# COMPUTATIONAL SECRET SHARING

By

**AmnaBatool**

**Filza Shahid**

**HaiderJaved**

**Urooj Fatima**

Submitted to the Faculty of Computer Science, Military College of Signals National University of Sciences and Technology, Islamabad in partial fulfilment for the requirements of a B.E Degree in Computer Software Engineering

JUNE 2019

## CERTIFICATE OF CORRECTNESS AND APPROVAL

*This is to officially state that the thesis work contained in this report*

**“Computational Secret Sharing”**

*is carried out by*

**AmnaBatool, Filza Shahid, HaiderJaved and Urooj Fatima**

*under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelors of Computer Software Engineering from National University of Sciences and Technology (NUST), Islamabad.*

Approved By:

Signature: ________________________________

Supervisor: **AP Waseem Iqbal**

MCS, Rawalpindi

# ABSTRACT

In today’s world everything is stored online. As the world moves towards cloud computing and distributed computing there is a great need for security. The data stored on cloud servers are of two kinds; Sensitive data and comparatively less sensitive data. Compromise on the security of any kind of data isn’t acceptable, especially in the case of sensitive information.Existing cloud system relies on Public Key Infrastructure to secure information, where there are risks around the possible existence of back-doors, front-doors, and a general lack of true understanding of encryption methods.Traditional methods for encryption are ill-suited for simultaneously achieving high levels of confidentiality and reliability.

Secret Sharing (also called secret splitting) refers to methods for distributing a secret amongst a group of participants, each of whom is allocated a share of the secret. The secret can be reconstructed only when a sufficient number, of possibly different types, of shares are combined together; individual shares are of no use on their own.

Computational Secret Sharing (**CSS**) is a scheme of one dealer and n players. The dealer gives a share of the secret to the players, but only when specific conditions are fulfilled will the players be able to reconstruct the secret from their shares.

## DECLARATION OF ORIGINALITY

We hereby declare that the work contained in this report and the intellectual content of this report are the product of the sole effort of our group, comprising of AmnaBatool, Filza Shahid, HaiderJaved and Urooj Fatima. No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere, nor does it include any verbatim of the published resources which could be treated as a violation of the international copyright decree. We also affirm that we do recognize the terms ‘plagiarism’ and ‘copyright’ and that in case of any copyright infringement or plagiarism established in this thesis, we will be held fully accountable of the consequences of any such violation.

*Dedicated to all those*

*who lead us on the journey*

*from ignorance to knowledge.*

# ACKNOWLEDGEMENTS

All of our gratitude is extended to Almighty Allah, the most beneficent, whose blessings empowered us to tread this journey.

We feel indebted to the benign faculty of Department of Computer Software Engineering, who raised us to these standards of knowledge; and especially to AP Waseem Iqbal, whose unending support, motivation and guidance made us capable of turning the concept of this project into reality.

We also extend our thankfulness to our families and friends, who kept our moral spirits high in the times of need.

# TABLE OF CONTENTS

# LIST OF TABLES

## LIST OF FIGURES

## KEY TO SYMBOLS OR ABBREVIATIONS

| | |
|---|---|
| **CSS** | Computational Secret Sharing |
| **App** | Application |
| **AS** | Assumption |
| **CO** | Constraints |
| **D** | Dependency |
| **HTML** | Hypertext Markup Language |
| **HTTP** | Hypertext Transfer Protocol |
| **HTTPS** | Hypertext Transfer Protocol Secure |
| **OE** | Operating Environment |
| **REQ** | Functional Requirement |
| **SDS** | Software Design Specification |
| **SE** | Security Requirements |
| **SF** | Safety Requirements |
| **SQL** | Structured Query Language |
| **SRS** | Software Requirements Specification |
| **STP** | Software Test Plan |
| **UD** | User Documentation |
| **UI** | User Interface |
| **URL** | Uniform Resource Locator |
| **MVC** | Model, View and Control |

| | |
|---|---|
| **N Shares** | Total Number of Key Shares produced from splitting the Original Key |
| **K Shares** | Required /Threshold Number of Key Shares required to reconstruct the Key |

CHAPTER 1

# Introduction

# 1. Introduction

## 1.1 Motivation

Cloud Computing has seen one of the most radical shifts within Information Technology, and theshift is most apparent both in the move from migrating private networks to virtualized networks, as wellas the move from private cloud systems onto public ones. Unfortunately these moves have not reallychanged the methods of providing security and robustness. Additionally, the publiccloud still suffers from doubts around large-scale outage risks and other security concerns.

Traditional methods for encryption are ill-suited for simultaneously achieving high levels of confidentiality and reliability.This is where Secret Sharing comes into play.

Computational Secret Sharing (**CSS**) is a technique, where the information to be made secret is encoded and distributed across multiple data shares, where each share of the information is distributed to the participants. The secret can be reconstructed only when a sufficient number, of possibly different types, of shares are combined together; individual shares are of no use on their own.

## 1.2 Problem Statement

Existing cloud system relies on Public Key Infrastructure to secure information, where there are risks around the possible existence of back-doors, front-doors, and a general lack of true understanding of encryption methods.

The Future Internet requires in-built data protection, either with a strong protective shell or with sharing methods which break data into secure fragments, and which have a strong enforcement policy to rebuild the data shares without relying on traditional encryption.

The Public Key Infrastructure is used nowadays is insufficient and the main objective of this project is to make the cloud systems more secure by breaking data into fragments and storing them at different servers.

## 1.3 Project Scope

Computational Secret Sharing (**CSS**) will be a Desktop Application that allows it's users to secure their sensitive data without compromising on space, confidentiality and minimizing the risk of loss.

Computational Secret Sharing (**CSS**) can be used for activities where simple passwords are an inadequate authentication method and more rigorous proof is required to confirm the identity of the parties involved in the communication and to validate the information being transferred.

### 1.3.1 External Scope

The scope of the project can be increased to be implemented and used in Pakistan Military Organizations, Government agencies and Intelligence Agencies.

## 1.4 Project Objectives

### 1.4.1 Academic Objectives

- To understand comprehensively and implement the concepts of Cryptography
- To understand comprehensively and implement the concepts of Encryption
- To understand comprehensively and implement the concepts of Decryption
- To understand the concepts of basic Application Development
- To go through the process of Professional Software Development

### 1.4.2 Application/End-Goal Objectives

The Application Objectives are as under:

- Users can choose between Encryption and Decryption
- The Application will Encrypt the data/file provided and generate a Key which shall be divided into the specified number of Key Shares (by the users)
- The Key Shares will be transferred to different users upon some authentication
- The Application will Decrypt the data/file provided upon feeding the required number of Key Shares (as specified by users while encryption)
- After some authentication, the Key Shares will be combined to form the original Key which will be used for Decryption

- An Interface will be provided for the execution of above steps
- A Database will be developed to store the Original Key's Hashes and other required information at the backend of the Application

## 1.5 Document Organization

The document looks at all the aspects of development and usage of Computational Secret Sharing(**CSS**) since its initial stage to its completion. Starting with Chapter 1 and 2 covering an Introduction and Literature Review of the topic, Chapter 3 and 4 further discusses the Process of Requirement Elicitation and Design Development. After completion of these stages, Chapter 5 and 6 of the document covers details regarding Implementation and Testing of the system.

## 1.6 Intended Audience and Reading Suggestions

The document is meant for the following stakeholders:

- **Project Supervisor:** to assist in project supervision and guiding the team in a better way.
- **Development Team:** to help in the development of product and trace-back of functional requirements.
- **Testing Team:** to help the testers to understand the applicable constraints.
- **Users:** The potential stakeholders of the system.

- **UG Project Evaluation Team:** to assist the evaluation committee in evaluatingthe progress of UG Projects.

## 1.7 Summary

This chapter gives a brief overview of the system and its functionalities, as well as this document, which covers all of the essential details regarding Computational Secret Sharing (**CSS**). This chapter also covers details regarding the scope and objectives of the system.

**CHAPTER 2**

# Literature Review

# 2. Literature Review

## 2.1 Introduction

One of the major risks in scaling existing systems into the Cloud is the current reliance on PKI (Public Key Infrastructure) to secure information. Protecting data in Cloud-based storage systems is a key challenge, and existing architectures often fail to properly control access rights to such data. The insider threat, especially from the likes of a System Administrator, and from unforeseen implementation issues, causes Cloud-based Systems to be flawed from many viewpoints. There are thus some key driving forces to ensure that data is protected in future Cloud-based infrastructures. Keeping in view the requirements of Cloud-based Systems, Computational Secret Sharing (**CSS**) will help with in-built data protectionwith sharing methods which break data into secure fragments which has a strong enforcement policy to rebuild the data shares without relying on traditional encryption. The users can use Computational Secret Sharing (**CSS**) for Encryption and Decryption which requires Key Shares of the Original Key. For Encryption the Key used to encrypt the data/file is divided into the required number of Key Shares and distributed among users so that security breachers cannot reveal much information even if they get a pass key for a user. For Decryption the Original Key is reconstructed from the threshold number of Key Shares and then used to decrypt the data/file. The project will be implemented in the form of a Desktop Application.

## 2.2 Product Perspective

Today Cloud Systems use Public Key Infrastructure to enable entities to securely communicate on an insecure public network and reliably verify the identity of an entity.The Public Key Infrastructure (PKI) is crumbling, partially due to the lack of a strong understanding of how encryption actually works, but also due to weaknesses in itsimplementation.As the computational of our computers is increasing along with the skill set of the people, the PKI (Public Key Infrastructure) is more liable to attacks now. The ways which we can assure security is either by making PKI (Public Key Infrastructure) more hard/strong, making a hybrid of it with some other algorithm or protocol, or just use another method for security.

This is where the Computational Secret Sharing (**CSS**) comes in to play. It aims to make sensitive data more secure by distributing a secret amongst a group of participants, each of whom is allocated a share of the secret. The secret can be reconstructed only when a sufficient number, of possibly different types, of shares are combined together (individual shares are of no use on their own).The semantics of the project will make it easier to provide further protection to the Cloud-based Systems. Furthermore, itis a new, stand-alone product.

It follows the following System Block Diagram:

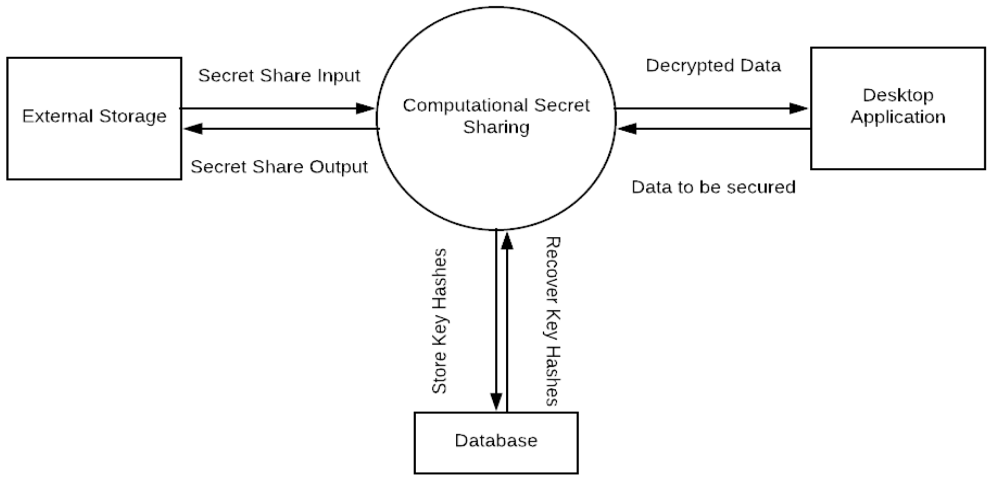


*Figure 2-1–System Block Diagram*

## 2.3 Product Features

Following are the major functions of the Computational Secret Sharing (**CSS**):

- Every user can use the Application for Encryption and Decryption
- For Encryption, the user will provide the data/file to be encrypted. The Application will generate a Key to encrypt the provided data/file
- The Original Key will be divided into a specified number of Key Shares and be distributed among different (specified) users after authentication
- For Decryption, the users will provide the required (threshold) number of Key Shares after authentication, which will be used to reconstruct the Original Key
- The reconstructed Key will be used to decrypt the provided data/file

- Failure to provide the required (threshold) number of Key Shares or entering incorrect Key Share will make the Application discard the old Key Shares and make new ones, thus providing further protection from attacks
- The Application will store the Key Shares, Hashes and other authentication specifications in a Database at the back-end

## 2.4 User Classes and Characteristics

The software has two types of users: Normal Users and Privileged Users. All the users have the same access level to the system and its data and can perform functions provided by the Application i.e. Encryption and Decryption; with the Privileged Users only having a few more functionalities available to them.

### 2.4.1 Normal Users

The Normal Users (such as testers, supervisors, other such users etc.) will use the Application for Encryption and Decryption of data/files. Each of these users can only get a single Key Share at the time of distribution of Key Shares of the Original Key.

### 2.4.2 Privileged Users

The Privileged Users, as the name suggests, have a few more advantages as compared to normal users. They can use the Application for Encryption and Decryption of data/files. Each of these users can get Key Shares up to the required (threshold) number of Key

Shares which is required for Decryption. Thus giving them the privilege to decrypt the data/file without the need of other users. These users are usually the ones at the top of the hierarchy of an organization.

## 2.5 Operating Environment

### 2.5.1 OE-01

The Desktop-based system of Computational Secret Sharing (**CSS**)will run on the computer system with following specifications:

- Pentium 4 or Higher CPU
- At least 512MB of RAM
- At least 1 GB of free disk space
- Windows 7 or later operating system
- Color monitor and a working internet connection

### 2.5.2 OE-02

Computational Secret Sharing (**CSS**) should be managed with Microsoft SQL Database Management System.

