FILE FOX

By

Sardar Waqas Umar

Nasira Hasnain

Samar Abbas

Asad Tariq Qureshi

Supervisor Name:

Dr. Rabia Latif

Submitted to the Faculty of Department of Electrical Engineering,
Military College of Signals, National University of Sciences and Technology, Islamabad
in partial fulfillment for the requirements of a B.E Degree in
Telecom Engineering

JUNE 2019

CERTIFICATE FOR CORRECTNESS AND APPROVAL

It is certified that work contained in the thesis

“ File Fox ”

is carried out by

Sardar Waqas Umar, Nasira Hasnain, Samar Abbas and Asad Tariq Qureshi

under my Supervision and in my Judgement. It is generous in scope and excellence, for the degree of Bachelors of Electrical (Telecomm) Engineering from Military College of Signals ,National University of Sciences and Technology (NUST). Plagiarism in document is 16%.

Approved by:

____

Asst Prof Dr. Rabia Latif
Electrical Engineering Dept.
MCS, NUST

ABSTRACT

FILE FOX

Cloud Computing is next generation enterprise architecture. It is demand of modern era. Due to any technical error, we may lose our important data on PC. Cloud is providing back up to our data. We can access cloud at anytime and anywhere. Public and Private cloud are two main types of cloud. There are numerous security and data issues on cloud. Therefore, a need exists to secure and manage our important data and files. FILE FOX is aimed at providing a solution to file security issues and data management on cloud. Security is great necessity for the World due to its increasing demand day by day. Through File Fox, we have been able to Manage data/files on cloud. GUI will provide user friendly environment which makes its use easier. There are numerous issues on Public Cloud that's why we are deploying private Cloud. Data on cloud is in plain text. In FILE FOX, data on cloud will be encrypted. Key is generated by Diffie-Hellman and shared through socket Programming. Both Software and Hardware implementations are the part of this project.

DECLARATION

No portion of the work presented in this dissertation has been submitted has been submitted earlier or qualification either at Military College of Signals, NUST or elsewhere.

DEDICATION

In the name of Allah, the Most Beneficent, the Most Merciful.

To our respected teachers whose guidance and support made this Project possible and to our parents whose prayers gave us power to complete our Final year project.

ACKNOWLEDGEMENTS

Despite the fact that only our name appears on the cover of this thesis, many people have contributed in our FYP. We are grateful to all those people who made all this possible.

Above of all, **ALMIGHTY ALLAH** who has always been there throughout this venture.

To our respected Supervisor **Dr. Rabia Latif** whose guidance and support made this project possible. She has been the main driving force that backing us morally. We are highly thankful to all of our teachers and staff of MCS who supported and guided us throughout our course work. Their knowledge, guidance and training enabled us to carry out this whole work.

To our parents, without whose firm support and unsparing cooperation, mammoth work would not have been possible.

Finally, we are grateful to the faculty of Electrical (Telecom) Department of the Military College of Signals, NUST.

TABLE OF CONTENTS

List of Figures

Key Symbols or Abbreviations

GUI	Graphical Use Interface
NIST	National Institute of Standards and technology
USB	Universal Serial Bus
CSP	Cloud service Provider
AES	Advanced Encryption Standard
GPU	Graphics Processing unit
CPU	Central Processing Unit
RAM	Random Access Memory
VPC	Visual Private Cloud
I/O	Input/Output
Fab	Frequency Allocation Bureau
FIA	Federal Investigation Agency
PIA	Pakistan International Airline
NADRA	National Data base and registration authority
DES	Data Encryption Standard
IP	Internet Protocol

CHAPTER 1
INTRODUCTION

1.1 Background

Cloud computing is emerged from large-scale distributed computing technology. NIST states cloud computing as “a model for enabling convenient, on demand network access to a shared pool of configurable computing resources (e.g. networks, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”. Due to any technical errors, we may lose precious data on PC. That’s why, cloud is necessary to provide secure backup to our data. In it, both files, data and software are not on user’s computer. Security issues become apparent because user’s application, program is in provider site.

Sending and receiving of file on cloud is shown in figure 1.1. Owner of file uploads his file on cloud. Encryption is done on cloud . When Client wants that file. He will request owner for key exchange. After entering key, file is decrypted on cloud. He will download this file.

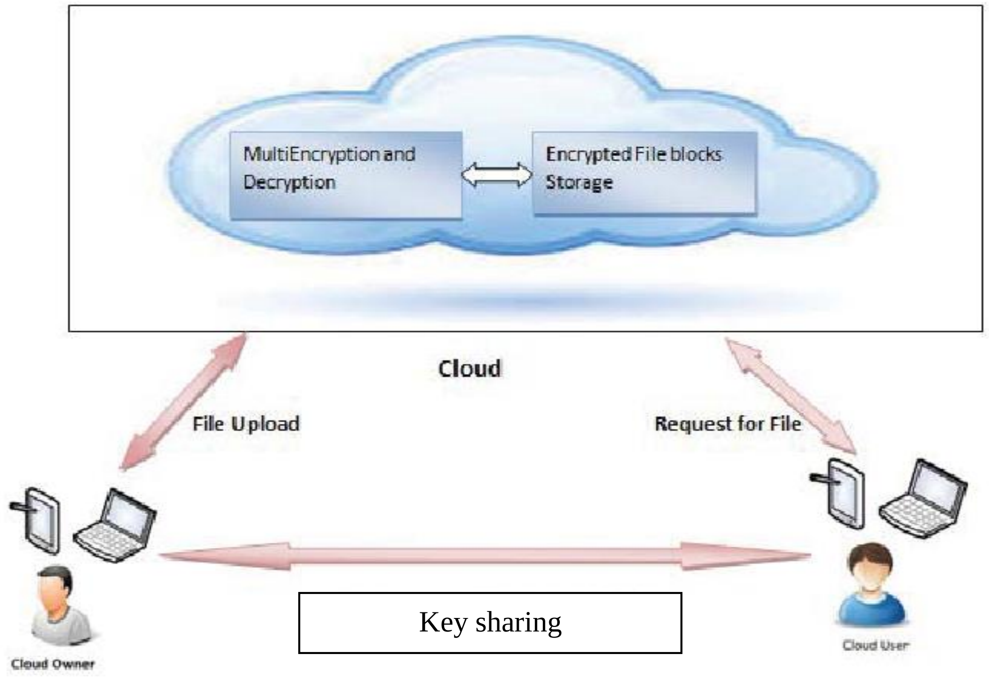


Figure 1.1: Sending and receiving of file in Cloud

1.2 Problem Statement

In cloud, data and files are shared among all servers, users and individuals. So, it is difficult for the cloud provider to make sure file security. Consequently, it becomes very easy for an invader to access and then misuse and destroy the integrity of important data. Cloud will be of no use if we compromise on these problems. There are authentication and authorization issues and there is no data integrity. Once a person uploads a file on cloud, he will lose control on its file. Control will be shifted to Cloud Service Provider(CSP).Any unauthenticated and unauthorized user can access important data.

1.3 Proposed Solution

File Fox is providing a solution that secures and manages data on cloud. We are sending encrypted data to cloud so that if hacker gets the file then he doesn't have any key so he/she can't decrypt the text .Hashing is used to check integrity of file. We are deploying Private cloud so that there will be less security issues. Only authorized users can access data and files. Our objective is to keep control of file owner on it's file.

1.4 Project Scope

The proposed system will secure important data and files deployed on cloud. It fulfils security needs of data centre of cloud with minimum time with maximum throughput. It facilitates authorization of user for file/data and files will be infeasible to get breached. It is demand of data centres. It facilitates authorization of user for file. In countries, like Pakistan where security is an untouched topic, this system can be helpful in protecting data of NADRA, PIA, FAB, FIA, Telephone Companies, Ministries etc.

1.5 Aim and Objectives

1.5.1 Research Objectives

Study of different encryption algorithms, key sharing protocols, Public and Private cloud.

