Sudan University of Science and Technology

College of Graduated Studies

A Versatile Encryption Scheme for Cloud Computing Security

نظام تشفير متعدد الأغراض لأمن بيئة الحوسبة السحابية

A Thesis Submitted in Partial Fulfillment of the Requirements of M.Sc. in Computer Science
(Information Security Track)

By

Amel SalahEldein Hassan

Supervisor

Dr. Awad Mohamed Awadelkarim

2014

DEDICATION

I dedicate this research to person whose prayer helps me my mother

To person whose encourage me to the way of success my father

My brothers and sisters for their support

My friends and my colleagues the people whom I love and respect

Everyone from him I learned.

ACKNOWLEDGEMENT

I would like to express my special thanks and gratitude to my supervisor Dr. Awad Mohamed Awadelkarim for constructive guidance.

It is great pleasure to thank my best friends and colleagues for their cooperation.

ABSTRACT

Although the cloud computing model is considered to be a very promising computing platform in world, because of its advantages such as flexibility and cost-effectiveness, but it leads to loss of control over the cloud-hosted assets in terms of data confidentiality, integrity, availability and access control, which increases the complexity of the loss of control. The data are stored in a computing environment owned by the party ,store in another party and processing may be a third party and all these parties may be assigned tasks to other parties means that the assignments of control is a transitive .

To retaining control of data we need to encrypt all data in cloud computing environment. The problem of traditional encryption limits the use of data. As a solution to this problem the cryptographers have recently invented new type of encryption fits the requirements of cloud computing environment called a versatile encryption schemes which distinct it, the versatile encryption allows processing data in ciphertext.

Thus, in this research, the adopted versatile encryption algorithm has been implemented and evaluated to study that such type of encryption is effectual and useful for cloud computing environment.

المستخلص

على الرغم من أن نموذج الحوسبة السحابية يعتبر منصة واعدة جدا في العالم للحوسبة لما له من مزايا مثل المرونة وفعالية التكلفة ، الاّ أنه يؤدي إلى فقدان السيطرة الأمنية على الأصول التي تستضيفها الحوسبه السحابية من ناحية سرية البيانات وتكامليتها ومدى اتاحيتها والتحكم في الوصول لها ومما يزيد تعقيد فقدان التحكم أن هذه البيانات التي تخزن في بيئة الحوسبة يملكها طرف ويقوم بتخزينها طرف آخر ومعالجتها قد تتم بطرف ثالث وكل هذه الأطراف قد تسند مهامها إلى أطراف أخري أي أن إسناد التحكم يكون متعدياً. ولنبقي على التحكم على البيانات نحتاج لتشفير جميع بيانات بيئة الحوسبة السحابية ومشكلة التشفير التقليدي أنه يحد من إستخدام البيانات . وكحل لهذه المشكلة إخترع مؤخراً خبراء التشفير نوع جديد من التشفير يلائم متطلبات بيئة الحوسبة السحابية يسمى التشفير متعدد الأغراض ومايميزه انه يسمح بمعالجة البيانات وهي مشفرة.

وبالتالي في هذا البحث، تم تطبيق وتقييم خوارزمية التشفير المتعدد التي تم اعتمادها لدراسة أن مثل هذا النوع من التشفير فعّال ومفيد في بيئة الحوسبة السحابية.

LIST OF FIGURES

LIST OF TABLES

Table of Contents

CHAPTER 1

Introduction

1.1 Introduction

Cloud computing, in general terms, is anything that delivers hosted IT services over the Internet, and allows consumers to access services and data via any device with Internet access [1].

Cisco defines cloud computing as “IT resources and services that are abstracted from the underlying infrastructure and provided “on-demand” and “at scale” in a “multitenant environment”. On-demand means the resource available to customer when needed and pays cost when used it, at scale means the resource size are not limited, and a multitenant environment means the environment of cloud shared between multiple customers.

Cloud computing stockholder are cloud provider, cloud consumer and service provider. Cloud provider owned the infrastructure of cloud, service provider put his service in the infrastructure of cloud provider, and the cloud consumer uses this service.

Cloud has three service models [2,3] this three service model represent the shape of provided service by provider, first model cloud software as services (SaaS) in this model the service provided to the consumer is application running in cloud infrastructure application such as collaboration, Customer Relationship Management (CRM)/ Enterprise Resource Planning(ERP)/ Human Resource(HR), industry application .in this model the consumer have not control to underling

infrastructure like servers, operating system, sometimes they have controlled access to application configuration settings. Salesforce CRM is an example of this model [1].

Second model is platform as services (PaaS) in this model the consumer have application deploy it in the cloud infrastructure or acquired application, this application to build it used language or tools supported by provider. Example of (PaaS) is java runtime, middleware, Database. The control in this model is like (SaaS) in addition to the consumer can changes of application hosting environment configuration. GoogleApps and MicrosoftAzure are examples of PaaS [1].

Third model is infrastructure as services (Iaas) in this model the services provided to consumer such as servers, networking, and storage. The consumer have not control the underling cloud infrastructure but has control to servers, operating system, and deployed application. Amazon Web Services is an example of IaaS [1].

The cloud has four deployment model [1] public cloud, private cloud, community cloud and hybrid cloud. Public cloud the infrastructure of cloud is public and owned by organization selling cloud services. Infrastructure in private cloud is owned by single organization, Intel, Hewlett Packard (HP) and Microsoft have their own internal private clouds. Although this model reduces the risk to data owned by organization this model is costly because the organization builds the cloud infrastructure from scratch. In the community cloud the infrastructure shared by several organization, Google Gov example of this model. Hybrid model the infrastructure consist public, private, community cloud. This model found when multiple organizations make agreement to use specific standards or technology to make their application or data portable.

1.2 Challenges in the Cloud

Security in the cloud inherited all challenges in secure IT System and open new challenges because in cloud computing the data hosted in third party. The transition to cloud need to degree of maturity of the organization, If the organization has not experience in the cloud infrastructure, need to increase the internal experience or search external helps to ensure the success **[1, 2]**.

The main challenges in the cloud computing is management of cloud environment, the organization must defines and implements suitable organizational structure, processes, polices, laws and controls to maintain effective governance and compliance it.

Cloud environment is not permanent, all parties must be adaptable to these changes, we need monitoring, testing and evaluation to ensure the privacy required, security criteria used and operation, polices is followed, in addition to the continuous change in the regularity, new attack appear and new laws must compliance it.

Security of data is the key issues of all systems and applications, cloud computing environment add new challenges to data such as multiple clients puts their data in the cloud environment, and the client have abstracted control to their data. Cloud provider must protect the data of clients from each other.

Client access to their data specified by Service Level Agreement between client, cloud provider and service provider, if the attack is found in cloud environment the client can not apply any tool to detect the attacker in the cloud environment because the client may disclose data owned by another client. That tools needs access to the hardware in cloud environment.

Cloud provider may resort to encrypt the data of clients in cloud environment to store it in protected manner. Although the encryption is effective method to prevent data from unauthorized client but is hinders the availability and uses of the data.

Cloud computing is very complex environment and need to standards to managed the relationship between cloud stakeholder in optimal manner, standards are determined the rights and obligations for stakeholder.

Interaction between the stakeholder in the cloud environment depends on the internet connection, cloud computing be impossible without internet connection or may face problem in environment that has internet connection unreliable.

1.3 Research Problem

Cloud computing environment is shared between multiple clients, where their data are manipulated. The data owned by (CCs), stored in other party (CPs), and may manipulate or processing by third party (CSs) which mean there are many problems in cloud environment from security viewpoint :loss of governance, cheap data and data analysis, transitive nature, (CPs) espionage, contractual obligation, management interface compromise, data protection, insecure or ineffective deletion of data, malicious insider

1.4 Research Objective

The main aim of this research is try to prove the versatile encryption scheme is have effective, useful and rational contribution to solve security problem in cloud computing environment.

1.5 Research Methodology

- Security requirements definition and determination

- Implementation of the adopted algorithm

- Validation and evaluation phase.

1.6 Structure of Research

Chapter 2 investigates literature review and related work. Chapter 3 displays background and definition of a versatile encryption schema.. Chapter 4 explains the research methodology to conduct this research. Chapter 5 explains and evaluates and discusses the implementation of the adopted algorithm. Chapter 6 shows the conclusion, future work and obstacles of this research.

CHAPTER 2

The Literature Review

2.1 Introduction

Nowadays there is a lot of information in business such as proprietary customer data, billing information, client orders, and supplier schedules need to protect from competitor's hand, thieves and hackers. Protecting business means enabling them to survive and grow. Because the data is a key element in the business you must store it in secure manner, encryption is the most popular techniques of security used. The challenge of protecting your data becomes more complex with the adoption of new business models such as cloud computing.

Cloud computing are services providing through the internet from the first party which is the service provider to the consumer which is the second party who benefits from this service. To use cloud resources by consumer needs to put their data in cloud infrastructure which mean transfer the control to cloud provider. To protect the data in cloud infrastructure, cloud consumer may apply encryption mechanism to it. Encryption researchers have recently invented versatile encryption schemes sufficiently practical for the cloud computing infrastructure.

In this chapter we introduce to literature review of cloud computing technology by defines architecture, operating and governance of it. Then show main points in information security related to it with concentrate in encryption techniques as most popular information security techniques.

2.2 Literature Review, Background and Definition

2.2.1 Cloud Computing

There are two basic assets supported by cloud computing data and applications. In the cloud computing may not need to reside data and application in the same location. Organization may transfer part of function to the cloud computing. As mentioned in [2] cloud computing can be divided into three aspects: cloud architecture, governing the cloud computing and operating in the cloud computing:

2.2.1.1 Cloud Architecture

According to National Institute of Standards and Technology (NIST) cloud computing defined as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

Cloud computing model composed of five essential characteristics which define in [3, 4] as:

- On-demand Self Service: Means the resources can be provisioned immediately when needed without requiring human interaction with each service provider, released when no longer required, and billed only when used.
- Broad Network Access: Services of cloud computing is available on the network and access through multiple devices such as mobile phones, laptops, and PDAs.
- Resource Pooling: Resources in the infrastructure of cloud are shared between multiple clients this knows as multi-tenant model, the resources assigns or reassigns depends on consumer demand. Consumer has no control and not knows the exact location of provided resource.

- Rapid Elasticity: Resources can be rapidly expanding, shrinking and available at any time. Resources often appear to be unlimited.
- Measured Service: Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

Cloud stakeholders including: cloud providers (CPs) which are owned the cloud infrastructure, service providers (SPs) which have a service, rent resources from cloud provider (CPs) and put their service in infrastructure owned by (CPs) and pay the cost of provided resource, and cloud consumers (CCs) which consumed the service and pay fees to (CPs) or (SPs) when used resources or services.

Although the cloud model is designed to reap uncountable benefits for all cloud stakeholders including (CPs), (CCs), and (SPs), the model still has a number of open issues that impact its credibility.

There are three models to deliver the services on the cloud computing with various ways of exposing the underlying infrastructure of cloud computing to (CCs) .The degree of controlling and management to (CCs) on the underlying cloud infrastructure various depends on the service models which define in[1, 4, 5, 6, 7] as :

- Software as a Service (SaaS): The (CPs) provides an application to (CCs) as a service on demand. The (CCs) can access to applications through various devices such as PC, and mobile. Most of the responsibility for security management depends on (CPs). (CCs) have some management operation such as management user identities to determine which user access to their application, configuration level to their application, and restrict their application access to specific IPs. Salesforce CRM is an example of this model.

- Platform as a Service (PaaS): The (CCs) can rent platform from (CPs) such as middleware, database software and application run-time environments. Instead of the (CCs) building platform from scratch, they can used it like they owned it. (CCs) may deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the (CPs).

  The (CCs) have more responsibilities for managing the configuration and security for the platform in the cloud infrastructure. In this model (CCs) have more control than (SaaS). The consumer does not controls the cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls). GoogleApps and Microsoft Azure are examples of PaaS.
- Infrastructure as a Service (IaaS): The (CCs) can rents hardware from (CPs) such as operating system, storage, and virtualized servers. Instead of the (CCs) purchasing that hardware, they can used it as they owned it. Most control and responsibility for security transfer from the (CPs) to the (CCs). Security provisions beyond the basic infrastructure are carried out mainly by the (CCs). Amazon Web Services is an example of IaaS.

In Figure 2.1 show the relation between resources, service delivery models, subscriber of service (CCs) and provider of service (SPs).

Some times (CPs) play the role of the (SPs). The cloud provider can obtain the services such as (IaaS, PaaS, or SaaS) from (SPs) and deploy it on their cloud infrastructure and then can provide the service to (CCs) with fees. Figure 2.2 show the differences in scope and control among cloud service models.

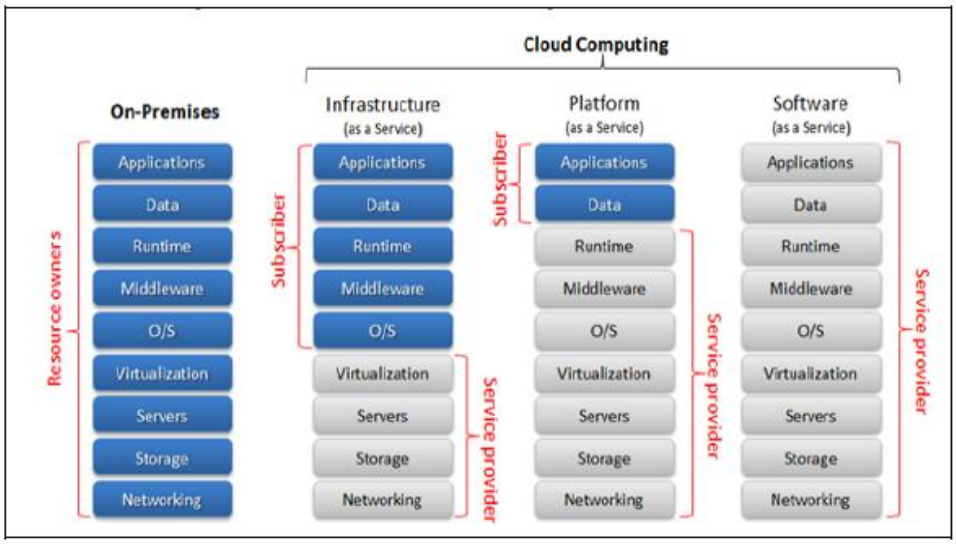

Figure 2.1: Service delivery models [1]

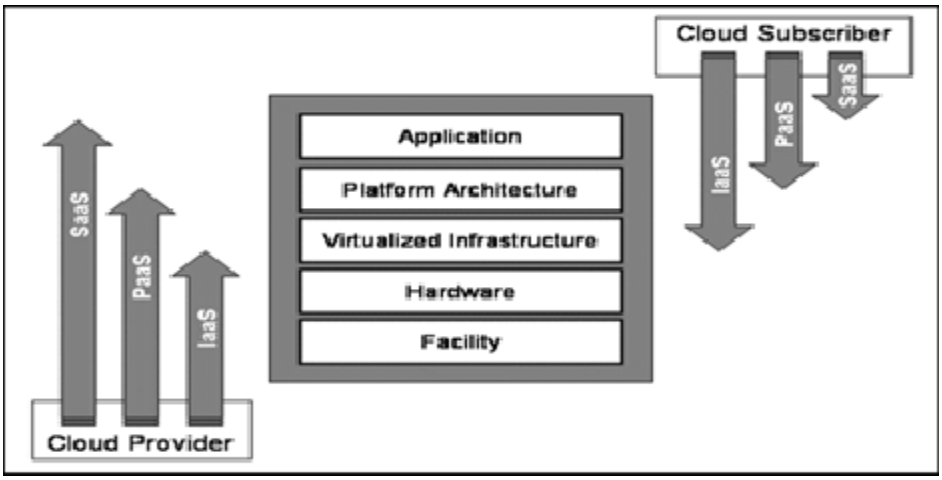

Figure 2.2 Differences in Scope and Control among Cloud Service Models [3]

There are four primary cloud deployment models define in [1, 4, 7]: public, community, private, and hybrid deployment. The different of deployment models affect an organization's scope and control over the computational environment of a cloud, the service model supported by the cloud affect the deployment model. (1) In the public deployment the cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud

services. Amazon Web Services (AWS) and Microsoft Azure are two examples of public deployment, (2) Private deployment means the underlying infrastructure of cloud computing is owned by organization. It may be managed by the organization or a third party and may exist on premise or off premise. In this case the organization plays three role of cloud stakeholder (CCs), (SPs), and (CP), organization owned infrastructure, services on it and used this services. A private cloud gives the organization greater control over the infrastructure and computational resources than does a public cloud. Intel, Hewlett Packard (HP) and Microsoft have their own internal private clouds. (3) Community cloud: The cloud infrastructure is shared by several organizations and supports a specific community, which are shared concerns (e.g., mission, security requirements, policy, and compliance considerations).

A community cloud is somewhat similar to a private cloud, but the infrastructure and computational resources are shared by several organizations that have common privacy, security, and regulatory considerations, rather than for the exclusive use of a single organization, an example of it is Google Gov. (4) Hybrid cloud: The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data or application portability.

2.2.1.2 Governing the Cloud Computing

This aspect defines the governance and enterprise risk management which means the ability of organization to governs and measures enterprise risk introduced by cloud computing. The ability of (CCs) to adequately measure risk of (CPs) and (SPs).

The organization to maintain effective information security governance, risk management and compliance they must identifies and implements appropriate organizational structure, process, policy and control. Major element in governance is agreement and negotiated about any things related to services between stakeholders in the cloud.

The organization also must define legal issue related to cloud computing with all stakeholders; stakeholder must discuss security breach disclosure laws, regulatory requirements, privacy requirements, and international laws that related to security of data when moved it from (CCs) environment to the (CPs) infrastructure.

When the organization migrate from traditional environment to cloud computing they faces new challenges. Stakeholder must defines issues dealing with evaluating how cloud computing affects compliance with internal security policies, as well as various compliance requirements (regulatory, legislative, and otherwise) and proving compliance during an audit.

The data is key elements in business. When the organization transition to the cloud, the traditional techniques of securing data are challenged by cloud computing because in the cloud there are new physical and logical architecture need new security strategies. The organization must classified the data to determines which data is sensitive to put polices to dealing with it. Polices such as what data can access, the user who accesses to data, at any time and from anywhere. Security of data in this aspect negotiates three movements important to data. Firstly protect data when move it the first time to cloud environment .Secondly protect data when transfer it between stakeholder (CPs) and (CSs) in the cloud infrastructure. Thirdly protecting data when store it within the cloud infrastructure.

Organization when adopts the cloud computing must understanding that they may have to be changes providers in the future. Gaining the benefits of this more elastic environment requires both portability and interoperability to be the design goals in any application in the cloud infrastructure.

2.2.1.3 Operations in the Cloud Computing

The evolution of cloud computing as new technologies, cloud services has enabled business entities do more with less, fewer resource and better operating efficiency but inherit new security risk not found in traditional security. In cloud computing, clients usually outsource their data to the cloud storage servers to reduce the management costs. While those data may contain sensitive personal information, the cloud servers cannot be fully trusted in protecting them. Manager should knowledge that is no measure hundred percent secure. This leads manager to implements security as layers of measure.

The (CCs) should conduct critical evaluation of (CPs) infrastructure, review the documentation about services and ensure that (CPs) are compliant with global standards of security like ISO 27001 ISMS, assessing security facilities of (CSs), (CPs) from various parameter, ensure that (CPs) and (SPs) can commits to service level agreement (SLA) which contains the profile of service provision by the cloud computing.

The (CCs) should ensure how (CPs) and (SPs) dealing with disaster recovery and backup, both are influences on the availability of the data which is most important element in the business. The (CPs) evolve cloud computing they must advance the enterprise data center which is considered as long-term cloud success.

Organization should define proper and adequate incident detection, response, notification, and remediation. This attempts to address items that should be in place at both provider and user levels to enable proper incident handling and forensics. Incident response is one of the cornerstones of information security management. Also they must discuss how they securing the application software that is running on or being developed in the cloud. This guidance is for all cloud stakeholders in the designing cloud computing application. Application must be monitoring in the cloud infrastructure, monitoring such as log, performance, malicious use and monitoring the access control to application.

Encryption is a promising way to protect the confidentiality of the outsourced data, but it also introduces much difficulty to performing effective searches over encrypted information.

When organization applies encryption to their data they must identify proper and scalable key management because the encryption is mainly depends on the keys.

The organization discusses how they managing identities and leveraging directory services to provide access control. The identity management is more complex in the cloud computing because the infrastructure is shared between customers which may conflict in their concerns. Virtualization is one of key elements of (IaaS) and also used in the back-end of the (SaaS) and (PaaS). This means all services in the cloud depend mainly or partially on the virtualization of resource. The benefits of the virtualization are multi-tenancy and server utilization.

In this aspect the organization discuss the security issues surrounding system/hardware virtualization, rather than a more general survey of all forms of virtualization.

Organization may benefit from other organization that provides security as a service which means delegation of detection, remediation, and governance of security to trusted third party with proper tools and expertise.

2.2.2 Information Security

The NIST Computer Security Handbook [NIST95] defines the term computer security as follows: "The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications)" [4]**.**

This definition contains three key objectives in computer security: (1) confidentiality: this term contains two means data confidentiality and privacy. The data confidentiality assures that private or confidential information is not disclosed by unauthorized person, while privacy means the person control the information related to him and by whom and to whom that information may be disclosed.(2) integrity: this terms also contains two means data integrity which means prevent the changes of information from unauthorized person. Another mean of it system integrity which means a system performs its intended function in correct manner free from deliberate or inadvertent unauthorized manipulation of the system.(3) availability: assures that systems work promptly and service is not denied to the authorized users.

To apply security in organization determines security needs of organization and search to security products and polices satisfy the need with specific speed, cost,

performance. The Open System Interconnections (OSI) model security architecture is defines such a systematic approach that organizing the task of providing security [4]. The OSI security architecture concentrates on security attacks, mechanisms, and services.

security service defines as any process that enhance the security of information by using one or more security mechanisms which are any process designed to detect, prevent, or recover from a security attack which defines as any action that compromises the security of information.

There are two types of security attack: active attack and passive attack: The aims of the attacker in the active attack are disclose the information that is being transmitted. The aims of the attacker in the passive attack not just disclose the information that is being transmitted but also changes the content of the message [4].

The security services can be divided into five categories:

- Authentication: the assurance that the communicating entity is the one that it claims to be.
- Access control: allow the use of resources to just authorized users.
- Data Confidentiality: prevent the data from person who unauthorized.
- Data Integrity: the assurance that data received as sent by an authorized entity.
- Non Repudiation: provides protection against repudiation of message from sender who claims that message is not send by him, or receiver who claims that message is not received by him [4].

The security mechanisms such as encipherment (encryption), which converts the data from readable form (plaintext) to unreadable form (ciphertext) using

mathematical algorithm with zero or more secret keys. The reverse operation of encipherment is decipherment (decryption) which converts unreadable form (ciphertext) to readable form (plaintext) using mathematical algorithm with zero or more secret keys [4].

There are many techniques used for encipherment such as:

- Encryption: this method used one or more keys with encryption algorithm to convert plaintext to ciphertext the common types of it are substitution and transposition in substitution to apply encryption they replace the character of plaintext by another character or numbers or symbols. While when used transposition they changes the places of character in plaintext by some sort of permutation.
- Steganography: area of study encipherment achieved by conceals the existence of message [4].

In enciphering operation the message is not hiding this area of study known as cryptography. Techniques used for deciphering a message without any knowledge of the enciphering details such as algorithm fall into the area of study known as cryptanalysis. Cryptography with cryptanalysis area of study knows as cryptology.

There are two types of encryption: symmetric and asymmetric encryption: in the symmetric encryption they used mathematical algorithm with one key to encrypt and decrypt, while they used mathematical algorithm with two keys one for encrypt and other for decrypt in asymmetric encryption which called public key encryption.

Definition of algorithm mentioned in this research [4]:

- **Advanced Encryption Standard (AES)**

The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001. AES is a symmetric block cipher. The cipher takes a plaintext block size of 128 bits, or 16 bytes. The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits). The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.

- **Cipher Block Chaining (CBC)**

In this mode the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block.

- **Electronic Codebook (CBC)**

The simplest mode in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key.

