,value>.* The *value* field contains the address of the first node of L_i in A and the key to encrypt this node.



.

Figure 3.5: Building lookup table T

This field is encrypted using the output of a pseudorandom function Blum Micali denoted by f. The *address* field is an index for the lookup table. All the entries are also permuted randomly using AES in counter mode. Then ***T*** also padded.

As I mentioned above the scheme has a client side and a server side, ***A*** and ***T*** generated by the client and then stored in the server along with the encrypted documents (Figure 3.6).

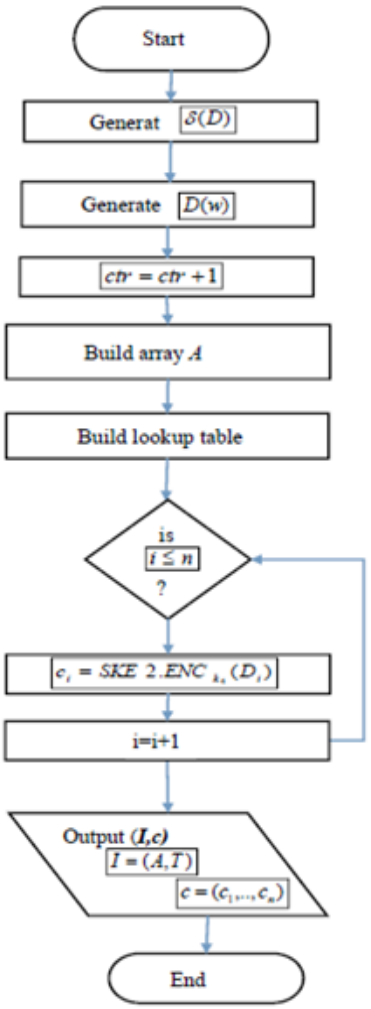


Figure 3.6: Client side flow chart

To search for w_i (Figure 3.7), the system computes the decryption key and the address for the corresponding entry in T and sends them to the server. The server locate the entry in T and decrypt it, got the address and the key for the first node. And since each node contains a pointer for the next node and the key to decrypt it, the server will be able to locate and decrypt all the nodes in L_i and get all the identifiers.

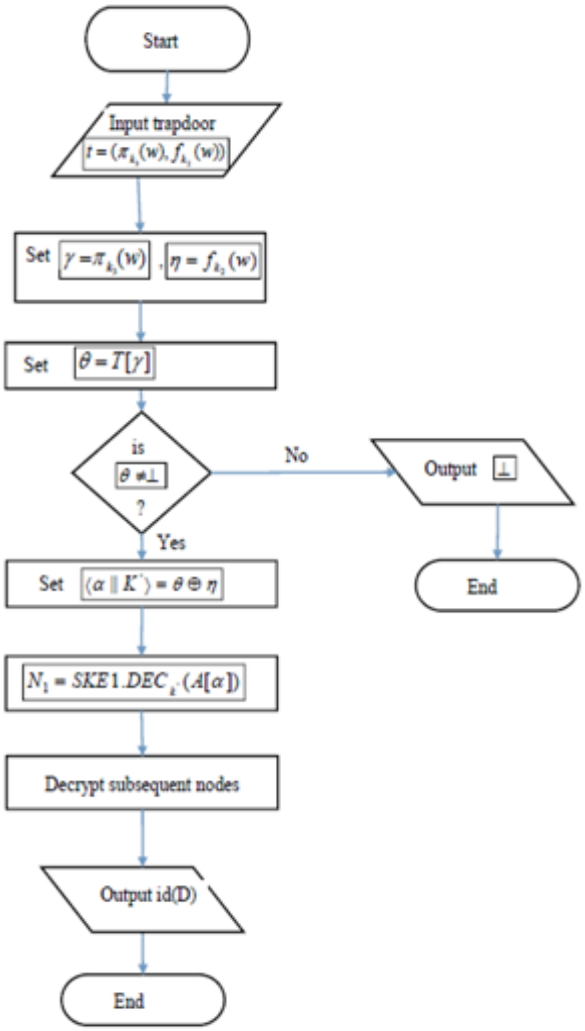


Figure 3.7 : Server side flowchart

3. 2. 2 System Modules

The modules below used to implement the algorithm (figure 3.8) of the proposed non- adaptive SSE.

$\mathsf{Gen}(1^k)$: sample $K_1, K_2, K_3 \xleftarrow{\$} \{0,1\}^k$, generate $K_4 \leftarrow \mathsf{SKE2.Gen}(1^k)$ and output $K = (K_1, K_2, K_3, K_4)$.