1.5.2 Academic Objectives

During this project we have developed sound knowledge and technical skills in the fields of:

- Working in the field of Cryptography
- GUI Development
- Implementation of different encryption algorithms, GUI in Python
- Programming Skills on Raspberry Pi (Practice on Python language)
- Cloud Deployment
- Key generation by Hashing
- Key sharing by Diffie-Hellman
- Synchronization of GUI with Cloud
- Use of Raspberry pi for alarm generation

The developed skills are both hardware and software in nature.

1.5.3 Other Objectives

- To support legislature and privately-owned businesses.

So, through this venture, we wish to coordinate our hypothetical information with reasonableness to increase further understanding and refine our abilities in all the fields referenced above.

1.6 Limitations

File Fox have certain limitations:

- Resources are fixed. When traffic is more than resources, we can’t manage it.
- Within network
- Internet Connection

1.7 Organization of thesis:

For clear understanding, thesis is comprised of 8 sections, short description of each section is as follows:

Chapter 1	It has covered a brief overview of File Fox. The problem statement and its solution, approach that File Fox has come up with.
Chapter 2	It deals with the brief overview of existing literature of this project which tells about algorithms, Cloud Computing.
Chapter 3	Hardware and Software requirements of project. A brief overview of design and explain methodology of project. Deliverables of project are discussed in it.
Chapter 4	Various applications and fields in which it can be used and resources used in doing project.
Chapter 5	Shows the future work and conclusion.
Chapter 6	Shows the references used.
Chapter 7	Appendix is given in which synopsis is also attached.

CHAPTER 2

LITERATURE REVIEW

2.1 Existing Literature

In this section, a short review of existing information with this venture is given. A basic knowledge regarding the Cryptography (Twofish, Rijndael) was a prerequisite before starting this project, hence various research papers and articles were studied.

A brief description of all the topics that were studied for the File Fox is given below.

2.1.1 Encryption Algorithms

2.1.1.1 Twofish:

Twofish is a well-known encryption algorithm generally used in cryptography. It is derived from Blowfish . Twofish is a 128-bit block cipher that acknowledges variable key length up to 256 bits. It has Feistel network and have 16 rounds with a bijective function made up of 4 key-subordinate S-boxes.

For secure and safe data, we should consider following parameters:

Imperceptibility: It is the property in which one is unable to separate between the original and the fixed information.

Robustness: It alludes to proportion of work required to end implanted data without ending spread information.

Embedding Capacity: It alludes to all clandestine data that can be implanted without foulness to nature of information.

Encryption: One portion of n-bit key is utilized as the genuine encryption key and the other portion of the n-bit key is utilized to change the encryption calculation.

Twofish Building Blocks

• **Feistel Network** It is a wide-ranging strategy for changing over capacity (normally called F network) into a stage. F does not change straightly and perhaps non surjective. In each round input to F is “source block” and yield is XORed with “bijective block” after which two blocks exchange for following round.

• **S-Boxes** An s-box is a table decided non direct substitution operation utilized in most block ciphers. S-boxes changes both input and yield measure and can be made either discretionarily or algorithmically. These s-boxes are made utilizing two fixed (8x8) bit permutations and key material.

• **MDS Matrices** It is a code over a field. It is a direct mapping from a field components to be field components, creating a composite vector of a + b components. The separation between two vectors created by MDS mapping is at any rate b + 1. It can without much strength be demonstrated that no mapping can have bigger most extreme separation between two vectors, subsequently the term MDS. Mapping can be signified by an MDS matrix comprise of (a x b) components.

• **Whitening** The method of XOR-ing key material before first round and after last round. It considerably expands the trouble of key persist assaults against the rest of the cipher.

• **Key Schedule** It is way by which the key bits are bent into round keys that the cipher can utilize. Twofish needs a ton of key material, and has a confounded key schedule. The key schedule utilizes the same crude as the round function.

• **Pseudo – Hadamard Transforms (PHT)** It is a simple blending task that keeps running in programming giving two sources, a and b.

Features It has strong keys. Extra key length can be utilized on a wide assorted variety of platforms and applications and be reasonable for a stream cipher, hash function. To encourage simplicity of examination and usage twofish has simple design. A completely improved usage of Twofish scrambles on a Pentium Pro at 17.8 clock cycles per byte, and an 8-bit smart card implementation encrypts at 1660 clock cycles per byte. By cryptanalyzing, we saw that our best assault breaks 5 rounds with 222.5 picked plaintexts and 251 exertion.

2.1.1.2 AES (Rijndael):

AES represents Advanced Encryption Standard, known by its unique name Rijndael. It is a detail for the encryption of electronic information set up by the NIST in 2001.AES is a subset of the Rijndael block cipher created by two Belgian cryptographers, Vincent Rijmen and Joan Daemen, who presented a proposition to NIST amid the AES determination process. Rijndael is a group of ciphers with various key and block sizes.

For AES, NIST choose three individuals from Rijndael family, each with a block size of 128 bits, however three diverse key lengths: 128, 192 and 256 bits as shown in figure 2.3.

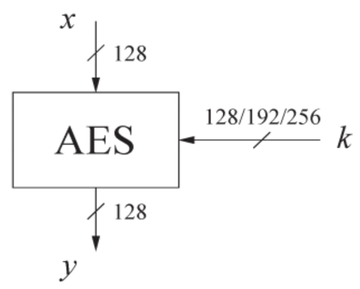


Figure 2.1: Rijndael Schematic Diagram

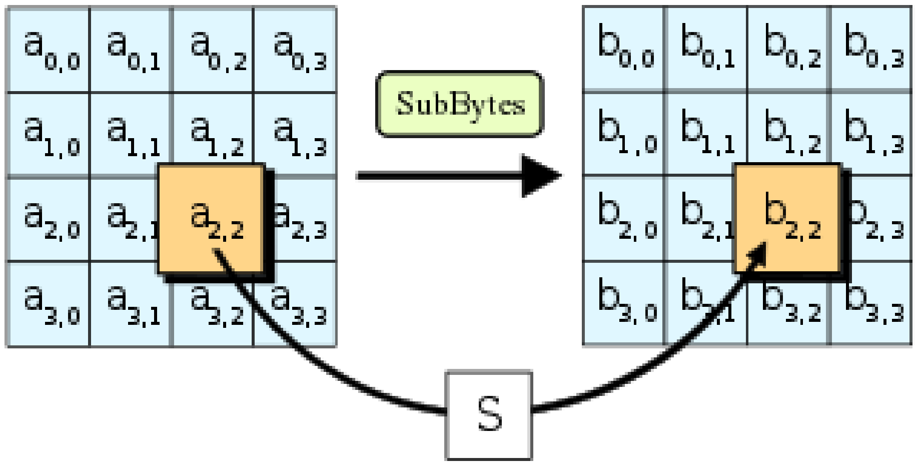
AES is presently utilized around world. It replaces the Data Encryption Standard (DES), which was distributed in 1977. The algorithm depicted by AES is a symmetric-key algorithm (same key is used for both scrambling and decoding the information).

It has four steps:

Step 1: Sub-Bytes step

Each byte in the state is replaced with its entry in a fixed 8-bit lookup table, S; bij = S(aij). In the Sub-Bytes step, each byte in the state array is replaced with a Sub Byte using an 8-bit substitution box as shown in figure below. It provides non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF (2^8), known to have good nonlinearity properties.

To maintain strategic distance from assaults dependent on straightforward algebraic properties, the S-box is developed by joining the inverse function with an invertible affine transformation. While decoding, the Inverse of Sub-Bytes step is utilized. It requires first taking the inverse of the affine transformation after that finding the multiplicative inverse.



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
S

Figure 2.2: AES Sub-byte step

Step 2: Shift Rows step

Bytes in each line of the state are moved consistently to the left as shown in figure below. Number of places every byte is moved contrasts for each line. Shift Rows step works on the rows of the state; it consistently moves the bytes in each row by specific balance.

For AES:

- The first row is left unaltered.
- Each byte of the second row is moved one to the left.
- The third row is moved by counterbalances of two.
- The fourth row is moved by balances of three respectively.