Definitions [4]

- **Pseudorandom Number**

a sequence of numbers which is algorithmically produced and not really Random. It can be fully repeated if the initial conditions and algorithm are known.

- **Pseudorandom Function**

A pseudorandom function is a family of functions with the property that the input-output behavior of a random instance of the family is “computationally indistinguishable” from that of a random function.

A PRF is used to produce a pseudorandom string of bits of some fixed length. Examples are symmetric encryption keys and nonces. Typically, the PRF takes as input a seed plus some context specific values, such as a user ID or an application ID.

- **Pseudorandom number generator**

An algorithm that is used to produce an open-ended sequence of bits is referred to as a (PRNG). A common application for an open-ended sequence of bits is as input to a symmetric stream cipher.

2.2.3 Security in the cloud Computing

In cloud computing, clients usually outsource their data to the cloud storage servers to reduce the management costs. While those data may contain sensitive personal information, the cloud servers cannot be fully trusted in protecting them. Encryption is a promising way to protect the confidentiality of the outsourced data, but it also introduces much difficulty to performing effective searches over encrypted information. Most existing works do not support efficient searches with complex query conditions, and care needs to be taken when using them because of the potential privacy leakages about the data owners to the data users or the cloud server.

2.2.3.1 Challenges of Security in the Cloud Computing:

The majority of (CPs) surveyed do not believe their organization views the security of their cloud services as a competitive advantage, they do not consider cloud computing security as one of their most important responsibilities and believe it is customer's responsibility [5].

The (CPs) are allocate ten percent or less of their operational resources to security, also they believe the primary reason why customers purchase cloud resources are lower cost and faster deployment of applications not improved security or

compliance with regulations but see this unlikely reason. Some (CPs) do not have dedicated security personnel to oversee the security of cloud applications, infrastructure or platforms. In addition, (CCs) admit they are not vigilant in conducting audits or assessments of cloud computing providers before deployment [5].

To benefits stakeholder of cloud computing in proper way they must negotiate any attributes related to security of data and operations, also they must believe the responsibility of security is shared between them. [5].

2.2.3.2 Encryption in the Cloud Computing

Encryption techniques are most techniques used in cloud environment to protect sensitive data from unauthorized user by store the data in unreadable form which discuss in [2, 6, 8]**.**

Data in cloud computing technologies must be encrypted in motion when transfer between cloud computing stakeholders, data in processing must also not disclosure and data when stored either in owner repositories or in cloud infrastructure.

The encryption of mobile media and the ability to securely share those encryption keys between the cloud service provider and consumer is an important and often overlooked need. Because moving large volumes of data quickly and cheaply over the internet is still not practical in many situations, many organizations must send mobile media, such as an archive tape, to the cloud provider. It is critical that the data is encrypted and that only the cloud provider and client have access to the encryption keys.

Encryption techniques in the few year ago when used it to protect data from unauthorized access are scarified the functionality and reduce the system performance because they need to decrypt and then used the data.

Cryptographers have recently invented versatile encryption schemes that allow operation and computation on the ciphertext. A versatile encryption schemes allows the data owner (CCs) to compute a capability from his secret key. A capability encodes a search query, and the cloud provider (CPs) can use this capability to decide which documents match the search query, without learning any additional information. Other cryptographic primitives such as homomorphic encryption and private information retrieval (PIR) this schema can enable cloud users (CCs) to benefit from one another's data in a controlled manner such as detecting user activities outside of the norm) and better data loss prevention [6]. When these cryptographic techniques mature, they may open up new possibilities for cloud computing security.

2.3 Summery

In this chapter we displayed the literature review, background and definition of cloud computing environment by demonstrated the architecture, governing and operating cloud computing. Then we defined the main points in information security. Then we presented security in cloud computing by determined challenges of cloud environment with encryption as better solution to most challenges.

CHAPTER 3

The Versatile Encryption Approach

3.1 Introduction

A versatile encryption schema in the origin are encryption techniques with new properties added to cope with the requirements of new technologies such as cloud computing.

Years ago when the organization encrypts its data the primary purpose is to protect it in their repositories in a confidential manner or sometimes data may transfer from one branch to another within the organization. This means the data is transferred in different places but them all ultimately under the control of the organization. In order to used the data by organization they need just to decrypt it.

Now, with the emergence of new technologies, notably the technology of cloud computing, if any organization adopts cloud technologies for benefits from services they will convey their data and store it in cloud environment in encrypted form. In this case if the organization encrypts data with traditional encryption techniques, they are need to bring the data from a third party (cloud computing) and then apply decryption operation to retrieve the data in readable form, apply desired operations, then are encrypts the data again and returns it to the cloud environment, that is, if it can conduct operations on the enterprise data in their environment rather than the environment of cloud computing.

Sometimes there are services require applications which exist only in a cloud environment. The organization if are decrypted the data in cloud environment this is means the data are disclose by the cloud owner, a versatile encryption schema emerged from this problem.

The organization when apply a versatile encryption schema they can conducting searches for information in their data on cloud environment and retrieve results without disclose the contents of their data to cloud provider.

A versatile encryption schemes allows customer to apply operations and computations on the encrypted data and return the result without decrypts the data.

3.2 Categories of a Versatile Encryption

There are three categories falls in a versatile encryption schema [6] which discuss in the following sections:

3.2.1 Searchable Encryption

This category allows the data owner to compute a capability from his secret key. A capability encodes a search query, and the cloud provider can use this capability to decide which documents match the search query, without learning any additional information. The secret key in this type may be the session key if the algorithm is symmetric, or it may be the private key in the case that the algorithm is asymmetric uses the public key for encryption.

Many research recently invented encryption schemes in this category which defines in the following section:

3.2.1.1 Public Key Encryption with keyword Search

In paper [7] build algorithm called a non-interactive searchable encryption scheme (PEKS) that allow the mail client to search for specific words on server mail that client stores mail on it in encrypted format. The goal on this paper is to enable client of mail to send a short secret key (trapdoor) T_w to the mail server that will enable the server to locate all messages containing the keyword (W), but learn nothing else. Client produces this trapdoor (T_w) using his private key. The server simply sends the relevant emails back to client.

And this scheme consists of the following polynomial time randomized algorithms:

- KeyGen(s): takes a security parameter s and generates a public private key pair C_{pub}, C_{priv}.
- PEKS (C_{pub}, W): for a public key C_{pub} and a word W produces a searchable encryption of W.
- Trapdoor (C_{priv}, W): given Client's private key and a word W produces a trapdoor T_w.
- Test (C_{pub}, S, T_w): given Client's public key, a searchable encryption $S = PEKS(C_{pub}; W'')$, and a trapdoor T_w = Trapdoor (C_{priv}; W), outputs 'yes' if W = W'' and 'no' otherwise.

The scenario of this algorithm client runs the KeyGen algorithm to generate his public/private key pair. He uses Trapdoor algorithm to generate trapdoors T_w for any keywords W that he wants the mail server or mail gateway to search for. The mail server uses the given trapdoors as input to the Test algorithm to determine whether a given email contains one of the keywords W specified by client or no.

There are two constructions for (PEKS) algorithm:

- Construction is based on a variant of the computational Diffie-Hellman problem: Abstractly, they use two groups G_1, G_2 of prime order p and a bilinear map $e: G_1 \times G_1 \longrightarrow G_2$ between them.

  The map satisfies the following properties:

  - Computable: given $g, h \in G_1$; there is a polynomial time algorithms to compute $e(g,h) \in G_2$.
  - Bilinear: for any integers $x, y \in [1, p]$ we have $e(g^x, g^y) = e(g,g)^{xy}$.
  - Non-degenerate: if g is a generator of G_1 then (g,g) is a generator of G_2.

  The size of G_1 ; G_2 is determined by the security parameter.

  They will need hash functions $H_1 : \{0,1\}^* \longrightarrow G_1$ and $H_2 : G_2 \longrightarrow \{0,1\}^{\log p}$.

  The PEKS works with first construction as follows:

  - KeyGen: The input security parameter determines the size, p of the groups G_1 and G_2.
  - The algorithm picks a random $\alpha \in Z_p^*$ and a generator g of G_1. It outputs $C_{pup} = [g, h = g^{\alpha}]$ and $A_{piv} = \alpha$.
  - PEKS (C_{pub}, W): First compute $t = e(H_1(W), h^r) \in G_2$ for random $r \in Z_p^*$. Output of PEKS $(A_{pub}, W) = [g^r, H_2(t)]$.
  - Trapdoor (C_{priv}, W): output $T_w = H_1(W)^{\alpha} \in G_1$.
  - Test (C_{pub}, S, T_w): let $S = [A, B]$. Test if $H_2(e(T_w, A)) = B$.

    If so, output yes; if not, output no.

  They prove that this system is a non-interactive searchable encryption scheme semantically secure against a chosen keyword attack in the random oracle model.

- Another construction is based on general trapdoor permutations, assuming that the total number of keywords that the user wishes to search for is bounded by some polynomial function in the security parameter.

3.2.1.2 Practical Techniques for Searches on Encrypted Data

Paper [8] presents cryptographic schemes with symmetric key that enable searching on encrypted data without leaking any information to the un-trusted server. This scheme has number of crucial advantages: Firstly provably secure for encryption which means that the un-trusted server cannot learn anything about the plaintext when only given the ciphertext. Secondly they provide query isolation for searches meaning that the un-trusted server cannot learn anything more about the plaintext than the search result; thirdly they provide controlled searching, so that the un-trusted server cannot search for an arbitrary word without the user's authorization; fourthly support hidden query so that the user may ask the un trusted server to search for a secret word without revealing the word to the server.

The algorithm that implements this schema is efficient and practical, for a document of length n the encryption and search algorithms only need $O(n)$ number of stream cipher and block cipher operations. Their schemes introduce essentially no space and communication overhead. Also flexible and can be easily extended to support more advanced searches.

Client has set of mails stored in un-trusted server on encrypted form, because client may have only a low-bandwidth network connection to the un-trusted server, he wishes to only retrieve the documents which contain the word W. There are two types of approaches: one of them is to build up an index that, for each word W of interest, lists the documents that contain W.

An alternative approach is to perform a sequential scan without an index. The advantage of using an index is that it may be faster than the sequential scan when the documents are large. The disadvantage of using an index is that storing and updating the index can be of substantial overhead. So the approach of using an index is more suitable for mostly-read-only data.

When using the sequential scan this paper proposed basic algorithm which are containing the following encryption scenario: client is generate pseudorandom value $S_1,.....,S_l$ using pseudorandom generator G and apply pseudorandom function F with K_i chosen by client to calculate pseudorandom stream $T_i = \left(S_i, F_{k_i}(W_i)\right)$. T_i is pseudorandom stream only client can generate it so, no one else can decrypt it. Then the client calculates ciphertext $C_i = (W_i \oplus T_i)$ and stores it on service provider.

The basic scheme supports searches over the ciphertext by this scenario: if client wants to search about W he can tell the service provider and send to it W and K_i corresponding to each location which W occur. Service provider can then search for W in the ciphertext by checking whether $C_i \oplus W_i$ is of the form of $\langle s, F_{K_i}(s)\rangle$ for some s.At the positions where service provider does not know K_i, he does not learns nothing about the plaintext. The basic scheme provides provable secrecy if the pseudorandom function F and pseudorandom generator G are secure.

If the client wants from un-trusted server to search for W either client must reveal all the K_i (which may reveal the entire document), or client must know in advance which locations may appear at.

Another schema provides controlled searching: In this schema the researcher add additional pseudorandom function f which will be keyed independently of F.

The main idea is to choose keys as $k_i = f_{k'}(W_i)$. require that k' be chosen uniformly randomly by client and never be revealed. Then if client wish to allow un trusted server to search for the word W, client reveals $f_{k'}(W)$ and W to server. This allows server to identify all the locations where W might occur, but reveals absolutely nothing on the locations i where $W_i \neq W$. This attains their desired goal of controlled searching. If the document to be encrypted consists of a series of chapters, an alternative approach is to generate the key k_i for the word W in chapter C as $k_i = f_{k'}((C,W))$ This allows client to control which chapters server may search in as well as controlling which words server may search for.

Scheme three provides hidden searches: suppose client would now like to ask server to search for a word W but he is not willing to reveal W to server. They proposed a simple extension to the above scheme to support this goal. Client should merely pre-encrypt each word of the clear text separately using a deterministic encryption algorithm. So they may think of this pre-encryption step as ECB (Electronic Code-Book) encryption of the words of the document using some block cipher. if the word is very long, internally the map $E_{k''}$ may be implemented by CBC (Cipher Block Chaning) encrypting W_i with a constant IV, or some such, but the point is that this process must be the same at every position of the document. they let $X_i = E_{k''}(W_i)$.

After the pre-encryption phase, client has a sequence of E-encrypted words $X_1,...,X_l$ Now he post-encrypts that sequence using the stream cipher construction described above to obtain $C_i = X_i \oplus T_i$ where $X_i = E_{k''}(W_i)$ and $T_i = \left(S_i, F_{k_i}(X_i)\right)$.To search for a word W,client compute $X = E_{k''}(W)$ and $k = f_{k'}(X)$ and sends (k, X) to server. Note that this allows server to search for W without revealing W itself. It is

easy to see that this scheme satisfies the hidden search property as long as the pre-encryption E is secure.

scheme three actually suffers from a small inadequacy: client can no longer recover the plaintext from just the ciphertext if client generates keys $k_i = E_{k''}(W_i)$ then client can no longer recover the plaintext from just the ciphertext because he would need to know $E_{k''}(W_i)$ This defeats the purpose of an encryption scheme, because even legitimate principals with access to the decryption keys will be unable to decrypt. In this paper they show a simple fix for this problem. In the fixed scheme,which arefinal schema they split the pre-encrypted word $X_i = E_{k''}(W_i)$ into two parts, $X_i = (L_i, R_i)$, where L_i (respetivly R_i). Instead of generating $k_i = f_{k'}(E(W_i))$, client should generate $k_i = f_{k'}(E(L_i))$.

To decrypt, client can generate S_i using the pseudorandom generator (since client knows the seed), and with S_i he can recover L_i by XORing S_i against the first $n-m$ bits of C_i . Finally, knowledge of L_i allows client to compute k_i and thus finish the decryption.

Alernative approch to sequantial scan is searching with an encrypted index, sequential scan may not be efficient enough when the data size is large. An index contains a list of key words; with each key word is a list of pointers to documents where the key word appears. The key words are words of interest that client may want to search for later. client can certainly build the index of his clear text documents and then encrypt the clear text and the index and store the ciphertext on server.

There are many aproaches to encrypt the index, one of them is encrypt the key words in the index and leave the lists of positions in clear. This approach makes it easy for server to perform search queries on client's behalf, but also leaks a lot of information to server and hence may allow him to apply various statistical attacks. Therefore, this naive approach is inefficient.

Another way is to also encrypt the document pointers in each list in the index. Consequently, when the server searches for encrypted word and finds a match, he returns to client the encrypted list of matching positions from the index. client may decrypt the encrypted entries and send to server another request to retrieve the relevant documents. One possible advantage for this scheme is that the request could be embedded in other retrievals so that server might have uncertainty about the correlation of the search request and the retrieval request for ciphertext. The disadvantage is that client has to spend an extra round-trip time to retrieve the documents. The server merge the results of several search queries for client to solve problem of wasted time.

Note that a general disadvantage for index search is that whenever client changes his documents, he must update the index. There is a trade-off between how much index client updates and how much information server might be able to learn.

3.2.1.3 Multi-dimensional Range Query over Encrypted Data

in paper [9] design an encryption scheme called Multi-dimensional Range Query over Encrypted Data (MRQED), to address the privacy concerns related to the sharing of network audit logs and various other applications. the scheme allows a network gateway to encrypt summaries of network flows before submitting them to an un-trusted repository. When network intrusions are suspected, an authority can

release a key to an auditor, allowing the auditor to decrypt flows whose attributes (e.g., source and destination addresses, port numbers, etc.) fall within specific ranges. Apart from network audit logs, their scheme also has interesting applications for financial audit logs, medical privacy, untrusted remote storage, etc.

Recently, the network intrusion detection community has made large-scale efforts to collect network audit logs from different sites. In this application, a network gateway or an Internet Service Provider (ISP) can submit network traces to an audit log repository. However, due to the presence of privacy sensitive information in the network traces, the gateway will allow only authorized parties to search their audit logs. They consider the following four types of entities: a gateway, an untrusted repository, an authority, and an auditor. they design a cryptographic primitive that allows the gateway to submit encrypted audit logs to the untrusted repository. Normally, no one is able to decrypt these audit logs.

This paper mentioned application example of (MRQED) such as Financial audit logs contain sensitive information about financial transactions. scheme (MRQED) allows financial institutions to release audit logs in encrypted format. When necessary, an authorized auditor can obtain a decryption key from a trusted authority. With this decryption key, the auditor can decrypt certain transactions that may be suspected of fraudulent activities. However, the privacy of all other transactions are preserved. Scheme provides the following properties:

Range query on attributes. An authority can issue a decryption key for all flows whose (t, a, p) where t is time stamp, a is a source address and p is destination port number falls within a certain range: $t \in [t_1, t_2]$ and $a \in [a_1, a_2]$ and $p \in [p_1, p_2]$.

A MRQED scheme consists of four (randomized) polynomial-time algorithms: Setup, Encrypt, DeriveKey and QueryDecrypt.

In the network audit log example, an authority runs setup to generate public parameters and a master private key; a gateway runs the encrypt algorithm to encrypt a flow. Encryption is performed on a pair (Msg, X). The message Msg is an arbitrary string, and X is a point in multi-dimensional space, representing the attributes.

Whenever necessary, the authority can run the DeriveKey algorithm, and compute a decryption key allowing the decryption of flows whose attributes fall within a certain range. Given this decryption key, an auditor runs the QueryDecrypt algorithm over the encrypted data to decrypt the relevant flows. Suppose that during time $[t_1, t_2]$, there is an outbreak of a worm characteristic by the port number p_1. Now the trusted authority issues a key for the range $t \in [t_1, t_2]$ and $p = p_1$ to a research group who has been asked to study the worm behavior. With this key, the research group should be able to decrypt only flows whose time-stamp and port number fall within the given range. The privacy of all other flows should still be preserved.

An (MRQED) scheme consists of the followingpolynomial-time randomized algorithms.

- Setup($\sum$, L_Δ): Takes a security parameter $\sum$ and D-dimensional lattice L_Δ and outputs public key PK and master private key SK.
- Encrypt(PK , X , Msg): Takes a public key PK, apoint X, and a message Msg from the message space M and outputs a ciphertext C.

- DeriveKey(PK , SK , B): Takes a public key PK , a master private key SK , and a hyper-rectangle B and outputs decryption key for hyper-rectangle B .
- QueryDecrypt(PK , DK , C): Takes a public key PK , a decryption key DK , and a ciphertext C and outputs either a plaintext Msg or $\perp$, signaling decryption failure.

  For each message $Msg \in M$ hyper-rectangle $B \subseteq L_{\Delta}$ and point $X \in L_{\Delta}$, the above algorithms must satisfy the following consistency constraints:
  QueryDecrypt (PK , DK , C) is equal Msg if $X \in B$ Otherwise is equal $\perp$.

3.2.1.4 Predicate Privacy in Encryption Systems

In paper [10] the researcher proposed new encryption paradigm called predicate privacy in encryption systems which allows for such fine-grained control over access to encrypted data.

In a predicate encryption scheme, the owner of a master secret key can create and issue secret key tokens to other users. Tokens are associated with predicates which can be evaluated over encrypted data. Specifically, an encryption of a data x can be evaluated using a token TK_f associated with a predicate f to learn whether $f(x) = 1$.

In this work, the researcher consider a different dimension of predicate encryption predicate privacy.

In addition to protecting the privacy of plaintexts, they also protect the description of the predicates encoded by tokens.

Unfortunately, predicate privacy is inherently impossible to achieve in the public-key setting. Since encryption does not require a secret key, an adversary can

encrypt any plaintext of his choice and evaluate a token on the resulting ciphertext to learn whether the plaintext satisfies the predicate associated with the token. In this way, an adversary can gain information about the predicate encoded in a token. Therefore, it does not make sense to consider the notion of predicate privacy for predicate encryption in the public-key setting.

In this paper they consider predicate privacy in the symmetric-key setting, in applications where you want to hide information about the predicate being tested from the party evaluating a token.

The researcher discussion details of Symmetric-Key Predicate-Only Encryption schema for the class of predicates f over the set of attributes consists of the following probabilistic polynomial time algorithms:

- Setup(1^{λ}): Takes as input a security parameter 1^{λ} and outputs a secret key SK.
- Encrypt(SK, x): Takes as input a secret key SK and a plaintext $x \in \sum$ and outputs a ciphertext CT .
- GenToken(SK, f): Takes as input a secret key SK and a description of a predicate $f \in F$ and outputs a token TK_f.
- Query(TK_f, CT): Takes as input a token TK_f for a predicate f and a ciphertext CT. It outputs either 0 or 1, indicating the value of the predicate f evaluated on the underlying plaintext. Correctness. For correctness, the researcher used the following condition.For all λ, all $x \in \sum$, and all $f \in F$
  - If $f(x) = 1$, then Query(TK_f, CT) = 1.
  - If $f(x) = 1$, then $\Pr\left[Query(TK_f, CT) = 0\right] \succ 1 - \varepsilon(\lambda)$ where ε is a negligible function.

In this paper also prove the security for a symmetric-key predicate-only encryption scheme.

3.2.1.5 Conjunctive, Subset, and Range Queries on Encrypted Data

Paper in [7] developed a mechanism for equality tests in the public-key settings. In paper [8] developed a mechanism for equality tests on data encrypted with a symmetric key system.

While in this paper [11] support query of equality conjunction, comparison conjunction, subset conjunction which mean extended the paper in [7]. Let ϕ be a set of predicates over $\sum$.

A searchable public-key systems in this paper comprises of the following algorithms:

- Setup(λ) : a probabilistic algorithm that takes as input a security parameter and outputs a public key PK and secret key SK.
- Encrypt(PK , I , M):encrypts the plaintext pair (I, M) using the public key PK. They veiw $I \in \sum$ as the searchable field, called an index, and $M \in M$ as the data..
- GenToken (SK , (P)) : takes as input a secret key SK and the description of a predicate $P \in \phi$. It outputs a token TK_p.
- Query(TK , C) algorithm takes a token TK for some predicate $P \in \phi$ as input and a ciphertext C. It outputs a message $M \in M$ or $\perp$. If C the ciphertext of (I, M) then the algorithm outputs M when $P(I) = 1$ and outputs $\perp$ otherwise.

In this paper introduce the concept of Hidden Vector Encryption (HVE) system which leads to complexity of query (comparison and subset queries) then proof the

security of both public-key systems without (HVE) system and public-key systems with (HVE) system.

This paper in construct public-key systems that support queries on encrypted data a secret key can produce tokens for testing any supported query predicate. The token lets anyone test the predicate on a given ciphertext without learning any other information about the plaintext.

3.2.2 Private Information Retrieval

In this category we can perform computations on encrypted data without decrypting.

3.2.2.1 Private Information Retrieval

Another paper in [12] proposed schema called Private Information Retrieval (PIR) this paper view the database as a binary string $x = x_1,...,x_n$ of length n. Identical copies of this string are stored by $K \geq 2$ servers. The user has some index i, and he is interested in obtaining the value of the bit x_i. To achieve this goal, the user queries each of the servers and gets replies from which the desired bit x_i can be computed. The query to each server is distributed independently of i and therefore each server gains no information about i.

Schemes are based on exclusive-or (linear summations, or sum) queries; this type of queries is very common and is actually implemented in several "real-world" databases.

In this paper they consider a randomized strategy for the user, which on input an index $i \in [n] \xrightarrow{\Delta} \{1,...,n\}$ and random input r (of length , l_{rnd}), produce k queries(of length l_q each), $Q_1(i.r),...,Q_k(i,r)$, one per server.

The servers respond according to strategies $A_1,...,A_k$ with replies (of length l_a) that depend on the contents of the database, denoted x, and the corresponding query. The user reconstructs the desired bit x_i from these k replies, together with i and r. The privacy requirement is that each individual query is distributed independently of i and thus the server gains no information about the identity of the desired item.