## 2.6 Design and Implementation Constraints

CO-01: Internet connection needed.

CO-02: Use of English language as the only means of communication in the system.

CO-03: Communication and speed of the system will be dependent on the network infrastructure specifications.

CO-04: The delivery of the key shares will only be available to the registered users over a secure network.

## 2.7 User Documentation

UD-01: Final release will be accompanied with auser guide to inform users how to use Computational Secret Sharing (**CSS**). User documentation that would be delivered along with the final product

- User Manual
- SRS Document
- SDS Document
- Test Plan Document
- Final Report

## 2.8 Assumptions and Dependencies

AS-01: Basic assumption for development of Computational Secret Sharing (**CSS**) is that the Application will be able to run on any System having Windows 7 (or onwards), thus being available for use 24/7.

AS-02: Users must know the User Interface for the better performance of the product.

AS-03: Limitations of the memory must be kept in mind by users.

AS-04: Users of Computational Secret Sharing (**CSS**) should be assumed to have access to a computer with internet access.

D-01: Overall performance of the product will depend on the network infrastructure.

D-02: Overall performance of the product will depend on the network speed.

CHAPTER 3

# Requirements

# 3. Requirements

## 3.1 Introduction

The purpose of this chapter is to present a detailed description of the Computational Secret Sharing (**CSS**). It will explain the purpose, features, interfaces, functionality, entire process, constraints and the application's reaction to external stimuli. It is intended for stakeholders and the system developers.

## 3.2 External Interface Requirements

### 3.2.1 User Interfaces

Computational Secret Sharing (**CSS**) consists of responsive Graphical User Interfaces (GUI) that will help the users to easily encrypt and decrypt data/files according to their needs/requirements. The learning curve for these Interfaces will be gradual, so as to make the Users of the Application feel at ease whenlearning about the options available to them.

### 3.2.2 Hardware Interfaces

#### 3.2.2.1 Computer System

The system shall have:

- Keyboard input

- Mouse input
- A monitor
- A working internet connection and the hardware requirements that come with it (Network card, Ethernet Port, Modem etc.)

#### 3.2.2.2 Database Server

- To retrieve/store data

### 3.2.3 Software Interfaces

- Primary Operating System supported by Computational Secret Sharing (**CSS**) Interface will be Windows 7
- Computational Secret Sharing (**CSS**) should be able to run on a Windows based platform, having Windows 7 or above
- Computational Secret Sharing (**CSS**) should be work with Microsoft SQL Database Management System

### 3.2.4 Communications Interfaces

- The system will be connected to a network via secure connection for communication
- The user must also enable firewall to enable communication

## 3.3 Functional Requirements

### 3.3.1 Encryption

#### 3.3.1.1 Description and Priority

The system will enable the user to encrypt his/her respective data/file. When the user opens the Application he/she will be provided with the option of Encryption and creating a new file for the cipher text.

Priority Level: High

#### 3.3.1.2 Stimulus/Response Sequences

Input: The user will open the Application and select the Encryption option

Output: The user will be led to a screen where he/she is to enter the data/file to be encrypted and also create a new file for the cipher text

#### 3.3.1.3 Functional Requirements

REQ-01: The system will allow the user to encrypt the required file on a single click

REQ-02: The system will allow the user to create a new file to store cipher text of the encrypted file

## 3.3.2 Key and Key Shares Generation

### 3.3.2.1 Description and Priority

This feature of the Application creates the key which is used to encrypt the respective data/file and divide the key into shares as specified by the users.When the user selects the encryption option, key generation at the back end begins. The key is used to encrypt the data/file and then divided into shares.

Priority Level: High

### 3.3.2.2 Stimulus/Response Sequences

Input: The user will select the Encryption option and enter the data/file to be encrypted along with the cipher text file

Output: The user is prompted to enter the total number and required number of Key Shares and then the Original Key is divided into the total number of Key Shares

### 3.3.2.3 Functional Requirements

REQ-03: The system will generate the Key on a single click of the Encryption button

REQ-04: The system will divide the Key into the total number of Key Shares when the total number of shares and required number of shares are entered

## 3.3.3 Key Distribution

### 3.3.3.1 Description and Priority

This feature of the Application transfers the Key Share to different users upon authentication. When the Key is divided into the total number of Key Shares, they are transferred to the users by means of USB after authentication of the users.

Priority Level: High

### 3.3.3.2 Stimulus/Response Sequences

Input: The Application asks for authentication to transfer the Key Shares

Output: The Key Shares are transferred to the users and Encryption is complete

### 3.3.3.3 Functional Requirements

REQ-05: The system will share the key shares with other users when some authentication is provided

REQ-06: The system will not transfer the Key Shares if the required authentication criteria is not met

### 3.3.4 Key Wrapping

#### 3.3.4.1 Description and Priority

This feature of the Application allows further protection of the Key Shares when they are transferred to the users by means of a USB. After the user has authenticated themselves, the Application further provides a protective shell so that no attacker can access the Key Share present in the USB.

Priority Level: High

#### 3.3.4.2 Stimulus/Response Sequences

Input: The user will authenticate themselves and the Key Share is further protected by Key Wrapping

Output: The system will transfer the now further protected Key Share into the USB provided by the user

#### 3.3.4.3 Functional Requirements

REQ-07: The system will wrap the Key Share with the Key Wrapping algorithm to provide another protective shell for the secret

### 3.3.5 Decryption

#### 3.3.5.1 Description and Priority

The system will enable the user to decrypt his/her respective data/file. When the user opens the Application he/she will be provided with the option of Decryption and creating a new file for the decrypted data.

Priority Level: High

#### 3.3.5.2 Stimulus/Response Sequence

| | |
|---|---|
| Input: | The user will open the Application and select the Decryption option |
| Output: | The user will be led to a screen where he/she is to enter the data/file to be decrypted and also create a new file for the decrypted data |

#### 3.3.5.3 Functional Requirements

| | |
|---|---|
| REQ-08: | The system will allow the user to decrypt the required file on a single click |
| REQ-09: | The system will allow the user to create a new file to store decrypted data |

### 3.3.6 Key Shares Transfer and Validation

#### 3.3.6.1 Description and Priority

This feature of the Application transfers the Key Shares upon authentication and then uses them to reconstruct the key which is used to decrypt the data. This feature also validates the key by comparing it to the hash of the original key. When the user has entered the data/file to be decrypted then the system transfers the Key Shares and uses them to reconstruct the Original Key.

Priority Level: High

#### 3.3.6.2 Stimulus/Response Sequences

Input: The user will select the Decryption option and enter the data/file to be decrypted along with the file for decrypted data

Output: The system will transfer the Key Shares upon authentication and reconstruct the Original Key for Decryption

#### 3.3.6.3 Functional Requirements

REQ-10:Use Case Diagram The system will reconstruct the key when the required key shares are entered

REQ-11: The system will compare the hash of the new key with the hash of the original key

REQ-12: The system will create a new Key when incorrect Key Shares are entered and distribute themamong the users

### 3.3.7 Event Log and Storage

#### 3.3.7.1 Description and Priority

This feature of the Application keeps a record of the activity of the users and stores all the relevant information of the users and all other activities such as the hashes of Keys.

Priority Level: High

#### 3.3.7.2 Stimulus/Response Sequences

| | |
|---|---|
| Input: | User performs any activity on the Application |
| Output: | The system will store the record of the activities on the Database at the back-end |

#### 3.3.7.3 Functional Requirements

REQ-13: The system will record the time of activity

REQ-14: The system will record the action the user selected

REQ-15: The system will record the locations of the file to be encrypted/decrypted and the new files created

REQ-16: The system will store the hashes of the Keys

REQ-17: The system will store the number of total and required Key Shares

REQ-18: The system will store the authentication requirements for the transfer and distribution of Keys by means of USB

## 3.4 Other Nonfunctional Requirements

### 3.4.1 Performance Requirements

Certain functionalities will be required, based on the performance and response of Computational Secret Sharing (**CSS**).The system will be fast in terms of performance. The system will respond to the user request in less than 400 mili-seconds.

### 3.4.2 Safety Requirements

SF-01: The system will be fast and responsive to user actions. However, while working with large data the system may lead the application to become unresponsive

SF-02: System will create a new Key when incorrect Key Share(s) is entered and then transfer that Key's Shares to the users

SF-03: Users can lodge a complaint in case of error

### 3.4.3 Security Requirements

SE-01: The system will permit users to onlyview the data that is intended for them

SE-02: The connection with the server must be a secure https connection

SE-03: The communication must be encrypted

SE-04: The System will provide confidentiality and integrity

### 3.4.4 Software Quality Attributes

Quality attributes of Computational Secret Sharing (**CSS**) are described below. By following these attributes, the quality of Computational Secret Sharing (**CSS**) will be improved.

#### 3.4.4.1 Runtime System Qualities

At runtime, Computational Secret Sharing (**CSS**) has to provide its users with functionalities so that they can search for the desired services. Some of the runtime qualities that should be considered in the development of Computational Secret Sharing (**CSS**) are described here.

##### *3.4.4.1.1 Functionality*

CSS must provide functions to search the different services. CSS must provide the functions of authentication of a user.

##### *3.4.4.1.2 Availability*

The system will be available through a suggested and well known platform. The cloud is expected to be available 99.9% of the time depending on the network connection.

##### *3.4.4.1.3 Usability*

Usability is an important criterion in the development of CSS. The system should present all functionalities in such a way that nothing is missed by the user. The graphical user interface of the app is to be designed with usability as the first priority. The application will be presented and organized in a manner that is both visually appealing and easy for the user to navigate.

#### 3.4.4.2 Non-Runtime System Qualities

These are qualities of Computational Secret Sharing (CSS) which are required to make this software useful for further enhancements. It will also be helpful for future development as well as extending the system to different environments.

##### *3.4.4.2.1 Modifiability*

CSS must support modifiability so any further improvements or features are easy to incorporate.

##### *3.4.4.2.2 Portability*

CSS should be able to run in different computer environments. The CSS server should be a platform-independent and should support interoperability.

##### *3.4.4.2.3 Testability*

Different quality tests should be performed so that CSS is free from faults and perform according to requirements.

CHAPTER 4

# Design

# 4. Design

## 4.1 Introduction

This chaptercovers all the functional requirements and demonstrates how they interrelate with each other abstractly. The low-level design also illustrates as to how all of these requirements have been implemented. This low-level design does not address any non-functional requirements that the system has and that has been mentioned in the SRS Document.

## 4.2 Overview of the Modules

The system will be architected mainly in four fundamental modules “Application”, “Encryption”, “Decryption” and the “Database”. It will further be having sub-modules. A brief overview of these modules is given below.

### 4.2.1 Application Module

The Application Module is the User Interface with which the user interacts with the Application for data/file Encryption/Decryption.

### 4.2.2 Encryption Module

The Encryption Module receives data/file to encrypt from the Application Module, generates a key, applies AES encryption to the data/file using this key, splits the key into

N shares and then hashes the original key. Finally, it sends the encrypted data, key shares and the hashed key back to the Application Module.

### 4.2.3 Decryption Module

The Decryption Module receives data/file to decrypt, at least K key shares and the hashed key from the Application Module. It then reconstructs the key using the key shares, verifies the key by hashing it and then matching it with the original hashed key, applies AES decryption to the encrypted data and then sends the decrypted data back to the Application Module.

### 4.2.4 Database Module

The key hash produced in the Encryption Module is received by the Database from the Application Module and stored in it. This key may also be retrieved when needed for key verification.

## 4.3 Structure andRelationships

This section covers the overall technical description of Computational Secret Sharing (**CSS**). It shows the working of application in perspective of different viewpoints and shows relationships between different components.

### 4.3.1 System Block Diagram

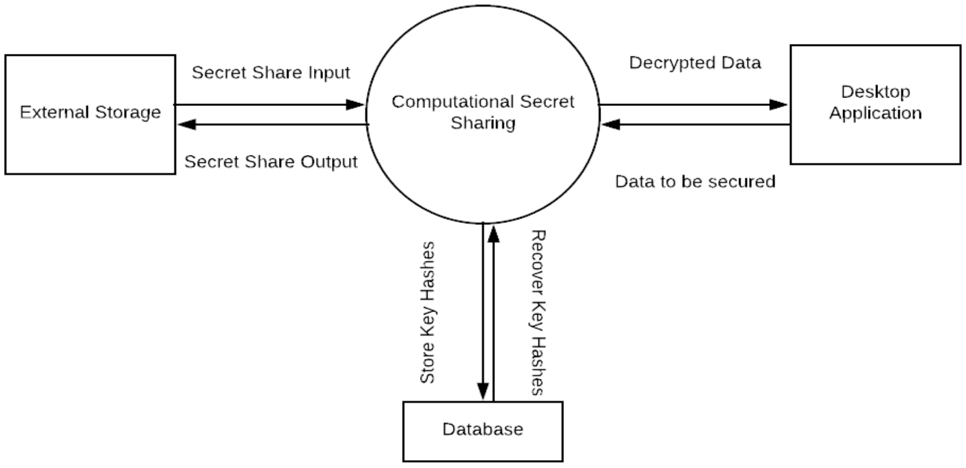


*Figure 4-1–System Block Diagram*

The Desktop Application Module sends the data to be encrypted/decrypted to Computational Secret Sharing (**CSS**). Computational Secret Sharing (**CSS**) then applies either AES Encryption or AES Decryption to the data received based on the request from user. The hashed key that is generated in the encryption process is then stored in the system database. This hashed key may also be retrieved for key verification in the decryption process. The key shares which are used for reconstructing the key are communicated via an external storage, both in and out of the system.

### 4.3.2 Use Cases

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.

