Project Made by:

Dhruv Patel

Shyam Patel

Bikramjeet Singh

**ABSTRACT**

* As Cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud.
* For the protection of data privacy, sensitive data usually have to be encrypted before outsourcing, which makes effective data utilization a very challenging task. Although traditional searchable encryption schemes allow a user to securely search over encrypted data through keywords and selectively retrieve files of interest, these techniques support only *exact* keyword search.
* That is, there is no tolerance of minor typos and format inconsistencies which, on the other hand, are typical user searching behavior and happen very frequently. This significant drawback makes existing techniques unsuitable in Cloud Computing as it greatly affects system usability, rendering user searching experiences very frustrating and system efficacy very low. In this paper, for the first time we formalize and solve the problem of effective fuzzy keyword search over encrypted cloud data while maintaining keyword privacy.
* Fuzzy keyword search greatly enhances system usability by returning the matching files when users’ searching inputs exactly match the predefined keywords or the closest possible matching files based on keyword similarity semantics, when *exact* match fails.
* In our solution, we exploit edit distance to quantify keywords similarity and develop an advanced technique on constructing fuzzy keyword sets, which greatly reduces the storage and representation overheads. Through rigorous security analysis, we show that our proposed solution is secure and privacy-preserving, while correctly realizing the goal of fuzzy keyword search.

**TABLE OF CONTENTS**

**Sr.no Contents Pg.no**

**1. INTRODUCTION**

1.1Motivation

1.2 Background

1.3 Proposed Project

**2. PROPOSED WORK**

2.1 Problem Definition

2.2 Features

2.3 Goal

2.4 Risk Management

2.5 Lifecycle of project

2.6 Feasibility Study

**3. SYSTEM DESCRIPTION**

3.1 Modules description in detail

3.2 Problem formation

3.3 Advantages and Disadvantages

3.4 Constructions over fuzzy keyword search

3.5 Security Analysis

**4. SCHEDULE OF WORK**

**5. PROJECT DESIGN**

5.1 Hardware and Software requirements

5.2 Analysis and data modeling

5.5.1 DFDs, Overflow diagram

5.5.2 Use case, URL diagrams

**6. IMPLEMENTATION**

6.1 Implementation

6.2 Scenario

6.3 Testing

6.3.1 Testing Strategies

6.3.2 Testing cases

6.4 Sreenshots

6.5 Sample code

**CONCLUSION AND FURTHER ENHANCEMENTS**

**REFERENCES**

**APPENDIX**

**LIST OF TABLES**

**Sr.no Contents Pg.no**

1. Project Schedule

2. Test Cases

**LIST OF FIGURES**

**Fig.no Contents Pg.no**

5.2.1 Module Diagram 41

5.5.2 UML Diagram 42

5.2.3 Class Diagram 43

5.2.4 Object Diagram 44

5.2.5 Dataflow Diagram 45

5.2.6 Projectflow Diagram 46

5.2.7 System Architecture 4

**CHAPTER 1 INTRODUCTION**

**1.1 Motivation**

In this paper, we address the problem of supporting efficient yet privacy-preserving fuzzy keyword search services over encrypted cloud data. Specifically, we have the following goals: i) to explore new mechanism for constructing storage efficient fuzzy keyword sets; ii) to design efficient and effective fuzzy search scheme based on the constructed fuzzy keyword sets; iii) to validate the security of the proposed scheme.

**1.2 Background**

Plaintext fuzzy keyword search. Recently, the importance of fuzzy search has received attention in the context of plaintext searching in information retrieval community.They addressed this problem in the traditional information accessparadigm by allowing user to search without usingtry-and-see approach for finding relevant information based on approximate string matching. At the first glance, it seems possible for one to directly apply these string matching algorithms to the context of searchable encryption by computing the trapdoors on a character base within an alphabet. However, this trivial construction suffers from the dictionary and statistics attacks and fails to achieve the search privacy.

**1.3 Proposed Project**

Fuzzy keyword search is needed as it provides tight security against the hackers and the trapdoors who want the access to the data in order to hack the information.This straightforward approach apparently provides fuzzykeyword search over the encrypted files while achieving searchprivacy using the technique of secure trapdoors. However, this approach has serious efficiency disadvantages. The simple enumeration method in constructing fuzzy keyword sets wouldintroduce large storage complexities, which greatly affect the usability. Recall that in the definition of edit distance, substitution, deletion and insertion are three kinds of operations in computation of edit distance. For example, assume there are 104 keywords in the file collection with average keyword length 10, d = 2, and the output length of hash function is 160 bits, then, the resulted storage cost for the index will be 30GB. Therefore, it brings forth the demand for fuzzy keyword sets with smaller size.

**CHAPTER 2 PROPOSED WORK**

**2.1 PROBLEM DEFINATION**

In this paper, we consider a cloud data system consisting of data owner, data user and cloud server.We assume the authorization between the data owner and users is appropriately done. An authorized user types in a request to selectively retrieve data files of his/her interest. The cloud server is responsible for mapping the searching request to a set of data files, where each file is indexed by a file ID and linked to a set of keywords. The fuzzy keyword search scheme returns the search results according to the following rules: 1) if the user’s searching input exactly matches the pre-set keyword, the server is expected to return the files containing the keyword. We formalize and solve the problem of effective fuzzy keyword search over encrypted cloud data while maintaining keyword privacy. Fuzzy keyword search greatly enhances system usability by returning the matching files when users’ searching inputs exactly match the predefined keywords or the closest possible matching files based on keyword similarity semantics, when exact match fails

**2.2 FEASIBILITY STUDY**

All projects are feasible given unlimited resources and infinite time. It is both necessary and prudent to evaluate the feasibility of the project at the earliest possible time. Feasibility and risk analysis is related in many ways. If project risk is great , the feasibility listed below is equally important.