$\mathsf{Enc}_K(\mathbf{D})$:

Initialization:

1. scan $\mathbf{D}$ and generate the set of distinct keywords $\delta(\mathbf{D})$
2. for all $w \in \delta(\mathbf{D})$, generate $\mathbf{D}(w)$
3. initialize a global counter $\mathsf{ctr} = 1$

Building the array A:

4. for $1 \leq i \leq |\delta(\mathbf{D})|$, build a list L_i with nodes $\mathtt{N}_{i,j}$ and store it in array $\mathtt{A}$ as follows:
   (a) sample a key $K_{i,0} \xleftarrow{\$} \{0,1\}^k$
   (b) for $1 \leq j \leq |\mathbf{D}(w_i)| - 1$:
   - let $\mathsf{id}(D_{i,j})$ be the j^{th} identifier in $\mathbf{D}(w_i)$
   - generate a key $K_{i,j} \leftarrow \mathsf{SKE1.Gen}(1^k)$
   - create a node $\mathtt{N}_{i,j} = \langle \mathsf{id}(D_{i,j}) \| K_{i,j} \| \psi_{K_1}(\mathsf{ctr}+1) \rangle$
   - encrypt node $\mathtt{N}_{i,j}$ under key $K_{i,j-1}$ and store it in $\mathtt{A}$:

$$\mathtt{A}[\psi_{K_1}(\mathsf{ctr})] \leftarrow \mathsf{SKE1.Enc}_{K_{i,j-1}}(\mathtt{N}_{i,j})$$

   - set $\mathsf{ctr} = \mathsf{ctr} + 1$
   (c) for the last node of L_i,
   - set the address of the next node to NULL: $\mathtt{N}_{i,|\mathbf{D}(w_i)|} = \langle \mathsf{id}(D_{i,|\mathbf{D}(w_i)|}) \| 0^k \| \mathsf{NULL} \rangle$
   - encrypt the node $\mathtt{N}_{i,|\mathbf{D}(w_i)|}$ under key $K_{i,|\mathbf{D}(w_i)|-1}$ and store it in $\mathtt{A}$:

$$\mathtt{A}[\psi_{K_1}(\mathsf{ctr})] \leftarrow \mathsf{SKE1.Enc}_{K_{i,|\mathbf{D}(w_i)|-1}}(\mathtt{N}_{i,|\mathbf{D}(w_i)|})$$

   - set $\mathsf{ctr} = \mathsf{ctr} + 1$
5. let $s' = \sum_{w_i \in \delta(\mathbf{D})} |\mathbf{D}(w_i)|$. If $s' < s$, then set the remaining $s - s'$ entries of $\mathtt{A}$ to random values of the same size as the existing s' entries of $\mathtt{A}$

Building the look-up table T:

6. for all $w_i \in \delta(\mathbf{D})$, set $\mathtt{T}[\pi_{K_3}(w_i)] = \langle \mathsf{addr}_{\mathtt{A}}(\mathtt{N}_{i,1}) \| K_{i,0} \rangle \oplus f_{K_2}(w_i)$
7. if $|\delta(\mathbf{D})| < |\Delta|$, then set the remaining $|\Delta| - |\delta(\mathbf{D})|$ entries of $\mathtt{T}$ to random values of the same size as the existing $|\delta(\mathbf{D})|$ entries of $\mathtt{T}$

Preparing the output:

8. for $1 \leq i \leq n$, let $c_i \leftarrow \mathsf{SKE2.Enc}_{K_4}(D_i)$
9. output $(I, \mathbf{c})$, where $I = (\mathtt{A}, \mathtt{T})$ and $\mathbf{c} = (c_1, \ldots, c_n)$

$\mathsf{Trpdr}_K(w)$: output $t = (\pi_{K_3}(w), f_{K_2}(w))$

$\mathsf{Search}(I, t)$:

1. parse t as (γ, η), and set $\theta \leftarrow \mathtt{T}[\gamma]$
2. if $\theta \neq \perp$, then parse $\theta \oplus \eta$ as $\langle \alpha \| K' \rangle$ and continue, otherwise return $\perp$
3. use the key K' to decrypt the list L starting with the node stored at address α in $\mathtt{A}$
4. output the list of document identifiers contained in L

$\mathsf{Dec}_K(c_i)$: output $D_i \leftarrow \mathsf{SKE2.Dec}_{K_4}(c_i)$

Figure 3.8 : A non-adaptively secure SSE scheme [16]

3.2.2.1 Common Modules

xor_this: To encrypt and decrypt the lookup table

TripleDES: Encryption module that uses the triple DES algorithm to encrypt and decrypt the nodes of the lists in array A.

3.2.2.2 Client side Modules

Keygen Module: A key generator that takes a security parameter as a seed and use it to generate a secret keys which used to encrypt the documents. The module uses openssl_random_pseudo_bytes to generate a cryptographically secure string of pseudo-random bytes.

random_int: A PHP function to generate cryptographic random integers that are suitable for use to encrypt the nodes of the linked list and the entries of the lookup table.

Permutation Modules**:** To randomly shuffle the linked lists array, Blum Blum Shub pseudo-random generator has been used. For the lookup table permutation, Blum Micali pseudo-random generator has been used.

Lookup Table Encryption Module: AES in counter mode AES-ctr function was used as a pseudo-random generator to xor the entries of the table with the output of this function.

AES Module: AES encryption algorithm was chosen to encrypt the client's document.