### 4.3.2.1 Use Case Diagram

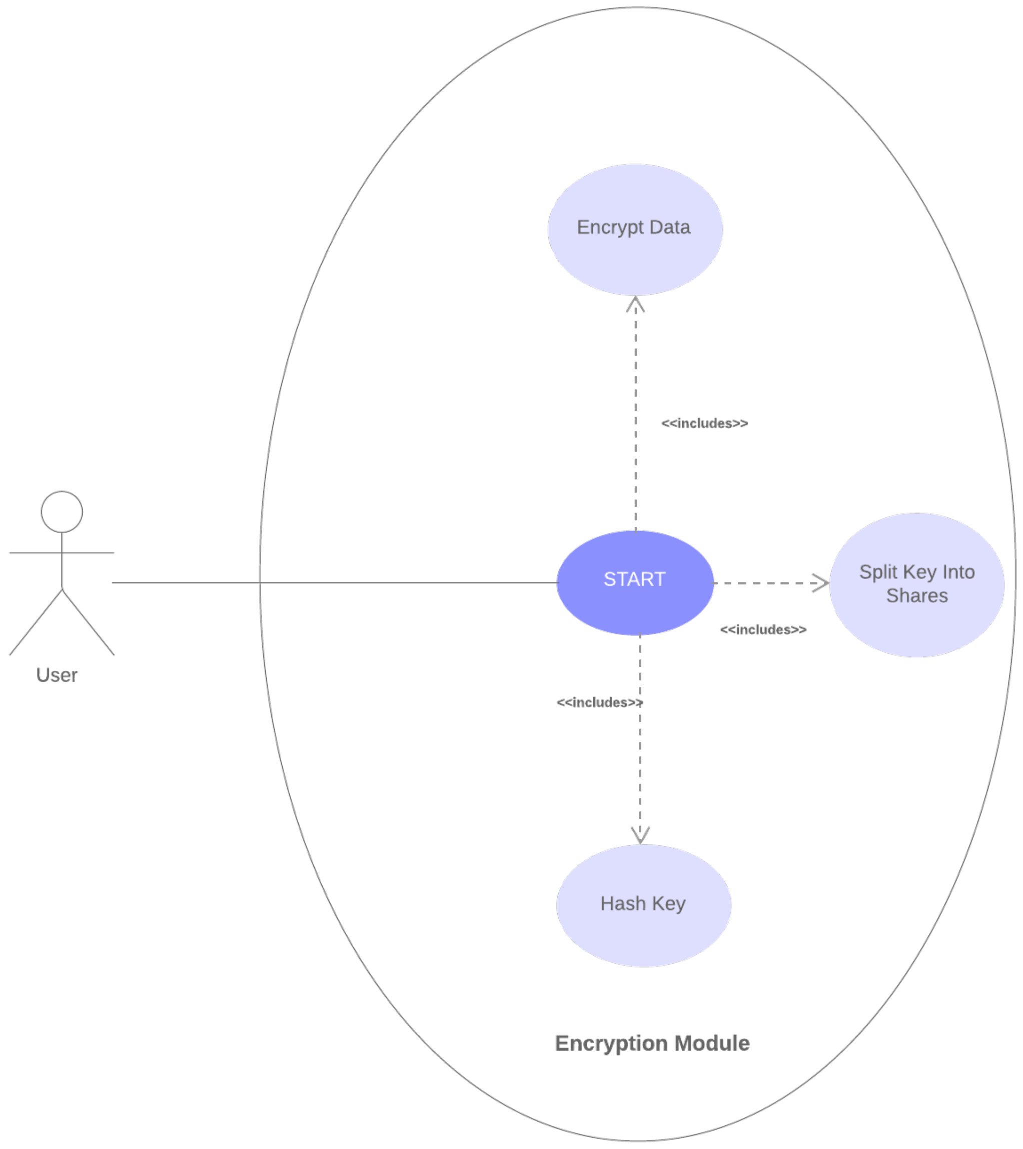


*Figure 4-2 - Use Case Diagram-Encryption Module*

### 4.3.2.2 Use Cases Description

| | |
|---|---|
| Use Case ID: | 01 |
| Use Case Name: | Encrypt Data |
| Actors: | Users |

| | | | |
|---|---|---|---|
| Created by: | Haider | Last Updated by: | Haider |
| Date Created: | 08/01/2019 | Date Last Updated: | 08/01/2019 |
| Description: | User opens the application to encrypt the data/file | | |
| Preconditions: | The user has to open the application and select the encryption option | | |
| Post-conditions: | Encrypts the file upon providing with the name and location of the file to encrypted and creating a file for cipher text | | |
| Normal Flow (Primary Scenario): | 1. The user will open the application and select the encryption option<br>2. The user will provide the name and location of the file to be encrypted and also create a file for cipher text<br>3. The file is encrypted successfully using the key generated | | |
| Alternative Flows: | 1. The user fails to enter the correct name or location of the file to be encrypted<br>2. An error occurs while encryption or during the generation of the key used for encryption | | |

*Table 4-1–Encrypt Data Use Case*

| | | | |
|---|---|---|---|
| Use Case ID: | 02 | | |
| Use Case Name: | Split Key into Shares | | |
| Actors: | Users | | |
| Created by: | Haider | Last Updated by: | Haider |
| Date Created: | 08/01/2019 | Date Last Updated: | 08/01/2019 |

|  |  |
|---|---|
| Description: | The system divides the key into N shares and distributes them among the users |
| Preconditions: | The system has generated a key for encryption |
| Post-conditions: | The key is divided into N shares and distributed among users |
| Normal Flow (Primary Scenario): | 1. The system generates the key which is to be used for encryption<br>2. The system splits the key into N shares<br>3. The system transfers the key shares to multiple users |
| Alternative Flows: | 1. There is an error in splitting the key<br>2. There is an error in distribution of the shares |

*Table 4-2–Split Key into Shares Use Case*

|  |  |  |  |
|---|---|---|---|
| Use Case ID: | 03 |  |  |
| Use Case Name: | Hash Key |  |  |
| Actors: | Users |  |  |
| Created by: | Haider | Last Updated by: | Haider |
| Date Created: | 08/01/2019 | Date Last Updated: | 08/01/2019 |
| Description: | The system generates the key used for encryption, hashes it and stores the hash for future authentication |  |  |
| Preconditions: | The system has generated the key for encryption |  |  |
| Post-conditions: | The hash of the key is stored in the database |  |  |
| Normal Flow (Primary Scenario): | 1. The system generates the key used for encryption<br>2. The system produces the hash of the key |  |  |

| | |
|---|---|
| | 3. The system stores the hash of the key for future authentication |
| Alternative Flows: | 1. An error occurs while creating the hash of the key<br>2. The hash is not stored in the database and is lost |

*Table 4-3–Hash Key Use Case*

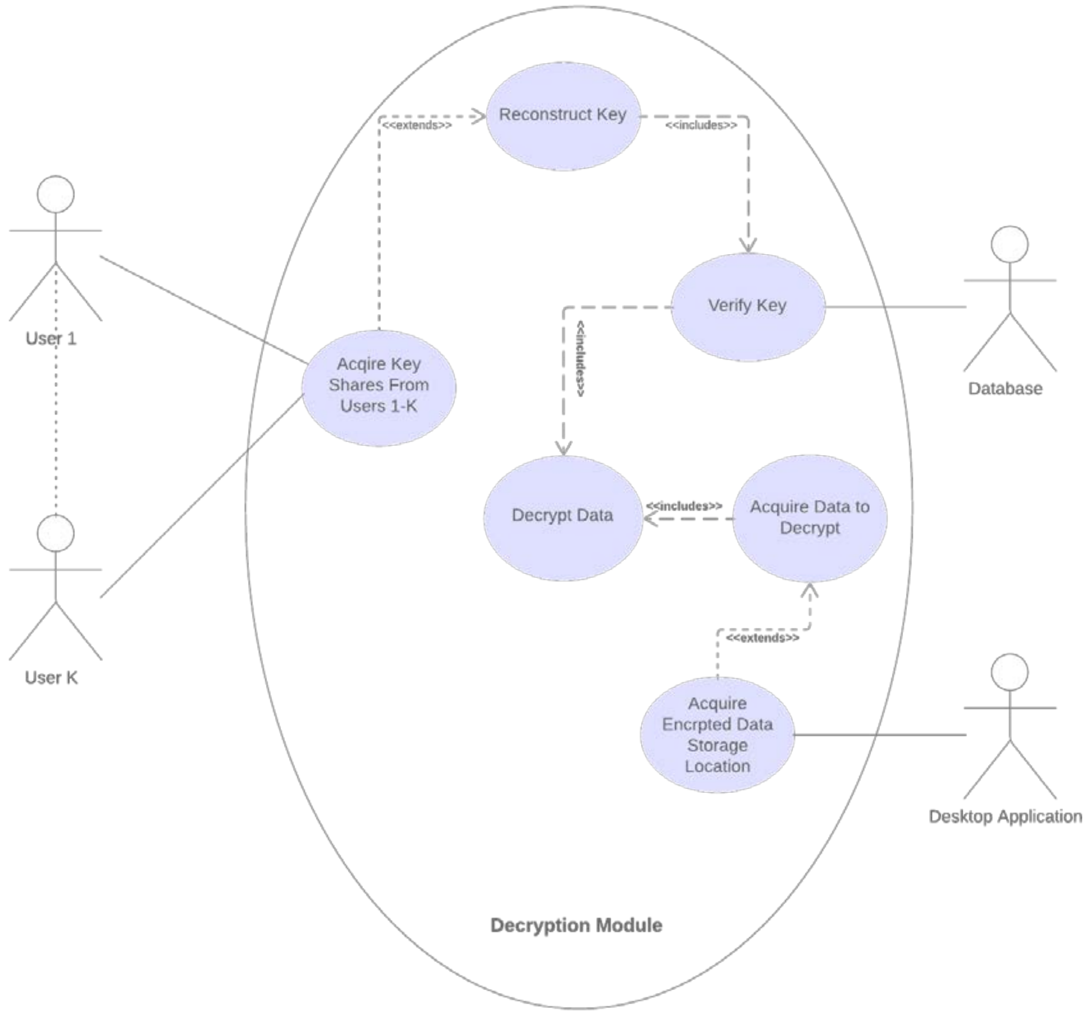


*Figure 4-1 - Use Case Diagram-Decryption Module*

| | |
|---|---|
| Use Case ID: | 04 |

| | | | |
|---|---|---|---|
| Use Case Name: | Acquire Encrypted Data/Data to Decrypt | | |
| Actors: | Users | | |
| Created by: | Haider | Last Updated by: | Haider |
| Date Created: | 08/01/2019 | Date Last Updated: | 08/01/2019 |
| Description: | The user acquires the data to decrypt and enters it into the system | | |
| Preconditions: | The user has encrypted data and has access to the application | | |
| Post-conditions: | Data to be decrypted is entered into the application | | |
| Normal Flow (Primary Scenario): | 1. The user has access to the application and has data to be decrypted<br>2. The user enters the data to be decrypted into the application | | |
| Alternative Flows: | 1. The user does not have access to the application<br>2. The user does not have data that needs to be decrypted | | |

*Table 4-4–Acquire Encrypted Data/Data to Decrypt Use Case*

| | | | |
|---|---|---|---|
| Use Case ID: | 05 | | |
| Use Case Name: | Decrypt Data | | |
| Actors: | Users | | |
| Created by: | Haider | Last Updated by: | Haider |
| Date Created: | O8/01/2019 | Date Last Updated: | 08/01/2019 |
| Description: | User opens the application to decrypt the data/file | | |
| Preconditions: | The user has to open the application and select the decryption option | | |

| | |
|---|---|
| Post-conditions: | Decrypts the file upon providing with the name and location of the file to decrypted and creating a file for decrypted data |
| Normal Flow (Primary Scenario): | 1. The user will open the application and select the decryption option<br>2. The user will provide the name and location of the file to be decrypted and also create a file for decrypted data<br>3. The file is decrypted successfully using the key generated by combining the key shares |
| Alternative Flows: | 1. The user fails to enter the correct name or location of the file to be decrypted<br>2. An error occurs while decryption or during the generation of the key used from key shares |

*Table 4-5–Decrypt Data Use Case*

| | | | |
|---|---|---|---|
| Use Case ID: | 06 | | |
| Use Case Name: | Acquire Key Shares | | |
| Actors: | Users | | |
| Created by: | Haider | Last Updated by: | Haider |
| Date Created: | 08/01/2019 | Date Last Updated: | 08/01/2019 |
| Description: | The system prompts the user to enter the required key shares for reconstructing the original key that will be used for decryption. The key shares are validated and then used for key generation | | |

| | |
|---|---|
| Preconditions: | The user has to select the decryption option and provide the file to be decrypted |
| Post-conditions: | The key is generated from the key shares and used for decryption |
| Normal Flow (Primary Scenario): | 1. The user will select the decryption option and provide the file to be decrypted<br>2. The system will prompt the users to enter the required number of key shares<br>3. The system will validate the key shares and use them to reconstruct the key used for decryption |
| Alternative Flows: | 1. The user fails to enter the required number of key shares<br>2. The user fails to enter the correct key shares |

*Table 4-6–Acquire Key Shares Use Case*

| | | | |
|---|---|---|---|
| Use Case ID: | 07 | | |
| Use Case Name: | Reconstruct Key | | |
| Actors: | Users | | |
| Created by: | Haider | Last Updated by: | Haider |
| Date Created: | 08/01/2019 | Date Last Updated: | 08/01/2019 |
| Description: | The system will use the entered key shares to reconstruct the original key which will be used for decryption | | |
| Preconditions: | The user has to enter the required number of key shares | | |
| Post-conditions: | The original key is reconstructed using the key shares and is used for decryption | | |

| | |
|---|---|
| Normal Flow (Primary Scenario): | 1. The user enters the required key shares<br>2. The system take those key shares and uses them to reconstruct the original key<br>3. The reconstructed key is then used for decryption |
| Alternative Flows: | There is an error in the reconstruction of the key |