A k server Private Information Retrieval (PIR) scheme for database length n consists of

- k query functions $Q_1,...,Q_k : [n] \times \{0,1\}^{l_{rnd}} \mapsto \{0,1\}^{l_q}$;
- k answer functions, $A_1,...,A_k : \{0,1\}^n \times \{0,1\}^{l_q} \mapsto \{0,1\}^{l_a}$;
- a reconstruction function, $R : [n] \times \{0,1\}^{l_{rnd}} \times \left(\{0,1\}^{l_a}\right)^k \mapsto \{0,1\}$;

These functions should satisfy correctness and privcy.

This schema has proprites such as perfect privacy, memoryless protocol, deterministic server strategeis, noncllousion and coalition.

3.2.3 Proofs of Retrievability

Apart from ensuring privacy, applied cryptography may also offer tools to address other security problems related to cloud computing. For example, in proofs of retrievability. The storage server can show a compact proof that it is correctly storing all of the client's data.

In the following section we mentioned an example of algorithm in the proofs of retrievability category.

3.2.3.1 Provable Data Possession

Researcher in [13] introduce model for provable data possession (PDP) that allows a client that has stored data at an un-trusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The paper present two provably-secure PDP . The overhead at the server is low (or even constant), as opposed to linear in the size of the data. Experiments using their implementation verify the practicality of PDP and reveal that the performance of PDP is bounded by disk I/O and not by cryptographic computation.

The model is allows the server to access small portions of the file in generating the proof; all other techniques must access the entire file. Within this model, they give the first provably-secure scheme for remote data checking. The client stores a small $O(1)$ amount of metadata to verify the server's proof. Also, the scheme uses $O(1)$ bandwidth. The challenge and the response are each slightly more than 1 Kilobit. they also present a more efficient version of this scheme that proves data possession using a single modular exponentiation at the server, even though it provides a weaker guarantee.

A PDP schemes has multiple featuers such as provide data format independence and put no restriction on the number of times the client can challenge the server to prove data possession. Also, a variant of their main PDP scheme offers public verifiability this feature allows anyone, not just the data owner, to challenge the server for data possession.

Figure 3.1 demostrate the senario of a PDP protocol .In the a PDP protocol the client C (data owner) pre-processes the file, generating a piece of metadata that is stored locally, transmits the file to the server S, and may delete its local copy. The server stores the file and responds to challenges issued by the client. Storage at the server is in $\Omega(n)$ and storage at the client is in $O(1)$, conforming to their notion of an outsourced storage relationship.

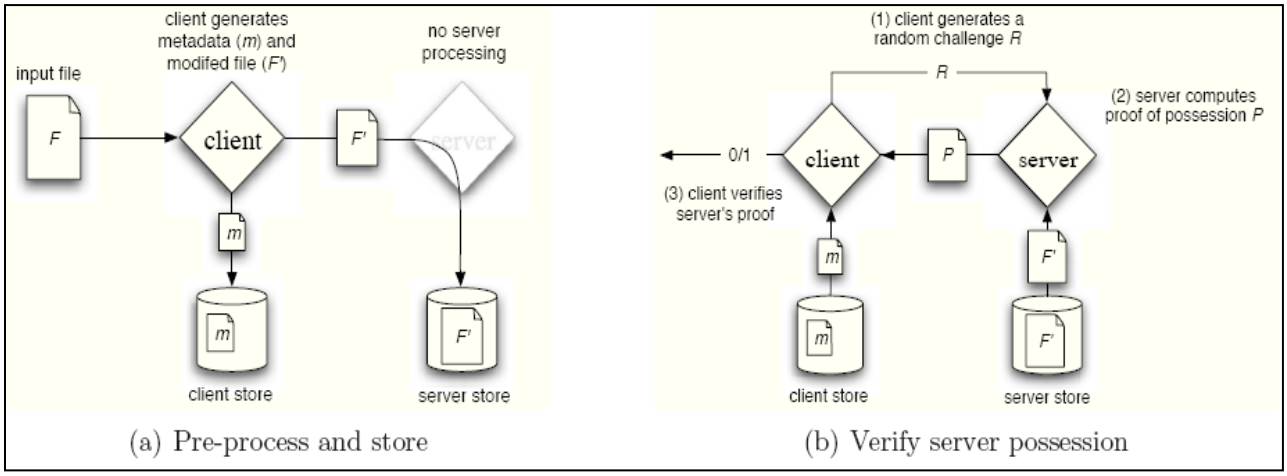

Figure 3.1: Protocol for Provable Data Possession [13].

As part of pre-processing, the client may alter the file to be stored at the server. The client may expand the file or include additional metadata to be stored at the server. Before deleting its local copy of the file, the client may execute a data possession challenge to make sure the server has successfully stored the file. Clients may encrypt a file prior to out-sourcing the storage. For their purposes, encryption is an orthogonal issue; the "file" may consist of encrypted data and our metadata does not include encryption keys.

At a later time, the client issues a challenge to the server to establish that the server has retained the file. The client requests that the server compute a function

of the stored file, which it sends back to the client. Using its local metadata, the client verifies the response. The goal of a PDP scheme that achieves probabilistic proof of data possession is to detect server misbehavior when the server has deleted a fraction of the file.

The importance parameter of PDP schema is computation complexity which is the computational cost to pre-process a file at client, to generate a proof of possession at server and to verify such a proof at client; another parameter is block access complexity which is the number of file blocks accessed to generate a proof of possession, and last parameter is communication complexity which is the amount of data transferred between client and server.

Any process in this schema relevant to the server, the schema try to minimized because the server concurently serve multiple clients and the overhead computation in the client is not important.

In this paper to meet requirment to minimized the computation in the server, client try to access random subset of the blocks. In this paper they introduce new concept of a homomorphic verifiable tags (HVTs) that will be used as a building block for their PDP schemes. The (HVTs) act as verification metadata for the file blocks and, besides being unforgeable they have other properites dicuss in details in [13]. Key components of this scheme are the homomorphic verifiable tags. They allow to verify data possession without having access to the actual data file.

A PDP scheme is a collection of four polynomial-time algorithms:

- KeyGen(1^k) is a probabilistic key generation algorithm that is run by the client to setup the scheme. It takes a security parameter k as input, and returns a pair of matching public and secret keys(pk , sk).

- TagBlock (pk , sk , m) is a (possibly probabilistic) algorithm run by the client to generate the verification metadata. It takes as inputs a public key pk , a secret key sk and a file block m , and returns the verification metadata V .
- GenProof (pk , F , $chal$, $\sum$) algorithm is run by the server in order to generate a proof of possession. It takes as inputs a public key pk , an ordered collection F of blocks, a challenge $chal$ and an ordered collection $\sum$ which the verification metadata corresponding to the blocks in F . It returns a proof of possession V for the blocks in F , that are determined by the challenge $chal$.
- CheckProof(pk , sk , $chal$, V) algorithm is run by the client in order to validate a proof of possession. It takes as inputs a public key pk , a secret key sk , a challenge $chal$ and a proof of possession V . It returns whether V is a correct proof of possession for the blocks determined by $chal$.

Th researcher measure the performance of E-PDP and the benefits of sampling based on their implementation of E-PDP in Linux and discussion the results by concrete example.

3.2.3.2 Compact Proofs of Retrievability

Researcher in [14] they are give the first proof-of-retrievability (POR)schemes with full proofs of security against arbitrary adversaries.

Their first scheme has the shortest query and response of any proof-of-retrievability with public verifiability and is secure in the random oracle model. The second scheme has the shortest response of any proof-of-retrievability scheme with private verifiability (but a longer query), and is secure in the standard model. Both schemes rely on homomorphic properties to aggregate a proof into one small authenticator value. The scenario of the first schema: the client breaks an erasure encoded file into n blocks $m_1, m_2 ..., m_n \in Z_p$ for some large prime p . The user

authenticates each block as follows. He chooses a random $\alpha \in Z_p$ and PRF key k for function f. These values serve as his secret key. he calculates an authentication value for each block i as: $\sigma_i = f_k(i) + \alpha m_i \in Z_p$. The blocks and authenticators are stored on the server. The proof of retrievability protocol is as follows. The verifier chooses a random challenge set I of l indices along with l random coeficients in Z_p Let Q be the set of $\{(i, v_i)\}$ of challenge index of coefficient pair. The verifier sends Q to the prover.

The prover then calculates the response, a pair (α, μ), as:

$\sigma \leftarrow \sum_{(i,v_i)\in Q} v_i.\sigma_i$ and $\mu \leftarrow \sum_{(i,v_i)\in Q} v_i.m_i$ .Now verifier can check that the response was correctly formed by checking that $\sigma = \alpha \cdot \mu + \sum_{(i,v_i)\in Q} v_i \cdot f_k(i)$.

The second scheme is publicly verifiable. It follows the same framework as the first, but instead uses The Boneh, Lynn, Shacham (BLS) scheme in paper [15] which called as BLS signatures for authentication values that can be publicly verified. The structure of these signatures allows for them to be aggregated into linear combinations as above.

They prove the security of this scheme under the computational Diffie-Hellman assumption over bilinear groups in the random oracle model.

Let $e: G \times G \longrightarrow G_t$ be a computable bilinear map with group G support being Z_p. A user's private key is $x \in Z_p$ and his public key is $v = g^x \in G$ along with another generator $u \in G$. The signature on block i is $\sigma_i = \left[H(i) u^{mi}\right]^x$.

On receiving query $Q = \{(i, v_i)\}$, the prover computes and sends back $\sigma \longleftarrow \prod_{(i,v_i)\in Q} \sigma_i^{v_i}$

and $\mu \leftarrow \sum_{(i,v_i)\in Q} v_i.m_i$ The verification equation is: $e(\sigma, g) = e(\prod_{(i,v_i)\in Q} H(i)^{v_i} \cdot u^{\mu}, v)$

This scheme has public verifiability: the private key i is required for generating the authenticators $\{\sigma_i\}$ but the public key v is sufficient for the verifier in the proof-of-retrievability protocol.

Proof-of-retrievability protocol.

3.2.3.3 Zero-knowledge proofs of retrievability

Another proofs of retrievability schema introduce in [16] the verification process of this scheme requires a low, constant amount of overhead, which minimizes communication complexity.

proofs of retrievability is a cryptographic proof technique for a storage provider to prove that clients' data remain intact. It is necessary for cloud service providers to make use of the (POR) technique to provide a secure management of their storage services. In this paper they introduce the first formal definition of interactive proofs of retrievability (IPOR) on the standard model of interactive proof systems. In terms of this definition, they provide a practical zero-knowledge POR (ZK-POR) solution to prevent data leakage in the public verification process. Also they prove the soundness and zero-knowledge propertis of this scheme by constructing a polynomial-time knowledge Extractor, having rewindable black-box access to the prover, under the computational Diffie-Hellman (CDH) assumption.

Construction of ((Interactive-POR schema)

An interactive proof of retrievability scheme S is a collection of two algorithms and an interactive proof system, $S = (K, T, P)$.

- KeyGen(1^k) : It takes a security parameter k as input, and returns a pair of matching public and secret keys(pk , sk).
- TagBlock (sk , F): It takes as inputs a secret key sk and a file F , and returns the triples $(\varsigma, \phi, \sigma)$, where ς denotes the secret used to generate the verification tags, ϕ is the set of public verification parameters u and index information X , i.e., $\phi = (u, X)$ = (u, χ); σ denotes the set of verification tags;

Proof (P , V) It is a protocol of proof of retrievability between a prover (P) and a verifier (V). At the end of the protocol run, V returns, $V\{0 \mid 1\}$where 1 means the file is correctly stored on the server. It includes two cases:

- $(P(F, \sigma), V(sk, \varsigma))$ is a private proof, where P takes as input a file F and a set of tags σ , and V takes as input a secret key sk and a secret of tags ς .
- $(P((F, \sigma), V), (pk, \varphi))$ is a public proof, where P takes as input a file F and a set of tags σ , and a public key pk and a set of public parameters ϕ are the common input between P and V , where $P(x)$ denotes the subject P holds the secret x and $(P, V)(x)$denotes both parties P and V share a common data x in a protocol.

An IPOR is called zero-knowledge proof of retrievability (ZK-POR) if the completeness, knowledge soundness, and zero-knowledge property hold.

3.3 The adopted Algorithm

After discuss in section 3.2 the general categories of a versatile encryption scheme. Then 3.2.1.2displayed the algorithms. In section 3.2.1.2 shows literature of the adopted algorithm.

In this section discuss the chosen verification of adopted algorithm as points of strengths in the adopted algorithm which as reported in [8]:

- ***Provably Secure***: the un-trusted server (CPs) cannot learn anything about the plaintext given only the ciphertext. The algorithm achieved this property by ensures that pseudorandom function (F) and the pseudorandom generator (G) which generate T_i are secure. Also that only (CCs) generate the pseudorandom stream $(T_i,.....,T_l)$ and no one else can decrypt.
- ***Controlled Searching***: the un-trusted server (CPs) cannot search for a word without the user's (CCs) authorization.

  To choose keys as $K_i = f_{ki2}(L_i)$ requires that $Ki2$ be chosen uniformly randomly by (CCs) and never be revealed.

  If the (CCs) want to allow (CPs) to search for the word W, he reveals $f_{ki2}(L_i)$ and W to (CPs). This allows (CPs) to identify all the locations where W might occur, but reveals absolutely nothing on the locations i where $W_i \neq W$. This attains the controlled searching.
- ***Hidden Queries:*** the (CCs) may ask the un-trusted server to search for a secret word without revealing the word to the server.

  This property achieved by pre-encrypt to all words. Also encrypts search word.

  The (CCs) to search about word W_i send to (CPs) just $< X_i, K_i >$ calculates by equation $(X_i = E_{k_3}(W_i), K_i = f_{k_2}(X_i))$.

- ***Query Isolation:*** meaning that the un-trusted server (CPs) learns nothing more than the search result about the plaintext.

  The algorithm provides query isolation, meaning that even when a single key is revealed, no extra information is leaked beyond the ability to identify the positions where the corresponding word W_i occurs.

This property mean the algorithm is prevents the (CPs) from disclosed any information than the search result.

As the result of these discussions the algorithm presents cryptographic schemes that enable searching on encrypted data without leaking any information to the untrusted server (CPs).

The algorithm described in [8] also are simple and fast (More specifically, for a document of length n, the encryption and search algorithms only need $O(n)$ stream cipher and block cipher operations); and they introduce almost no space and communication overhead.

The schemes take the form of probabilistic searching: a search for the word W returns all the positions where W occurs in the plaintext, as well as possibly some other erroneous

positions. They may control the number of errors by adjusting a parameter in the encryption algorithm; each wrong position will be returned with probability about $1/2^m$ so for a ℓ -word document, they expect to see about $\ell/2^m$ false matches. The user will be able to eliminate all the false matches (by decrypting), so in remote searching applications, false matches should not be a problem so long as they are not so common that they overwhelm the communication channel between the user and the server.

3.4 Related Work

in this section we discuss some related work to this research.

3.4.1 Related Work in Cloud Computing

In this paper [17], Ming et al proposed a novel and practical framework for fine-grained data access control to personal health record (PHR) data in cloud computing environments, under multi owner settings.

The framework divides the users in the whole PHR system into multiple security domains (SDs), and for each SD they introduced one or more authorities which govern attribute-based credentials for users within that SD.

There are two categories of SDs: public domains (PUDs) and personal domains (PSDs). A PUD usually contains a large number of professional users, and multiple public attribute authorities (PAA) that distributive govern a disjoint subset of attributes to remove key.

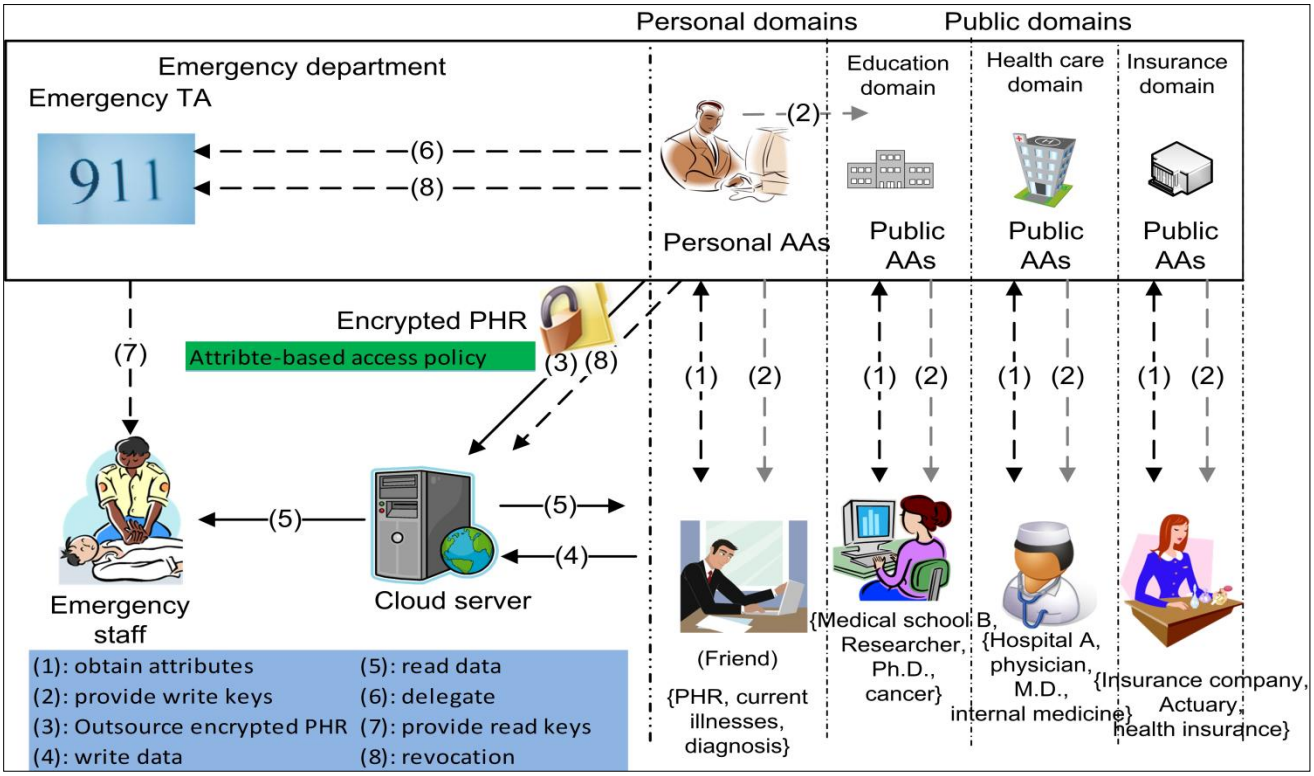

Figure 3.2: The Proposed Multi-Owner, Multi-Authority, and Multi-User Framework for access control of PHR in cloud computing. [17]

The scenario of the framework is: step 1) In Figure 3.2 key distribution. Users first obtain attribute-based keys from their AAs. In Step 2) the AAs distribute write keys that permit users in their SD to write to some patients' PHR.

PHR Access illustrated in step 3) first, the owners upload ABE-encrypted PHR files to the cloud server. Step 5) the readers download PHR files from the server, and they can decrypt the files only if they have suitable attribute-based keys. Step 4) the writers will be granted write access to someone's PHR, if they present proper write keys.

Step 8) shows user revocation, when an emergency happens, the regular access policies may no longer be applicable. To handle this situation, break-glass access is needed to access the victim's PHR. In their framework, each owner's PHR's access right is also delegated to an emergency department (ED, step 6). To prevent from abuse of break-glass option, the emergency staff needs to contact the ED to verify her identity and the emergency situation, and obtain temporary read keys step 7). After the emergency is over, the patient can revoke the emergent access via the ED.

The framework addresses the unique challenges brought by multiple PHR owners and users, in that we greatly reduce the complexity of key management when the number of owners and users in the system is large.

In this paper [18], Remya proposed an efficient, secure and privacy preserving keyword search scheme which supports multiple users with low computation cost and flexible key management. The proposed model called efficient and privacy preserving multi user keyword search for cloud storage services.

The system consists of four different entities: data owner, data user, key server and the cloud server .Proposed scheme works as shown in Figure 3.3.

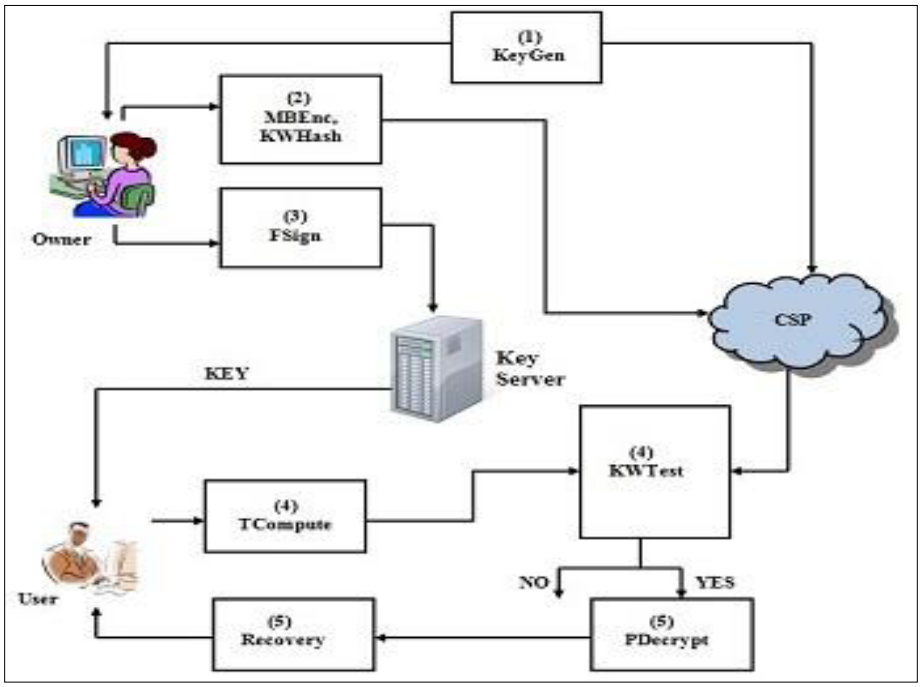

Figure 3.3: Working Process of Proposed Scheme [18]

It has the following advantages: 1) it supports keyword search in encrypted form. The cloud server could determine which all documents contain the specified keyword without revealing anything about the contents of document or the keyword searched. 2) In this scheme, the service provider will participate in the partial decipherment of the cipher text, thus reducing the computational overhead of the user. 3) In this scheme, same keywords are encrypted to different cipher text for different documents thus reducing redundancy and avoiding the chance of statistical attack on keyword cipher text. 4) The scheme also supports multiple users. The user who searches for document may be different from the users who encrypt and store it in cloud. User authorization is also provided.

Gowri et al [19] proposed a patient-centric model for data access control to PHRs stored in semi-trusted servers. Figure 3.4 show the framework of this model.