The Client Side Database:

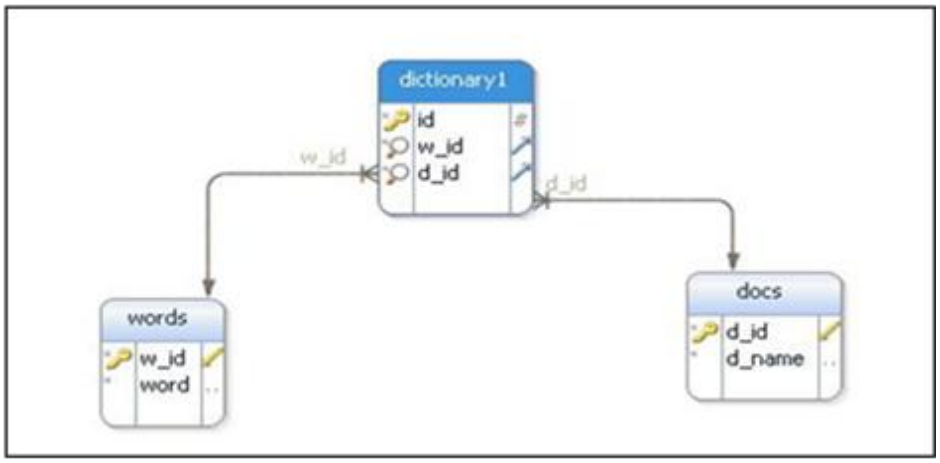


Figure 3.9 : The client database scheme

Three tables was created (figure 3.9): Documents table which contains the documents and their indexes, Words table contains all the distinct words in the documents and finally the Dictionary1 table, each record of this table contains a word identifier and the identifiers of all the documents that contain the word.

3.2.2.3 Server side Modules

Search: The search module is simply parsing the input to access a specific record in the lookup table, then xor the output of a pseudo-random function *AES-ctr* to decrypt the entry. The key extracted will be used to decrypt the node of the linked list in array *A*.

The Server Side Database: Only one table for the documents and their identifiers.

3.3 Technical Information

3.3.1 Program Reliability

The user may be inexperienced and have little or no access to technical support so, system reliability is important. In general, program reliability can be achieved by avoiding the introduction of faults and bugs and by including fault tolerant facilities in the system. Defensive programming [12] involves incorporating checks for faults and fault recovery code in the program.

Defensive programming is approach to improving software quality by making the software behave in a predictable manner despite unexpected input or user actions. In this project, the validation technique used to prevent the user from uploading wrong files type and to ensure that all required values are provided. JQuery libraries used to implement this technique.

3.3.2 Reusability

A modular approach was adapted to the development of the PHP files (the main engine of the system). Care was taken with naming conventions. Individually reusable components were extracted and imported into individual modules.

3.3.3 Design Decisions

Main reason for choosing a web interface is that it is a known interface. With the continued growth of the Internet, people are becoming more accustomed to the look and feel of web pages.

3.3.4 PHP 5

PHP has several features that lead me to choose it for this project. First of all it has support for object-oriented programming, it has also a fast load time results in faster site loading speeds. The code in PHP runs much faster than ASP because it runs in its own memory space while ASP uses an overhead server and a COM based architecture.

In addition, PHP is an open source, with all of the advantages of the open source there will be no embargo, so there is no problem of accessing sources to improve the design or for further researches to design an adaptive model.

Most tools associated with the PHP are not just open source software, it's also free. And for the hosting its not expensive. ASP programs need to run on Windows servers with IIS installed. Hosting companies need to purchase both of these components in order for ASP to work, this often results in a more expensive cost for monthly hosting services. On the other hand, a PHP would only require running on a Linux server, which is available through a hosting provider at no additional cost.

Another essential reason is that most of the popular cloud storage providers such as Dropbox are using PHP so, it won't be complicated to integrate this system to those providers.

With reference to the database, PHP has a lightweight and consistent interface for accessing databases (PDO extension). It is also flexible for database connectivity. It can connect to several databases, the most commonly used is the MySQL.

3. 3. 5 MySQL

Mysql is also open source, with a community version available for free download. A little technical know-how is enough to get MySQL set up and configured on commodity hardware at very low cost. Furthermore MySQL's AB provides support and maintenance services such as code updates and bug fixes.

MySql is flexible and scalable, so you can start with small database and increase the size and the performance any time you need it. MySQL can be configured to run tiny embedded applications using a footprint no larger than one megabyte, or scale it up to handle many terabytes of data. One way MySQL achieves this scalability is through a popular feature called stored procedures, mini, pre-compiled routines that reside outside of the application. These procedures are stored and run on the database server to reduce the processing footprint on the client and make the most of processing power, since the database server is usually faster. Stored

procedures aren't unique to MySQL, but the recent addition of this feature set makes the database that much more attractive.

3. 3. 6 Apache

Because of several reasons, Apache became the most used web server over the internet . First, it’s is a freely available Web server that is distributed under an "open source" license and from Version 2.0 it runs on most platforms such as UNIX-based operating systems, UNIX/POSIX-derived systems, on AmigaOS, and on Windows .

Second Apache server has a high performance. It can handle a large files (namely, greater than 2GB) on 32-bit platforms and can also serve high concurrent requests with a high efficiency. The reason is that Apache support proxy load balancing and Multi-Processing allow the server to serve more concurrent requests.

Chapter 4. Results & Analysis

4. 1 Introduction

In this chapter, the results will be shown and the success of the project will be evaluated. This can be measured using the performance of the implementation, in terms of time, space overheads and security.