*Table 4-7–Reconstruct Key Use Case*

| | | | |
|---|---|---|---|
| Use Case ID: | 08 | | |
| Use Case Name: | Verify Key | | |
| Actors: | Users | | |
| Created by: | Haider | Last Updated by: | Haider |
| Date Created: | 08/01/2019 | Date Last Updated: | 08/01/2019 |
| Description: | The system uses the hash of the original key and compares it with the hash of the reconstructed key and verifies if it is the correct key | | |
| Preconditions: | 1. There exists a record of hash of the original key<br>2. The key is reconstructed and its hash is produced to be compared | | |
| Post-conditions: | The reconstructed key is verified to be the same key and is then used for decryption | | |
| Normal Flow (Primary Scenario): | 1. The system reconstructs the key<br>2. The system produces hash of the reconstructed key<br>3. The system compares the hash of original and | | |

| | |
|---|---|
| | reconstructed keys<br>4. The reconstructed key is verified and then used for decryption |
| Alternative Flows: | 1. The hash of the original does not match the hash of the reconstructed key<br>2. The system generates a new key and distributes its shares among the users |

*Table 4-8–Verify Key Use Case*

### 4.3.3 Class Diagram with Description

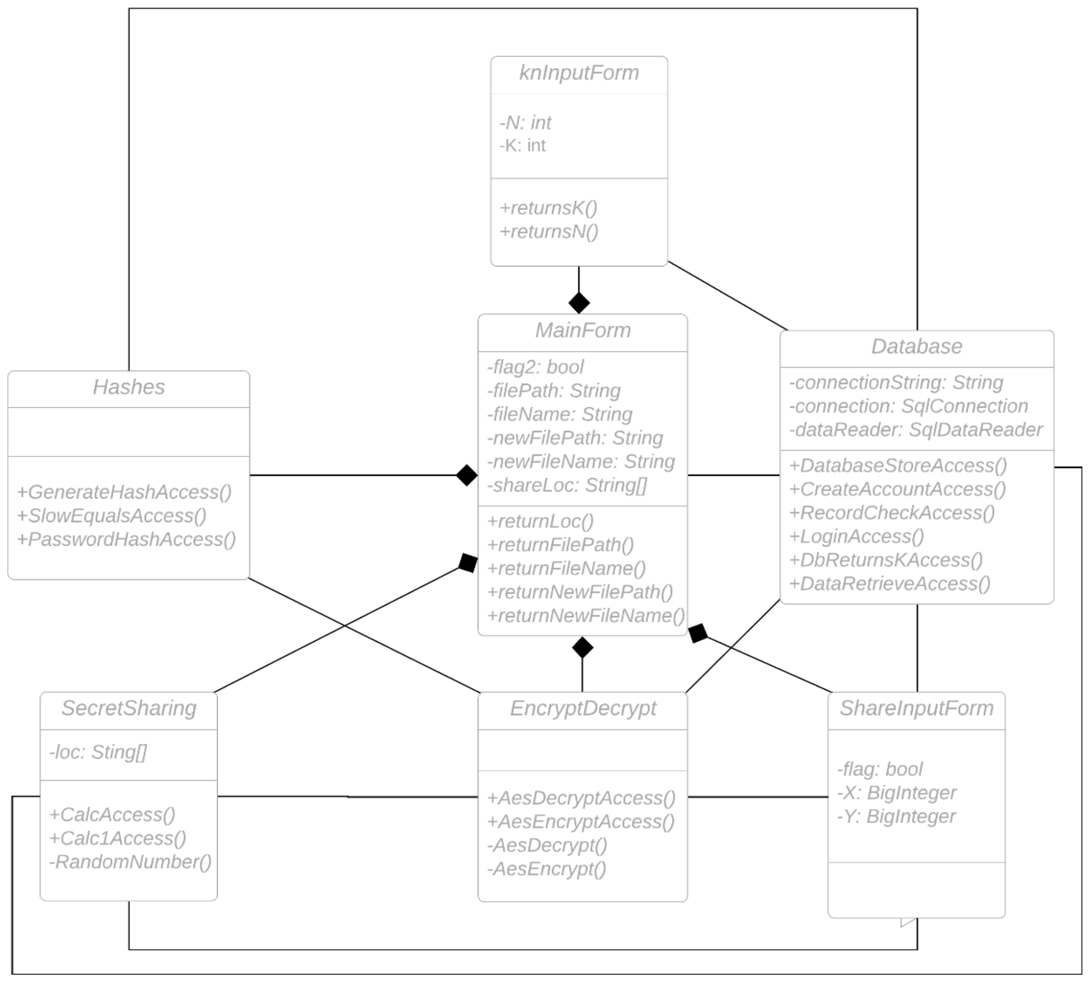


*Figure 4-2 - Class Diagram*

| Class Name | Description |
|---|---|
| Main | Main Class contains the origin for the function CSS to perform. It is the main class acting as a gateway to all the other classes. |
| User | User Class contains all the information related to user management. It |

| | |
|---|---|
| | has aggregation with other classes of user categorization and the functions that perform all the user management functions. |
| Encrypt | Encrypt Class contains all the information related to encryption and whatever is required for it. It also has aggregation with other classes. |
| Decrypt | Decrypt Class contains all the information related to decryption and whatever is required for it. It also has aggregation with other classes. |
| Secret Sharing | Secret Sharing Class contains all the information related to the algorithm of Secret Sharing employed by the application. It also has aggregation with other classes. |
| Hashes | Hashes Class contains all the information about how hashes of keys are generated and compared. It also has aggregation with other classes. |
| Database | Database Class contains all the information about the database present at the back-end of the application that keeps a record of all the activities. It also has aggregation with all other classes. |
| Input | Input Class contains all the information about every input related to the application. It also has aggregation with other classes. |

*Table 4-9- Class Diagram Description*

## 4.4 User Interface Issues

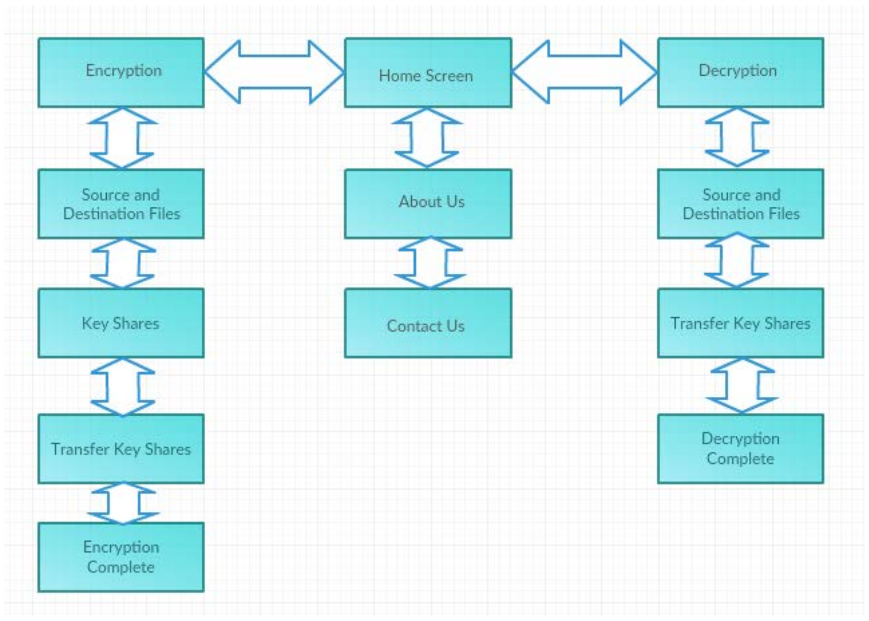


*Figure 4-3 - Interface Diagram*

### 4.4.1 Description of the Diagram

#### 4.4.1.1 Home Screen

This is the initial home screen, which gives the useroptions to choose either Encryption or Decryption, or just use the navigation menu.

#### 4.4.1.2 Encryption

If the user selects the Encryption option, he/she is taken to the next screen so that Encryption can be performed.

#### 4.4.1.3 Source and Destination Files

This feature prompts the user to enter the name and destination of the source (file to be encrypted) and destination (file to store the cipher text) files.

#### 4.4.1.4 Key Shares

This feature prompts the user to enter the total number and required number of Key Shares.

#### 4.4.1.5 Transfer Key Shares

This feature transfers the Key Shares to the users upon authentication.

#### 4.4.1.6 Encryption Complete

This screen signals that the Encryption has been successfully completed.

#### 4.4.1.7 Decryption

If the user selects the Decryption option, he/she is taken to the next screen so that Decryption can be performed.

#### 4.4.1.8 Source and Destination Files

This feature prompts the user to enter the name and destination of the source (file to be decrypted) and destination (file to store the decrypted data) files.

#### 4.4.1.9 Transfer Key Shares

This feature transfers the Key Shares from the users to the system upon authentication and uses them for reconstructing the key for decryption.

#### 4.4.1.10 Decryption Complete

This screen signals that the Decryption has been successfully completed.

#### 4.4.1.11 About Us

This feature provides information about the Application that will help the user in better understanding how it works and all the effort put into making it.

#### 4.4.1.12 Contact Us

This feature provides information about how to contact the developers for queries.

### 4.4.2 Activity Diagrams

#### 4.4.2.1 Encryption Module

The diagram below displays how the system performs Encryption.

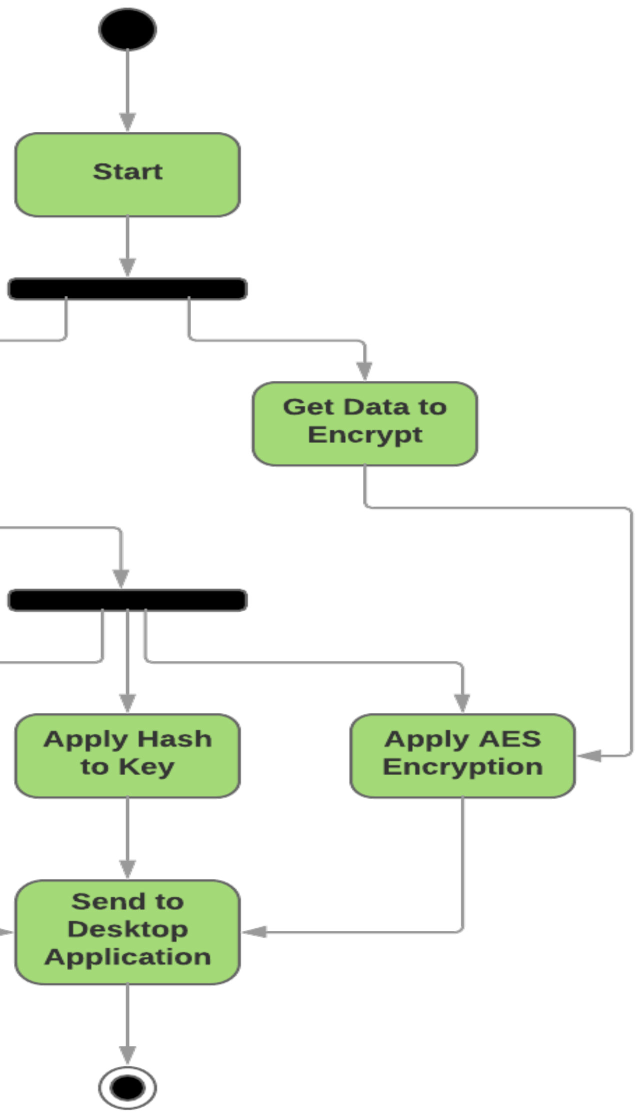


*Figure 4-4–Encryption Module – Activity Diagram*

### 4.4.2.2 Decryption Module

The diagram below displays how the system performs Decryption.