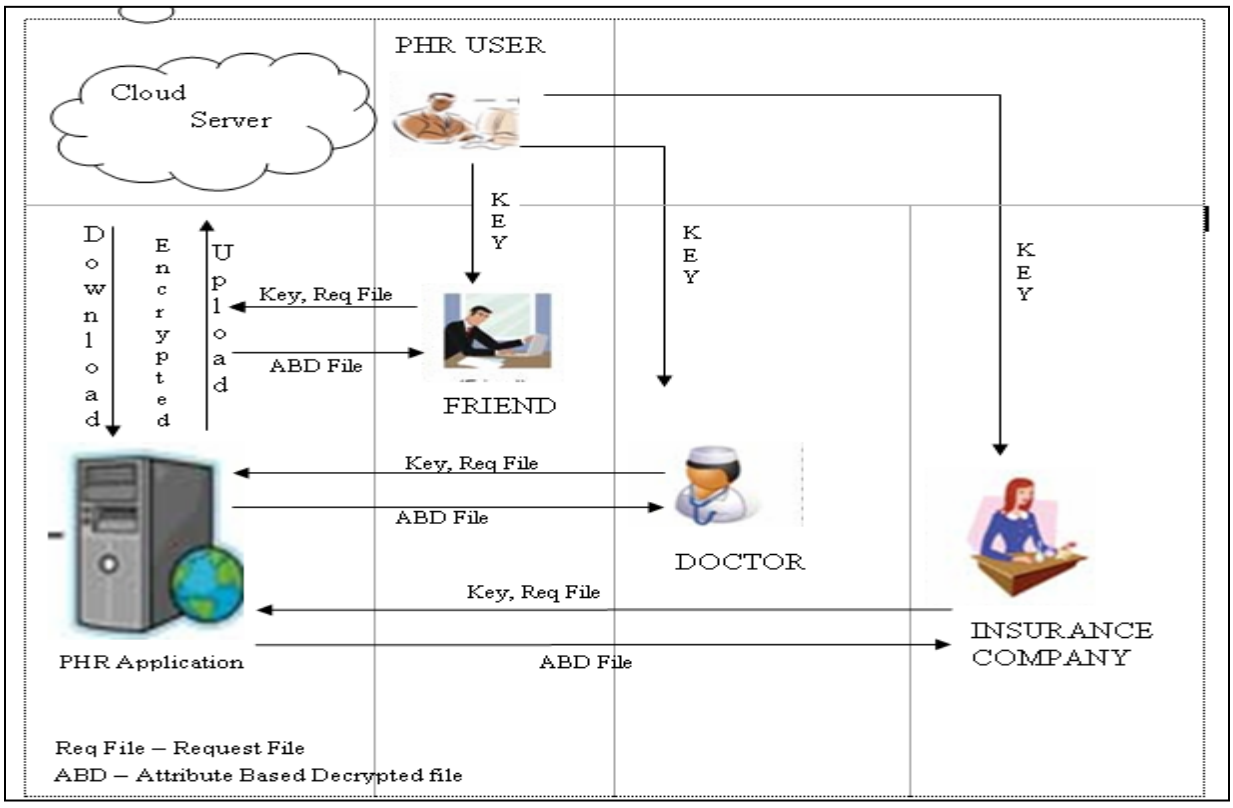

Figure 3.4: Framework of Patient Centric Model [19]

The advantages of proposed system are:

1) There is policy management for file access, data access member can able to access the files which they have rights set by the policy. 2) Files stored in the semi-trusted cloud are in encrypted form and there is no chance of others to view the file content. 3) There is a structured way to access the file for personal & professional purpose through attribute policies and attribute based encryption and decryption.

Their main design goal of this model is to help the data owner achieve fine-grained access control on files stored by cloud servers. Also prevent cloud servers from being able to learn both the data file contents and user access privilege information.

In [20] Shaheen et al proposed system model called A Novel Method for Patient Centric Secure and Scalable sharing of PHR in Cloud Computing using Encryption. The block diagram of PHR system is shown in Figure 3.5.

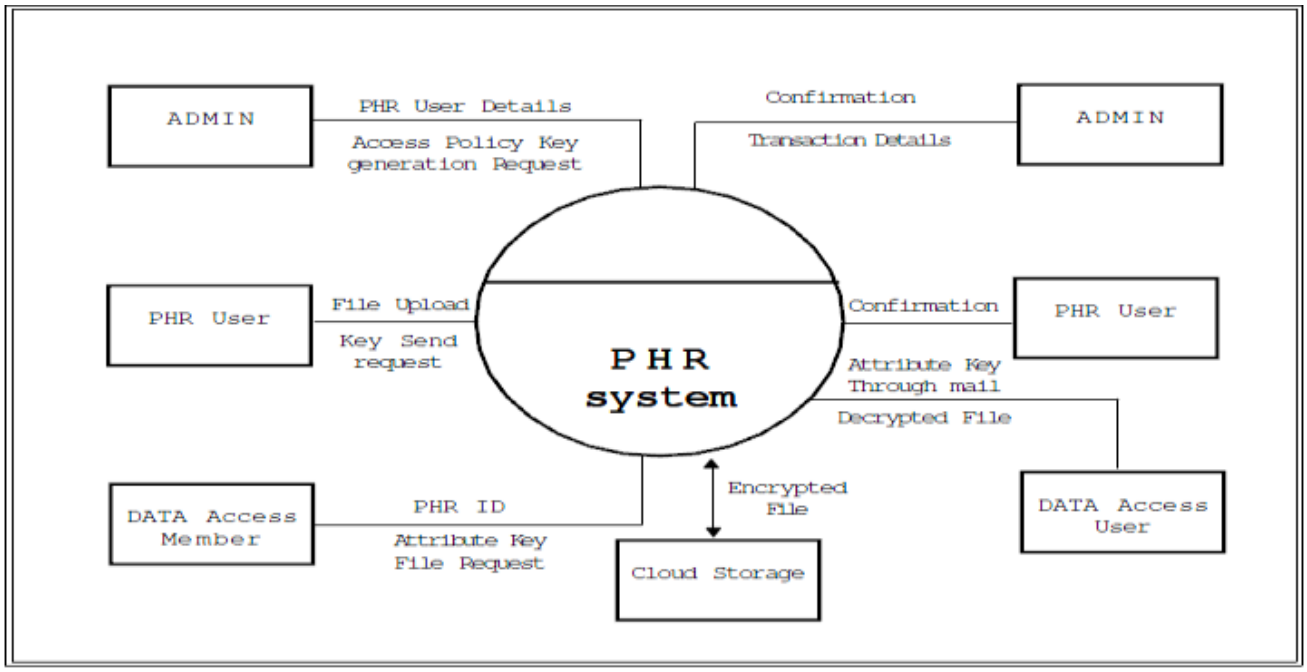

Figure 3.5: Block Diagram [20]

Algorithm for proposed system: Step 1) PHR owner will input the files he/she wants to share. Step 2) Encryption of input files is done using DES algorithm. Step 3) Encrypted files uploaded into the cloud server. Step 4) User or data access member input the attributed key. Step 5) verification of the key for its validity. Step 6) if the key is invalid, files will not decrypt. Step 7) if the key is valid, files will be decrypted. Step 8) Decrypted files will be downloaded into the user's local system.

Josh et al [21] proposed a design that they refer to as Patient Controlled Encryption (PCE) as a solution to secure and private storage of patients' medical records. PCE

allows the patient to selectively share records among doctors and healthcare providers. The design of the system is based on a hierarchical encryption system.

The design prevents unauthorized access to patients' medical data by data storage providers, healthcare providers, pharmaceutical companies, insurance companies, or others who have not been given the appropriate decryption keys.

The goals of PCE are: 1) the patient should have confidence that the administrators of the health data server will not learn anything about the patient's record. 2) Guarantee security in the case of server compromise: even if the server is compromised or stolen, the patient should be certain that his data has not been leaked and 3) guarantee correctness of the health record: the patient should be able to verify that no one has tampered with his record. 4) Efficient access to patient health records, 5) easy sharing of parts of the record, and 6) efficient searching over records.

In paper [22] Ming et al they address the problem of authorized private keyword searches (APKS) over encrypted data in cloud computing, where multiple data owners encrypt their records along with a keyword index to allow searches by multiple users. Figure 3.6 show the multiple owner.

To limit the exposure of sensitive information due to unrestricted query capabilities, They propose a scalable, fine-grained authorization framework called authorized private keyword searches over encrypted Personal Health Records in cloud computing where users obtain their search capabilities from local trusted authorities according to their attributes.

Then they propose two novel solutions for APKS over encrypted data based on a recent cryptographic primitive, hierarchical predicate encryption (HPE), where in

the first one they enhance the search efficiency using attribute hierarchy, and in the second they enhance the query privacy via the help of proxy servers which is shown in Figure 3.7.

Their solutions also support efficient multi-dimensional range query, search capability delegation and revocation. In addition, They implement their solution on a modern workstation; the results show that APKS achieves reasonable search performance. Figure 3.7 show APKS schema.

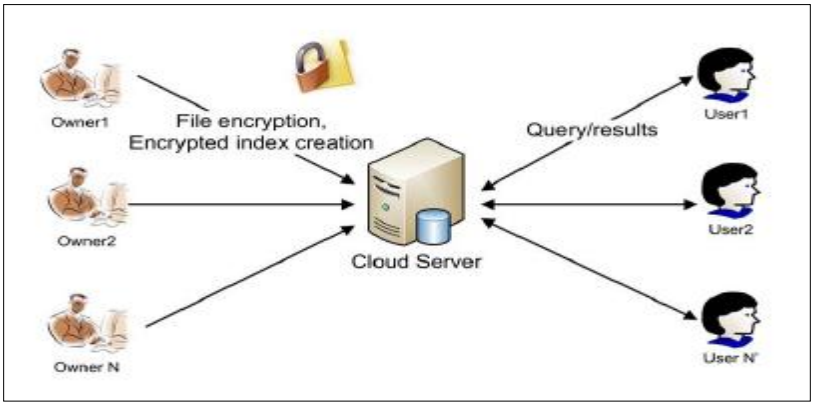

Figure 3.6: System Model for multi-owner data outsourcing in cloud computing [22]

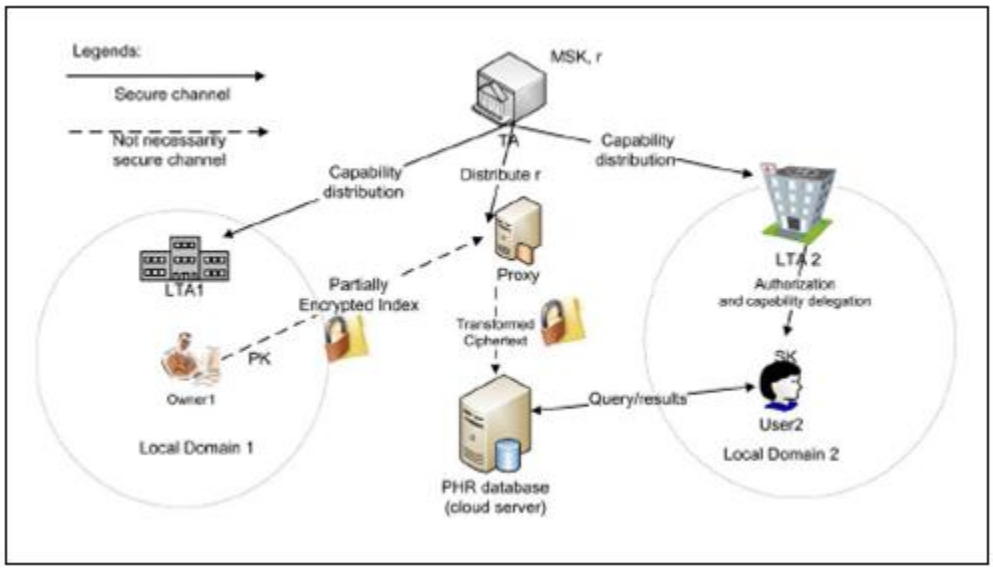

Figure 3.7: The Enhanced Framework for APKS+ that preserves query privacy [22]

Ahmed et al in paper [23], they address the challenge of data management in wireless sensor networks for patient supervision. They proposed a secure and scalable architecture for collecting and accessing large amount of data generated by medical sensor networks. They leverage cloud computing technology to dynamically scale storage resources via on demand provisioning.

Their contributions in this work are many folds: First, They proposed a new cloud based architecture for medical wireless sensor networks. Second, they showed how they guarantee the confidentiality and the integrity of outsourced medical data without involving patients or doctors interventions. Third, they proposed an innovative access control which allows implementing complex and dynamic security policies necessary to medical application while reducing the management and processing overhead. More specifically, they combine Cipher text Policy Attribute Based Encryption (CPABE) and symmetric encryption to achieve fine grained access with low computation overhead.

Unlike existing patient-centric systems, security configuration and key management in their solution are totally transparent to users (patients and doctors) and do not require their interventions.

The proposed architecture described in Figure 3.1Figure 3.8. This architecture considers two categories of users, healthcare professionals and patients, and is composed of the following components: (1) the wireless Sensor Network (WSN) which collects health information from patients, (2) the monitoring applications which allow healthcare professionals to access to stored data, (3) the Healthcare Authority (HA) which specifies and enforces the security policies of the healthcare institution and (4) the cloud servers which ensure data storage. By storing data on the cloud, their architecture offers virtually infinite storage capacity and high scalability.

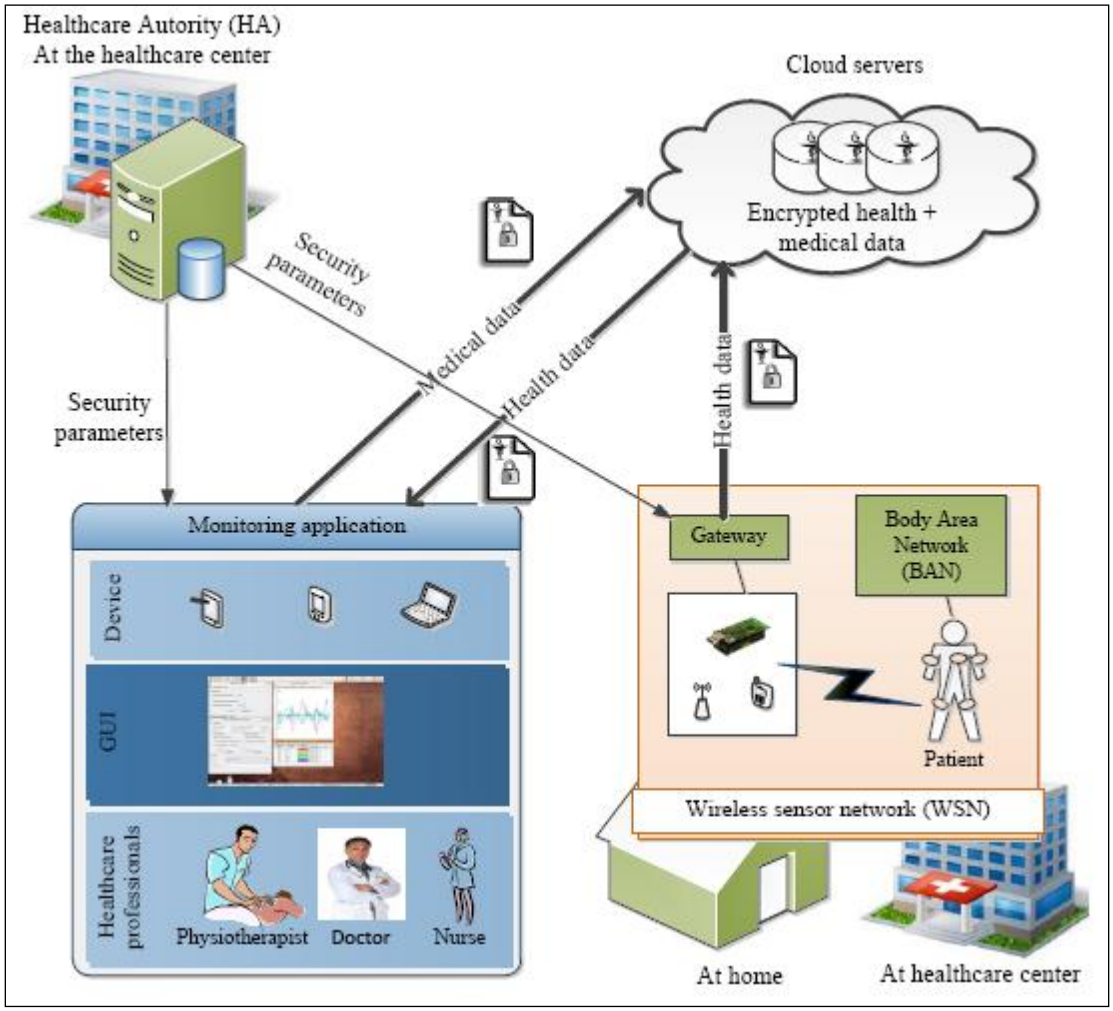

Figure 3.8: the Proposed Architecture [23]

Mehmet in paper [24], consider the problem of searching on encrypted data. He presents an extensive literature survey on this subject and provides detailed analyses for the existing solutions; keyword and non-keyword based approaches. He identify memory overhead as one of the problems and suggest some practical modifications for reducing it. He proposes to extend the existing non-keyword based scheme by using a keyed hashing function. Furthermore, he shows that we can substantially reduce the memory overhead by combing the non-keywords in

the document. These improvements are motivated by the practical applications and optimally combine the advantages of both the keyword and non-keyword based techniques. He implements the proposed improvements on the ENRON Email Corpus and show that the effectiveness of the proposed scheme is greater than the existing schemes, especially in terms of the memory overhead and encryption time.

In paper [25], Seny et al discuss how to build Cryptographic Cloud Storage. They consider the problem of building a secure cloud storage service on top of a public cloud infrastructure where the service provider is not completely trusted by the customer. They describe at a high level, several architectures that combine recent and non-standard cryptographic primitives in order to achieve their goal. They survey the benefits such architecture would provide to both customers and service providers and give an overview of recent advances in cryptography motivated specifically by cloud storage. To address the concerns outlined above and increase the adoption of cloud storage, they discuss designing a virtual private storage service based on recently developed cryptographic techniques. Such a service should aim to achieve the best of both worlds by providing the security of a private cloud and the functionality and cost savings of a public cloud. More precisely, such a service should provide (at least): confidentiality: the cloud storage provider does not learn any information about customer data and integrity: any unauthorized modification of customer data by the cloud storage provider can be detected by the customer while retaining the main benefits of a public storage service: availability, reliability, efficient retrieval and data sharing.

Seny et al in paper [26], they focus on the problem of constructing practical SSE schemes for the purpose of designing practical searchable cryptographic cloud storage systems [25].

Searchable symmetric encryption (SSE) allows a client to encrypt its data in such a way that this data can still be searched. The most immediate application of SSE is to cloud storage, where it enables a client to securely outsource its data to an un-trusted cloud provider without sacrificing the ability to search over it.

Any practical SSE scheme, however, should (at a minimum) satisfy the following properties: sub linear search time, security against adaptive chosen keyword attacks, compact indexes and the ability to add and delete files efficiently. Unfortunately, none of the previously-known SSE constructions achieve all these properties at the same time. This severely limits the practical value of SSE and decreases its chance of deployment in real-world cloud storage systems. To address this, they propose the first SSE scheme to satisfy all the properties outlined above.

Mehmet et al, in paper [27], proposed an efficient similarity searchable symmetric encryption scheme to search over encrypted data. To do so, they utilized locality sensitive hashing (LSH) which is widely used for fast similarity search in high dimensional spaces for plain data. They proposed LSH based secure index and a search scheme to enable fast similarity search in the context of encrypted data. The basic idea of LSH is to use a set of hash functions to map objects into several buckets such that similar objects share a bucket with high probability, while dissimilar ones do not. LSH uses locality sensitive function families to achieve this goal. In addition, they provide a real world application of the proposed scheme and verify the theoretical results with empirical observations on a real dataset.

The performance analysis shows that our commitment/challenge/response protocol transmits a small, constant amount of data, which minimize network communication.

3.5 Summery

This chapter defined a versatile encryption schema, categories and some algorithm for each category as example. Then display the strength points of adopted algorithm as chosen verification. Then discuss and displayed related work.

CHAPTER 4

Research **Methodology**

4.1 Introduction

This chapter presents and lists down all the steps that associated and embraced in order to conduct this research.

4.2 Research Process

This research divides the work into three main steps as shown in Figure 4.1: starts by determines security requirements in cloud computing environments. Step two discusses the implementation of versatile encryption algorithm. Step three validates and evaluates versatile encryption algorithm. These steps sometimes may be interleaved. We will discuss these steps in the following sections.

4.2.1 Security requirements definition and determination

In this step, investigates the information available to the literature view of the cloud computing environment and finds out the details in section 2.2. The greatest challenge facing institutions to maintain the confidentiality of the data and the data is the key to investment. This challenge limits the adoption of cloud computing technology in spite of the consensus of institutions on the benefits of them. In this research we will try to resolve this problem by uses a versatile encryption scheme which we discussed in CHAPTER 3. Then we have presented the related work in section 3.43.4.

Before discusses how to use a versatile encryption scheme in this research must firstly defines the security requirements accurately for the cloud computing

environment that we will need to achieve them through an analysis of the expected risk which will be introduced in section 5.2.1.

4.2.2 Implementation of the adopted algorithm

This step explains the implementation of versatile encryption algorithms to solve the problem mentioned in previously step. The output of this step program used this type of encryption. This implementation will present their components and discuss in detail in section 5.3.

4.2.3 Validation and evaluation phase

In this step validates and evaluates the effectiveness of versatile encryption algorithms with cloud computing environment. Assessment is based on the requirements discussed in the first step. From this discussion we will come out of the results which will be displayed in the section 5.4.

4.3 Summary

In this chapter we demonstrate how the research process was carried out in three steps: Security requirements definition and determination, Develop the proposed scheme and third step is validation and evaluation (including the case study implementation).

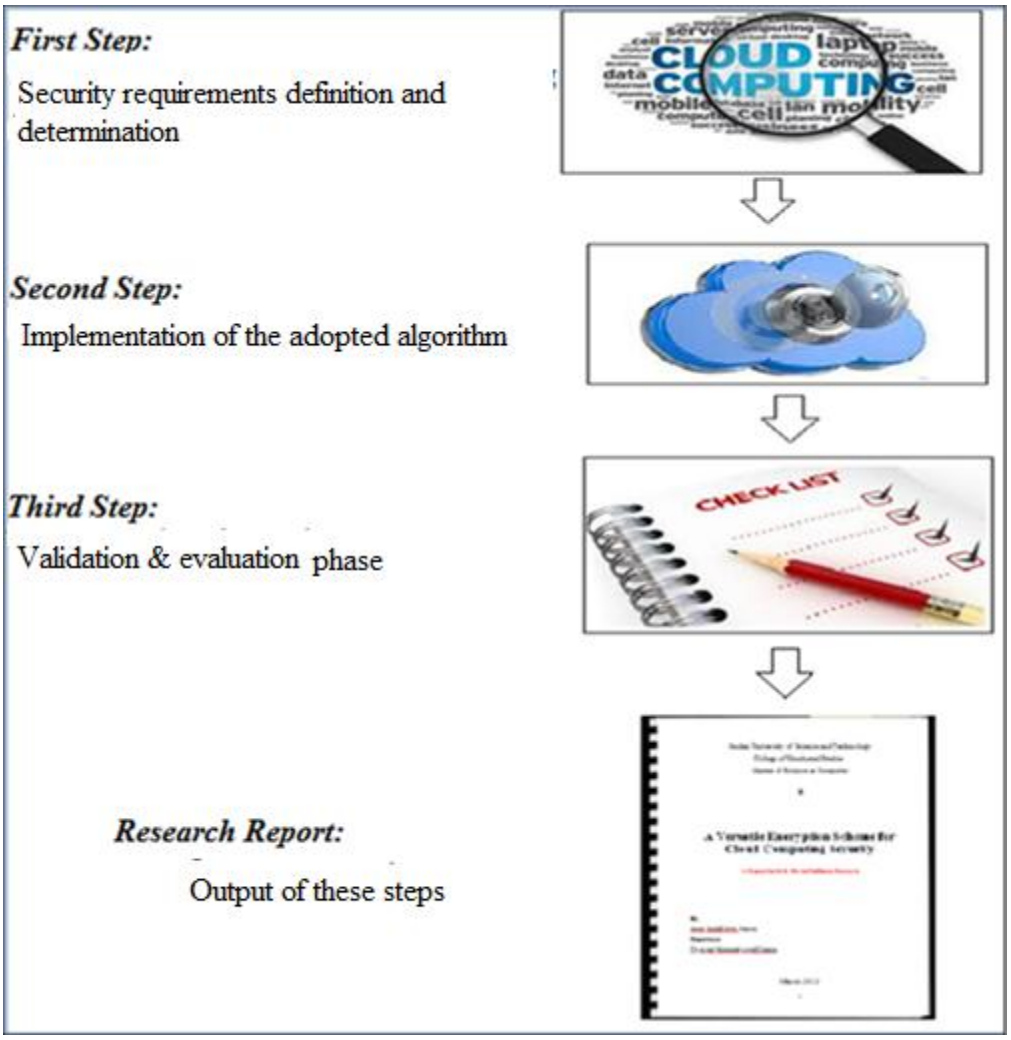

Figure 4.1: Research Process

CHAPTER 5

Implementation & Evaluation of the Adopted Algorithm

5.1 Introduction

In this chapter we determine security requirements in cloud computing environments by introduces risk analysis in this environment .Then we discuss the implementation of versatile encryption algorithm as effective solution to risk in cloud environment which achieved the security requirement.

5.2 Security requirements definition and determination

In this section we determine security requirement by three steps: first step analyze expected risk of data in cloud environment. Second step we determine security services to avoid the expected risk. Third step we list down security mechanisms which achieves through them the security services.