Thus, each column of the yield state of the Shift Rows step is made out of bytes from each column of the information source.

The significance of this progression is to stay away from columns being scrambled freely, in which case AES deteriorates into four autonomous block ciphers.

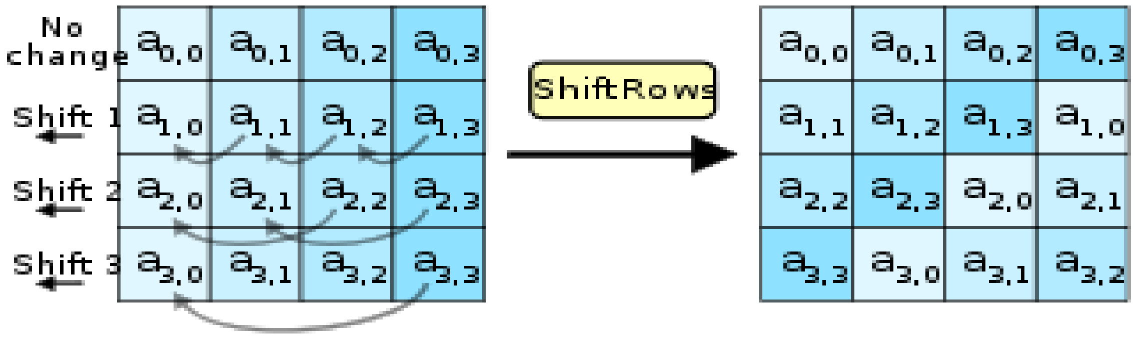
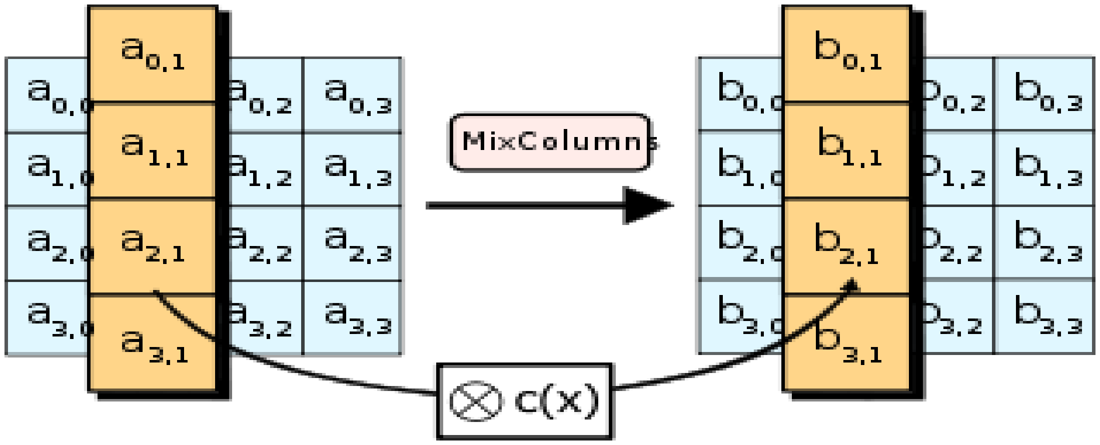

No change
Shift 1
Shift 2
Shift 3
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

Figure 2.3: Shift row step

Step 3: The Mix-Columns step

Each column of the state is multiplied with a fixed polynomial as shown in figure below. The four bytes of each column of the state are combined using an invertible linear transformation. The Mix Columns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with Shift Rows, Mix Columns provides diffusion in the cipher. During this operation, each column is transformed using a fixed matrix (matrix left-multiplied by column gives new value of column in the state): Matrix multiplication is composed of multiplication and addition of the entries. Addition is simply XOR. Multiplication is modulo irreducible polynomial. If processed bit by bit, then, after shifting, a conditional XOR with $1B_{16}$ should be performed if the shifted value is larger than FF_{16} (overflow must be corrected by subtraction of generating polynomial).



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$\otimes c(x)$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Figure 2.4: MixColumn step

Step 4: Add Round Key step

The subkey is combined with the state. For each round, a subkey is derived from the main key using AES's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.

2.1.2 Cloud Computing

It is sort of web-based computing which gives shared PC handling, assets and information to PCs and different gadgets. In order to access cloud services, we need active internet connection. Cloud computing gives following facilities:

- **IaaS (Infrastructure as a service)**
  It helps in virtualization so that we can create instances and build up infrastructure the way user wants. Infrastructure as a service are online services that provide high-level APIs used to dereference various low-level details of underlying network infrastructure like physical computing resources, location, data partitioning, scaling, security, backup etc.

- **PaaS (Platform as a service)**
  It gives platform so as to create applications. It is a cloud computing model in which a third-party provider delivers hardware and software tools (usually those needed for application development) to users over the internet. A **PaaS** provider hosts the hardware and software on its own infrastructure.

- **SaaS (Software as a service)**
  Software is available in cloud and we can place data(information) into it. Google, Gmail are examples of it. It is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. It is sometimes referred to as "on-demand software", and was formerly referred to as "software plus services" by Microsoft.

Openstack lies in category of IaaS. IaaS and PaaS are quite similar. In IaaS we have a type of platform in which we can create our application. In PaaS, platform like operating is already present while in IaaS platform is of user's choice. Openstack is an open source programming that gives cloud environment (Public as well as Private cloud). It is overseen by Openstack foundation. Openstack is fate of cloud computing. It was created in 2010 by NASA. Openstack community is huge. IBM, suse, hp, CISCO, DELL, redhat, VMware are contributors of openstack.

Features of openstack:

- It is opensource. We can get it free.
- We can modify anything we need based upon infrastructure that we want to create.

There are five deployment models of openstack as shown in figure 2.12.

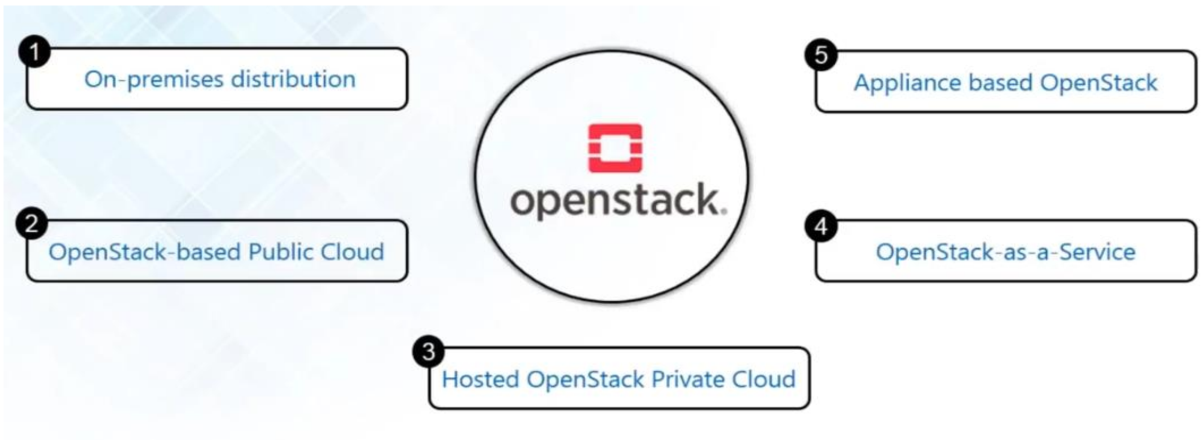

1
On-premises distribution
2
OpenStack-based Public Cloud
openstack.
3
Hosted OpenStack Private Cloud
4
OpenStack-as-a-Service
5
Appliance based OpenStack

Figure 2.5: Openstack Deployment Models

- **Openstack based Public cloud** It supports all users who want to make use of a computing resource, such as hardware (OS, CPU, memory, storage) or software (application server, database) on a subscription basis. Most common uses of public clouds are for application development and testing, non-mission-critical tasks such as file-sharing, and e-mail service.