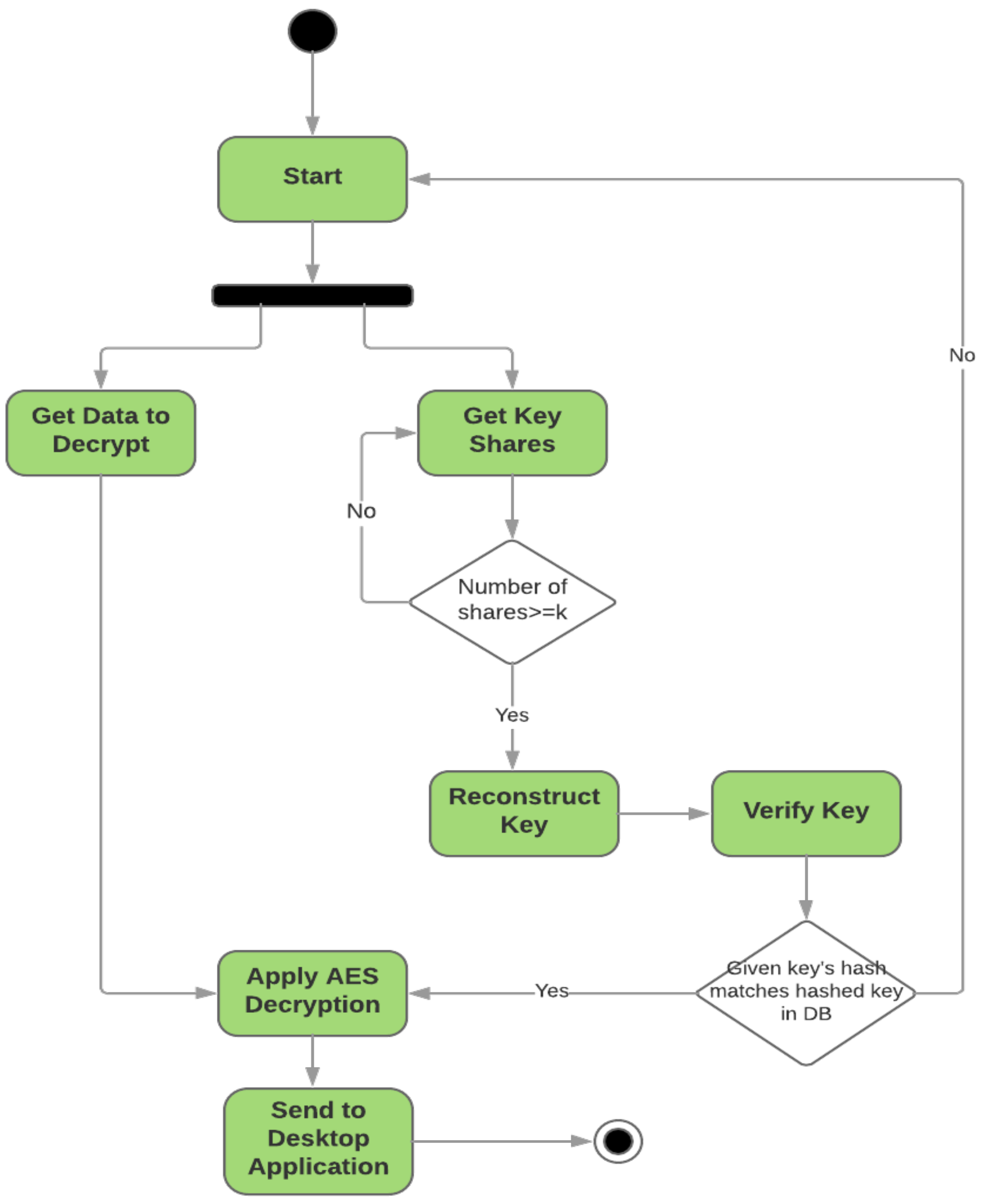


*Figure 4-5 – Decryption Module – Activity Diagram*

#### 4.4.2.3 Desktop Application

The diagram below shows how the Desktop Application functions.

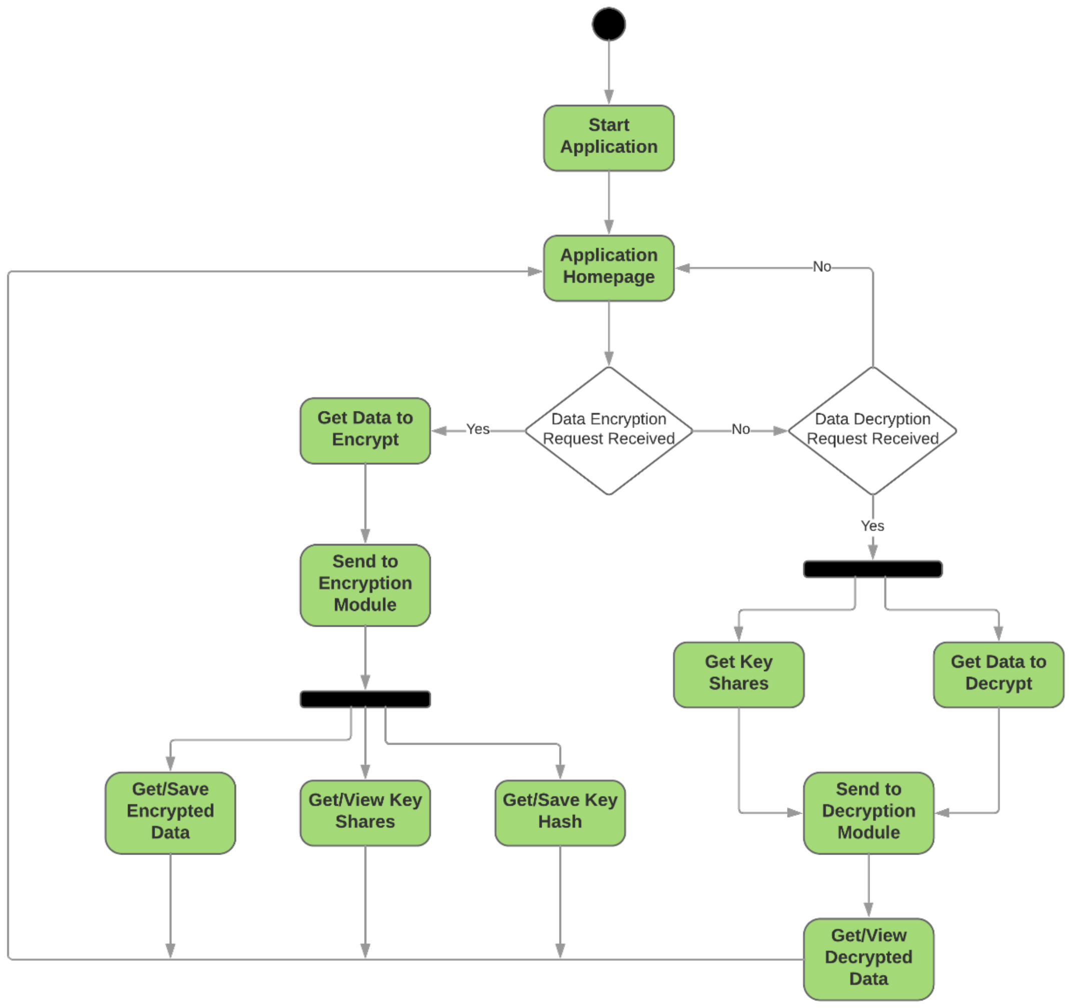


*Figure 4-6 – Desktop Application – Activity Diagram*

### 4.4.3 Sequence Diagrams

Following sequence diagrams show the sequence of activities performed in all use cases described above.

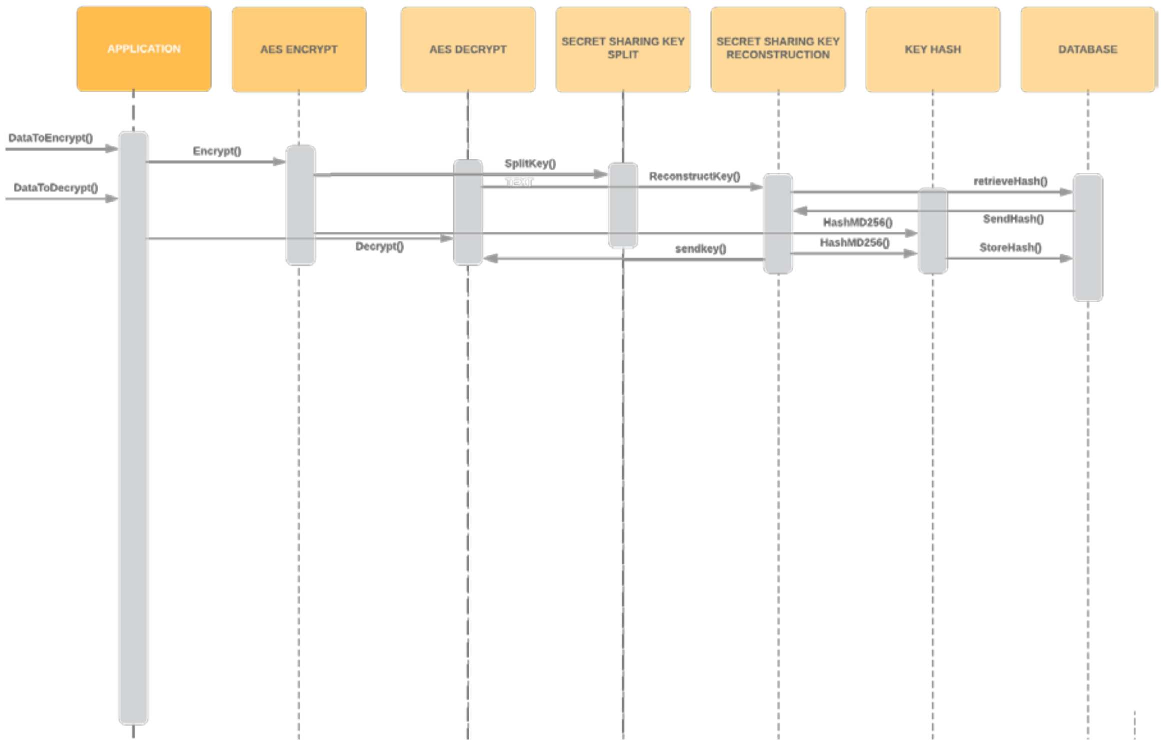
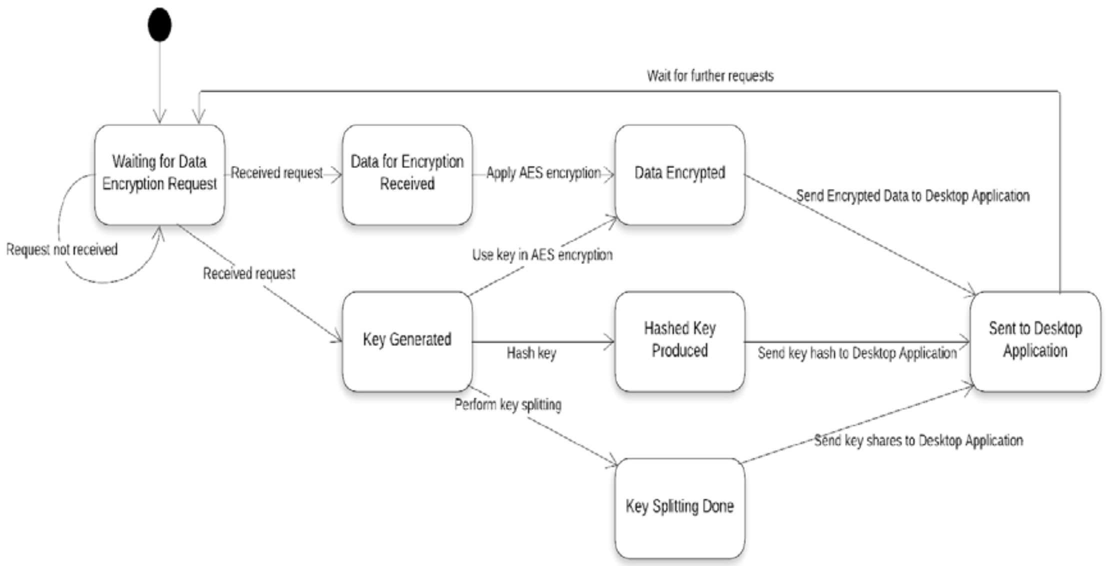


*Figure 4-9 – Sequence Diagram*

### 4.4.4 State Diagrams

Following are the state diagrams of Computational Secret Sharing (**CSS**) showing all the states that the system may have during its course of action.

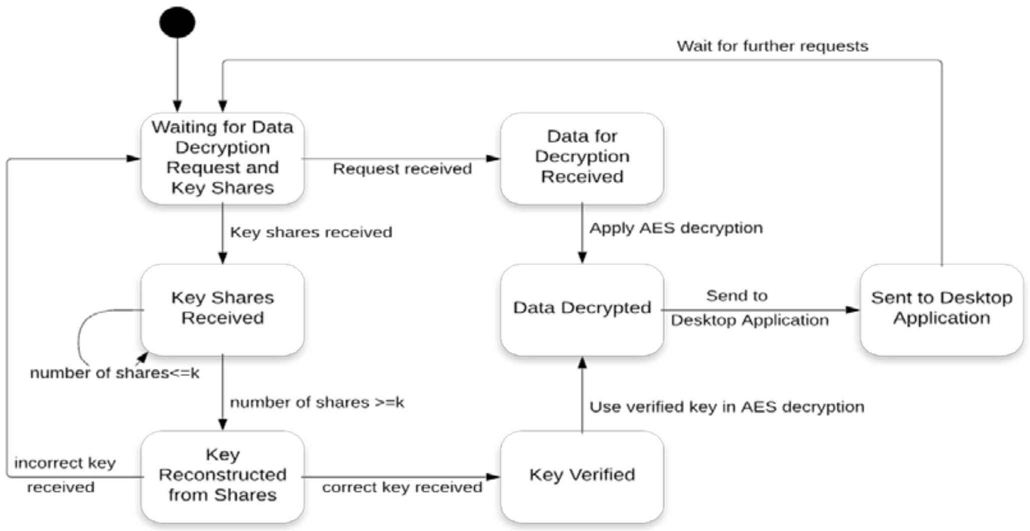
#### 4.4.4.1 Encryption Module



Wait for further requests
Waiting for Data Encryption Request
Received request
Data for Encryption Received
Apply AES encryption
Data Encrypted
Send Encrypted Data to Desktop Application
Request not received
Received request
Use key in AES encryption
Key Generated
Hash key
Hashed Key Produced
Send key hash to Desktop Application
Sent to Desktop Application
Perform key splitting
Send key shares to Desktop Application
Key Splitting Done