5.2.1 Risk Analysis

To determine the security requirements for the cloud computing environment first we need to analyze the potential risks to see what the vulnerabilities. Top security risks in cloud environment are [10, 32]:

5.2.1.1 Loss of Governance

In using cloud infrastructures, the (CCs) necessarily cedes control to the (CPs) on a number of issues which may affect security. At the same time, SLAs may not offer a commitment to provide such services on the part of the cloud provider, thus leaving a gap in security defenses.

5.2.1.2 Cheap Data and Data Analysis

The (CCs) to benefit from the services provided by the (CPs), the (CCs) is compelled to store their data in the cloud environment. The rise of cloud computing has created enormous data sets that can be monetized by applications such as advertising. The (CPs) may use the (CCs) data for the purposes of data mining without the (CCs) permission.

Google, for instance, leverages its cloud infrastructure to collect and analyze consumer data for its advertising network [6].

5.2.1.3 Transitive Nature

Another possible concern is that the contracted (CPs) might itself use subcontractors, over whom the (CCs) has even less control, and who also must be trusted.

5.2.1.4 (CPs) Espionage

The (CCs) are worried from theft the company proprietary information by the (CPs). Corporate users of these services are concerned about confidentiality and availability of their data

5.2.1.5 Contractual Obligations

One problem with using another company's infrastructure besides the uncertain alignment of interests is that there might be legal implications. The (CPs) cannot provide evidence of their compliance with the relevant requirements.

5.2.1.6 Management Interface Compromise:

Customer management interfaces of a public cloud provider are accessible through the internet and mediate access to larger sets of resources (than traditional hosting

providers) and therefore pose an increased risk, especially when combined with remote access and web browser vulnerabilities.

5.2.1.7 Data Protection

Because the data is a key element in the business you must store it in secure manner. Cloud computing poses several data protection risks for (CCs), (CPs) and (SPs). In some cases, it may be difficult for the cloud customer to be sure that the data is handled in a lawful way. This problem is exacerbated due to multiple reasons are as follows

First reason, the (CCs) in a cloud computing environment participates the resources with other (CCs), making (CCs) data stores in the same place. (CPs) must be a secure (CCs) data and not to allow the (CCs) access to other (CCs) data.

second reason, the (CCs) how to make sure that the (CPs) who adopted are applied the requirements in the contract and the problem is that the even if (CPs) applied requirements the (CCs) does not guarantee that the data kept safe the (CCs) need monitoring of their data in (CPs).

Third reason, The (CCs) must be adopt (CPs) to achieve the required confidentiality requirements to data in addition to that, the (CCs) also secures its data because as we mentioned in the first problem that the (CPs) is himself considered the risk to the (CCs) should be bench him.

Fourth reason, the (SPs) in the cloud computing may be another party other than the (CPs) and thus we have three different entities that must secure each side than the other. The (CCs) became addition to securing data from the (CPs) and other (CCs) it secured data also from the (SPs) in an environment of cloud computing,

(SPs) it may be several various parties provided the services to (CPs) which increases the risk of (CCs) data.

And also a (CPs) in addition to securing the environment of computing from (CCs) and then securing the (CCs) data from each other, try to securing the environment of computing and (CCs) data from the (SPs) as well as the (SPs) must securing environment from (CCs) and the (CPs).

Fifth reason, environment of cloud combines between different parties, where large amounts of data, making it vulnerable to hacker.

5.2.1.8 Insecure or Ineffective Deletion of Data

In the case of multiple tenancies and the reuse of hardware resources, this represents a higher risk to the customer than with dedicated hardware.

5.2.1.9 Malicious Insider

Examples include (CPs) system administrators and managed security service providers.

5.2.2 Security Services

The second step after we analyze the potential risks in the above section we show the services that we want to achieve. The main services solve most of risk: 1) Confidentiality this service means assures that private or confidential information owned by (CCs) is not made available or disclosed to unauthorized individuals such as (CPs), (SPs) or other (CCs) in environment of cloud computing.

2) Integrity is another service which means assures that data and programs owned by (CCs) are change only in a specified and authorized manner in cloud computing.

There are many security mechanisms in each security service which solve one or more security risk.

5.2.3 Security mechanisms

In this research we proposed a versatile encryption schema highlights in CHAPTER 2 as mechanism to achieve the services mentioned in the above section which to solve all risk mentioned in section 5.2.1.

5.3 Implementation of the Adopted Algorithm

After review the researches in a versatile encryption schema in CHAPTER 3, this section describe the implementation of the adopted algorithm which are described in paper [8].

Scenario to explain the steps of algorithm: client when he wants to store his data in secure manner, try to dealing with the (CPs) which applies a versatile encryption algorithms to achieve this goal. In this research has been applied one of them which are mentioned in the paper [8]. When apply this algorithm there are steps to be implemented by the (CCs) and part of the steps are performed at the (CPs).

Primary objective of the (CCs) is secure data in (CPs) in this case; the service provided by (SPs) to the client is a storage space. As example assume the (CCs) wants to store his email in this storage space.

In the following sections we display and describe the adopted algorithm in more details with standards: firstly we display pseudocode of algorithm with show the key part of algorithm in source code. Secondly we demonstrate the steps of algorithm by using flow chart. Lastly we show snapshot of implementation.

5.3.1 The Pseudocode of Proposed Adopted Algorithm

The adopted algorithm has four schemes, starts by basic scheme, and then each of them improvements to the previous scheme.

The following steps describe the pseudocode of the final scheme:

The (CCs) performs the following steps to encrypt documents by using adopted algorithm and then stores the encrypted document in server:

1. The (CCs) dealing with documents as series of L words $(W_1,.....,W_l)$. The length of each word is n bits. Method called *EnterWords()* splits the document into words ;
2. The (CCs) generate L pseudorandom values $(S_1,.....,S_l)$ with length 16 bytes.

   In our implementation used algorithm SHA1PRNG to generates pseudorandom values: Method called *GenerateSi()* do this. Part of method:

   ```
   SecureRandom secureRandom = SecureRandom.getInstance("SHA1PRNG");
   byte[] si = new byte[16];
   secureRandom.nextBytes(si);
   ```

3. The (CCs) generate L K_{i3} $(Ki3_1,.....,Ki3_l)$ with length 16 bytes.

   Method called *GenerateKi3Scheme3 ()* do this. Part of method :

   ```
   KeyGenerator keygen = KeyGenerator.getInstance("AES");
   keygen.init(128);
   byte[] key = keygen.generateKey().getEncoded();
   Ki3Bytes[g]=key;
   ```

4. The (CCs) encrypt each word by K_{i3} to generate X_i with length 16 bytes.

   Using this equation $X_i = E_{ki3}(W_i)$.

   Method called *Encrypt_Wi_Using_Ki3_Scheme3 ()* do this. Part of method:

   ```
   SecretKeySpec skeySpec = new SecretKeySpec(Ki3Bytes[g], "AES");
   byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
   IvParameterSpec ivspec = new IvParameterSpec(iv);
   Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
   ```

cipher.init(Cipher.ENCRYPT_MODE, skeySpec, ivspec);

XiBytes[g] = cipher.doFinal(WiBytes[g]);

5. The (CCs) split X_i into two parts $\langle L_i, R_i \rangle$ with length bytes for each part.

   Method called *Split_Xi()* do this.

6. The (CCs) generate L K_{i2} $(Ki2_1,.....,Ki2_l)$ with length 16 bytes.

   Method called *GenerateKi2Scheme2()* do this.

7. The (CCs) generate L K_i $(Ki_1,.....,Ki_l)$ with length 16 bytes , by encrypts L_i using K_{i2}.Using this equation $K_i = f_{ki2}(L_i)$.

   Method called *Encrypt_Li_Using_Ki2_Scheme2()* do this.

8. The (CCs) encrypt S_i using K_i with length 16 bytes.

   Using this equation $encryptedS_i = F_{ki}(S_i)$.

   Method *Encrypt_Si_Using_Ki ()* do this.

9. Then generates L T_i $(T_i,.....,T_l)$object with length 32 bytes by concatenates the original S_i with encrypted S_i. Using this equation $T_i = (S_i, F_{ki}(S_i))$.

   Method *PrepareTi_Scheme3 ()* do this. Part of method:

```
for(int k=0;k<SiBytes[g].length;k++){
        Ti[k]=SiBytes[g][j1];
        j1++;
 }
 for(int k=SiBytes[g].length;k<Ti.length;k++){
        Ti[k]=encryptedSi[g][j2];
        j2++;
 }
```

10. Then generate cipher object C_i with length 32 bytes by XOR between T_i and X_i .Using this equation $C_i = W_i \oplus T_i$ .Method *GenerateCi ()* do this. Part of method:

```
for(int o=0;o<wordsindex;o++){
    CiBytes[o]=XOR(Ti2Bytes[o],XiBytes[o]);
}
```

as we mentioned earlier these 10 steps are executed by the (CCs) when they want to store their a document in a (CPs) .

In (CCs) side, when the (CCs) in any time want to search about W they can tell the (CPs) and send to it W and K_i corresponding to each location which W occur. The (CPs) can then search for W in the ciphertext by checking whether $C_i \oplus W_i$ is of the form of $\langle s, F_{K_i}(s) \rangle$ for some s .

The following steps demonstrate the search operation in the (CPs):

Method called *Search_Scheme3()* do this.

1. When the (CCs) wants to search about W sends to server W and K_i .
2. The (CPs) to search about W in the cipher documents recalculates T_i for each C_i with search word W using this equation $T_i = C_i \oplus W$. Part of search method:

   Ti2[i]=XOR(CiBytes[i],searchword);
3. Then the (CPs) split each T_i into two parts: S_i and encrypted S_i $\langle F_{k_i}(S_i) \rangle$. Part of search method:

   ```
   Ti2Half1=new byte[SiBytes[i].length];
   Ti2Half2=new byte[Ti2Bytes[i].length-SiBytes[i].length];
   for(int b=0;b<SiBytes[i].length;b++)
           Ti2Half1[b]=Ti2[i][b];
   int y=0;
   for(int b2=SiBytes[i].length;b2<Ti2Bytes[i].length;b2++){
           Ti2Half2[y]=Ti2[i][b2];
           y++;
   }//for
   ```
4. Each T_i calculated and split then decrypts S_i using K_i which send by (CCs). Method called *Decrypt_Si_Using_Ki()* do this. Part of method:

   SecretKeySpec skeySpec = new SecretKeySpec(K, "AES");

```
byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
IvParameterSpec ivspec = new IvParameterSpec(iv);
Cipher cipher=null;
 cipher = Cipher.getInstance("AES/CBC/NoPadding");
cipher.init(Cipher.DECRYPT_MODE, skeySpec, ivspec);
decryptedS= cipher.doFinal(encryptedS);
```

5. The (CPs) test whether the S_i equals decrypted S_i or no. if the test return yes the word found and otherwise the word is not found after test all the C_i. Part of search method:

```
decryptedS=Decrypt_Si_Using_Ki(key,encryptedS);
if(new String(Ti2Half1).equals(new String(decryptedS))){
        foundword=i;
       Search=true;
       break;
}
```

5.3.2 The Flow Chart of Adopted Algorithm

Steps of algorithm describes by flow chart as shown in Figure 5.2 which demonstrates the steps in details.

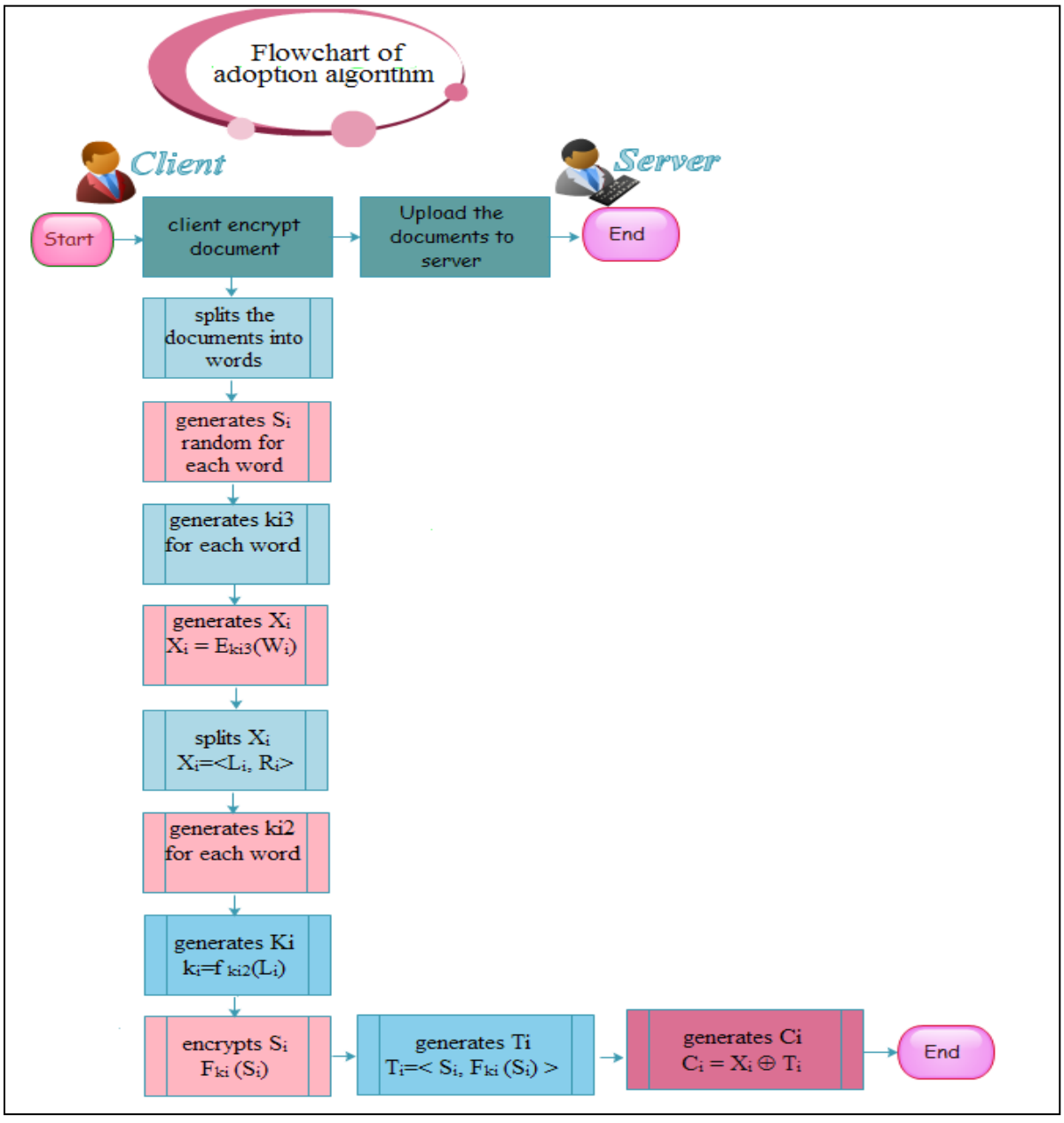

Figure 5.1: Flow Chart of Adopted Algorithm

5.3.3 Flow Chart for generate cipher in Adopted algorithm

This flow chart displays the steps which generate cipher in the Adopted algorithm in details.

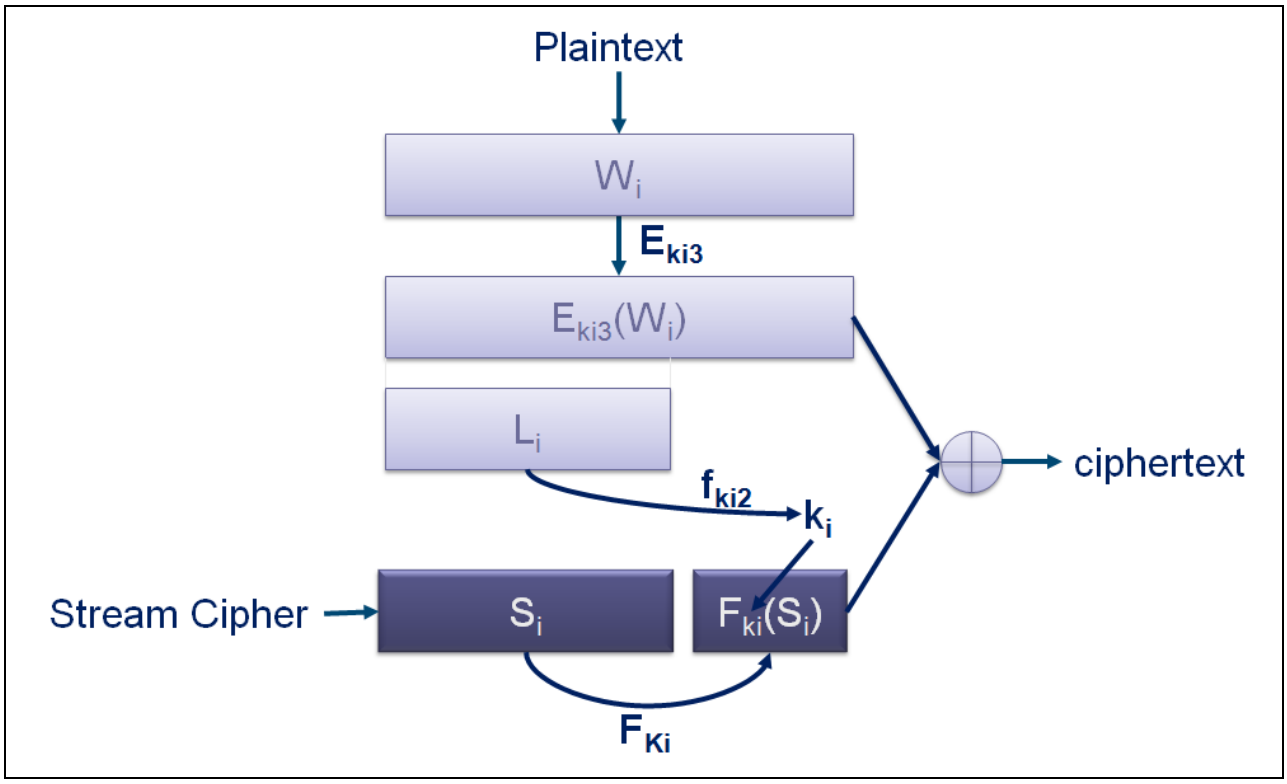

Figure 5.2: Flow Chart for generate cipher in Adopted Algorithm

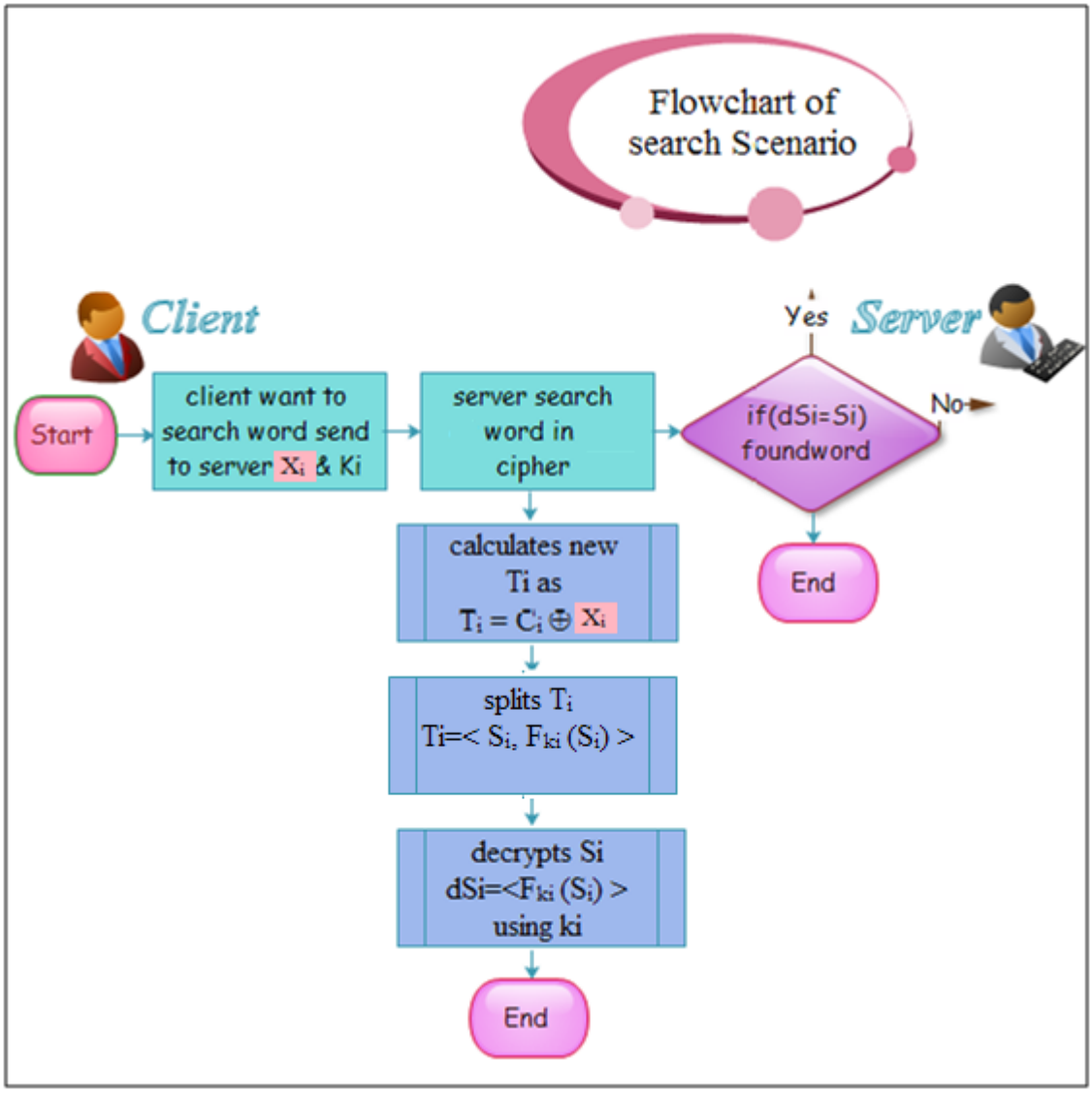

Figure 5.3: Flowchart of Search Scenario

5.3.4 Snapshot of Implementation

In this section we display snapshot of implementation.

1. When the (CCs) have file wants to store it in (CPs) executes this program as shown in Figure 5.4

Figure 5.4: implementation of versatile Encryption algorithm

2. Then click browse file to select the file from his computer this is shown in the Figure 5.5.

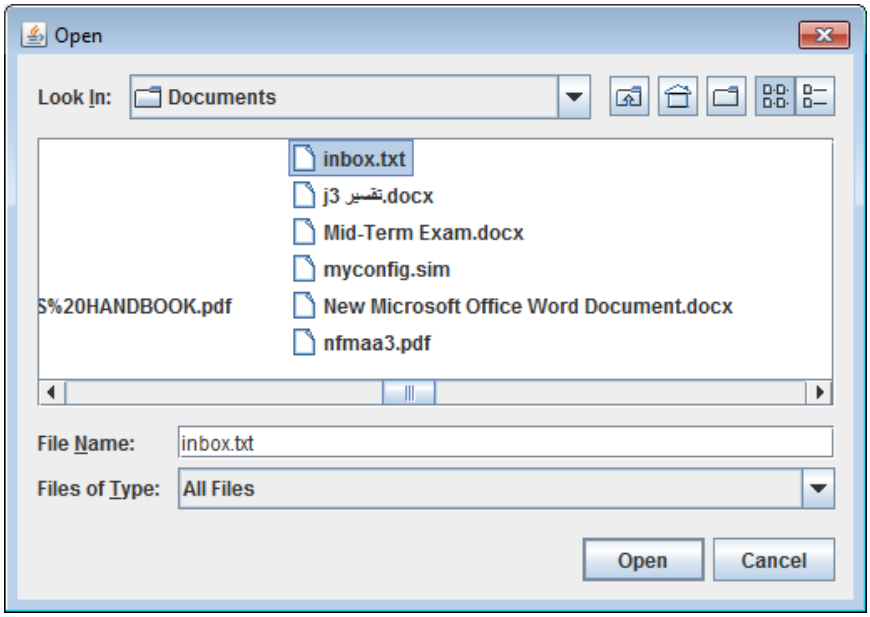

Figure 5.5: Browse File

3. After the (CCs) choose the file as shown in Figure 5.6. Chose file is named inbox.txt.

Figure 5.6: Chose File

4. Then after select file (CCs) click button Encryption & Upload File to apply the algorithm in file. Figure 5.7 shows the first step of adopted algorithm (splits file to words). Figure 5.8 last step of adopted algorithm (generates cipher).Figure 5.9 shows ciphertext for chosen file.