4. 2 The results

4. 2. 1 Client side procedures

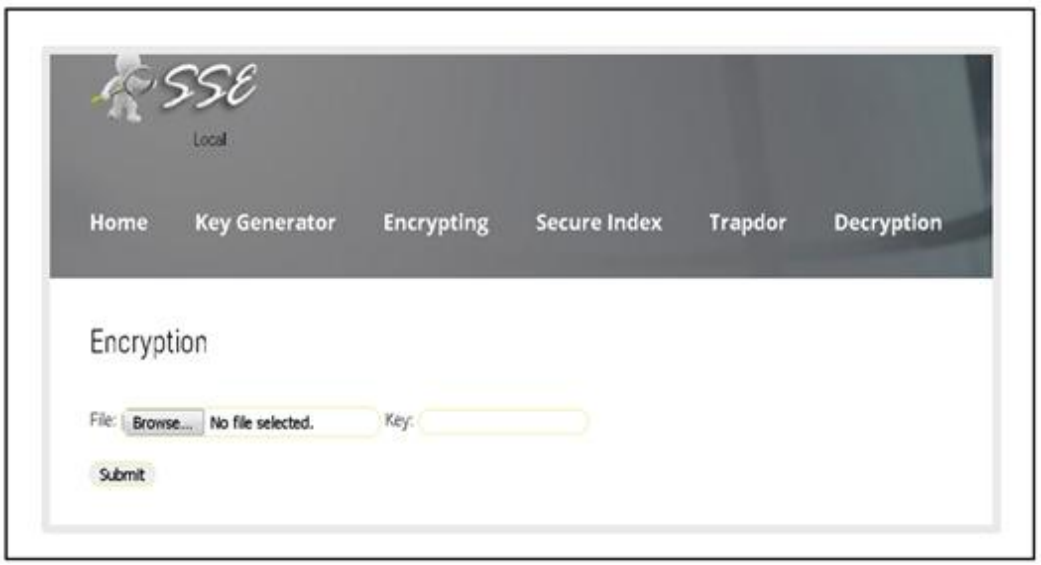


Figure 4.1 : System screens

Figure 4.13 explain the procedures in the client side which start by creating the key then use this key to encrypt the documents. The system then will create both the secure index and the trapdoor keys. Figure 4.1 is

shows the system screen for the first procedure after the key generations. It shows also the tabs of the other procedures.

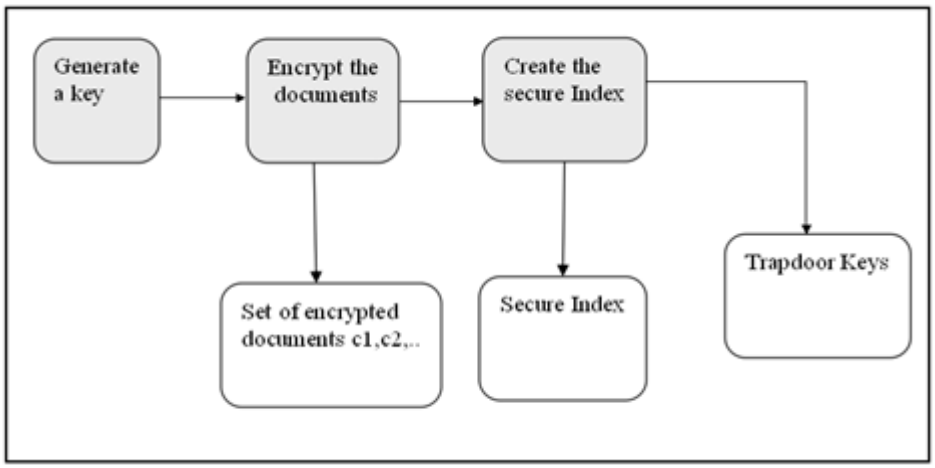


Figure 4.2 : Client side procedures

4. 2. 2 Server side procedures

Figure 4.3 : Server side screen

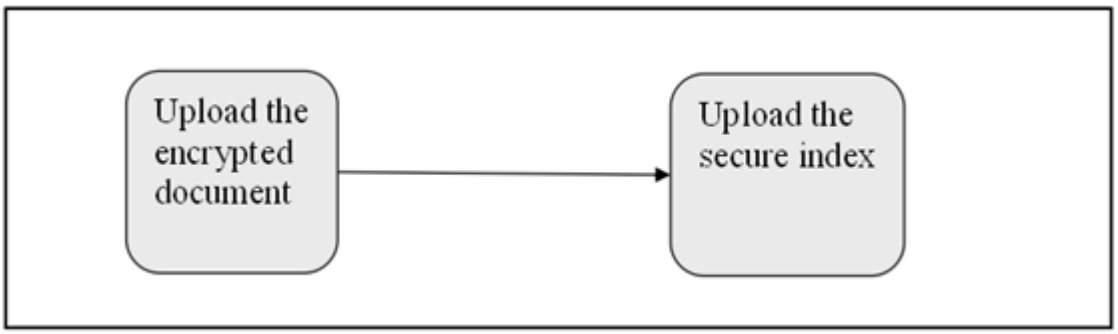


Figure 4.4 : Server side procedures

Figure 4.3 is a screen from the server side shows the procedures tabs and the search tab. Figure 4.4: illustrates the procedures in the server side.