- **Hosted Openstack Private cloud** It is typically infrastructure used by a single organization. Such infrastructure may be managed by the organization itself to support various user groups, or it could be managed by a service provider that takes care of it either on-site or off-site. Private clouds are more expensive than public clouds due to the capital expenditure involved in acquiring and maintaining them. However, private clouds are better able to address the security and privacy concerns of organizations today.

- **Hybrid cloud** In a hybrid cloud, an organization makes use of interconnected private and public cloud infrastructure. Many organizations make use of this model when they need to scale up their IT infrastructure rapidly, such as when leveraging public clouds to supplement the capacity available within a private cloud. For example, if an online retailer needs more computing resources to run its Web applications during the holiday season it may attain those resources via public clouds.

- **Community cloud** It supports multiple organizations sharing computing resources that are part of a community; examples include universities cooperating in certain areas of research, or police departments within a county or state sharing computing resources. Access to a community cloud environment is typically restricted to the members of the community.

We are using Private cloud model in project. In private cloud, seller has openstack based private cloud including fundamental equipment based and openstack programming. Servers support this and data and application that we are implementing on cloud is managed by us.

- **Architecture**

Components that make up openstack are as follows:

- Horizon (Dashboard)
- Nova (Compute domain)
- Glance (Service for launching instances)
- Swift (Object store service)
- Neutron (Networking service)
- Cinder (Block stockpiling administration)
- Heat (Orchestration purposes)
- Ceilometer (Billing administration)
- Keystone (Authentication and authorization purposes)

CHAPTER 3
DESIGN AND DEVELOPMENT

In this chapter we discuss about the design of system and briefly explain every depended part to produce end deliverable, followed by this go through each part in detail with discussing methodology. Besides this working of each part is theoretically analyze in this chapter.
pter.

3.1 - Approach:

- Study about different encryption algorithms
- Selection of algorithms
- Study about key sharing
- Selection of protocols for key sharing
- Selection of Hash for Key generation
- Study about GUI (Graphical User Interface)
- Research about Public cloud and Private cloud
- Storage of files/data on cloud
- Synchronization of GUI with Cloud and with other component
- Bug Testing

3.2 – Preliminary Design:

The details of design are given below:

3.2.1 Hardware Specification

The project consists of the following modules:

- ➢ Raspberry Pi
- ➢ Rechargeable Battery
- ➢ PC

3.2.1.1 Raspberry Pi:

It is also known as RPi and is released on 29 February 2012.It is intended for industrial application. It's introductory price was $35.It's Model B is a small credit card size PC and also has a keyboard, mouse, display, power supply, micro SD card with installed Linux Distribution. We can run applications from word processors and spreadsheets to games. It is a progression of tiny single-board PC created in UK by Raspberry Pi Foundation. The first model ended up unquestionably more mainstream than evaluated selling outside its objective market for utilizations, for example robotics. In any case, a few accessories have been incorporated into official and unofficial bundles.

It works on Linux, FreeBSD, NetBSD, OpenBSD, Plan 9, RISC OS, Windows 10 IoT Core, Windows 10 ARM64.

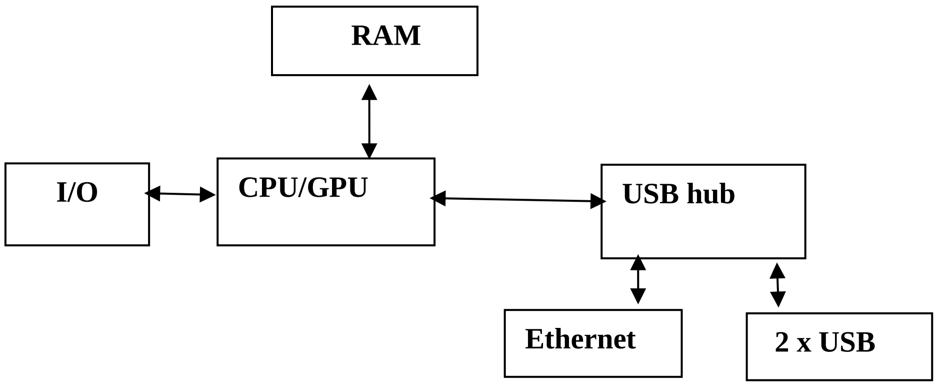


Figure 3.1: Raspberry Pi Working Model

Working of raspberry pi is shown in figure 3.1.We give input to raspberry pi and input is stored in RAM of pi.We connect pi through internet to our system.

Raspberry Pi may be operated with USB computer, keyboard and mouse. It may be likewise utilized with USB storage, USB to MIDI converters, and virtually any other gadget/component with USB capabilities.

Different peripherals can be joined through the various pins and connectors on the outside of Raspberry Pi.

Applications of Raspberry pi:

- Use in   education
- Use in home   automation
- Use in   industrial automation
- Use in commercial products

Figure 3.2: Raspberry Pi

Working:

Log files of our data will be stored in Neutron. Our raspberry pi will continuously monitor those log files. Whenever there is any change in the log files, it means invader is trying to destroy integrity. At that instant, raspberry pi will generate an alarm; alerting the admin to have a look on it's data. In this way we will secure our data.

3.2.1.2 Rechargeable Battery:

It is sort of electrical battery which can be charged, discharged, and revived ordinarily. It is also known as secondary cell or accumulator. The term "accumulator" is used as it gathers and stores energy through reversible electrochemical response.It is made up of at least one electrochemical cells.

It is used in

- automobile starters
- portable consumer gadgets
- light vehicles (mechanized wheelchairs, electrical bikes)
- uninterruptible power supplies
- battery capacity control stations.

3.2.1.3 PC

It is a multi-purpose computer whose size, capacities, and value make it plausible for individual use. These are planned to be worked straightforwardly by an end client, rather than by a computer expert or technician. In contrast to huge expensive minicomputers and centralized computers, time-sharing by numerous individualism the mean time is not utilized with PCs.

Institutional or corporate PC proprietors during 1960s needed to compose their own projects to do any helpful work with the machines. While PC clients may build up their very own applications, more often than not these systems run business programming, free-of-charge software ("freeware") or free and open-source programming, which is given in ready-to-run structure. Programming for PC is commonly created and disseminated freely from the hardware or operating system producers.

Requirement for cloud deployment is 8th Generation PC .1TB of Hard disk is required for it. We need 2x2 processor.

3.2.2 Software Specifications:

Software used in File Fox is as follows:

- **Openstack** (It is open source platform that provides complete public as well as private cloud environment and mostly configured as IaaS)
- **Ubuntu** (Operating system for cloud deployment)
- **Visual studio** (Implementation of AES, Twofish, GUI is done in it)

  Code is available in Appendix A.

3.3 Cloud Deployment:

Requirements for Installation of Openstack:

- Memory 4GB
- Hard disk 45 GB
- Ubuntu 16.04.2 LTS
- Processor 4 (2x2)

Steps for installation:

1. Login into ubuntu using account with super user. You may also login with sudo privilages.
2. To update and upgrade operating system, run these commands:
   ```
   $ sudo apt-get update
   $ sudo apt-get upgrade/ sudo apt-get distro upgrade
   ```
3. Add user stack using command given below:
   ```
   $ sudo adduser stack
   ```
4. To get into root mode, run commands given below:
   ```
   $ sudo –i
   $ echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
   ```
5. In order to verify the command has been run and added into /etc/sudoers, run the following command :
   ```
   $ more /etc/sudoers
   ```

Line has been added what we have echo'ed above.

6. Sign in and out again with user "stack" which we have created.
7. Install git by following command:

   $ sudo apt-get install git
8. Clone devstack files using given commands:

   $ git clone https://git.openstack.org/openstack-dev/devstack
9. Enter into devstack directory and create file " local.conf " using these command:

   $ cd devstack

   $ nano local.conf

   Add given lines into above created file. It includes passwords which will be used during installation and login.

   ```
   [[local|localrc]]
   ADMIN_PASSWORD=secret
   DATABASE_PASSWORD=$ADMIN_PASSWORD
   RABBIT_PASSWORD=$ADMIN_PASSWORD
   SERVICE_PASSWORD=$ADMIN_PASSWORD
   ```
10. Run shell script by following command:

    $ ./stack.sh

    We will get host IP address of openstack that will be used in browser to access openstack dashboard after execution of this script successfully.
11. IP address that we got is used to openstack dashboard terminal. Use username "Admin" and password "secret" to login.