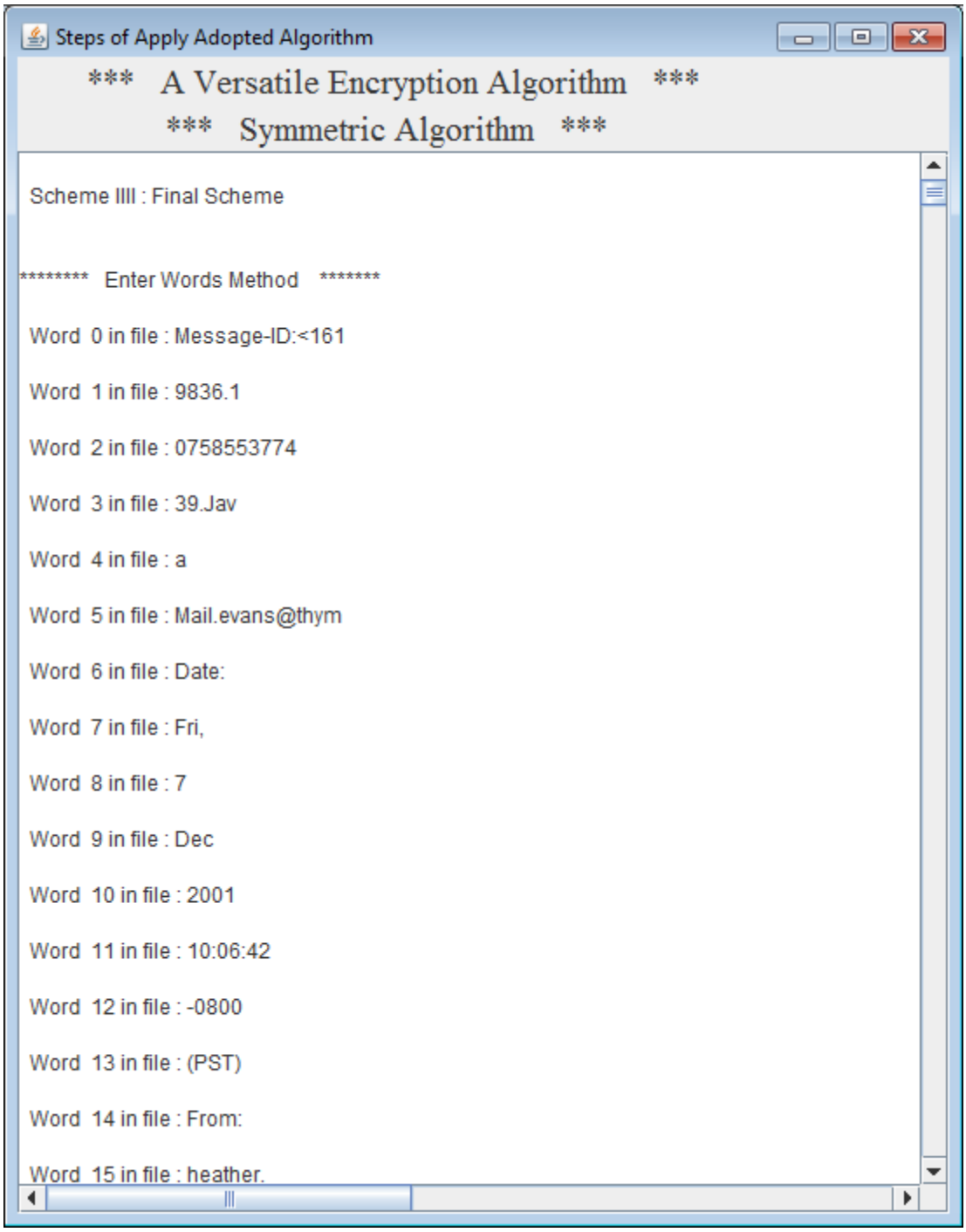

Figure 5.7: First Step of the Adopted Algorithm (splits file to words)

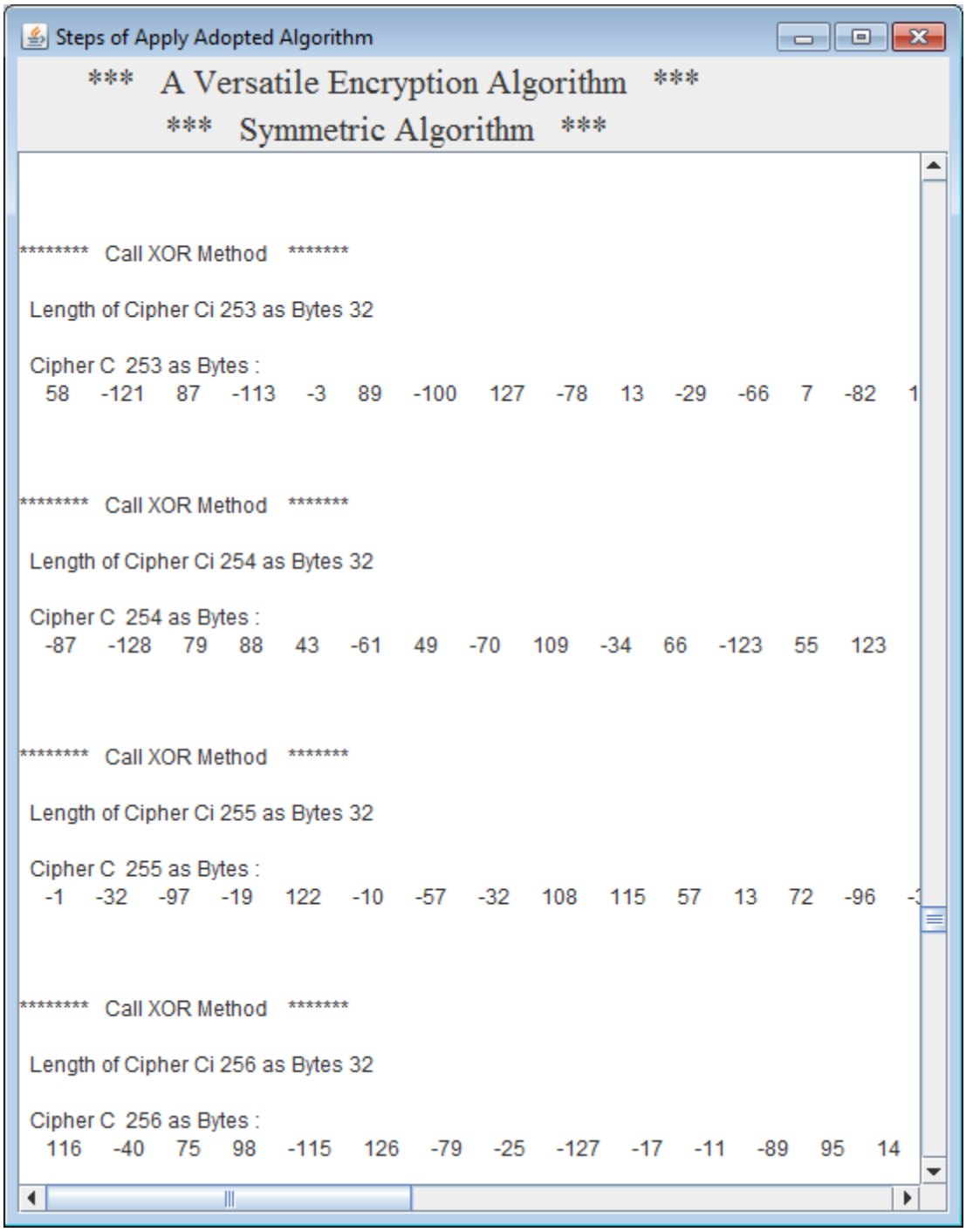

Figure 5.8: Last Step of the Adopted Algorithm (generates cipher)

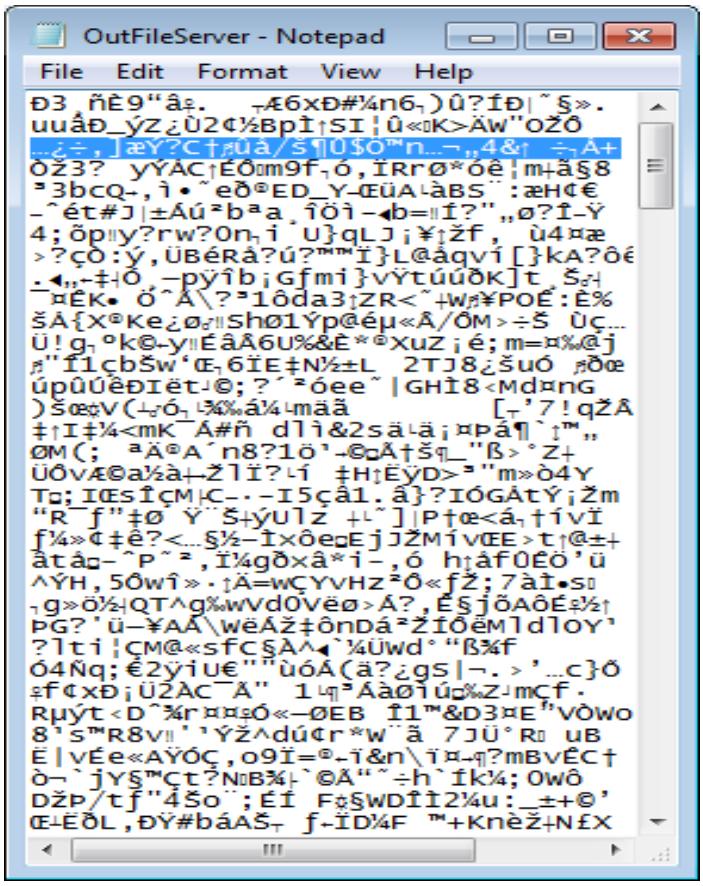

Figure 5.9: Ciphertext for chosen file

5. The (CCs) to search about specific word in cipher document. The word enters as shown in Figure 5.10 .

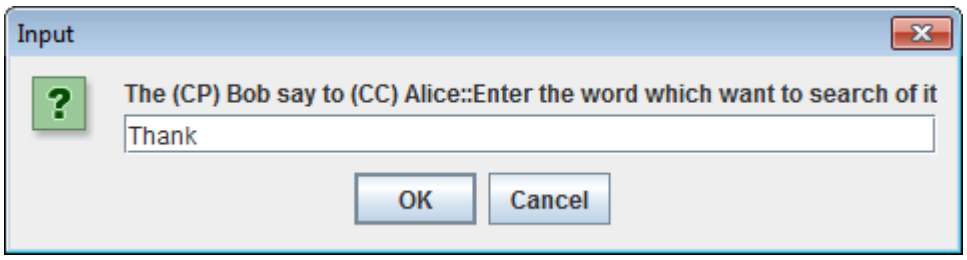

Figure 5.10: Enter Search Word

6. After the (CCs) click button ok, the (CPs) search about word in cipher then return the result of search operation to (CCs). Figure 5.11 shows the result of search about word (Thank).

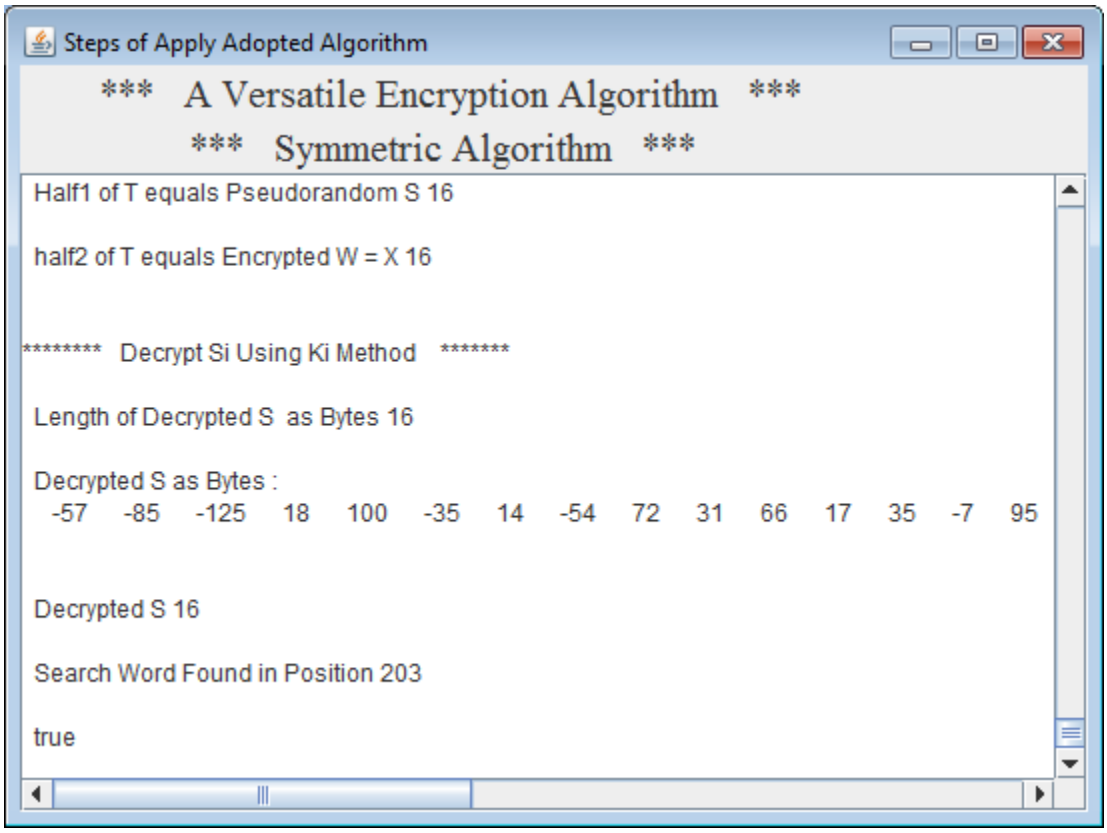

Figure 5.11: The Result of the Search Operation

5.4 Validation and evaluation phase (including the case study implementation)

The following table discusses the security risk with security requirement from the viewpoint of data only not consider the rest of the cloud computing environment:

Security risk and security services and security mechanism discussed in section 5.2.

Table 5.1: Security Risk & Security Requirment with Adopted Algorithm

Security Risk	Security Requirement	Security Mechanism Adoption Algorithm
1. Loss of Governance 2. Cheap data and data analysis 3. Transitive nature 4. (CPs) Espionage	The (CCs) control access to their data	The (CCs) control accesses to their data by encrypt it and no one can access to it. This makes (CCs) not loss governance of data and prevents the (CPs) to analysis data or espionage. Also any level of transitive the (CCs) ensure no one can access to their data in all level.
5. Contractual obligations	The (CCs) secure their data before export it to (CPs).	The (CCs) apply encryption to their data to solve contractual obligations problem by (CPs). The (CCs) ensure the data protects because it encrypts by himself not depends on obligates of contract from (CPs).
6. Management interface compromise	The (CCs) reduce the risk of their data in (CPs).	The (CCs) reduce the risk (interface compromise) when apply encryption on their data.
7. Data protection	The (CCs) ensure the data cannot be changed or disclosed by (CPs)	The (CCs) protect their data by encrypts it this mean no one can change or disclose.
8. Insecure or Ineffective Deletion	The (CCs) ensure the data secure when (CPs) reuse	The (CCs) secure their data by encrypts this means reduce the

of Data	of hardware resources.	risk of resource reusability in (CPs). Also the ineffective deletion is not problem because the data is encrypts this mean the data is not useful.
9. Malicious Insider	The (CCs) prevent their d data from malicious insider.	The (CCs) manipulate with data in encryption form which protects the data from malicious insider

As the result from the implementation and evaluation of a versatile encryption algorithm we found it protects the data from security risk which chives security features.

5.5 Background and Definitions

This section contains the background and definitions as mentioned in [8] . These definitions needed for proof of security to adopted algorithm.

The scheme requires several fundamental primitives from classical symmetric-key cryptography. Because they will prove their scheme secure, they use only primitives with a well-defined notion of security. They will list here the required primitives, as well as reviewing the standard definitions of security for them.

They measure the strength of the cryptographic primitives in terms of the resources needed to break them. They will say that an attack R-breaks a cryptographic primitive if the attack algorithm succeeds in breaking the primitive with resources specified by R, and they say that a crypto primitive is R-secure if there is no algorithm that can R-break it. Let $A:\{0,1\}^n \to \{0,1\}$ be an arbitrary algorithm and let X and Y be random variables

distributed on $\{0,1\}^n$ .The distinguishing probability of A-sometimes called the advantage of A—for X and Y is

Adv $A = |\Pr[A(X)=1] - \Pr[A(Y)=1]$ With this background, the list of required primitives is as follows:

Definitions

1. A *pseudorandom generator* G, i.e., a stream cipher.

They say that $G:\kappa_G \to S$ is a (t,e)-secure pseudorandom generator if every algorithm A with running time at most t has advantage Adv $A < e$ .The advantage of an adversary A is defined as Adv $A = |\Pr[A(G(U_{\kappa_G}))=1] - \Pr[A(U_S)=1]|$ where U_{κ_G}, U_S are random variables distributed uniformly on κ_G, s.

2. A *pseudorandom function* F .They say that $F:\kappa_F \times \chi \to \gamma$ is a (t,q,e)-secure pseudorandom function if every oracle algorithm A making at most q oracle queries and with running time at most t has advantage Adv $A < e$ The advantage is defined as Adv $A = |\Pr[A^{F_k}=1] - \Pr[A^R=1]|$ where R represents a random function selected uniformly from the set of all maps from χ to γ, and where the probabilities are taken over the choice of k and R.
3. A *pseudorandom permutation* E, i.e., a block cipher. They say that $E:\kappa_E \times Z \to Z$ is a (t,q,e)-secure pseudorandom function if every oracle algorithm A making at most q oracle queries and with running time at most t has advantage Adv $A < e$ the advantage is defined as Adv $A = |\Pr[A^{E_k, E_k^{-1}}=1] - \Pr[A^{\Pi,\Pi^{-1}}]=1|$ where Π represents a random permutation selected uniformly from the set of all bijections on Z, and where the probabilities are taken over the choice of k and Π.

Notice that the adversary is given an oracle for encryption as well as for decryption; this corresponds to the adaptive chosen-plaintext/ciphertext attack model.

In general, the intuition is that (t,q,e)-security represents resistance to attacks that use at most t offline work and at most q adaptive chosen-text queries.

There is of course no fundamental need for three separate primitives, since in practice all three may be built out of just one off-the-shelf primitive. For instance, given any block cipher, they may build a pseudorandom generator using the counter mode or a pseudorandom function using the CBC-MAC.

They rely on the following notation. If $f:\kappa\times\chi\to\gamma$ represents a pseudorandom function or permutation, they write $f_k(x)$ for the result of applying f to input x with key $k\in\kappa$.

They write a $\langle x,y\rangle$ for the concatenation of x and y, and $x\oplus y$ for the bitwise XOR of x and y. Let $G:\kappa_G\to\chi^\ell$ be a pseudorandom generator for some ℓ, $F:\kappa_F\times\chi\to\gamma$ be a pseudorandom function, and $E:\kappa_E\times Z\to Z$ be a pseudorandom permutation. Typically they will have $\chi=\{0,1\}^{n-m}, \gamma=\{0,1\}^m$ and $Z=\chi\times\gamma=\{0,1\}^n$.

5.6 Proof of Security

This section displays proof of security to adopted algorithms as mentioned in [8].

Define $H':\kappa_F\times\kappa_G\to(\chi\times\gamma)^\ell$ by $H'(\kappa,\kappa_G)=(s_1,F_k(s_1),...,s_\ell,F_k(s_\ell))$ where $s_j\in\chi$ as shorthand for the j-th block of $G(\kappa_G)$.

Lemma A.1

If F *is a* (t,ℓ,e_F)-secure pseudorandom function and G is a (t,e_G)-secure pseudorandom generator, then H' (defined as above) is a $(t,-\varepsilon,e_{H'})$-secure pseudorandom generator, where $e_{H'}=e_F+e_G+\ell(\ell-1)/(2|\chi)$ and the constant ε is negligible compared to t.

Proof. Define $\Psi(K) = (u_1, F_k(u_1),...,u_\ell, F_k(u_\ell))$ where $u_1,...,u_\ell$ are ℓ independent random variables each drawn from the uniform distribution on χ. Also, let U be a random variable with the uniform distribution on $(\chi \times \gamma)^\ell$. Where H', Ψ, U written for the random variables obtained by choosing k, k_G uniformly at random from $\kappa_F \times \kappa_G$.

The goal is to show that H' and U are indistinguishable to any computationally-bounded adversary. The proof will proceed by showing first that H' and Ψ are indistinguishable, and second that Ψ and U are indistinguishable.

First, they show that no algorithm with running time $t - \varepsilon$ can distinguish between H' and Ψ with advantage better than e_G suppose not, i.e., there exists an algorithm A with

running time at most $t - \varepsilon$ and Adv $A = |\Pr[A(H') = 1] - |\Pr[A(\Psi) = 1] \geq e_G$.

Then they exhibit an algorithm B with running time at most t which distinguishes the output of G from a truly random bit string with advantage at least e_G .The algorithm B works in the following way: on input $s = \langle s_1,..., s_\ell \rangle \in \chi^\ell$, it runs A on input

$I = (s_1, F_k(s_1),..., s_\ell, F_k(s_\ell))$ and halts with the output $A(I)$ from A. Note that

$|\Pr[B(G(k_G)) = 1] = \Pr[A(H') = 1]$ and $\Pr[B(U') = 1] = |\Pr[A(\Psi) = 1]$, where U' is a uniformly-distributed random variable on χ^ℓ .Thus they calculate

Adv $B = |\Pr[B(G(k_G)) = 1] - \Pr[B(U') = 1] = |\Pr[A(H') = 1] - \Pr[A(\Psi) = 1] =$ Adv $A \geq e_G$, which contradicts our assumption that Adv A was large.

Second, they show that no algorithm with running time $t - \varepsilon$ can distinguish between Ψ and U with advantage better than $e_F + \ell(\ell - 1)/(2|\chi)$ .Suppose not, i.e., there exists an algorithm A with Adv $A = |\Pr[A(\Psi) = 1] - |\Pr[A(U) = 1] \geq e_F + \dfrac{\ell(\ell - 1)}{2|\chi|}$

Then they construct an oracle algorithm B which distinguishes F from a truly random function, as follows:

B chooses $u_1,...,u_\ell$ uniformly and independently at random, queries its oracle f a total of ℓ times with the inputs u_i to receive the outputs $f(u_i)$, runs A on the string $I = (u_1, f(u_1),...,u_\ell, f(u_\ell))$ and halts with $A(I)$ as its output. They want to show that Adv B is large. Of course, $\Pr[B^{F_k} = 1] = \Pr[A(\Psi) = 1]$, so it remains only to characterize $\Pr[B^R = 1]$ where R is a truly random function selected uniformly from the set of all maps $\chi \to \gamma$.

Let E denote the event that the values $u_1,...,u_\ell$ are all distinct. Also, write $\overline{\mathrm{E}}$ for the complementary event, i.e., the case where there exist i, j with $1 \le i \prec j \le \ell$ such that $u_i = u_j$ Now they may compute

$$\begin{aligned}
&\Pr[B^R = 1] = \Pr[B^R = 1|\mathrm{E}] \cdot \Pr[\mathrm{E}] \\
&+ \Pr[B^R = 1|\overline{\mathrm{E}}] \cdot \Pr[\overline{\mathrm{E}}] \\
&\le \Pr[A(U) = 1|\mathrm{E}] . \Pr[\mathrm{E}] + \Pr[\overline{\mathrm{E}}] \\
&\le \Pr[A(U) = 1|\mathrm{E}] . \Pr[\mathrm{E}] \\
&+ \Pr[A(U) = 1)|\overline{\mathrm{E}}] . \Pr[\overline{\mathrm{E}}] + \Pr[\overline{\mathrm{E}}] \\
&\le \Pr[A(U) = 1] + \frac{\ell(\ell - 1)}{2|\chi|}
\end{aligned}$$

Without loss of generality, they may assume $\Pr[B^{F_k} = 1] \ge \Pr[B^R = 1]$ Thus, Adv $B =$

$$\begin{aligned}
&|\Pr[B^{F_k} = 1] - \Pr[B^R = 1]| \\
&\ge \Pr[A(\Psi) = 1] - \Pr[A(U) = 1] \\
&- \ell(\ell - 1)/(2 | \chi)| \\
&\ge AdvA - \ell(\ell - 1)/(2 | \chi |) \ge e_F
\end{aligned}$$

which contradicts our assumption that Adv A was large.