*Figure 4-10 –Encryption Module – State Diagram*

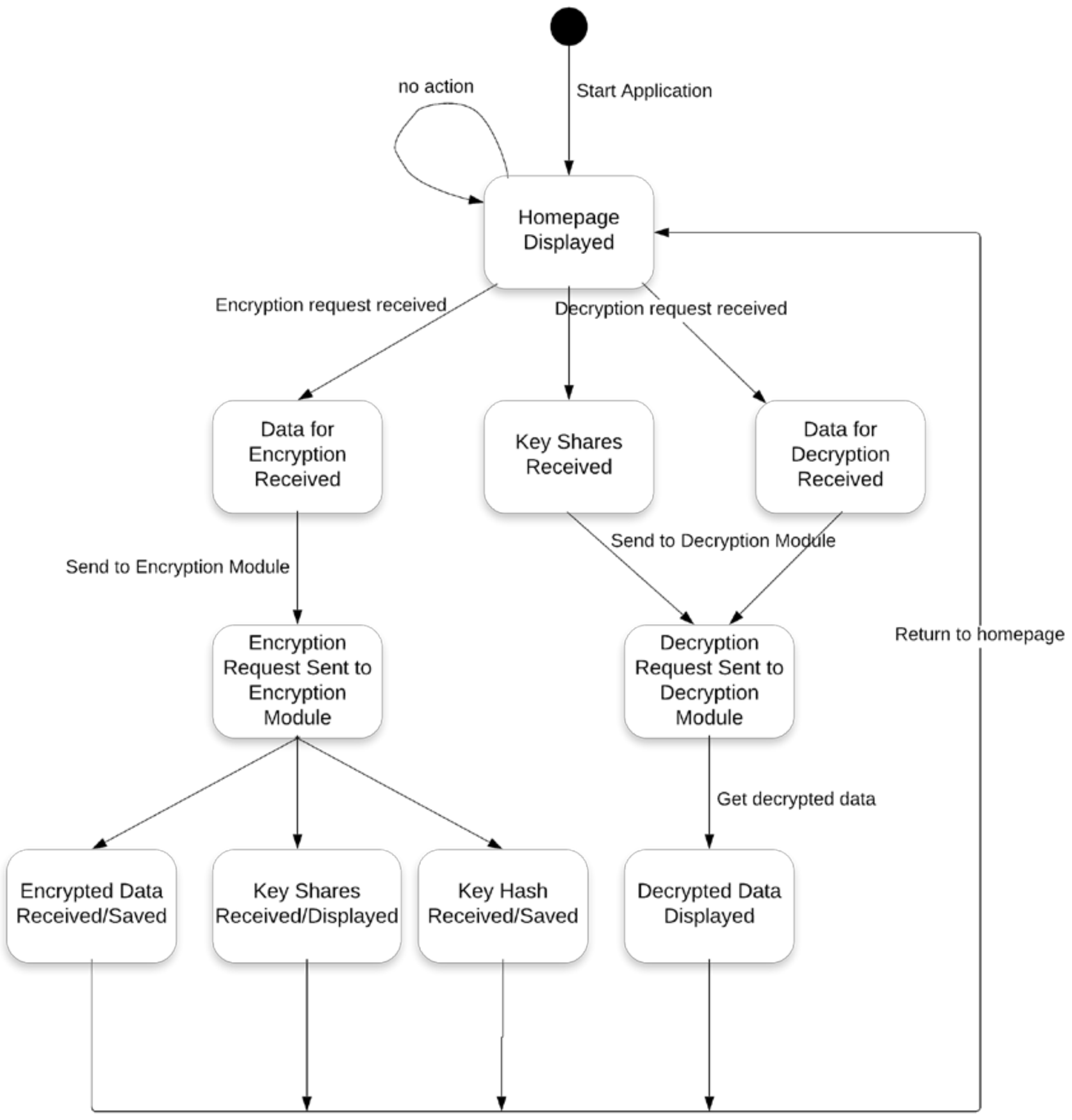
#### 4.4.4.2 Decryption Module



Wait for further requests
Waiting for Data Decryption Request and Key Shares
Request received
Data for Decryption Received
Key shares received
Apply AES decryption
Key Shares Received
Data Decrypted
Send to Desktop Application
Sent to Desktop Application
number of shares<=k
number of shares >=k
Use verified key in AES decryption
incorrect key received
Key Reconstructed from Shares
correct key received
Key Verified

*Figure 4-11 –Decryption Module – State Diagram*

#### 4.4.4.3 Desktop Application



Start Application
no action
Homepage Displayed
Encryption request received
Decryption request received
Data for Encryption Received
Key Shares Received
Data for Decryption Received
Send to Encryption Module
Send to Decryption Module
Encryption Request Sent to Encryption Module
Decryption Request Sent to Decryption Module
Return to homepage
Get decrypted data
Encrypted Data Received/Saved
Key Shares Received/Displayed
Key Hash Received/Saved
Decrypted Data Displayed

*Figure 4-12 –Desktop Application – State Diagram*

## 4.5 Detailed Description of Components

### 4.5.1 User Management

| | |
|---|---|
| Identification | **User Management** |

| | |
|---|---|
| Type | Component |
| Purpose | • To manage a set of users. It will manage the user credential database and validation of user credentials in case of login activity<br>• All of this functionality is hidden from the user - the nitty-gritty details of managing the users are the job of the System class |
| Function | • VerifyDetails()<br>• GrantsAccess() |
| Subordinates | User database will be used to hold the user records |
| Dependencies | All user management functions depends on it |
| Interfaces | Represents User management options to the user |
| Resources | User Database |
| Processing | • VerifyDetails() verify user credentials from user database<br>• GrantsAccess() grants user access in the user database |
| Data | • Graphics for interface support<br>• A database repository for maintaining the user record |

*Table 4-10- User Management Component*

### 4.5.2 User Access Rights

| | |
|---|---|
| Identification | **User Access Rights** |
| Type | Component |
| Purpose | • To classify the users into categories on the basis of privileges |

| | |
|---|---|
| | assigned to each user category<br>• The privileges contain user rights of interacting the app. |
| Function | • Categorize(User)<br>• GrantAccess() |
| Subordinates | User database will be used to hold the user records |
| Dependencies | All user management functions depend on it |
| Interfaces | • Sends user to the profile specified for its category |
| Resources | User Database |
| Processing | • Categorize (User) is used to categorize the user into one of the defined categories<br>• GrantAccess () grants access according to that category |
| Data | Database repository for maintaining the user record, based on their individual access rights |

*Table 4-11- User Access Rights Component*

### 4.5.3 Button

| | |
|---|---|
| Identification | **Button** |
| Type | Component |
| Purpose | • Class to allow a user to choose various Menu options<br>• Graphic user interface representation |
| Function | • Represents various options available to the user<br>• ButtonCallback (button) – a Callback function, used when the button is clicked in order to call the requisite function related |

| | |
|---|---|
| | to that button |
| Subordinates | A graphical picture will be used for the button's graphic |
| Dependencies | Depends on the menu class |
| Interfaces | • button->text() [text overlay on the button]<br>• button->bitmap() [graphic for the button]<br>• button->screenX(), button->screenY() [screen coordinates for the button] |
| Resources | Nil |
| Processing | • ButtonCallback(button()<br>• Callback functions for the button |
| Data | A graphical picture to hold the visual graphic of the button |

*Table 4-1 - Button Component*

### 4.5.4 Menu Component

| | |
|---|---|
| Identification | **Menu Component** |
| Type | Component |
| Purpose | • User interface to access different features of the Application<br>• Separate menu class instances for each menu screen |
| Function | • Display a window of buttons representing different menu choices<br>• For each menu choice, the menu class will call the function display (button) to display button onscreen |
| Subordinates | An array will hold different buttons, one for each option of the menu |

| | |
|---|---|
| Dependencies | Calls instances of the button class |
| Interfaces | Represented in UIs as menus |
| Resources | Nil |
| Processing | display(button) displays the button onscreen |
| Data | Array of button classes |

*Table 4-13 - Menu Component*

## 4.6 Summary

The purpose of this chapterwas to deliver a portrayal of the design of the system suitable enough to allow for software development, with an understanding of what is built and how it is developed. It provides information essential to getting a description of the details for the software and the system to be built. It presents a design view and detailed description of Computational Secret Sharing (**CSS**). It explains the purpose, features, interfaces, what the system do, its entire processes in detail, the constraints under which it must operate and how the system reacts to inputs and what will be its outputs.

CHAPTER 5

# Implementation

# 5. Implementation

## 5.1 Introduction

The preceding chapter discoursed comprehensive design of the Computational Secret Sharing (**CSS**). This design is converted into an application by utilizing numerous technologies and tools. The implementation details are conferred in the subsequent divisions providing minutiae of the system's inner functioning.

## 5.2 Tools & Technologies

### 5.2.1 C#

C# (C Sharp) is a general-purpose, multi-paradigm programing language.

### 5.2.2 .NET

Microsoft .NET is a Software Framework that runs primarily on Microsoft Windows. It includes a large class library named as Framework Class Library and provides language interoperability across several programming languages.

### 5.2.3 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment from Microsoft. It is used to develop computer programs, as well as websites, web applications, web services and mobile applications.

### 5.2.4 SQL

SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.

### 5.2.5 Microsoft SQL

Microsoft SQL is a relational database management system developed by Microsoft. As a database server it is a software product with the primary function of storing and retrieving data as requested by other software applications.

### 5.2.6 XAML

Extensible Application Markup Language is a declarative XML-based language developed by Microsoft that is used for initializing structured values and objects.

## 5.3 UI Design

The system supports an intuitive and easy to use User Interface (UI) that has an extremely shallow learning curve and require minimum training to be operated at maximum efficiency.

### 5.3.1 Home Screen

Following are the sketches of UI implementation for CSS. This will be the first screen that the user sees upon opening the CSS. The interface is simple and self-explanatory.

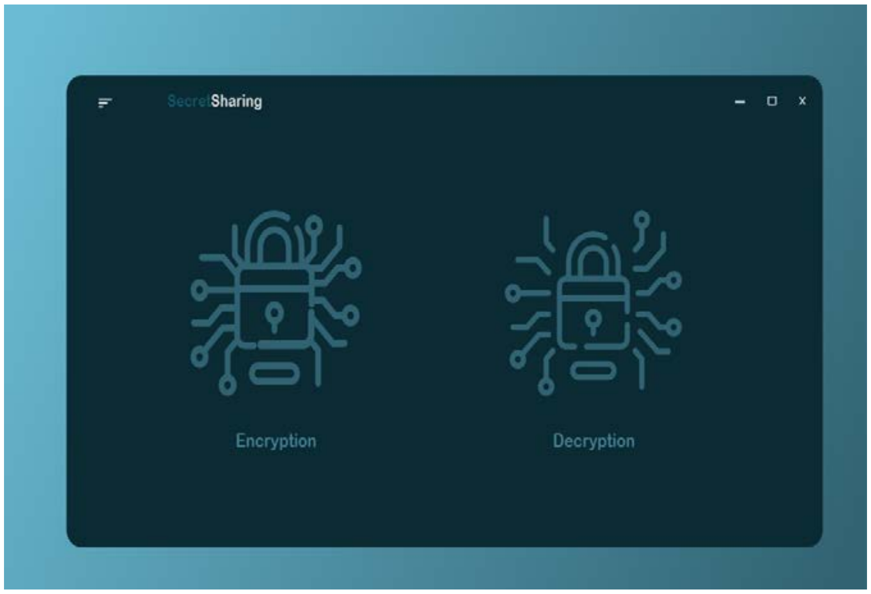
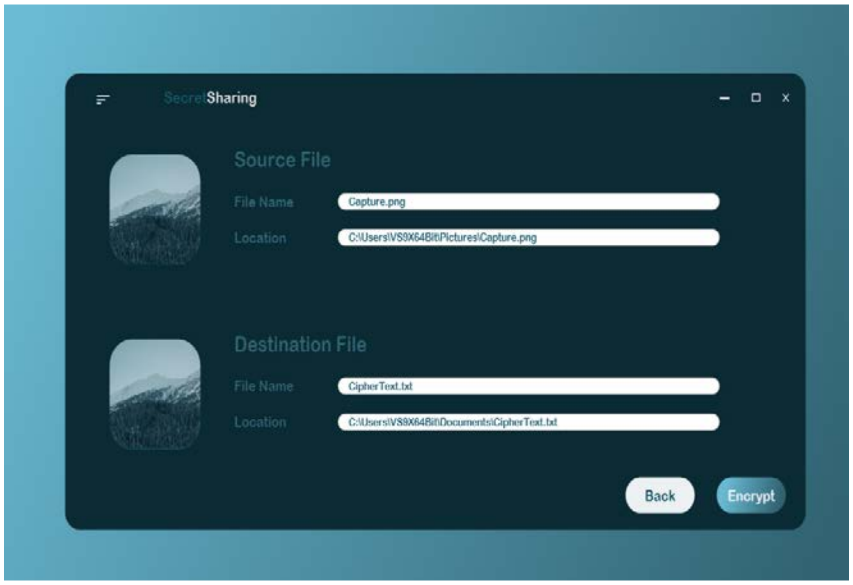


*Figure 5-1 - Home Screen UI*

### 5.3.2 Encryption

If a user selects the Encryption option, they are led to the following screen to enter the source and destination files.