Configuration:

To configure Iaas, we have to configure network, computing and storage facility for users.

Steps:

- IP address which we got in installation is used to open Openstack dashboard terminal. Use username " Admin " and password " secret" to login.
- Click on Project>>System>>Hypervisor to launch instances by nova computing

services.

- Unlaunch the instances in openstack. Create the volume. Cirros operating system is also mounted in it. In this step, required steps are Volume name, size and type.
- Create network. For it, use neutron service. We have to configure virtual router and IP subnets.
- Click on Create Network and type Network type and state should be configured to up.
- For external network, Create virtual router. External network is used as public network from list. Configure Router name and router working state. Also configure Network topology.

  Also configure Network topology.
- Configure instances. Configure instances name, flavor, network and all other necessary options which are required for it.

  In Network tab, configure stack network and launch instances. Instances page will appear.
- Allocate floating Ip address.
- Configure Security group to apply security rules or filtration on set of IP's. Edit existing group specific for ingress traffic for TCP port 22. It permits SSH from host machine on which openstack is installed.

  A keypair is created that is used on private network. It is generated by clicking on key pair option.

  Download the key into the /home/devstack .

  To use the cirros virtual machine remotely, use following command:

$sh –i name-keypair.pemcirros@192.168.187.149.

3.4 Methodology

Owner of file encrypts his file by using AES or Twofish as shown in figure 3.5 and then he uploads his encrypted file on cloud. When another user wants this file, he will request owner for key exchange. Look-up table helps to determine whether that user is authorized user or not. If he is authorized user then 128 bit session key is generated by Diffie-Hellman and is sent to other user by socket programming. User will download requested file and decrypt it with key provided as shown in figure 3.7. If he is unauthorized/unregistered user then raspberry pi will generate an alarm to tell owner that it is unauthorized user and request of user will be declined.

Complete methodology is shown in figure 3.3.

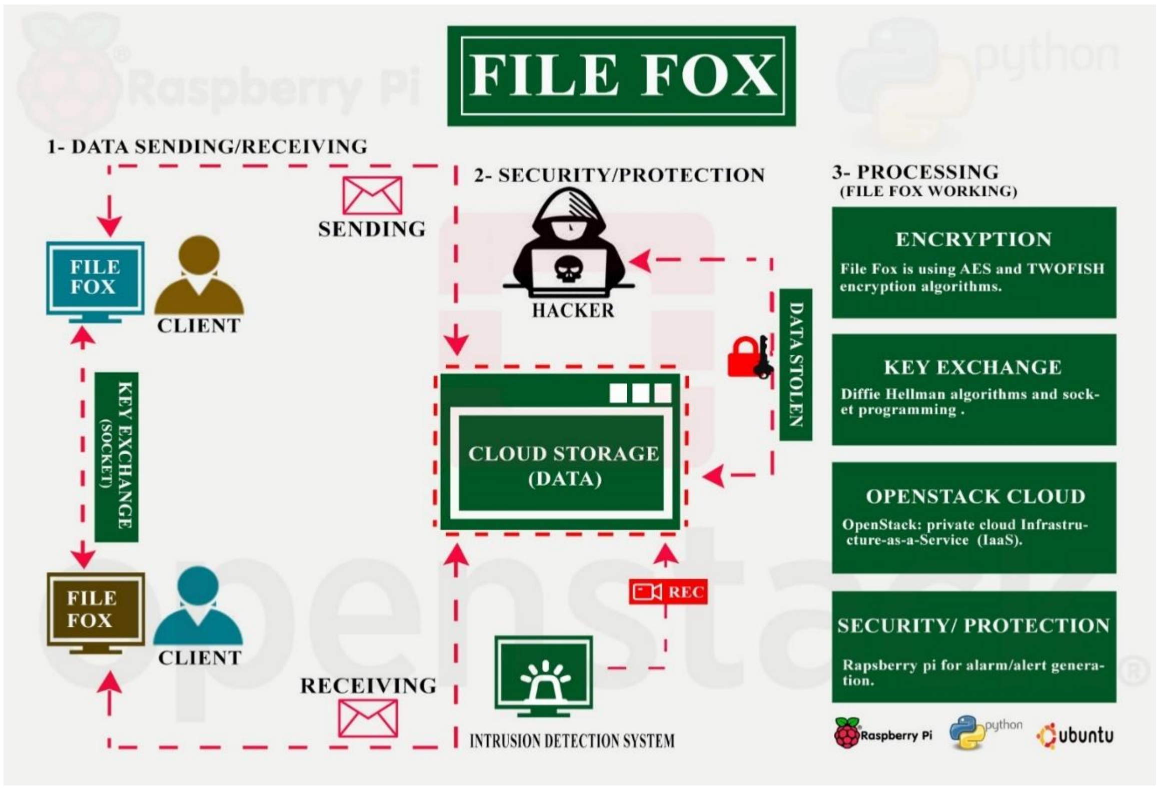

FILE FOX
1- DATA SENDING/RECEIVING
2- SECURITY/PROTECTION
3- PROCESSING
(FILE FOX WORKING)
SENDING
FILE FOX
CLIENT
HACKER
DATA STOLEN
KEY EXCHANGE (SOCKET)
CLOUD STORAGE (DATA)
REC
FILE FOX
CLIENT
RECEIVING
INTRUSION DETECTION SYSTEM
ENCRYPTION
File Fox is using AES and TWOFISH encryption algorithms.
KEY EXCHANGE
Diffie Hellman algorithms and socket programming .
OPENSTACK CLOUD
OpenStack: private cloud Infrastructure-as-a-Service (IaaS).
SECURITY/ PROTECTION
Rapsberry pi for alarm/alert generation.
Raspberry Pi
python
ubuntu

Figure 3.3: File Fox Block diagram

3.4.1 Lookup Tables:

Keystone is a component in our dashboard of cloud that provides us authentication and authorization. It manages look up table which has MAC addresses of users and their authority rules. Whenever any user tries to access our data, keystone will check which files that user can access and then authorize user accordingly.

Table 1: Lookup table

File Name	**Data Owner**	**Hash**	**IP**	**Encryption method**
File1	X	XXXXXXXX	192.168.10.2	AES
File2	Y	XXXXXXX	192.168.10.3	TWOFISH
File3	Z	XXXXXXX	192.168.10.4	AES

3.4.2 GUI:

We have made GUI in python 3.7 on Visual studio. We have used tkinter library while coding. Then we binded encryption codes with it. When we run the code, GUI will ask whether user wants to encrypt or decrypt data.

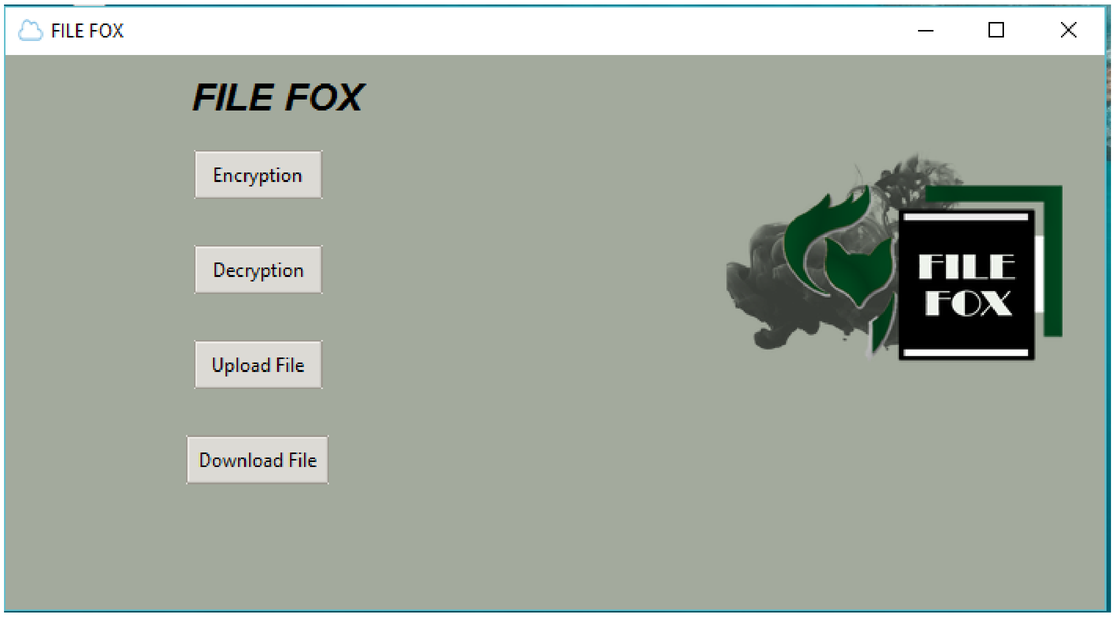


Figure 3.4: GUI

If user wants to encrypt data, he will click on encrypt option on GUI.