Next, they note that the above two results suffice to show that no algorithm with running time $t-\varepsilon$ can distinguish between H' and U with advantage better than $e_F + e_G + \ell(\ell-1)/(2|\chi)$ Consider any algorithm that attempts to distinguish H' from U; then Adv $A =$

$$
\begin{aligned}
&|\Pr[A(H')=1] - \Pr[A(U)=1] \\
&= |\Pr[A(H')=1] - \Pr[A(\Psi)=1] \\
&+ |\Pr[A(\Psi)=1] - \Pr[A(U)=1] \\
&\le |\Pr[A(H')=1] - \Pr[A(\Psi)=1] \\
&+ |\Pr[A(\Psi)=1] - \Pr[A(U)=1] \\
&\prec e_G + e_F + \frac{\ell(\ell-1)}{2|\chi|}
\end{aligned}
$$

where the final line follows by applying the previous two parts of the proof.

Next define $H:(\kappa_F)^\ell \times \kappa_G \to (\chi \times \gamma)^\ell$ by $H(\kappa, \kappa_G) = (s_1, F_k(s_1), ..., s_\ell, F_k(s_\ell))$ where s_j is defined as before. This is an independently keyed version of the construction H' analyzed in Lemma A.1. In other words, the key $k = (k_1, ..., k_\ell)$ is a vector of ℓ independent random variables that are uniformly distributed on κ_F.

Lemma A.2

If F *is* $a(t, I, e_F)$-secure pseudorandom function and G is a (t, e_G)-secure pseudorandom generator, then H (defined as above, with independent keys) is a $(t-\varepsilon, e_H)$

-secure pseudorandom generator, where $e_H = \ell \cdot e_F + e_G$ and the constant ε is negligible compared to t.

Proof. The outline of the proof is as in Lemma A.1, except the second part (the treatment of the indistinguishability of Ψ and U) must be modified slightly. They define $\Psi_i (0 \le i \le n)$ by $\Psi_i(k) = \langle u_1, F_{k_1}(u_1), ..., u_i, F_{k_i}(u_j), u_{i+1}, w_{i+1}, ..., u_\ell, w_\ell \rangle$ where they

$w_{i+1}, w_{i+2}, ..., w_\ell$ are independent, uniformly distributed random variables on γ and the u_i are as above (i.e., uniform on χ and independent of everything else).

Note, for example, that $\Psi_0 = U$ and $\Psi_\ell = \Psi$. Now they show that each pair of neighbors in the sequence $\Psi_0, \Psi_1, ..., \Psi_\ell$ are $(t - \varepsilon, e_F)$-indistinguishable.

Suppose not, i.e., there exists some i, and some algorithm A distinguishing Ψ_i from Ψ_{i-1} with advantage Adv $A \geq e_F$. Then they construct an oracle algorithm B that distinguishes F from a truly random function, as follows: B chooses $u_1, ..., u_\ell \in \chi, k_1, ..., k_{i-1} \in \kappa_F$, and $w_{i+1}, ..., w_\ell \in \gamma$ independently and uniformly at random; B uses its own key material to compute $F_{k_1}(u_1), ..., F_{k_{i-1}}(u_{i-1})$ uses its oracle f to compute $f(u_i)$; then B runs A on the string $I = \langle u_1, F_{k_1}(u_1), ..., u_{i-1}, F_{k_{i-1}}(u_{i-1}), u_i, f(u_i), u_{i+1}, w_{i+1}, ..., u_\ell, w_\ell \rangle$ and finally halts with $A(I)$

as its output. By the definition of B, they have $\Pr[B^{F_k} = 1] = \Pr[A(\Psi_i) = 1]$ Also, they see easily that $\Pr[B^R = 1] = \Pr[A(\Psi_{i-1}) = 1]$, since the output of a random function R that is invoked only once is uniform and independent of everything else in sight. Therefore, they find that Adv $B =$

$$\begin{aligned} &\left|\Pr[B^{F_k} = 1] - \Pr[B^R = 1]\right| \\ &= \Pr[A(\Psi_i) = 1] - \Pr[A(\Psi_{i-1}) = 1]| \\ &= AdvA \geq e_F \end{aligned}$$

which contradicts our assumption that Adv A was large.

Now a simple application of the triangle inequality suffices to show that no algorithm can $(t - \varepsilon, \ell.e_F)$-distinguish $\Psi_\ell = \Psi$ from $\Psi_0 = U$: if A is any such algorithm, then Adv $A =$

$$
\begin{aligned}
&\left|\Pr\left[A(\Psi_\ell)=1\right]-\Pr\left[A(\Psi_0)=1\right]\right| \\
&\le \sum_{1\le i\le \ell}\left|\Pr\left[A(\Psi_i)=1\right]-\Pr\left[A(\Psi_{i-1})=1\right]\right| \\
&< \ell.e_F .
\end{aligned}
$$

This suffices to complete the proof, since the rest of the proof of Lemma A.1 now carries through.

They are, at last, ready to consider the construction H above where now the keys k_i are not necessarily chosen independently, but instead are chosen according to some distribution D on $(\chi\times\gamma)^\ell$ .They require D to have the following property:

Definition 1. They say that a distribution _ on the keys $k_1,...,k_l$ has the twining property if, for all j either (a) there $i<j$ exists such that $\Pr_D\left[k_j=k_i\right]=1$ or (b) D selects k_j uniformly at random from κ_F *independently of* $k_1,...,k_{j-1}$.

Proof of Security –Scheme 1

Theorem A.3

If F is a (t,l,e_F) -secure pseudorandom function and G is a (t,e_G) -secure pseudorandom generator, and if the key $k\in(\kappa_F)^n$ is chosen according to a distribution D with the twining property, then H I a $(t-\in,e_H)$ -secure pseudorandom generator, where $e_H=\ell.e_F+e_G+\ell(\ell-1)/(2|\chi)$ and the constant ε is negligible compared to t.

Proof. First observe that they may, without loss of generality, reorder the keys so that, for all j , either

a) $\Pr_D\left[k_j=k_{j-1}\right]=1$ or

b) k_j is selected uniformly and independently of $k_1,..,k_{j-1}$. Thus, they obtain a sequence of keys of the form $\left\langle\left\langle k_1',...,k_1'\right\rangle,\left\langle k_2',...,k_2'\right\rangle,...,\left\langle k_m',...,k_m'\right\rangle\right\rangle$ where the $k_1',k_2',...,k_m'$ are all

independent. Let ℓ_i denote the number of times that key k_i' is repeated in k, and let χ be a function which associates to each i the first j such that $\Pr_D[k_j = k_i'] = 1$.

Now they simply combine the techniques used in the proofs of Lemma A.1 and Lemma A.2. They use a hybrid argument as in Lemma A.2, this time defining ψ_i by

$\Psi_i(k) = \langle u_1, F_{k1}(u_1), ..., u_{j'-1}, F_{k_{j'-1}}(u_{j'-1}), u_{j'}, w_{j'}, ..., u_l, w_l \rangle$ Where they define $j' = \chi(i+1)$. They can see (using the arguments presented in the first part of the proof of Lemma A.1) that $\Psi_m = \Psi(t - \varepsilon, e_G)$-indistinguishable from H.

To obtain the desired result, they next show that each pair of neighbors Ψ_{i-1}, Ψ_i in the sequence $\Psi_0, \Psi_1, ..., \Psi_m$ is $t - \in, e_F + \ell_i(\ell_i - 1)/(2|\chi)$-indistinguishable.

Suppose not, i.e., there exists some i and some algorithm A distinguishing Ψ_i from Ψ_{i-1} with advantage Adv $A \geq e_F + \ell_i(\ell_{i-1})/2 \mid \chi$. Let $j = \chi(i)$ and $j' = \chi(i+1)$ Then they construct an oracle algorithm B that distinguishes F from a truly random function, as follows:

B chooses $u_1, ..., u_\ell \in \chi, k_1', ..., k_{i-1}' \in \kappa_F$ and $w_{j'}, ..., w_\ell \in \gamma$ independently and uniformly at random; B uses its own key material to compute $F_{k_1}(s_1), ..., F_{k_{j-1}}(s_{j-1})$ and uses its oracle f to compute $f(s_j), f(s_{j+1}), ..., f(s_{j'-1})$; then B runs A on the string

$I = \langle \langle u_1, F_{k_1}(u_1), ..., u_{j-1}, F_{j-1}(u_{j-1}) \rangle, \langle u_j, f(u_j), ..., u_{j'-1}, f(u_{j'-1}) \rangle, \langle u_{j'}, w_{j'}, ..., u_\ell, w_\ell \rangle \rangle$ and finally halts with $A(I)$ as its output. They have $\Pr[B^{F_k} = 1] = \Pr[A(\Psi_i) = 1]$ Also, using the argument in the second part of the proof of Lemma A.1, they obtain the bound

$\Pr[B^R = 1] \leq \Pr[A(\Psi_{i-1}) = 1] + \ell_i(\ell_i - 1)/(2|\chi|)$, from which they may conclude that Adv $B \geq$ Adv $A - \ell_i(\ell_i - 1)/(2|\chi|) \geq e_F$; and this contradicts our assumption that Adv A was large.

Finally, using the triangle inequality, and noting that $\sum_{1\le i\le m} \frac{\ell_i(\ell_i-1)}{2|\chi|} \le \frac{\ell(\ell-1)}{2|\chi|}$ they obtain the desired result.

Next, they consider the security of H when the key material is chosen using a pseudorandom function $f : \kappa_F \times \{0,1\}^* \to \kappa_F$ instead of using truly random bits. They will require $k_f \in \kappa_F$ to be chosen uniformly at random, independent of everything else.

Proof of Security –scheme II

Theorem A.4

Suppose F is a (t, ℓ, e_F) -secure pseudorandom function, f is a (t, ℓ, e_f) -secure pseudorandom function, and G is a (t, e_G)-secure pseudorandom generator.

Suppose moreover that they choose the keys k_i as $k_i = f_{k_f}(W_i)$. Then H will be a $(t-\varepsilon, e_H)$-secure pseudorandom generator, where $e_H = \ell.e_F + e_f + e_G + \ell(\ell-1)/(2|\chi)$.

Proof. They will show that the resulting distribution D on the keys has an 'approximate twining property', in the sense that D is computationally indistinguishable from a distribution with the twining property. In particular, the latter distribution is given by the random variables $k_i^R = R(W_i)$, where R is a truly random function selected uniformly from the set of all maps from $\{0,1\}^*$ to κ_F.

A straightforward simulation argument shows that the random variables $H(k, k_G)$ and $H(k^R, k_G)$ are $(t-\varepsilon, e_f)$-indistinguishable. Suppose not, so that there exists an algorithm A that distinguishes those two random variables with running time at most $t-\varepsilon$ and advantage Adv $A \ge e_f$ .Then they show that they can construct an adversary B which (t, ℓ, e_f)-breaks f .The oracle algorithm works as follows: B picks $k_G \in \kappa_G$ uniformly at

random; B computes $k_i = g(W_i)$ using its oracle g and computes $\langle s_1,...,s_\ell \rangle = G(k_G)$; then B runs A on the string $I = \langle s_1, F_{k_1}(s_1),..., s_\ell, F_{k_\ell}(s_\ell) \rangle$ and finally halts with $A(I)$ as its output. They have $\Pr\left[B^{f_{k_f}} = 1\right] = \Pr[A(H(k, k_G)) = 1]$ and $\Pr[B^R = 1] = \Pr[A(H(k^R, k_G)) = 1]$

Thus Adv B = Adv $A \geq e_f$, which contradicts our assumption that Adv A is large.

Finally, they note that k^R as the twining property, so by Theorem A.3, the random variable $H(k^R, k_G)$ and U are $(t - \varepsilon, \ell.e_F + e_G + \ell(\ell - 1)/(2|\chi))$ indistinguishable.

Applying the triangle inequality completes the proof.

A final goal is to show that the final scheme is secure even after they reveal one key k_i so that a server may perform a search on our behalf. Let $\rho\chi : \chi \times \gamma \to \chi$ be a projection onto the first component, so that $\rho\chi(\langle x, y \rangle) = x$.

Proof of Security –Final Scheme

Theorem A.5

Suppose E *is a* (t, ℓ, e_E) -secure pseudorandom permutation, F is a (t, ℓ, e_F) -secure pseudorandom function, f is a (t, ℓ, e_f)-secure pseudorandom function, G is a (t, e_G)

-secure pseudorandom generator and they choose the keys k_i as $k_i = f_{k_f}(L_i)$ where $L_i = \rho\chi(E_{k''}(W_i))$. Then will be a $(t - \varepsilon, e_H)$ -secure pseudorandom generator, where $e_H = \ell.e_F + e_f + e_G + \ell(\ell - 1)/(2|\chi)$.

Moreover, if they disclose one k_i and consider the projection of H where they discard all outputs at positions j where $W_j = W_i$ then they obtain a $(t - \varepsilon, e'_H)$-secure pseudorandom generator, where $e'_H = e_H + e_E + \ell/(2|\chi|)$.

Proof. The techniques used in the proof of Theorem A.4 apply directly (replacing the W_i 's with L_i 's), and one may readily see that H is a $(t-\varepsilon, e_H)$-secure pseudorandom generator.

Now, suppose they disclose k_i Let E denote the event that there exists some j with $W_j \neq W_i$ but $L_j = L_i$. They may observe that $\Pr[\overline{E}] \leq e_E + \ell/|\chi|$. Let D' be the distribution D modified by projecting away all keys at positions j where $W_j = W_i$. Note that D' represents the distribution of key material for the projected version of H Also, when conditioned on the event E, the distribution D' has the 'approximate twining property' used in the proof of Theorem A.4, and thus those results apply to the projected version of H.

5.7 Summery

This chapter determined the security requirement by identified security risk, security services and security mechanism. It discussed the implementation with specific standards: pseudocode, flowchart and snapshot of implementation. In the end of this chapter we validated and evaluated the versatile encryption algorithm by table show security features with versatile encryption algorithm.

CHAPTER 6

Conclusions & Future work

6.1 Conclusions

At the conclusion of the work the main contribution can be abbreviated as follows:

At first we investigated the challenges in cloud computing environment. Then the research discussed a versatile encryption scheme. And also this research identified security risk associated with data in cloud computing environment and selected one of a versatile encryption scheme as mechanism to achieve security requirement. Then we implemented the adoption algorithm using java language. The details of adoption algorithm mentioned in section 5.3 with standards such pseudocode, key points in source code, flowchart and snapshot of implementation and also we identified some of the related work in section 3.4.

Finally in this research the versatile algorithm was evaluated and discussed through table in section 5.4 shows to what extent the versatile encryption scheme is useful and effective in cloud computing environment.

6.2 Future work

A number of issues could be done in future paper:

1. Build cloud computing environment adopts this versatile encryption schema.
2. Enable the (CCs) to change setting of algorithm such as algorithms which generating key, pseudorandom function, length of pseudorandom value and schedule to change key.
3. Improve the adopted algorithm to process complex query not just equality query.

4. Make output of this algorithm more useful to (CCs) by return information such as document contains the word.
5. Try to implements this algorithm using public keys algorithm to encrypt the key which increased the security.
6. Implements of all versatile algorithms and try to assemble strengths point for each algorithm and build new algorithm combines the strengths points.

6.3 Obstacles in this Research

The problem we faced on this research how we find real environment to test the implementation and evaluate it using real data. Also the algorithm is complex in search operation and not described in clear form.

Another problem we try to implements another versatile algorithm using asymmetric key but found the equation is complex.

References

[1] G. Conway, "Introduction Cloud Computing," 2011.

[2] C. S. Alliance, "Security Guidance for Critical Areas of Focucs on Cloud Computing," 2011.

[3] J. Wayne and G. Timothy, "Guidelines on Security and Privacy in Public Cloud Computing," 2011.

[4] W. Stallings, Cryptography and Network Security principles and practice, Pearson Education, Inc., publishing as Prentice Hall, 2011.

[5] P. Institute, "Security of Cloud Computing Providers Study," 2010.

[6] C. Richard, G. Philippe, J. Markus, S. Elaine, S. Jessica, M. Ryusuke and M. Jesus, "Controlling Data in the Cloud: Outsourcing Computation without Outsourcing Control," *ACM,* November 2009.

[7] D. Boneh, D. Giovanni, O. Rafail and P. Giuseppe, "Public Key Encryption with keyword Search," 2004.

[8] X. Dawn, W. David and P. Adrian, "Practical Techniques for Searches on Encrypted Data," 2000.

[9] S. Elaine, B. John, H. C. T-H, S. Dawn and P. Adrian, "Multi-Dimensional Range Query over Encrypted Data," 2007.

[10] S. Emily, S. Elaine and W. Brent, "Predicate Privacy in Encryption Systems," 2008.

[11] B. Dan and W. Brent, "Conjunctive, Subset, and Range Queries on Encrypted Data," 2007.

[12] C. Benny, G. Oded, K. Eyal and S. Madhu, "Private Information Retrieval," *Journal of the ACM*, vol. 45, no. 6, p. 965–982, November 1998.

[13] A. Giuseppe, B. Randal and C. Reza, "Provable Data Possession at Untrusted Stores," in *ACM Conference on Computer and*, 2007.

[14] S. Hovav and W. Brent, "Compact Proofs of Retrievability," 2008.

[15] B. Dan, L. Ben and S. Hovav, "Short Signatures from the Weil Pairing," in *Asiacrypt*, 2001.

[16] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn and H. Hu, "Zero-knowledge proofs of retrievability," *SCIENCE CHINA*, December 2010.

[17] S. Y. a. K. R. a. W. L. Ming Li, "Securing Personal Health Records in Cloud Computing: Patient-centric and Fine-grained Data Access Control in Multi-owner Settings," *SecureComm*, vol. 10, pp. 98-106, Septemper 2010.

[18] R. Rajan, "efficient and privacy preserving multi user keyword search for cloud storage services," *International Journal of Advanced Technology & Engineering Research (IJATER)* , vol. 2, no. 4, pp. 48-51, July 2012.

[19] S. M. B. Gowri. N. Dixit, "Patient Centric Frame Work For Data Access Control Using Key Management In Cloud," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 4, pp. 1869-1872, April 2013.

[20] P. K. E. Shaheen Taj S.A, "a novel method for patient centric secure and scalable sharing of PHR in cloud computing using encryption," *International Journal of Soft Computing and Engineering (IJSCE),* vol. 3, no. 4, pp. 226-232, may 2013.

[21] M. C. E. H. a. K. L. Josh Benaloh, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records," Chicago, Illinois, USA., 2009.

[22] L. Ming, Y. Shucheng, C. Ning and L. Wenjing, "Authorized Private Keyword Search over Encrypted Data in Cloud Computing," 2011.

[23] A. H. A. B. a. Y. C. Ahmed Lounis, "Secure and Scalable Cloud-based Architecture for e-Health Wireless sensor networks," Compi`egne Cedex, 2011.

[24] M. Ucal, "Searching on Encrypted Data," 2005.

[25] K. L. Seny Kamara, "Cryptographic Cloud Storage," in *Workshop Real-Life Cryptographic Protocols and Standardization (RLCPS)*, 2010.

[26] C. P. T. R. Seny Kamara, "Dynamic Searchable Symmetric Encryption," 2012.

[27] M. S. I. M. K. Mehmet Kuzu, "Efficient Similarity Search over Encrypted Data," USA, 2011.

[28] I. Cisco Systems, "Cisco Cloud Computing - Data Center Strategy, Architecture,and Solution," 2009.

[29] G. Corp, "Cloud Computing," 2009.

[30] V. Kundra, " Federal Cloud Computing Strategy," 2011.

[31] I. Corporation, "IBM Point of View:," United States of America, 2009.

[32] G. H. Daniele Catteddu, "Cloud Computing Benefits, risks and recommendations for information security," Europe, 2009.