4. 2. 3 Search procedures

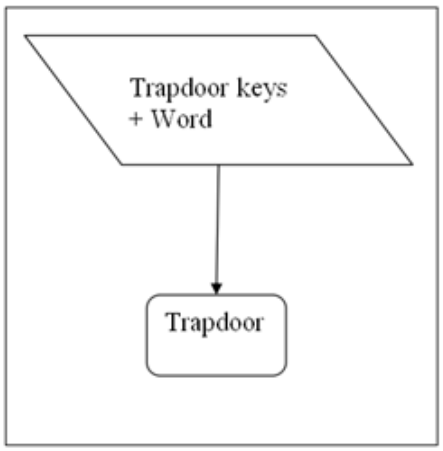


Figure 4.5 : Search client side procedures

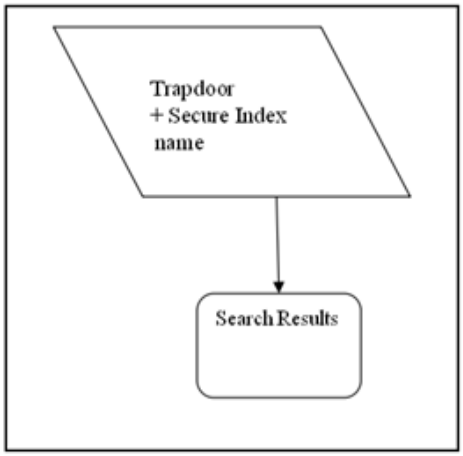


Figure 4.6 : Search Server side procedures

The search process starts at the client where the key word entered with the trapdoor keys to generate the trapdoor (Figure 4.5). Then the trapdoor should be sent to the server with the secure index name to retrieve the search results. Figure 4.6 shows the system screen for the search.

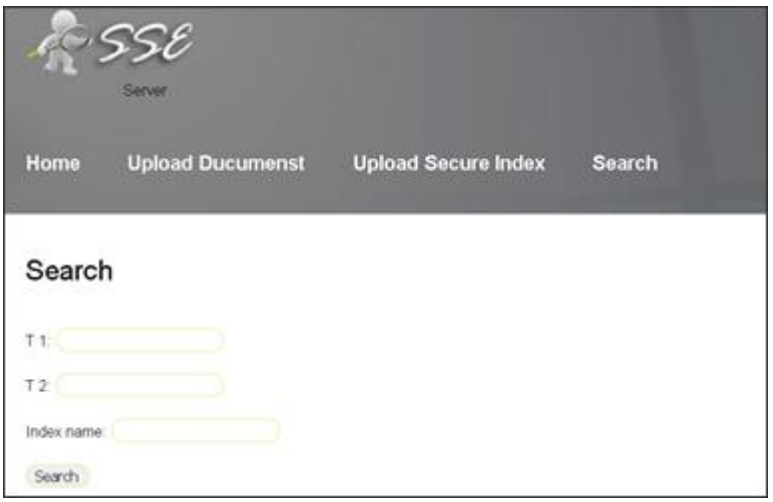


Figure 4.7 : Search screen

4. 3 Evaluation Decisions

To make it simple to calculate, the system will be evaluated without taking into account the size of the retrieved documents.

4. 3. 1 Scheme evaluation

In this section, the complexity will be detailed. The search computation, number of round and the communication cost will be discussed in comparison with the precious SSE schemes. Then, the space needed in the server will be declared.

4. 3. 2 Search

While the construction in [5] performs searches in one round, it can induce false positives, which is not the case for this construction. Additionally, all the constructions in [5, 7] require the server to perform an amount of work that is linear in the total number of documents in the collection. This construction needs one round per query to access the lookup table directly the retrieve the documents.

In table 4.1, the server computation row shows the costs per returned document for a query. All previous work requires an amount of server computation at least linear with the number of documents in the collection, even if only one document matches a query. In contrast, in this construction the server computation is constant per each document that matches a query, and the overall computation per query is proportional to the number of documents that match the query.

In regard to the communication overhead we need only to communicate with the server one time per query.

4. 3. 3 Space

The scheme in [7][8] needs linear logarithmic storage, while the other considered schemes need linear storage,. No need for more than the encrypted documents size space.

4. 3. 4 Security

The SSE scheme in this project does not leak anything beyond the outcome and the pattern of a search. Unlike the notion of IND2-CKA [5], this scheme does not give the adversary access to the encryption algorithm of the documents or the trapdoor oracle. .It provides security for the indexes by encryption and for the trapdoors by hiding the keyword and sends instead a reference. The scheme leaks the access pattern but nothing else.

Table 4.1 : Properties and performance (per query) of various SSE schemes. n denotes the number of documents in the collection. Consider only the overhead and omit the size of the retrieved documents, which is the same for all schemes. For server computation, we show the costs per returned document.[16]

Properties	oblivious RAM	oblivious RAM **light**	Song,	**Goh**	Chang and M. Mitzenmacher	**SSE-1**
server computation	$O(\log^3 n)$	$O(\sqrt{n})$	$O(n)$	$O(n)$	$O(n)$	$O(1)$

server storage	$O(n \log n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
number of rounds	$\log n$	2	1	1	1	1
communication	$O(\log^3 n)$	$O(\sqrt{n})$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
adaptive	Yes	yes	no	no	no	no

` [1]

Chapter 5. Conclusion and Recommendation

5.1 Conclusion

This project focus on the development of a secure non-adaptive searchable symmetric encryption scheme that allow the client to store encrypted data on an un-trusted server and still be able to securely perform server-side searches within the documents without needing to download and decrypt these documents . In Chapter 5, the system was shown to be efficient and practical enough to be used within industry and that the performance overhead is optimal. Issues with previous attempts were pointed out and a new security requirement is met so, the system guarantees security even when users perform more realistic searches.