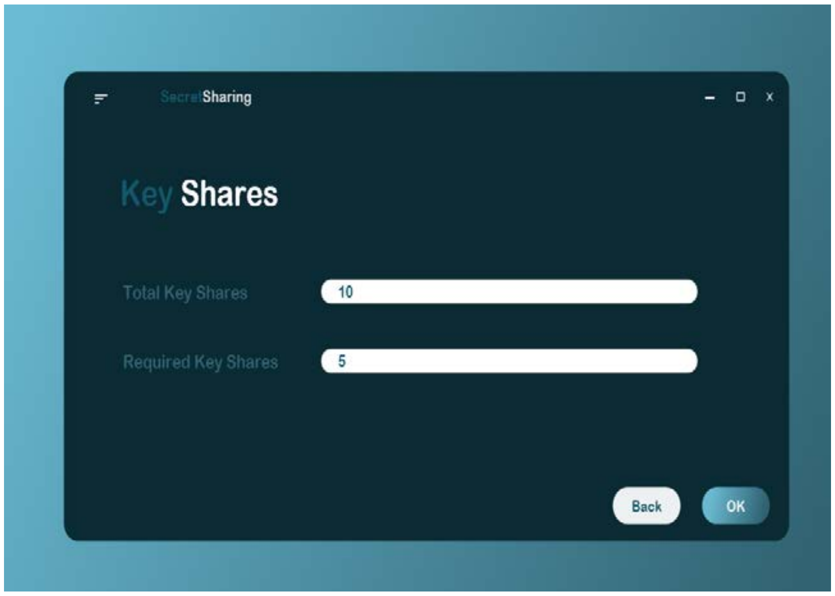


File Name: Capture.png
Location: C:\Users\VS9X64Bit\Pictures\Capture.png

File Name: CipherText.txt
Location: C:\Users\VS9X64Bit\Documents\CipherText.txt

*Figure 5-2 - Encryption UI*

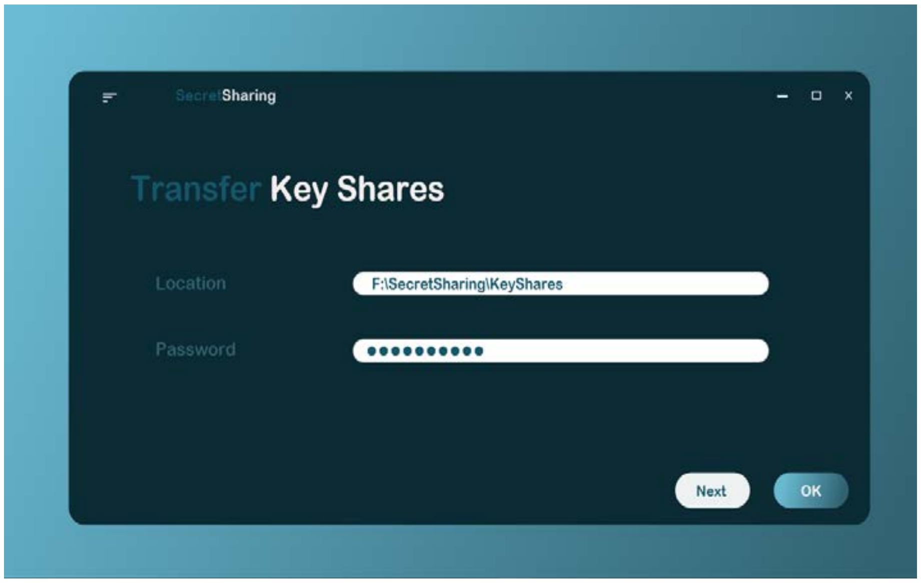
The system then prompts the user to enter the number of Key Shares.



Total Key Shares: 10
Required Key Shares: 5

*Figure 5-3 – Key Shares UI*

The system then transfers the Key Shares upon authentication.



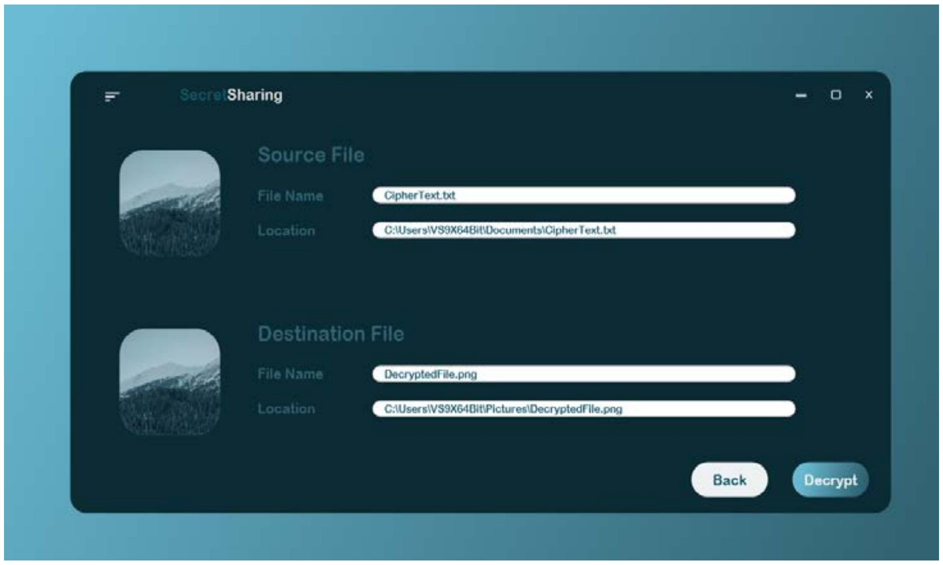
Location: F:\SecretSharing\KeyShares

Password: ●●●●●●●●●●●

*Figure 5-4 – Key TransferUI*

### 5.3.3 Decryption

If a user selects the Decryption option, they are led to the following screen to enter the source and destination files.



Source File

File Name: CipherText.txt

Location: C:\Users\VS9X64Bit\Documents\CipherText.txt

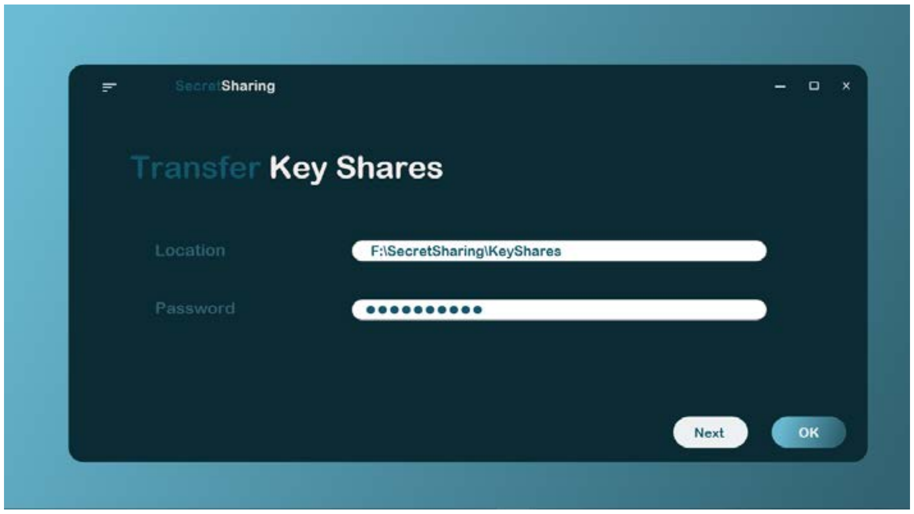
Destination File

File Name: DecryptedFile.png

Location: C:\Users\VS9X64Bit\Pictures\DecryptedFile.png

*Figure 5-5–Decryption UI*

The system then prompts the user to enter the required number of keys with authentication.



Location F:\SecretSharing\KeyShares

Password ●●●●●●●●●●●

*Figure 5-6 – Key Transfer UI*

## 5.4 Summary

Implementation details of Computational Secret Sharing (**CSS**) are discussed in this chapter. Different functionalities and strategies to develop the system have also been pondered upon.A brief introduction to different tools and technologies employed is also given.

CHAPTER 6

# Testing

# 6. Testing

## 6.1 Introduction

The purpose of this document is to elicit all material that is essential to plan and control the test efforts for the development of this project. It specifies the test plan for Computational Secret Sharing (**CSS**) application during the development phase and provides rationale behind necessity of these tests. This document provides an overview of the tests that were implemented, the items that were targeted by the tests, along with the testing approach that was deployed. This testing is being done according to the elicited requirements in Software Requirements Specification Document (SRS) for Computational Secret Sharing (**CSS**).

## 6.2 Test Items

- Encryption
- Decryption
- Key Division
- Key Distribution
- Key Re-construction
- Storing Hashes
- Comparing Hashes

## 6.3 Tested Features

- Encryption
- Decryption
- Storing Original File
- Storing Encrypted File
- Storing Decrypted File
- Storing Original Key
- Key Division
- Managing Key Shares
- Key Distribution
- Key Reading
- Key Re-construction
- View Encrypted File
- View Decrypted File

## 6.4 Features not to be tested

Following tests are not done on the system since they are out of the scope of this project.

- Security Testing
- Accuracy Testing

## 6.5 Approach

The system is working in modules so the testing phase will be initiated by testing each module separately i.e. unit testing, and then step by step integrating modules to test them with each other i.e. integration testing, followed by the testing of complete application as a whole.

## 6.6 Item Pass/Fail Criteria

- Items will pass the test if the actual output of each of the test case is same as the desired output of the system
- Any transfer of data between any modules is updated in the database

## 6.7 Suspension Criteria and Resumption Requirements

### 6.7.1 Suspension Criteria

- The build contains many serious defects which seriously limit testing progress
- Software/hardware problem
- Assigned resources are not available when needed to be tested

### 6.7.2 Resumption Requirements

- Resumption will only occur when the problems that caused the suspension have been resolved

## 6.8 Test Deliverables

- Test Plan Document

## 6.9 Testing Tasks

- Development of test cases
- Executionof tests based on the developed test cases
- Report defects from the executed test cases, if any
- Provision of complete test report
- Incorporate changes later in the stage of the project development

## 6.10 Environmental Needs

### 6.10.1 Hardware Requirements

- PC/Laptop
- Internet Connection

### 6.10.2 Software Requirements

- Operating System: Windows 7 and above
- MySQL database management system

## 6.11 Responsibilities

- The Computational Secret Sharing (**CSS**) development team is responsible for testing the system
- Project supervisor is responsible for the approval of the test document

## 6.12 Staffing and Training Needs

- A 1-hour training session can be arranged to guide the staff and faculty that will be managing this application
- The portfolio for students is self-explanatory and doesn't require any special training

## 6.13 Risks and Contingencies

- Delay in delivery of test items might require increased night shift scheduling to meet the delivery date
- Understanding requirements
- Domain and project knowledge

## 6.14 Test Cases

| | |
|---|---|
| Test Case Name | Application Startup Testing |

| | |
|---|---|
| Test Case ID | 01 |
| Description | This feature sends the user to the home screen of the application when he or she clicks on the application's icon. |
| Testing Technique Used | Black Box Testing |
| Preconditions | The computer is on and is connected to the internet. |
| Input Values | Mouse Click |
| Valid Inputs | Mouse Click |
| Steps | 1. Double click (desktop)<br>2. Single click (task bar) |
| Expected Output | The user will be sent to the home screen of Computational Secret Sharing. |
| Actual Output | Successful opening of the application |
| Status | PASS |

*Table 6-1- Application Startup Testing*

| | |
|---|---|
| Test Case Name | Start Encryption Testing |
| Test Case ID | 02 |
| Description | This feature allows the user to encrypt his or her file. This feature leads the user to start encryption by leading him or her to the next screen. This test case is aimed to check that feature |

| | |
|---|---|
| | works according to user requirement. |
| Testing Technique Used | Black Box Testing |
| Preconditions | System is running and connected to database. User has opened the application. |
| Input Values | Mouse click |
| Valid Inputs | Mouse Click |
| Steps | 1. Open application<br>2. Click on the Encryption icon |
| Expected Output | The user is lead to the screen where he or she are required to enter details of the source and destination files. |
| Actual Output | Successful. The user is lead to the expected screen. |
| Status | PASS |

*Table 6-2- Start Encryption Testing*

| | |
|---|---|
| Test Case Name | Enter Source and Destination for Encryption Testing |
| Test Case ID | 03 |
| Description | This feature allows the user to enter the source file i.e. the file to be encrypted and the destination file i.e. the place to store the encrypted file. When he or she presses encrypt, the source file is encrypted and the user is taken to the key shares prompt screen. This test case is aimed to check that feature works according to user requirement. |

| | |
|---|---|
| Testing Technique Used | Black Box Testing |
| Preconditions | System is running and connected to the internet. User has opened the application and clicked on the Encryption icon. |
| Input Values | 1. Source File's Name<br>2. Source File's Location<br>3. Destination File's Name<br>4. Destination File's Location |
| Valid Inputs | Alphanumeric values for the fields stated above. |
| Steps | 1. Click on Encryption icon<br>2. Enter the Source File's Name and Location<br>3. Enter the Destination File's Name and Location<br>4. Click Encrypt |
| Expected Output | The file is encrypted and user is taken to the key shares prompt screen. |
| Actual Output | The file is encrypted and user is taken to the key shares prompt screen after clicking Encrypt. |
| Status | PASS |

*Table 6-3- Enter Source and Destination for Encryption Testing*

| | |
|---|---|
| Test Case Name | Key Shares Prompt Screen Testing |
| Test Case ID | 04 |
| Description | This test case checks that the user can enter the total number |

| | |
|---|---|
| | of keys he or she requires and the minimum number of keys required for reconstruction. It also checks that after clicking OK the original key is divided into the said total number of shares and the user is taken to the next phase. |
| Testing Technique Used | Black Box Testing |
| Preconditions | System is running and connected to the internet. User has opened the application, clicked on the Encryption icon and has entered the Source and Destination Files' details. |
| Input Values | 1. Total Number of Key Shares<br>2. Required Number of Key Shares |
| Valid Inputs | Numeric values for the fields stated above |
| Steps | 1. Click on Encrypt (after entering details)<br>2. Enter the Total Number of Key Shares<br>3. Enter the Required Number of Key Shares<br>4. Click OK |
| Expected Output | Key is divided according to the specified Key Shares and user is taken to the next phase. |
| Actual Output | Key is divided according to the specified Key Shares and user is taken to the next phase i.e. Key Transferring. |
| Status | PASS |