Appendices

```
import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;
//import java.util.Random;
import java.security.*;
import javax.crypto.spec.IvParameterSpec;
//import java.io.UnsupportedEncodingException;
import java.util.*;
import java.io.*;
/**
 *
 * @author Amal
 */

public class Symmetric_Algorithm {
  String Wi; // String Ti;
  byte WiBytes[][];
  byte TiBytes[][];
  byte Ti2Bytes[][];
  byte KiBytes[][];
  byte CiBytes[][];
  String CiString[];
  byte SiBytes[][];
  byte encryptedSi[][];
  byte Ki2Bytes[][];
  byte Ki3Bytes[][];
  byte encryptedXi[][];
  byte XiBytes[][];
  byte LiBytes[][];
  byte RiBytes[][];
  StringBuffer AllCipher=new StringBuffer(9999);
```

```
Scanner sc=new Scanner(System.in);
int flageSchema=0,wordsindex=0;
Amaltransaction f;
protected  Symmetric_Algorithm(){
          flageSchema=1;
  f=new Amaltransaction("Symmetric Algorithm");
  f.setVisible(true);
 }
protected  void EnterWords(){
  f.setData("\n********   Enter Words Method    *******");
  File file1=null,fil=null;
  try{
    fil=new File(MyFrame.fileChooser.getSelectedFile().toString());
    file1=new File("inbox.txt");
    Scanner inFile = new Scanner(fil);
    String line;int i=0;
    while (inFile.hasNextLine())
      {
                        line = inFile.nextLine();
                        Scanner words = new Scanner(line);
                        while (words.hasNext())
                         {
                                String word = words.next();
                                f.setData("Word  "+i+" in file : "+word);
                                //f.setData(word);
                                wordsindex++;
                                i++;
                         }
      }
      //fil.close();file1.close();

                WiBytes=new byte[wordsindex][];
                TiBytes=new byte[wordsindex][];
```

```
                Ti2Bytes=new byte[wordsindex][];
                KiBytes=new byte[wordsindex][];
                CiBytes=new byte[wordsindex][];
                SiBytes=new byte[wordsindex][];
                encryptedSi=new byte[wordsindex][];
                CiString=new String[wordsindex];
 }catch(FileNotFoundException e){
     f.setData("File " + file1.getName() + " not found.");
 }
  /*catch(IOException e){
   f.setData("Error reading from file " + file.getName());
   }*/
File file=null;
try{
                file=new File(MyFrame.fileChooser.getSelectedFile().toString());
                //file=new File("inbox.txt");
                Scanner inFile = new Scanner(file);
                String line;int i=0;
                while (inFile.hasNextLine())
                  {
                        line = inFile.nextLine();
                        Scanner words = new Scanner(line);
                        while (words.hasNext())
                          {
                                String word = words.next();
                                WiBytes[i]=word.getBytes();
                                f.setData("Length of Word  "+i+" as Bytes "+WiBytes[i].length);
        f.setData("Word  "+i+" as Bytes : ");
                                printValue(WiBytes[i]);
                                f.setData(" ");
                                i++;
                        }
                  }
```

```
                //file.close();
        }catch(FileNotFoundException e){
                //f2.setData("File " + file.getName() + " not found.");
        }
                /*catch(IOException e){
                        f2.setData("Error reading from file " + file.getName());
    }*/
    f.setData("\n********   word    *******"+WiBytes[0]);
}//End of EnterWords

//_____________________________________________________________________________
____

 protected  void GenerateSi(){
   //int randomInt = randomGenerator.nextInt(100);
        //log("Generated : " + randomInt);
        f.setData("\n********   Generate Si Method    *******");
        try{
          for(int g=0;g<wordsindex;g++){

                        SecureRandom secureRandom = SecureRandom.getInstance("SHA1PRNG");
                        f.setData("seed Number : "+g);
                        printValue(SecureRandom.getSeed(1));
                        f.setData(" ");
                        // Method 1 - Calling nextBytes method to generate Random Bytes
                        byte[] si = new byte[16];
                        secureRandom.nextBytes(si);


/** Converts Si to array of bytes  */
                SiBytes[g] = si;
     f.setData("Length of Pseudorandom S  "+g+" as Bytes "+SiBytes[g].length);
     f.setData("Pseudorandom S "+g+" as Bytes : ");
```

```
        printValue(SiBytes[g]);
        f.setData(" ");
        //Integer Si=new Integer(randomInt);
        }//for
     }catch(NoSuchAlgorithmException e){
          f.setData(e.toString());
    }
 }//End of GenerateSi


//_____________________________________________________________________________
_____

protected  void GenerateKi3Scheme3(){
    f.setData("\n********   Call Generate Ki3 in Scheme III Method    *******");
    try{
      Ki3Bytes=new byte[wordsindex][];
      for(int g=0;g<wordsindex;g++){
                              KeyGenerator keygen = KeyGenerator.getInstance("AES");
                              keygen.init(128);  // To use 256 bit keys, you need the "unlimited strength"
encryption policy files from Sun.
                              byte[] key = keygen.generateKey().getEncoded();
                              Ki3Bytes[g]=key;
                              f.setData("Length of K"3 "+g+" in Scheme II as Bytes "+Ki3Bytes[g].length);
                              f.setData("K"3 "+g+" as Bytes : ");
                              printValue(Ki3Bytes[g]);
                              f.setData(" ");

       }
     }catch (NoSuchAlgorithmException e) {
          f.setData("Exception in Catch Block "+e.toString());
     }
}//End of GenerateKi3Scheme3
```

```
//_____________________________________________________________________________
_____

protected  void Encrypt_Wi_Using_Ki3_Scheme3(){
  f.setData("\n********   Call Encrypt Wi Using Ki2 Method  in Schema III  *******");
  // build the initialization vector.  This example is all zeros, but it
  // could be any value or generated using a random number generator.
  try{
     XiBytes=new byte[wordsindex][];
     for(int g=0;g<wordsindex;g++){
                              SecretKeySpec skeySpec = new SecretKeySpec(Ki3Bytes[g], "AES");
                              byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
                              IvParameterSpec ivspec = new IvParameterSpec(iv);

          /** Steps of Encrypt Si */

                               //String SiString=Integer.toBinaryString(Si);
                              Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                              cipher.init(Cipher.ENCRYPT_MODE, skeySpec, ivspec);
                              XiBytes[g] = cipher.doFinal(WiBytes[g]);
                              f.setData("Length of X "+ g+" in Scheme III as Bytes "+XiBytes[g].length);
                              f.setData("Encrpted W Which are called X"+g+" as Bytes : ");
                              printValue(XiBytes[g]);
                              f.setData(" ");
                              //f.setData("Ciphertext: " + hexEncode(encrypted) + "\n");
     }
   }catch (NoSuchAlgorithmException e) {
     f.setData("Exception in Catch Block "+e.toString());
   }catch(NoSuchPaddingException e){
     f.setData("Exception in Catch Block "+e.toString());
   }catch(IllegalBlockSizeException e){
     f.setData("Exception in Catch Block "+e.toString());
```

```
    }catch (InvalidKeyException e) {
       f.setData("Exception in Catch Block "+e.toString());
    } //catch (UnsupportedEncodingException e) {//}
    catch(InvalidAlgorithmParameterException e){
       f.setData("Exception in Catch Block "+e.toString());
    }catch(BadPaddingException e){
       f.setData("Exception in Catch Block "+e.toString());
   }
  }//End of Encrypt_Wi_Using_Ki3_Scheme3
//_____
_____

protected  void Split_Xi(){
   LiBytes=new byte[wordsindex][];
   RiBytes=new byte[wordsindex][];
   byte Ti2Half1[];
   byte Ti2Half2[];
   //StringBuffer Ti2String=new StringBuffer();
   for(int i=0;i<wordsindex;i++){
            Ti2Half1=new byte[XiBytes[i].length/2];
            Ti2Half2=new byte[XiBytes[i].length/2];
            //String s=new String(Ti2[i]);
            //f.setData("Ti2 in s "+s.length());
           for(int b=0;b<Ti2Half1.length;b++){
                    //f.setData("b in search "+b);
                    Ti2Half1[b]=XiBytes[i][b];
           }//for
           int y=0;
           for(int b2=Ti2Half1.length;b2<XiBytes[i].length;b2++){
                    Ti2Half2[y]=XiBytes[i][b2];
                    y++;
           }//for
           LiBytes[i]=Ti2Half1;
```

```
            RiBytes[i]=Ti2Half2;
            f.setData("Half1 of T equals Pseudorandom S "+LiBytes.length);
            f.setData("half2 of T equals Encrypted Pseudorandom S "+RiBytes.length);
  }
 }//End of Split_Xi

//_____
_____
protected void GenerateKi2Scheme2(){
   f.setData("\n********   Call Generate Ki2 in Scheme II Method    *******");
   try{
             Ki2Bytes=new byte[wordsindex][];
      for(int g=0;g<wordsindex;g++){
                              KeyGenerator keygen = KeyGenerator.getInstance("AES");
                              keygen.init(128);  // To use 256 bit keys, you need the "unlimited strength"
encryption policy files from Sun.
                              byte[] key = keygen.generateKey().getEncoded();
                              Ki2Bytes[g]=key;
                              f.setData("Length of K'2 "+g+" in Scheme II as Bytes "+Ki2Bytes[g].length);
                              f.setData("K "+g+" as Bytes : ");
                              printValue(Ki2Bytes[g]);
                              f.setData(" ");

      }
    }catch (NoSuchAlgorithmException e) {
      f.setData("Exception in Catch Block "+e.toString());
    }
}//End of GenerateKi2Scheme2

//_____
_____

protected void Encrypt_Li_Using_Ki2_Scheme2(){
```

```
f.setData("\n********   Call Encrypt Li Using Ki2 Method  in Schema III *******");
// build the initialization vector.  This example is all zeros, but it
// could be any value or generated using a random number generator.
Cipher cipher=null;
try{
  for(int g=0;g<wordsindex;g++){
                        SecretKeySpec skeySpec = new SecretKeySpec(Ki2Bytes[g], "AES");
                        byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
                        IvParameterSpec ivspec = new IvParameterSpec(iv);

    /** Steps of Encrypt Si */
                         cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                         cipher.init(Cipher.ENCRYPT_MODE, skeySpec, ivspec);
                         KiBytes[g] = cipher.doFinal(LiBytes[g]);

                        f.setData("Length of K1 "+ g+" in Scheme II as Bytes "+KiBytes[g].length);
                        f.setData("Encrpted L "+g+" as Bytes : ");
                        printValue(KiBytes[g]);
                        f.setData(" ");
                        //f.setData("Ciphertext: " + hexEncode(encrypted) + "\n");
  }
}catch (NoSuchAlgorithmException e) {
  f.setData("Exception in Catch Block "+e.toString());
}catch(NoSuchPaddingException e){
  f.setData("Exception in Catch Block "+e.toString());
}catch(IllegalBlockSizeException e){
  f.setData("Exception in Catch Block "+e.toString());
}
      catch (InvalidKeyException e) {
      f.setData("Exception in Catch Block "+e.toString());
      } //catch (UnsupportedEncodingException e) {//}
      catch(InvalidAlgorithmParameterException e){
  f.setData("Exception in Catch Block "+e.toString());
```

```
    }
    catch(BadPaddingException e){
      f.setData("Exception in Catch Block "+e.toString());
    }
}//End of Encrypt_Li_Using_Ki2_Scheme2

//_____
_____

protected void Encrypt_Si_Using_Ki(){
  f.setData("\n********   Call Encrypt Si Using Ki Method  in Scheme III  *******");
  // build the initialization vector.  This example is all zeros, but it
  // could be any value or generated using a random number generator.
  try{
            encryptedSi=new byte[wordsindex][];
     for(int g=0;g<wordsindex;g++){
                    SecretKeySpec skeySpec = new SecretKeySpec(KiBytes[g], "AES");
                        byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
                        IvParameterSpec ivspec = new IvParameterSpec(iv);
                        Cipher cipher=null;
       /** Steps of Encrypt Xi */

         //String SiString=Integer.toBinaryString(Si);

         //PKCS5Padding
         cipher = Cipher.getInstance("AES/CBC/NoPadding");
         //f.setData("Length of Ki   "+ g+" as Bytes "+KiBytes[g].length);
         cipher.init(Cipher.ENCRYPT_MODE, skeySpec, ivspec);
         encryptedSi[g] = cipher.doFinal(SiBytes[g]);
         f.setData("Length of Encrypted S  "+ g+" as Bytes "+encryptedSi[g].length);
         f.setData("Encrypted S "+g+" as Bytes : ");
         printValue(encryptedSi[g]);
         f.setData(" ");
```

```
            //f.setData("Ciphertext: " + hexEncode(encrypted) + "\n");
          }
       }catch (NoSuchAlgorithmException e) {
          f.setData("Exception in Catch Block "+e.toString());
       }catch(NoSuchPaddingException e){
          f.setData("Exception in Catch Block "+e.toString());
       }catch(IllegalBlockSizeException e){
          f.setData("Exception in Catch Block "+e.toString());
       }catch (InvalidKeyException e) {
          f.setData("Exception in Catch Block "+e.toString());
       } //catch (UnsupportedEncodingException e) {//}
       catch(InvalidAlgorithmParameterException e){
          f.setData("Exception in Catch Block "+e.toString());
       }catch(BadPaddingException e){
          f.setData("Exception in Catch Block "+e.toString());
       }
    }//End of Encrypt_Si_Using_Ki
    //_____
    _____

    protected  void PrepareTi_Scheme3(){
              f.setData("\n********   Call Prepare Ti Method in Scheme III    *******");
              for(int g=0;g<wordsindex;g++){
              // Ti=new String(newSi).concat(new String(result));
               byte Ti[]=new byte[SiBytes[g].length+encryptedSi[g].length];
               int j1=0,j2=0;

               for(int k=0;k<SiBytes[g].length;k++){
                                 Ti[k]=SiBytes[g][j1];
                                 j1++;
              }
              for(int k=SiBytes[g].length;k<Ti.length;k++){
                                 Ti[k]=encryptedSi[g][j2];
```

```
                        j2++;
            }
            Ti2Bytes[g]=Ti;
            f.setData("Length of T 2 "+g+" in Scheme III as Bytes "+Ti2Bytes[g].length);
            f.setData("T 2  "+g+" in Scheme III as Bytes : ");
            printValue(Ti2Bytes[g]);
            f.setData(" ");
           }//for
 }//End of PrepareTi_Scheme3
//_____
_____

protected  void GenerateCi(){
           try{
           //FileOutputStream fi=new FileOutputStream("E://Cipher.txt");
           //PrintWriter fid=new PrintWriter(fi);
           PrintWriter fid=new PrintWriter("E://Cipher.txt");
           f.setData("\n********   Call Generate Ci Method    *******");
           for(int o=0;o<wordsindex;o++){
                    CiBytes[o]=XOR(Ti2Bytes[o],XiBytes[o]);
                    /*outFile.write(CiBytes[o],0,CiBytes[o].length);*/
                    f.setData("Length of Cipher Ci "+o+" as Bytes "+CiBytes[o].length);
                    f.setData("Cipher C  "+o+" as Bytes : ");
                    printValue(CiBytes[o]);
                    f.setData(" ");
                    //fid.write(CiBytes[o]);
                    CiString[o]=new String(CiBytes[o]);
                    fid.println(CiString[o]);
                    //AllCipher.append(CiString[o]+"\n");
           }//for
           fid.close();
           }catch (IOException ex) {
              System.err.println(ex);
```

```
}

 //=========================================================================
   /* byte CiBytesserver[][];
    Scanner sc=null;
          try {
                        sc = new Scanner("E://Cipher.txt");
                        CiBytesserver = new byte[lineNumber][];
                        while (sc.hasNext())
                        {
                                 CiBytesserver[lineNumber]=line.nextToken().getBytes();
                        }
                        sc.close();
                        System.out.println("in symmetric first   "+token);
                        stream2 = new FileInputStream("E://Cipher.txt");
                        BufferedReader buf2 = new BufferedReader(new InputStreamReader(new
DataInputStream(stream2)));
                        lineNumber = 0;
                        while ((line = buf2.readLine()) != null)   {
                                 // StringTokenizer lin=new StringTokenizer(line);
                                 // while (lin.hasMoreTokens())
                                     //CiBytesserver[lineNumber]=line.nextToken().getBytes();
              CiBytesserver[lineNumber]=line.getBytes();

           lineNumber++;
                        }
                         System.out.println("in symmetric  line number is  "+lineNumber);
                         System.out.println("in symmetric  word index  "+wordsindex);

                        stream.close();
                        //stream.read(CipherArray);
                       // CiBytes=CipherArray;
                  } catch (Exception e) {
```

```
                    e.printStackTrace();
    }*/

 //=========================================================================
}//End of GenerateCi
//_____________________________________________________________________________
_____

protected  byte[] Encrypt_W_Using_Ki3_Scheme3(byte searchWord[],int pos){
        f.setData("\n********   Call Encrypt Wi Using Ki2 Method  in Schema II  *******");
        // build the initialization vector.  This example is all zeros, but it
        // could be any value or generated using a random number generator.
                byte encryptedSearchWord []=null;
         try{
           //for(int g=0;g<3;g++){
                  SecretKeySpec skeySpec = new SecretKeySpec(Ki3Bytes[pos], "AES");
                        byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
                        IvParameterSpec ivspec = new IvParameterSpec(iv);

         /** Steps of Encrypt Si */

                         //String SiString=Integer.toBinaryString(Si);
                        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                        cipher.init(Cipher.ENCRYPT_MODE, skeySpec, ivspec);
                        encryptedSearchWord = cipher.doFinal(searchWord);
                        f.setData("Length of Search Word in Scheme III as Bytes "+searchWord.length);
                        f.setData("Search Word in Scheme III  as Bytes : ");
                        printValue(searchWord);
                        f.setData(" ");
                        f.setData("Length of Encrypted Search Word in Scheme III as Bytes
"+encryptedSearchWord.length);
                        f.setData("Encrypted Search Word in Scheme III Which are called X as Bytes :
");
```

```
                    printValue(encryptedSearchWord);
                    f.setData(" ");
                    //f.setData("Ciphertext: " + hexEncode(encrypted) + "\n");
        //   }
                }catch (NoSuchAlgorithmException e) {
                f.setData("Exception in Catch Block "+e.toString());
                }  catch(NoSuchPaddingException e){
                f.setData("Exception in Catch Block "+e.toString());
                }
                catch(IllegalBlockSizeException e){
                        f.setData("Exception in Catch Block "+e.toString());
                }
                catch (InvalidKeyException e) {
                f.setData("Exception in Catch Block "+e.toString());
                } //catch (UnsupportedEncodingException e) {//}
                 catch(InvalidAlgorithmParameterException e){
                        f.setData("Exception in Catch Block "+e.toString());
                }
                catch(BadPaddingException e){
                        f.setData("Exception in Catch Block "+e.toString());
                }
                return encryptedSearchWord;
        }//End of Encrypt_W_Using_Ki3_Scheme3
//_____
_____

protected byte[] Encrypt_Li_Using_Ki2(byte Li[],int pos){
        f.setData("\n********   Call Encrypt Xi Using Ki Method  in Scheme III  *******");
        // build the initialization vector.  This example is all zeros, but it
        // could be any value or generated using a random number generator.
          byte encryptedLi[]=null;
         try{
          //for(int g=0;g<3;g++){
```

```
            SecretKeySpec skeySpec = new SecretKeySpec(Ki2Bytes[pos], "AES");
                    byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
                    IvParameterSpec ivspec = new IvParameterSpec(iv);
                    Cipher cipher=null;
/** Steps of Encrypt Li */

                     //String SiString=Integer.toBinaryString(Si);


          //PKCS5Padding
                              cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                    //f.setData("Length of Ki   "+ g+" as Bytes "+KiBytes[g].length);
                    cipher.init(Cipher.ENCRYPT_MODE, skeySpec, ivspec);
                    encryptedLi = cipher.doFinal(LiBytes[pos]);
                    f.setData("Length of Encrypted Li  as Bytes "+encryptedLi.length);
                    f.setData("Encrypted Li as Bytes : ");
                    printValue(encryptedLi);
                    f.setData(" ");
                    //f.setData("Ciphertext: " + hexEncode(encrypted) + "\n");
 // }
          }catch (NoSuchAlgorithmException e) {
          f.setData("Exception in Catch Block "+e.toString());
          }  catch(NoSuchPaddingException e){
          f.setData("Exception in Catch Block "+e.toString());
          }
          catch(IllegalBlockSizeException e){
                    f.setData("Exception in Catch Block "+e.toString());
          }
          catch (InvalidKeyException e) {
          f.setData("Exception in Catch Block "+e.toString());
          } //catch (UnsupportedEncodingException e) {//}
           catch(InvalidAlgorithmParameterException e){
                    f.setData("Exception in Catch Block "+e.toString());
          }
```

```
                catch(BadPaddingException e){
                        f.setData("Exception in Catch Block "+e.toString());
                }
            return encryptedLi;
}//End of Encrypt_Li_Using_Ki2
//_____
_____

public  void printValue(byte b[]){
            //f.setData("Print Value Method ");
            for(int i=0;i<b.length;i++)
        f.setDataInLine(" "+b[i]);
        //f.setData(" ");
        //for*/
        /* String s="";
                    for(int i=0;i<b.length;i++){
                            byte byteVal=b[i];
           int intVal=new Byte(byteVal).intValue();
           s=s.concat(Integer.toBinaryString(new Integer(intVal)));
           f.setData(" "+s);
        }//for*/
            //f.setData("\n length as binary "+s.length());
}//End of printValue
//_____
_____

protected  byte[] XOR(byte [] Ti,byte [] Wi) {
           f.setData("\n********   Call XOR Method    *******");
           //f.setData("Word "+Wi.length);
           //f.setData("Ti  "+Ti.length);
           int u=0,maxlength=0;
           byte [] XOR_out,Tixor,Wixor;
           int lenWi = Wi.length;
```

```
int lenTi = Ti.length;

int j1=0,j2=0;
// For each character in our string, encrypt it...
if(lenWi>lenTi){
        maxlength=lenWi;
        Tixor =new byte [lenWi];
        for(int k=0;k<lenTi;k++){
                Tixor[k]=Ti[j1];
                j1++;
        }
        for(int k=lenTi;k<lenWi;k++){
                Tixor[k]=0;
        }
        Wixor=Wi;
}//if
else if(lenTi>lenWi){
        Wixor =new byte [lenTi];
        maxlength=lenTi;
        for(int k=0;k<lenWi;k++){
                Wixor[k]=Wi[j1];
                j1++;
        }
        for(int k=lenWi;k<lenTi;k++){
                Wixor[k]=0;
        }
       Tixor=Ti;
}//if
 else{
        maxlength=lenWi;
        Wixor=Wi;
        Tixor=Ti;
 }
```

```
          XOR_out =new byte [maxlength];
        for ( int i = 0, j = 0; i < maxlength; i++, j++ )
        {
          //f.setData("Word byte "+Wixor[i]);
          //f.setData("Ti byte "+Tixor[i]);

          XOR_out[u]=(byte)(Wixor[i] ^ Tixor[j]);
          u++;
        }

        return XOR_out;
}//End of XOR
//_____
_____

protected boolean Search_Scheme3(byte key[],byte [] searchword) {
        f.setData("\n********   Call Search Method    *******");
        boolean Search=false;
        //String S="";
        byte encryptedS[];
        byte decryptedS[];
        byte Ti2[][]=new byte[wordsindex][];
        byte Ti2Half1[];
        byte Ti2Half2[];
        int foundword=0;
        //StringBuffer Ti2String=new StringBuffer();
         for(int i=0;i<wordsindex;i++){
                Ti2Half1=new byte[SiBytes[i].length];
                Ti2Half2=new byte[Ti2Bytes[i].length-SiBytes[i].length];
                Ti2[i]=XOR(CiBytes[i],searchword);
                f.setData("Length of Calculated T "+i+" in Search Method in Scheme III
"+Ti2[i].length);
                f.setData("Calculated T "+i+" in Search Method in Scheme III as Bytes");
```

```
 printValue(Ti2[i]);
 f.setData(" ");
 f.setData("Length of Pseudorandom S in serach "+SiBytes[i].length);

 //String s=new String(Ti2[i]);
 //f.setData("Ti2 in s "+s.length());
 for(int b=0;b<SiBytes[i].length;b++){
        //f.setData("b in search "+b);
        Ti2Half1[b]=Ti2[i][b];
}//for
 int y=0;
 for(int b2=SiBytes[i].length;b2<Ti2Bytes[i].length;b2++){
        Ti2Half2[y]=Ti2[i][b2];
        y++;
}//for
 f.setData("Half1 of T equals Pseudorandom S "+Ti2Half1.length);
 f.setData("half2 of T equals Encrypted W = X "+Ti2Half2.length);
 //f.setData("Encrypted S "+i+" in Search"+Ti2Half2.length);
 //f.setData("Si length in serach "+SiBytes[i].length);
 encryptedS=Ti2Half2;
 decryptedS=Decrypt_Si_Using_Ki(key,encryptedS);
 f.setData("Decrypted S "+decryptedS.length);
 if(new String(Ti2Half1).equals(new String(decryptedS))){
          foundword=i;
          Search=true;
          break;
 }
 decryptedS=new byte[8];
}//for
  if(Search==true)
                f.setData("Search Word Found in Position "+foundword);
  else
                f.setData("Search Word is Not Found ");
```

```
        return Search;
}//End of Search_Scheme3
//_____
_____

protected  byte[] Decrypt_Si_Using_Ki(byte K[],byte encryptedS[]){
        f.setData("\n********   Decrypt Si Using Ki Method    *******");
       // build the initialization vector.  This example is all zeros, but it
       // could be any value or generated using a random number generator.
        byte decryptedS[]=new byte[8];
        try{

                SecretKeySpec skeySpec = new SecretKeySpec(K, "AES");
                byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
                IvParameterSpec ivspec = new IvParameterSpec(iv);
                Cipher cipher=null;
       /** Steps of Decrypt Si */

                        //String SiString=Integer.toBinaryString(Si);
               /*if(flageSchema==1){
                                cipher =Cipher.getInstance("AES/CBC/NoPadding");
                        }
               else if(flageSchema==2)
                                cipher = Cipher.getInstance("AES/CBC/NoPadding");
               else if(flageSchema==3)*/
                                cipher = Cipher.getInstance("AES/CBC/NoPadding");
               cipher.init(Cipher.DECRYPT_MODE, skeySpec, ivspec);

               decryptedS= cipher.doFinal(encryptedS);
               //f.setData("Block Size "+cipher.getBlockSize());
               //f.setData("Output Size "+cipher.getOutputSize(8));
               //f.setData(" Dycryyyyyyyyyypt "+K.length);
```

```
                //f.setData(" Dycryyyyyyyyyypt "+S.length);
                f.setData("Length of Decrypted S  as Bytes "+decryptedS.length);
                f.setData("Decrypted S as Bytes : ");
                printValue(decryptedS);
                f.setData(" ");
                        //f.setData("Ciphertext: " + hexEncode(encrypted) + "\n");

                }catch (NoSuchAlgorithmException e) {
                f.setData("Exception in Catch Block "+e.toString());
                }  catch(NoSuchPaddingException e){
                f.setData("Exception in Catch Block "+e.toString());
                }
                catch(IllegalBlockSizeException e){
                        f.setData("Exception in Catch Block "+e.toString());
                }
                catch (InvalidKeyException e) {
                f.setData("Exception in Catch Block "+e.toString());
                } //catch (UnsupportedEncodingException e) {//}
                 catch(InvalidAlgorithmParameterException e){
                        f.setData("Exception in Catch Block "+e.toString());
                }
                catch(BadPaddingException e){
                        f.setData("Exception in Catch Block "+e.toString());
                }
          return decryptedS;
 }//End of Decrypt_Si_Using_Ki
//_____
_____

protected int get_word_position(byte word[]){
        boolean found=false;
        int i=0;
        for(i=0;i<wordsindex;i++){
```

```
                if((new String(word)).equals(new String(WiBytes[i]))){
                        found=true;
                        System.out.println("gggg  "+new String(word));
                        System.out.println("gggg4  "+new String(WiBytes[i]));
                        break;
                }
        }
        System.out.println("found  "+found);
        if(found==true)
                return i;
        else
                return -1;
}//End of get_word_position
//_____
_____
}
```