5.2 Recommendation

There are two areas in which this project could be extended. First implementing the adaptive searchable symmetric encryption in which the current search depends on the previous search. The implementation should be secure under the same security definitions as this one. Second implementing a multi-user SSE, which extends the searching ability to parties other than the owner.

Reference

1. P. Mell and T. Grance, “Draft nist working definition of cloud computing,” http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, June 2009.

2. M. Armbrust, “Above the clouds: A berkeley view of cloud computing,” http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html, February 2009.

3. Stallings, William. “Cryptography and network security: principles and practices. 4th ed.” Upper Saddle River, N.J.: Pearson/Prentice Hall, 2006.

4. D. Song, D. Wagner, and A. Perrig. Practical techniques for searching on encrypted data. In IEEE Symposium on Research in Security and Privacy, pages 44{55. IEEE Computer Society, 2000.

5. E-J. Goh. Secure indexes. Technical Report 2003/216, IACR ePrint Cryptography Archive, 2003.See http://eprint.iacr.org/2003/216.

6. Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In Applied Cryptography and Network Security (ACNS '05), volume 3531 of Lecture Notes in Computer Science, pages 442{455. Springer, 2005.

7. O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. Journal of the ACM, 43(3):431{473, 1996.

8. R. Ostrovsky. E_cient computation on oblivious RAMs. In ACM Symposium on Theory of Computing (STOC '90), pages 514{523. ACM, 1990.

9. D. Boneh, G. D. Crescenzom, R. Ostrovsky, and G. Rersiano, \Public key encryption with keyword search," in Advances in Cryptology – EUROCRYPT 2004, Lecture Notes in Computer Science, vol. 3027, pp. 506{522, Interlaken, Switzerland, 2004. Springer Berlin/Heidelberg.

10. J. Baek, R. Safavi-Naini, and W. Susilo, \Public key encryption with keyword search revisited," in ICCSA 2008, vol. 5072 of Lecture Notes in Computer Science, pp. 1249{1259, Perugia, Italy, 2008. Springer Berlin/Heidelberg.

11. H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee,\Improved searchable public key encryption with designated tester," in ASIACCS '09 Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, pp. 376{ 379, Sydney, NSW, Australia, 2009. ACM New York, NY, USA.

12. J. Black and P. Rogaway. Ciphers with arbitrary nite domains. In B. Preneel, editor, The Cryptographers' Track at the RSA Conference (CT-RSA '02), volume 2271 of Lecture Notes in Computer Science, pages 114{130. Springer-Verlag, 2002.

13. M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. In Proc.of Selected Areas in Cryptography '09, volume 5867 of Lecture Notes in Computer Science, pages 295{312. Springer-Verlag, 2009. full version available as ePrint report 2009/251.

14. Cryptographically Secure Pseudorandom Number Generators. S.l.: General Books, 2010.

15. Oppliger, Rolf. Contemporary Cryptography. 2nd ed. Norwood: Artech House, 2011. 225-226.

16. Reza Curtmola, Garay, Kamara and M. Ostrovsky.(2006,October) Improved Definitions and Efficient Constructions. paper presented at 13th ACM Cnference on Computer and Communications Security(ccs'06).

Appendix A

Functions

```
<?php
function Gen($secpar)
{
$bytes = openssl_random_pseudo_bytes($secpar, $cstrong);
    $seckey   = bin2hex($bytes);
return ($seckey);
}
function keygen($length=10)
{
   $key = '';
   list($usec, $sec) = explode(' ', microtime());
   mt_srand((float) $sec + ((float) $usec * 100000));
   $inputs                                                    =
array_merge(range('z','a'),range(0,9),range('A','Z'));
   for($i=0; $i<$length; $i++)
   {
       $key .= $inputs{mt_rand(0,61)};
   }  return $key;
}
function xor_this($string,$key) {
    $i = 0;
    $encrypted = '';
    foreach (str_split($string) as $char) {
        $encrypted   .=   chr(ord($char)   ^   ord($key{$i++   %
strlen($key)}));
```

```
    }
    return $encrypted;
}
function xor_Decrypt($string, $key)
{
$stringarray = str_split($string);
$keyarray = str_split($key);
  for($i=0; $i<strlen($string); $i++)
  {
    for($j=0; $j<strlen($key); $j++)
    {
      $stringarray[$i] = $keyarray[$j]^$stringarray[$i];
    }
  }
  return $stringarray;
}
function bbs($seed)
{
$p=67;
$q=47;
$m=$p*$q;
return $seed=(($seed*$seed)%$m);
}
function modExp($g, $x, $p){
      $r = 1;
      while($x > 0){
         if($x % 2 == 1)
```

```
            $r = $r*$g % $p;
         $x = floor($x/2);
         $g = $g*$g % $p;
      }
      return $r;
   }
function bm($seed)
{
$p=349;
$g=13;
return $seed=(modExp($g,$seed,$p));
}
function lcg($seed,$m)
{
$a=7;
$c=11;
return $seed=((($a*$seed)+$c)%$m);
 }
?>
```