*Table 6-4- Key Shares prompt Testing*

| | |
|---|---|
| Test Case Name | Transferring Key Shares Testing (for Encryption) |
| Test Case ID | 05 |
| Description | This test case checks that the Key Shares are transferred securely. The user keeps on transferring the shares until all of them are transferred and then he or she is taken to the next screen. |
| Testing Technique Used | Black Box Testing |
| Preconditions | System is running and connected to the internet. User has opened the application, clicked on the Encryption icon, has entered the Source and Destination Files' details and also entered the Total and Required Key Shares. |
| Input Values | 1. Location of the place where the Key Share shall be transferred<br>2. Password |
| Valid Inputs | Alphanumeric values for the fields stated above. |
| Steps | 1. Enter the Total and Required Number of Key Shares<br>2. Click OK<br>3. Enter the Location (where to transfer)<br>4. Enter Password<br>5. Click Next to transfer next Key Shares<br>6. Click OK when all of the Key Shares are transferred |
| Expected Output | Key Share Shall be transferred and the user shall be prompted |

| | |
|---|---|
| | to enter the next one until all of them are transferred and then he or she shall be taken to the next screen. |
| Actual Output | Key Share Shall be transferred and the user shall be prompted to enter the next one until all of them are transferred and then he or she shall be taken to the next screen i.e. completion screen. |
| Status | PASS |

*Table 6-5- Transferring Key Shares (for Encryption) Testing*

| | |
|---|---|
| Test Case Name | Completion of Encryption Testing |
| Test Case ID | 06 |
| Description | This test case checks that the Encryption is complete and that the Key Shares have been transferred. |
| Testing Technique Used | Black Box |
| Preconditions | System is running and connected to the internet. User has opened the application, clicked on the Encryption icon, has entered the Source and Destination Files' details, also entered the Total and Required Key Shares and transferred the Key Shares. |
| Input Values | None. |
| Valid Inputs | None. |
| Steps | 1. Transfer the Key Shares |

| | |
|---|---|
| | 2. Click OK (on the last one) |
| Expected Output | Encryption is complete and the user can now return to the home screen for further use. |
| Actual Output | Encryption is complete and the user can now return to the home screen for further use. |
| Status | PASS |

*Table 6-6- Completion of Encryption Testing*

| | |
|---|---|
| Test Case Name | Return to Home (after Encryption) Testing |
| Test Case ID | 07 |
| Description | This test case checks that the user can return to the Home Screen after Encryption is complete. |
| Testing Technique Used | Black Box |
| Preconditions | User is using the application and Encryption is completed. |
| Input Values | Mouse Click |
| Valid Inputs | Mouse Click |
| Steps | 1. Encryption is Complete<br>2. Click on Return to Home |
| Expected Output | User is taken to the Home Screen. |
| Actual Output | User is taken to the Home Screen. |
| Status | PASS |

*Table 6-7- Return to Home Screen (after Encryption) Testing*

| | |
|---|---|
| Test Case Name | Start Decryption Testing |
| Test Case ID | 08 |
| Description | This feature allows the user to decrypt his or her file. This feature leads the user to start decryption by leading him or her to the next screen. This test case is aimed to check that feature works according to user requirement. |
| Testing Technique Used | Black Box |
| Preconditions | System is running and connected to database. User has opened the application. |
| Input Values | Mouse Click |
| Valid Inputs | Mouse Click |
| Steps | 1. Open application<br>2. Click on the Decryption icon |
| Expected Output | The user is lead to the screen where he or she are required to enter details of the source and destination files. |
| Actual Output | Successful. The user is lead to the expected screen. |
| Status | PASS |

*Table 6-8- Start Decryption Testing*

| | |
|---|---|
| Test Case Name | Enter Source and Destination for Decryption Testing |
| Test Case ID | 09 |
| Description | This feature allows the user to enter the source file i.e. the file |

| | |
|---|---|
| | to be decrypted and the destination file i.e. the place to store the decrypted file. When he or she presses decrypt, the source file is decrypted and the user is taken to the transfer key shares prompt screen. This test case is aimed to check that feature works according to user requirement. |
| Testing Technique Used | Black Box |
| Preconditions | System is running and connected to the internet. User has opened the application and clicked on the Decryption icon. |
| Input Values | 1. Source File’s Name<br>2. Source File’s Location<br>3. Destination File’s Name<br>4. Destination File’s Location |
| Valid Inputs | Alphanumeric values for the fields stated above. |
| Steps | 1. Click on Decryption icon<br>2. Enter the Source File’s Name and Location<br>3. Enter the Destination File’s Name and Location<br>4. Click Decrypt |
| Expected Output | The user is prompted to enter the Required Number of Key Shares. |
| Actual Output | The user is prompted to enter the Required Number of Key Shares. |
| Status | PASS |

*Table 6-9- Enter Source and Destination for Decryption Testing*

| | |
|---|---|
| Test Case Name | Transferring Key Shares Testing (for Decryption) |
| Test Case ID | 10 |
| Description | This test case checks that the Key Shares are transferred securely. The user keeps on transferring the shares until all of them are transferred and then he or she is taken to the next screen. |
| Testing Technique Used | Black Box |
| Preconditions | System is running and connected to the internet. User has opened the application, clicked on the Decryption icon and has entered the Source and Destination Files' details. |
| Input Values | 1. Location of the place where the Key Share shall be transferred<br>2. Password |
| Valid Inputs | Alphanumeric values for the fields stated above. |
| Steps | 1. Enter the Total and Required Number of Key Shares<br>2. Click OK<br>3. Enter the Location (where to transfer)<br>4. Enter Password<br>5. Click Next to transfer next Key Shares<br>6. Click OK when all of the Key Shares are transferred |
| Expected Output | Key Share Shall be transferred and the user shall be prompted |

| | |
|---|---|
| | to enter the next one until all of them are transferred and then he or she shall be taken to the next screen. |
| Actual Output | Key Share Shall be transferred and the user shall be prompted to enter the next one until all of them are transferred and then he or she shall be taken to the next screen i.e. the completion screen. |
| Status | PASS |

*Table 6-10- Transferring Key Shares (for Decryption) Testing*

| | |
|---|---|
| Test Case Name | Completion of Decryption Testing |
| Test Case ID | 11 |
| Description | This test case checks that the Decryption is complete and that the Key Shares have been transferred. |
| Testing Technique Used | Black Box |
| Preconditions | System is running and connected to the internet. User has opened the application, clicked on the Decryption icon, has entered the Source and Destination Files' details and transferred the Key Shares. |
| Input Values | None |
| Valid Inputs | None |
| Steps | 1. Transfer the Key Shares<br>2. Click OK (on the last one) |

| | |
|---|---|
| Expected Output | Encryption is complete and the user can now return to the home screen for further use. |
| Actual Output | Encryption is complete and the user can now return to the home screen for further use. |
| Status | PASS |

*Table 6-11- Completion of Decryption Testing*

| | |
|---|---|
| Test Case Name | Open Decrypted File Testing |
| Test Case ID | 12 |
| Description | This test case checks if the file has been Decrypted Properly and opens the Decrypted File on our device. |
| Testing Technique Used | Black Box |
| Preconditions | System is running and connected to the internet. User has opened the application, clicked on the Decryption icon, has entered the Source and Destination Files' details, transferred the Key Shares and Decryption is complete. |
| Input Values | Mouse Click |
| Valid Inputs | Mouse Click |
| Steps | 1. Decryption is complete<br>2. Click on Open File |
| Expected Output | The Decrypted File is opened and the user can view the results. |

| | |
|---|---|
| Actual Output | The Decrypted File is opened and the user can view the results. |
| Status | PASS |

Table 6-12 Open Decrypted File Testing

| | |
|---|---|
| Test Case Name | Return to Home (after Decryption) Testing |
| Test Case ID | 13 |
| Description | This test case checks that the user can return to the Home Screen after Encryption is complete. |
| Testing Technique Used | Black Box |
| Preconditions | User is using the application and Encryption is completed. |
| Input Values | Mouse Click |
| Valid Inputs | Mouse Click |
| Steps | 1. Encryption is Complete<br>2. Click on Return to Home |
| Expected Output | User is taken to the Home Screen. |
| Actual Output | User is taken to the Home Screen. |
| Status | PASS |

Table 6-13 Return to Home (after Decryption) Testing

| | |
|---|---|
| Test Case Name | Hamburger Menu/ Navigation Menu Testing |
| Test Case ID | 14 |

| | |
|---|---|
| Description | This test case checks the navigation menu, to see if it functions properly. |
| Testing Technique Used | Black Box |
| Preconditions | User has opened the application and is using it. |
| Input Values | Mouse Click |
| Valid Inputs | Mouse Click |
| Steps | 1. Open the Application<br>2. Click on the Hamburger Menu icon |
| Expected Output | The Navigation Menu opens. |
| Actual Output | The Navigation Menu opens. |
| Status | PASS |

*Table 6-14 Hamburger Menu/Navigation Menu Testing*

| | |
|---|---|
| Test Case Name | Navigation Menu’s Home Button Testing |
| Test Case ID | 15 |
| Description | This test case checks the feature of Home in the Navigation Menu. |
| Testing Technique Used | Black Box |
| Preconditions | User is using the application and has opened the Navigation Menu. |
| Input Values | Mouse Click |
| Valid Inputs | Mouse Click |

| | |
|---|---|
| Steps | 1. Click on Hamburger Menu icon<br>2. Click on Home |
| Expected Output | The user is directed to the Home Screen. |
| Actual Output | The user is directed to the Home Screen. |
| Status | PASS |

*Table 6-15 Navigation Menu's Home Button Testing*

| | |
|---|---|
| Test Case Name | Navigation Menu's About Us Button Testing |
| Test Case ID | 16 |
| Description | This test case checks for the About Us feature in the Navigation Menu. |
| Testing Technique Used | Black Box |
| Preconditions | User is using the application and has opened the Navigation Menu. |
| Input Values | Mouse Click |
| Valid Inputs | Mouse Click |
| Steps | 1. Click on Hamburger Menu icon<br>2. Click on About Us |
| Expected Output | User is directed to the About Us Screen of the Application. |
| Actual Output | User is directed to the About Us Screen of the Application |
| Status | PASS |

*Table 6-16 Navigation Menu's About Us Button Testing*

| | |
|---|---|
| Test Case Name | Navigation Menu’s Contact Us Button Testing |
| Test Case ID | 17 |
| Description | This test case checks for the Contact Us feature in the Navigation Menu. |
| Testing Technique Used | Black Box |
| Preconditions | User is using the application and has opened the Navigation Menu. |
| Input Values | Mouse Click |
| Valid Inputs | Mouse Click |
| Steps | 1. Click on Hamburger Menu icon<br>2. Click on Contact Us |
| Expected Output | User is directed to the Contact Us Screen of the Application. |
| Actual Output | User is directed to the Contact Us Screen of the Application |
| Status | PASS |

*Table 6-17 Navigation Menu’s Contact Us Button Testing*

## 6.15 Summary

From the extensive black box testing, it can be concluded that the system is working fine and in accordance with the functional requirements stated in the SRS document. The system can be further improved and enhanced during maintenance and upgrade phase. A detailed white box testing can also highlight the errors and bugs, if present, in the code of the application.

CHAPTER 7

# Conclusion and Future Work

# 7. Conclusion& Future Work

## 7.1 Conclusion

The modern cloud-based systems that are commonly produced are often scaled from private platforms into public cloud infrastructures, with the addition of some degree of encryption. As the data is more accessible, many systems now suffer from a loss of the symmetric key which had been used to encrypt the data. Too often a single key is used to encrypt large-scale data storage elements, where a single breach could potentially release vast amounts of sensitive information.

The usage of Computational Secret Sharing (**CSS**) will provide new and better methods of protecting data and providing robustness. Thus this application achieves what a single key encryption cannot achieve, i.e. breaching of this method of encryption is highly difficult and even if it is breached key shares below the number of threshold key shares do not release any sensitive information.

## 7.2 Design Decisions & Tradeoffs

The user interface has been kept simple and friendly so even a user with only basic knowledge of desktopapplications can use it effectively.

Clearly, components can do their work independently, but, in a certain flow (data as well as control). This leads us to **high cohesion.**

Moreover, component don’t have much interaction, once a component has completed its work system will generate an event for further action, consequently, the component registered for that event will come into action. This leads us to **low coupling**.

MVC pattern will be used for the implementation of this application. General behavior of MVC is shown below.

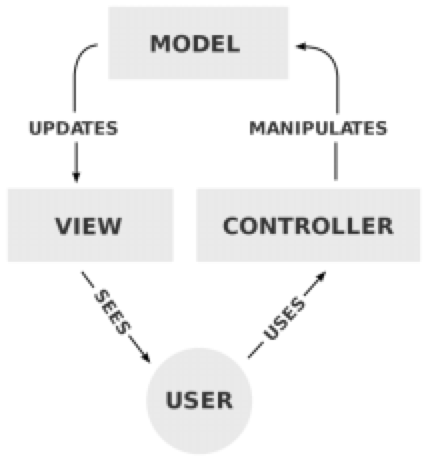


*Figure 7-1 – MVC*

## 7.3 Future Work

As specified in the external scope of the product, in the coming years, this software can be modified and adjusted to be implemented and used by the Pakistan Military Organizations, Government agencies and Intelligence Agencies.

The project can be further enhanced by incorporating confidentiality and shifting it over to HTTPS server. A survey can be conducted among the users of the application, and considering their reviews, changes can be made in the application on annual basis.

# Bibliography

[1]The Future Internet: A World of Secret Shares, William J. Buchanan, David Lanc, ElochukwuUkwandu, Lu Fan, Gordon Russell andOwen Lo

[2]Backups with Computational Secret Sharing, Master Thesis, Rasmus W. Lauritsen

[3]MVC architecture from HCI book Building Interactive Systems- Principles of HCI-Dan Olsen

[4]P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, 1st ed. Boston: Addison-Wesley Professional, 2015.

[5]B. Boehm, "Software Risk Management: Principles and Practices", *IEEE Software*, vol. 8, no. 1, pp. 32-41, 1991.

[6]https://medium.com/vault12/understanding-shamirs-secret-sharing-6a4bd27768c9