Then GUI will ask about encryption algorithm (AES, Twofish) as shown in figure 3.5.

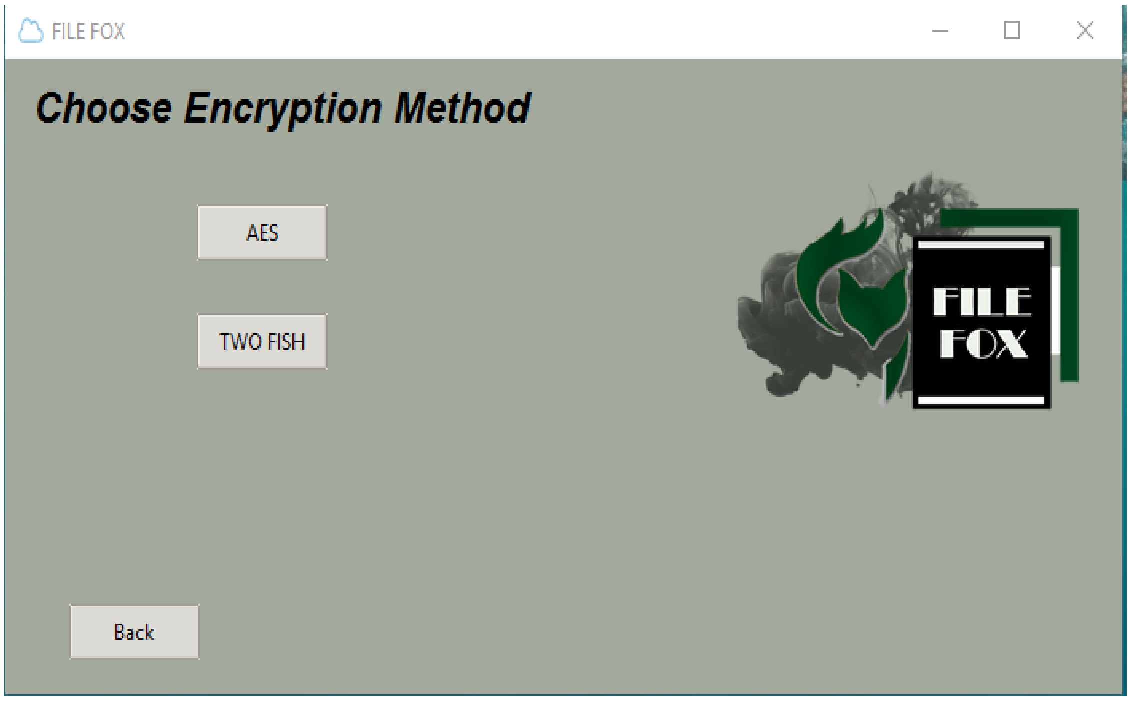


Figure 3.5: Selection of encryption method

After selecting encryption algorithm, user will browse the file that he wants to encrypt and user will add 16-byte key.

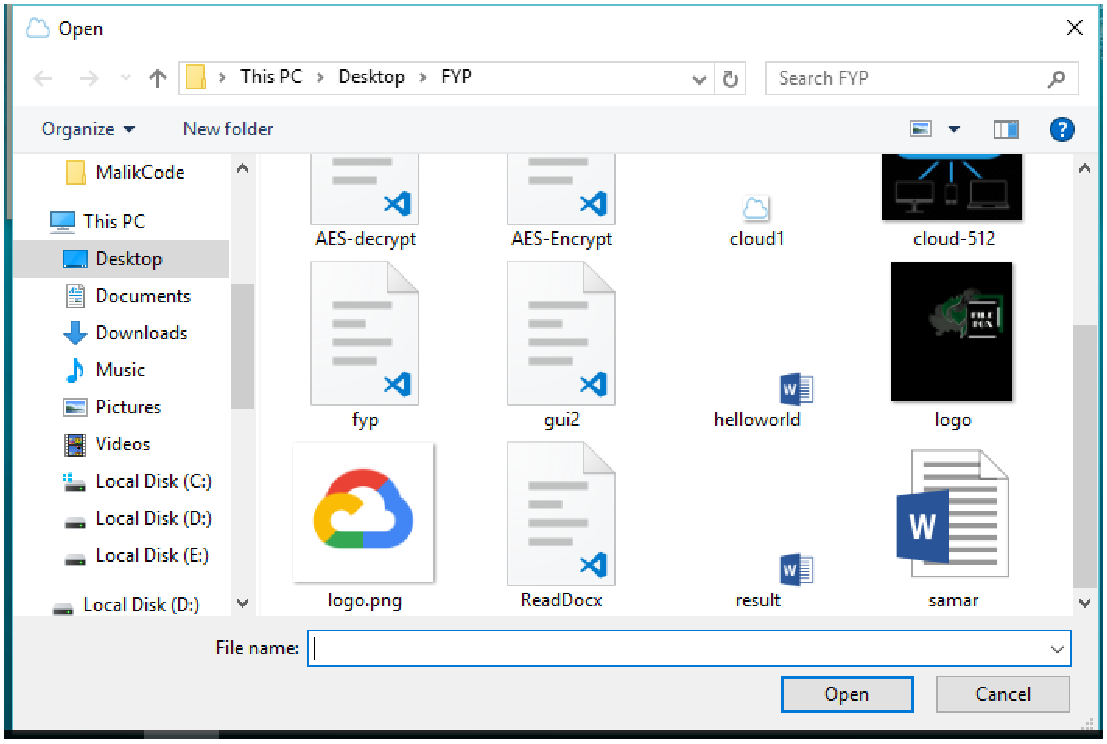


Figure 3.6: Browsing file

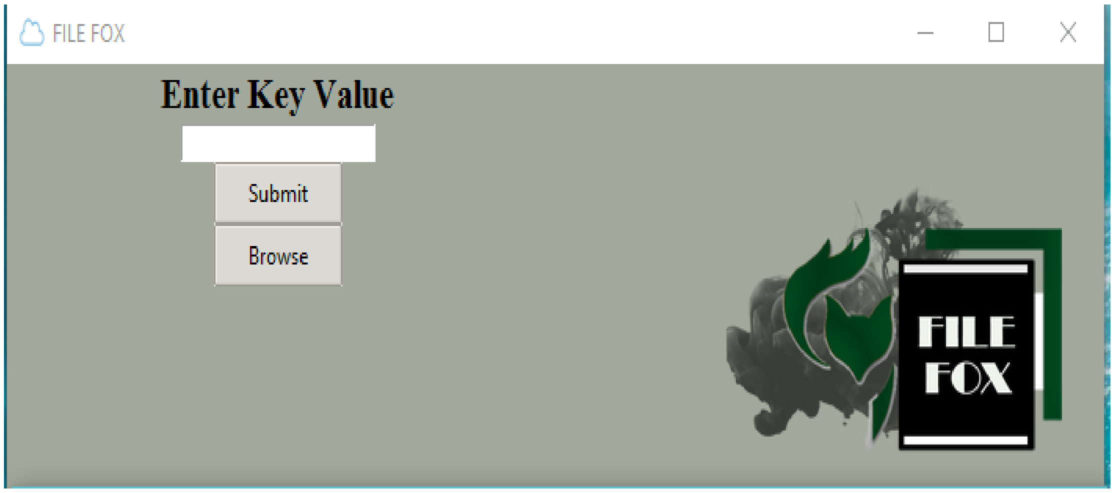


Figure 3.7: Entering Key

File will be encrypted and be stored on Cloud. If user wants to decrypt file, he will open GUI. He will select decrypt option. He will select decryption method (AES, Twofish). He will browse the encrypted file to decrypt and add a 16-byte key to decrypt key.