Dictionary

```
<?php

include "aes.php";

include "functions.php";

include "connection.php";

?>

  <?php

    /*

    ########################################################

    #                    Gen(1k)                           #

    ########################################################

    */

    ?>

       <form      method="POST"      enctype="multipart/form-data"
action="<?php $_SERVER['PHP_SELF'] ?>">

    <p><input     type="submit"     value="Generate     Dictionary"
name="submit"></p>

    </form>

    <?php

if(isset($_POST["submit"])) {

 $k1=rand(20, 30);

 $k2=keygen();

 $k3=rand(1, 30);

 $a_ind[0]=$k1;

/*

    ########################################################

    #                    Global counter                    #
```

```
   ##########################################################
   */
$ctr=1;
$a_ind[$ctr]=bbs($a_ind[$ctr-1]);
/*
   ##########################################################
   #              Building the array A                      #
   ##########################################################
   */
$result = mysql_query("SELECT * FROM words");
 $w = mysql_num_rows($result);

  $blockSize = 256;
$a_max=47*67;

for($j=0;$j<$a_max;$j++)
{
   $a[$j]=0;

}

for($i=1;$i<=$w;$i++)
{
   //echo $i."**";
   $k[$i][0]=keygen();
   $lim=$i-1;
```

```
   $result1 = mysql_query("SELECT * FROM words ORDER BY w_id ASC LIMIT
$lim,1");

   $row1 = mysql_fetch_array($result1);

   $widi=$row1["w_id"];

   $result2 = mysql_query("SELECT * FROM dictionary1 where
w_id=$widi");

    $dw = mysql_num_rows($result2);

   $l=1;

      while ($row2 = mysql_fetch_array($result2))

      {

         $d[$i][$l]=$row2["d_id"];

         $l++;

         // print_r($row2["d_id"]);

      }

       $addr[$i]= $a_ind[$ctr];

   for($j=1;$j<$dw;$j++)

   {

      $k[$i][$j]=keygen();

      $a_ind[$ctr+1]=bbs($a_ind[$ctr]);

      $n[$i][$j]=array($d[$i][$j],$k[$i][$j],$a_ind[$ctr+1]);

      //Encryption

      $node=json_encode($n[$i][$j]);

      $aes = new AES($node, $k[$i][$j-1], $blockSize);

       $enc = $aes->encrypt();

       $aes->setData($enc);

      $a[$a_ind[$ctr]]= $enc;

      $ctr++;

   }
```

```
   $n[$i][$dw]=array($d[$i][$j],0,NULL);
         //print_r($n[$i][$j]);
   //Encryption
   $nodel=json_encode($n[$i][$dw]);
   $aes = new AES($nodel, $k[$i][$dw-1], $blockSize);
   $enc = $aes->encrypt();
   $aes->setData($enc);
   $a_ind[$ctr+1]=bbs($a_ind[$ctr]);
   $a[$a_ind[$ctr]]= json_encode($enc);
   $ctr++;
}
   /*
   ########################################################
   #           Building the lookup table                  #
   ########################################################
   */
    while(pow(2,$k3)<=$w)
$k3=rand(1, 30);

   for($i=0;$i<=pow(2,5);$i++)
{
   $t[$i]=0;
}
   $t_ind[0]=$k3;
for($i=1;$i<=$w;$i++)
{
```

```
     $plain=$addr[$i].",".$k[$i][0];
    $x=xor_this($plain,$k2);
$x=json_encode($x);
$t_ind[$i]=lcg($t_ind[$i-1],pow(2,5));
$t[$t_ind[$i]]= $x;
    $x=json_decode($x, true);


    $td[$t_ind[$i]]=xor_this($x,$k2);
}
//print_r($t);
//print_r($td);
//$asenc=implode( '' , array_map('strval', $x ) );
//$td[$i]=xor_this($x,$k2);
//$x1=xor_decrypt($asenc,$k2);
 ////$asdec=implode( '' , array_map('strval', $x1 ) );
 //$td[$i]=$asdec;
//eval($t);
    /*
    ########################################################
    #              Secure Index file                       #
    ########################################################
    */
$txt="";
$target_dir_ind = "sec_ind/";
   $target_file_ind      =      $target_dir_ind      ."sec_ind".
$filename=mt_rand().".php";
 // echo $target_file_enc="enc_".$target_file_enc;
```

```
   $myfile = fopen($target_file_ind, "w")or die("Unable to open
file!");

$txt = "<?php ";

$txt   =   $txt."   \$a=Array('".   $asenc=implode(      "','"   ,
array_map('strval', $a  ) )."') ; Echo '<BR>'; ";

   //echo"^^^^^^^^";


//print_r( array_map('strval', $a  ));


 $txt   =   $txt."   \$t=Array('".   $asenc=implode(      "','"   ,
array_map('strval', $t ) )."') ;  ?>";

$txt = mb_convert_encoding($txt, 'UTF-8', 'auto');

file_put_contents("file.txt", "\xEF\xBB\xBF" . $txt);

fwrite($myfile, $txt);


fclose($myfile);

//foreach ( $a as $v){ echo $v .",";}


   /*

   ########################################################

   #                    Output                            #

   ########################################################

   */

$ak=mt_rand();

    $target_file_enc = "keys/trapdoorkey".$ak.".txt";

   //       $target_file_enc       =       $target_dir_enc       .
basename($_FILES["fileToUpload"]["name"]);

     $myfile = fopen($target_file_enc, "w")or die("Unable to open
file!");
```

```
   $txt = " Trapdoor key 1 is :".$k3."\n  Trapdoor key 2 is :".$k2."
Trapdoor w is :".$w."\n" ;

fwrite($myfile, $txt);


fclose($myfile);

   //echo "Now, your Documents has been encrypted you will find them
at this <a href='./encrypted'target='_blank'> LINK </a><br/>";

   echo "The secure index has been created <a
href='".$target_file_ind."'target='_blank'> DOWNLOAD </a><br/>";

echo "Key generated successfully. To download please <a
href=".$target_file_enc." target='_blank'>CLICK HERE</a><br/>";


?>

      </div>

   <?php

include "footer.php";

?>
```