3.4.3 Socket Programming

In socket programming, we have two clients, client 1 and client 2. If clients 1 want the key then he will establish a session with client 2 through socket programming. After establishing the connection, client 1 will request for the exchange of parameters.Then client 2 will listen the request and as a result he will exchange the parameters.

3.5 Special Skills Required

- Python
- HTML
- SQL
- Raspberry pi coding
- Cloud Computing knowledge

3.6 Deliverables

- The stored file is completely secured.
- The key is also safe.
- The system is very secure and robust in nature.
- Developed a system which is easy to use.
- User authentication and authorization.
- Deployed Private Cloud

CHAPTER 4

APPLICATIONS AND RESOURCES REQUIRED

4.1- Applications

Following are the applications of our project:

4.1.1 Industry

In industrial sector, there are important files like Product sheets, Repairing and Maintenance data, Progress reports are secured on cloud.

4.1.2 Military

In Military, in order to secure our Intelligence reports, Military data about different missiles and weapons, different military plans on cloud, we use this technique.

4.1.3 Media

Web based life channels, corporate structures and individual purchasers are for most part changing to heavenly universe of distributed computing.

4.1.4 Academics

In academics, universities can store and secure student's personal and academic records on cloud.

4.1.5 Government

In Government sector, offices like:

- **NADRA** (They have huge amount of important data e.g. Biometric records, refugee records, incoming and outgoing records.)
- **PIA** (Data about flights, Details of passengers on a flight is on cloud.)
- **FAB (**There is important data like details of Allocation of frequency spectrum to

different telecommunication or broadcast service providers; that should be secured.**)**

- **FIA** (Important data like Criminal reports, investigation reports, intelligence reports of different operations.)
- **Telephone Companies (**They have contacts, package and subscription details of all their customers.)
- **Shopkeeper (**Financial details, business correspondence details, customer order details)
- **Ministries**
- **Legislative reports**

4.1.6 Business

In business, we use it to store and secure business reports, planning, company details on cloud. Only authorized person like company's employers can access these files and data.

4.2 Resources Required

List of all resources used is given below :

Table 2: List of Resources

Serial #	Items	Quantity	Cost (Rs)
1	PC	1	86,500
2	Raspberry pi	1	5,500
		Total	92,000

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Future Work

- Resources can be enhanced so that more people may use it.
- We have done work on private cloud. In future. data can be managed on Public cloud that would benefit much people.

5.2 Conclusion

File Fox is security need of data centers of Cloud.It facilitates authorization of valid user for accessing a file. As data sent to Cloud is encrypted so it is infeasible to get breached. Hence it is concluded that it is demand of modern era to store files on cloud as it is providing backup to our files on PC.

CHAPTER 6

REFERENCES

[1] C. Burwick, D. Coppersmith, E. D‟Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas, L. O‟Connor, M. Peyravian, D. Safford and N. Zunic, MARS – a candidate cipher for AES, NIST AES Proposal, Jun 98. http://csrc.nist.gov/encryption/aes/round2/AESAlgs /MARS/mars.pdf, Aug. 1999.

[2] Schneier, J. Kelsey, D. Whiting, D. Wagner,C. Hall, and N. Ferguson, "TwoFish: A 128-bit Block Cipher", AES submission, june 1998.

[3] Dr. S.A.M Rizvi, Dr. Syed Zeeshan Hussain et.al "Performance Analysis of AES and TwoFish Encryption Schemes", International Conference on Communication Systems and Network Technologies 2011

[4] Shun-Lung Su, Lih-Chyau Wuu, and Jhih-Wei Jhang, "A New 256-bits Block Cipher –Twofish256"

[5] Shiho Moriai, Yiqun Lisa Yin. "Cryptanalysis of Twofish (II)".Technical Report, IEICE, ISEC2000-38, 2000

[6] S. Hesham and Klaus Hofmann , "High Throughput Architecture for the Advanced Encryption Standard Algorithm" IEEE,International Symposium on Design and Diagnostics of Electronic Circuits & Systems, pages 167- 170,April 2014.

[7] M. Nagle, D. Nilesh, "The New Cryptography Algorithm with High Throughput",IEEE, ICCCI ,pages 1-5,January 2014.

[8] N. Sharma ,A.Hasan, "A New Method Towards Encryption Schemes (Name-Based-Encryption Algorithm)",IEEE, International Conference on Reliability, Optimization and Information Technology,pages 310-313,Feb 2014.

[9] Inder Singh, M. Prateek," "Data Encryption and Decryption Algorithms using Key Rotations N. Sharma ,A.Hasan, "A New Method Towards Encryption Schemes (Name-Based-Encryption Algorithm)",IEEE, International Conference on Reliability, Optimization and Information Technology,pages 310-313,Feb 2014.

[10]] S. Munjal1, S. Garg, “Enhancing Data Security and Storage in Cloud Computing Environment”, IJCSIT, Vol. 6, ISSN 0975- 9646, pages 2623-2626,2015

[11] P. Kanchan, “ Use of Digital Signature with Diffie Hellman Key Exchange ”, Volume 5, Issue 6, ISSN 2250- 3153 ,pp 1-4, June 2015

CHAPTER 7

APPENDICES

7.1 Appendix A

Synopsis

Extended Title: Secure file storage on Cloud

Brief Description of The Project /Thesis with Salient Specifications: In this project, we have introduced security mechanism in which symmetric key cryptography algorithm are used to encrypt our data, this cipher text will be sent to cloud where it will be stored. Only authentic user can access this data. If an unauthentic user wants to fetch data, Pie will generate the alarm.

Scope of Work: Aim of this project is to secure our data from unauthorized people. As we will deploy our own private cloud which will work within network therefore only authorized user can access the data. Data will be secured and remains under the control of user.

Academic Objectives:

- Study of cloud and its applications
- Data security on cloud
- Study of different techniques of encryption/decryption
- GUI

Application / End Goal / Objectives:

- The stored file is completely secured.
- The key is also safe.
- The system is very secure and robust in nature.
- Develop a system which is easy to use.

Material / Resources Required:

- **Software Requirements:**
  1. Visual Studio
  2. Ubuntu
  3. Openstack

- **Hardware Components:**
  1. Raspberry Pi
  2. PC

No of Students Required: 04

Group Members:	NC Sardar Waqas Umar (Group Leader)
	NC Nasira Hasnain
	NC Samar Abbas
	NC Asad Tariq Qureshi

Special Skills Required:

- Python
- Cloud computing knowledge
- HTML
- SQL

7.2 Appendix B

7.2.1 Twofish code

```
Spyder Editor
"""
import chilkat
import sys

#  Create two instances
dhA = chilkat.CkDh()
dhB = chilkat.CkDh()

#  Unlock the component once at program startup...
success = dhB.UnlockComponent("Anything for 30-day trial")
if (success != True):
    print(dhB.lastErrorText())
    sys.exit()

#  Prime  P, and a generator, G.
#  These are Public and may be sent over insecure channel.
#  G < P and it might be 2 or 5
dhB.UseKnownPrime(2)

#  Shared secret  will be equal to the size of the prime (in bits).
#  Bob will  send P and G to Alice.
```

```
p = dhB.p()

g = dhB.get_G()


#  Alice calls SetPG to set P and G.  SetPG checks

#  the values to make sure it's a safe prime and will

#  return False if not.

success = dhA.SetPG(p,g)

if (success != True):

    print("P is not a safe prime")

    sys.exit()


#  E should be set to twice the size of the number of bits

#  in the session key.

#  Bob generates a random E (which has the mathematical

#  properties required for DH).

eB = dhB.createE(256)


#  Alice does the same:

eA = dhA.createE(256)


#  The "E" values are sent over the insecure channel.

#  Bob sends his "E" to Alice, and Alice sends her "E" to Bob.

#  "K" is the shared-secret.

#  Bob computes the shared secret from Alice's "E":

kB = dhB.findK(eA)
```

```
#  Alice computes the shared secret from Bob's "E":
kA = dhA.findK(eB)

#  kBob should be equal to kAlice and the expected
#  length (260 characters).  The strings contain the hex encoded bytes of
#  our shared secret:
print("Bob's shared secret:")
print(kB)
print("Alice's shared secret (should be equal to Bob's)")
print(kA)

#  For 128-bit session key for encryption, Bob and Alice should
#  both transform the raw shared secret using a hash algorithm ( MD5) that produces
#  the size of session key desired
            crypt = chilkat.CkCrypt2()
success = crypt.UnlockComponent("Anything for 30-day trial.")
if (success != True):
   print(crypt.lastErrorText())
   sys.exit()
crypt.put_EncodingMode("hex")
crypt.put_HashAlgorithm("md5")
sessionKey = crypt.hashStringENC(kB)
print(" Session Key:")
print(sessionKey)
```

```
#  Encrypt something...
crypt.put_CryptAlgorithm("Twofish")
crypt.put_KeyLength(128)
crypt.put_CipherMode("cbc")
iv = crypt.hashStringENC(sessionKey)
print("Initialization Vector:")
print(iv)
crypt.SetEncodedKey(sessionKey,"hex")
crypt.SetEncodedIV(iv,"hex")

#  Encrypt some text:
crypt.put_EncodingMode("base64")
cipherText64 = crypt.encryptStringENC("The quick brown fox jumps over the lazy dog")
print("Encrypted text")
print(cipherText64)

#  Decrypt some text:
crypt.put_EncodingMode("base64")
plain text = crypt.decryptStringENC(cipherText64)
print("Decrypted text")
print(plain text)
```

7.2.2 Diffie-Hellman code

```
import sys

import chilkat

import os

import random

#  Create two separate instances of the DH object.

dhBob = chilkat.CkDh()

dhAlice = chilkat.CkDh()

#  Unlock the component once at program startup...

success = dhBob.UnlockComponent("Anything for 30-day trial")

if (success != True):

    print(dhBob.lastErrorText())

    sys.exit()

#  The DH algorithm begins with a large prime, P, and a generator, G.

#  These don't have to be secret, and they may be transmitted over an insecure channel.

#  The generator is a small integer and typically has the value 2 or 5.

#  The Chilkat DH component provides the ability to use known

#  "safe" primes, as well as a method to generate new safe primes.

#  This example will use a known safe prime.  Generating

#  new safe primes is a time-consuming CPU intensive task

#  and is normally done offline.

#  Bob will choose to use the 2nd of our 8 pre-chosen safe primes.

#  It is the Prime for the 2nd Oakley Group (RFC 2409) --

#  1024-bit MODP Group.  Generator is 2.
#  The prime is: 2^1024 - 2^960 - 1 + 2^64 * { [2^894 pi] + 129093 }
dhBob.UseKnownPrime(2)
#  The computed shared secret will be equal to the size of the prime (in bits).
#  In this case the prime is 1024 bits, so the shared secret will be 128 bytes (128 * 8 =
1024).
#  However, the result is returned as an SSH1-encoded bignum in hex string format.
#  The SSH1-encoding prepends a 2-byte count, so the result is going  to be 2 bytes
#  longer: 130 bytes.  This results in a hex string that is 260 characters long (two chars
#  per byte for the hex encoding).
#  Bob will now send P and G to Alice.
p = dhBob.p()
g = dhBob.get_G()

#  Alice calls SetPG to set P and G.  SetPG checks
#  the values to make sure it's a safe prime and will
#  return False if not.
success = dhAlice.SetPG(p,g)
if (success != True):
    print("P is not a safe prime")
    sys.exit()

#  Each side begins by generating an "E"
#  value.  The CreateE method has one argument: numBits.
#  It should be set to twice the size of the number of bits
#  in the session key.
#  Let's say we want to generate a 128-bit session key
#  for AES encryption.  The shared secret generated by the Diffie-Hellman
#  algorithm will be longer, so we'll hash the result to arrive at the
#  desired session key length.  However, the length of the session
#  key we'll utlimately produce determines the value that should be
#  passed to the CreateE method.
#  In this case, we'll be creating a 128-bit session key, so pass 256 to CreateE.
#  This setting is for security purposes only -- the value
#  passed to CreateE does not change the length of the shared secret
#  that is produced by Diffie-Hellman.
#  Also, there is no need to pass in a value larger
#  than 2 times the expected session key length.  It suffices to
#  pass exactly 2 times the session key length.
#  Bob generates a random E (which has the mathematical
#  properties required for DH).
eBob = dhBob.createE(128)

#  Alice does the same:
eAlice = dhAlice.createE(128)

#  The "E" values are sent over the insecure channel.
#  Bob sends his "E" to Alice, and Alice sends her "E" to Bob.
#  Each side computes the shared secret by calling FindK.
#  "K" is the shared-secret.
#  Bob computes the shared secret from Alice's "E":
kBob = dhBob.findK(eAlice)

#  Alice computes the shared secret from Bob's "E":
kAlice = dhAlice.findK(eBob)

#  Amazingly, kBob and kAlice are identical and the expected
#  length (260 characters).  The strings contain the hex encoded bytes of
#  our shared secret:
print("Bob's shared secret:")
print(kBob)
print("Alice's shared secret (should be equal to Bob's)")
print(kAlice)

#  To arrive at a 128-bit session key for AES encryption, Bob and Alice should
#  both transform the raw shared secret using a hash algorithm that produces
#  the size of session key desired.   MD5 produces a 16-byte (128-bit) result, so
#  this is a good choice for 128-bit AES.
#  Here's how you would use Chilkat Crypt (a separate Chilkat component) to
#  produce the session key:
crypt = chilkat.CkCrypt2()
success = crypt.UnlockComponent("Anything for 30-day trial.")
if (success != True):
    print(crypt.lastErrorText())
    sys.exit()
crypt.put_EncodingMode("hex")
crypt.put_HashAlgorithm("md5")

#sessionKey = crypt.hashStringENC(kBob)
#key= random.getrandbits(52)
x=chilkat.CkPrng()
key=x.genRandom(8,"hex")
print("128-bit session Key:")
print(key)
```

7.2.3 AES code

```
from Crypto.Cipher import blockalgo
from Crypto.Cipher import _AES

class AESCipher (blockalgo.BlockAlgo):
    def __init__(self, key, *args, **kwargs)
        blockalgo.BlockAlgo.__init__(self, _AES, key, *args, **kwargs)
def new(key, *args, **kwargs):.

    return AESCipher(key, *args, **kwargs)

#: Electronic Code Book (ECB). See `blockalgo.MODE_ECB`.
MODE_ECB = 1

#: Cipher-Block Chaining (CBC). See `blockalgo.MODE_CBC`.
MODE_CBC = 2
```

```
#: Cipher FeedBack (CFB). See `blockalgo.MODE_CFB`.
MODE_CFB = 3

#: This mode should not be used.
MODE_PGP = 4

#: Output FeedBack (OFB). See `blockalgo.MODE_OFB`.
MODE_OFB = 5

#: CounTer Mode (CTR). See `blockalgo.MODE_CTR`.
MODE_CTR = 6

#: OpenPGP Mode. See `blockalgo.MODE_OPENPGP`.
MODE_OPENPGP = 7

#: Size of a data block (in bytes)
block_size = 16

#: Size of a key (in bytes)
key_size = ( 16, 24, 32 )
```