Search:

```
<?php
include "aes.php";
include "functions.php";
include "connection.php";
include "header.php";
?>
</header>
<!--======================
          Content
=======================-->
<section id="content"><div class="ic"></div>
  <div class="container">
    <div class="row">
      <div class="grid_12">
        <h3>Search</h3>
       <div class="extra_wrapper">
       <form id="form"  method="POST" enctype="multipart/form-data"
action="<?php $_SERVER['PHP_SELF'] ?>">
<p>T 1: <input type="text" id="t1" name="t1"></p>
<p>T 2: <input type="text" id="t2" name="t2"></p>
<p>Index name: <input type="text" id="index" name="index"></p>
   <p><input type="submit" value="Search" name="submit"></p>
   </form>
    <?php
if(isset($_POST["submit"])) {
 $t1=$_POST["t1"];
```

```
  $t2=$_POST["t2"];
 $index=$_POST["index"];
 //echo"sec_ind/$index";
 include "sec_ind/$index";
 $t[$t1]=json_decode($t[$t1], true);
//echo $jdt;
 $td=xor_this($t[$t1],$t2);
$add=explode(",",$td);
 if (is_numeric($add[0])&& array_key_exists($add[0],$a))
 {
 $inputText = $a[$add[0]];
 $inputKey = $add[1];
$blockSize = 256;
//echo "*";
$i=0;
do {
$aes = new AES($inputText, $inputKey, $blockSize);
//$enc = $aes->encrypt();
//$aes->setData($enc);
$dec=$aes->decrypt();
//$inputText=json_decode($inputText, true);

$n=json_decode($dec, true);
//$n=explode(",",$n);
//$n[2]=json_decode($n[2], true);
//echo "*".$n2=intval($n[2]);
$d[$i]=$n[0];
```

```
$inputKey=$n[1];

if($n[2]!=NULL)

 $inputText=$a[$n[2]];

$i++;

}while ($n[2]!=NULL);

?>

 <h3>Search Result</h3>

 <?php

$num=1;

foreach($d as $doc)

{

   $result2 = mysql_query("SELECT * FROM docs where d_id=$doc");

    $dw = mysql_fetch_array($result2);

   echo     $num."-     <a     href='encrypted/".$dw["d_name"]."'
target='_blank'>".$dw["d_name"]."</a>";

   $num++;

}

 }else

 {

    echo"oops!! something went wrong!!";

 }

 }

 ?>
```

National Ministry of Health Sudan

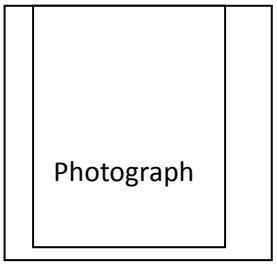
House-man Training period

The Final Assessment Report

Guidelines to Evaluators:

1. This report is strictly confidential.
2. It should be filled immediately after the end of the shift.
3. It should be filled after review of the house officer logbook& must be consistent with it.
4. The rating scale for each criteria& domains must be adhered to.
5. House officers who were performing unsatisfactory in the first half of the shift must be counselled about their shortcomings and helped to overcome them. This must be documented in writing.
6. The report should be put in an envelope labeled strictly confidential and dispatched to the hospital administration.
7. In case of an adverse report, the candidate should be informed by the evaluator.

Final Assessment Report



Grade: House- Officer

Specialty: _____

Name : _____

Period:From: _____ To: _____

Domains	**Unsatisfactory**	**Good**	**Very Good**
Knowledge: • Basic • Clinical Integration			
Clinical: • History taking • Physical examination • Identifying problems and priorities • Discrete use of lab tests • Suggestion Appropriate management plans • Data interpretation • Good records keeping			
Clinical& Technical Skills:			

• Clinical skills • Technical skills (Procedures)			
Professional attitudes: • Work habits& Ethics • Attendance& punctuality • Inter-professional relationship • Patients& parents relationship • Academic contribution& continuous professional development.			

Overall Assessment

Recommendation:

Completed the shift satisfactorily ☐ Grade ☐

Unsatisfactory ☐

- State reasons:

_____

_____

_____

- Was He/She counselled and advised about his unsatisfactory performance midway during the shift:

YES ☐ NO ☐

- in case of an adverse report was the candidate informed by the evaluator

YES ☐ NO ☐

Recommendation:

A- To repeat the shift ☐

B- To spend extra (remedial) time ☐ months

Name of Evaluator: _____

Professional Status: _____

Signature : _____

Date : _____

Hospital Director: _____

signature:

_____

Date : _____

Stamp