The following feasibility techniques has been used in this project

* + - * Operational Feasibility
      * Technical Feasibility
      * Economic Feasibility

**Operational Feasibility:**

Proposed system is beneficial since it turned into information system analyzing the traffic that will meet the organizations operating requirements.IN security, the file is transferred to the destination and the acknowledgement is given to the server. Bulk of data transfer is sent without traffic.

**Technical Feasibility:**

Technical feasibility centers on the existing computer system (hardware , software, etc..) and to what extent it can support the proposed addition. For example, if the current computer is operating at 80% capacity. This involves, additional hardware (RAM and PROCESSOR) will increase the speed of the process. In software, Open Source language that is JAVA and is used. We can also use in Linux operating system . The technical requirement for this project are Java tool kit and Swing component software and normal hardware configuration is enough , so the system is more feasible on this criteria.

**Economic Feasibility:**

Economic feasibility is the most frequently used method for evaluating the effectiveness of a candidate system. More commonly known as cost / benefit analysis, the procedure is to determine the benefits and saving that are expected from a candidate and compare them with the costs. If the benefits outweigh cost. Then the decision is made to design and implement the system. Otherwise drop the system. This system has been implemented such that it can be used to analysis the traffic. So it does not requires any extra equipment or hardware to implement.

**RELATED WORK:**

1)Plaintext fuzzy keyword search. Recently, the importance of

fuzzy search has received attention in the context of plaintext

searching in information retrieval community [11]–[13].

2)They addressed this problem in the traditional information access

paradigm by allowing user to search without using

try-and-see approach for finding relevant information based

on approximate string matching. At the first glance, it seems

possible for one to directly apply these string matching

algorithms to the context of searchable encryption by

computing the trapdoors on a character base within an

alphabet.

3).However, this trivial construction suffers from the

dictionary and statistics attacks and fails to achieve the search

privacy.

4)Note that all these existingschemes support only exact keyword search,

and thus are notsuitable for Cloud Computin

**CHAPTER 3 SYSTEM DESCRIPTION**

**3.1 MODULES DESCRIPTION IN DETAIL:**

1. Server
2. User
3. Key Generation
4. Fuzzy Search

**MODULE 1 : SERVER**

* Input – Select the file to Encrypt and Decrypt.
* Output –Those encrypted/Decrypted files are stored into cloud server.It Will be Send to Fuzzy Set.

Algorithm Used – Encryption/Decryption Algorithm(Data Encryption Standard-DES)

**MODULE 2 : USER**

* Input – User search the files.
* Output – Searched files are displayed in an encrypted form.

Algorithm Used – String Matching Algorithm and Spell Check Algorithm

**MODULE 3 : KEY GENERATION**

* Input – Authorized User send the request to admin.
* Output – Admin Generate the key ID.

**MODULE 4 : FUZZY SEARCH**

* Input – User give the Keyword of Fuzzy Set.
* Output – User Receive The Decrypted form of files like Text format,Image format,Video Format.

Algorithm Used – String Matching Algorithm and Spell Check Algorithm.

**PROBLEM FORMATION**

**A. System model**

It includes owner, data user and cloud server. Given a collection

of encrypted data files storedin the cloud server, a predefined set

of distinct keywords the cloud server provides the searchservice for

the authorized users over the encrypted data C. Weassume the

authorization between the data owner and users

is appropriately done An authorized user types in a request

to selectively retrieve data files of his/her interest. The cloud

server is responsible for mapping the searching request to a set

of data files, where each file is indexed by a file ID and linked

to a set of keywords. The fuzzy keyword search scheme returns

the search results according to the following rules:

1) If the user’s searching input exactly matches the pre-set keyword, the

server is expected to return the files containing the keyword1;

2) If there exist typos and/or format inconsistencies in the

searching input, the server will return the closest possible

results based on pre-specified similarity semantics .

**B. Threat Model**

We consider a semi-trusted server. Even though data files are

encrypted, the cloud server may try to derive other sensitive

information from users’ search requests while performing

keyword-based search over C. Thus, the search should be

conducted in a secure manner that allows data files to be

securely retrieved while revealing as little information as

possible to the cloud server. In this paper, when designing

fuzzy keyword search scheme, we will follow the security

definition deployed in the traditional searchable encryption [8].

More specifically, it is required that nothing should be leaked

from the remotely stored files and index beyond the outcome

and the pattern of search queries.

**C. Design Goals**

In this paper, we address the problem of supporting efficient

yet privacy-preserving fuzzy keyword search services over

encrypted cloud data. Specifically, we have the following

goals:

1)to explore new mechanism for constructing storage efficient

fuzzy keyword sets;

2)to design efficient and effective fuzzy search scheme based on the constructed fuzzy keywords etc.

3) to validate the security of the proposed scheme.

**ADVANTAGES:**

1.Before introducing our construction of fuzzy

keyword sets,we first propose a straightforward approach that achieves

all the functions of fuzzy keyword search, which aims at

providing an overview of how fuzzy search scheme works

over encrypted data.

2. This straightforward approach apparently provides fuzzy

keyword search over the encrypted files while achieving search

privacy using the technique of secure trapdoors.

**DISADVANTAGES:**

1.The simple enumeration method in constructing fuzzy keyword sets would

introduce large storage complexities, which greatly affect the

usability.

2.Recall that in the definition of edit distance, substitution,deletion and insertion are three kinds of operations in computation of edit distance.

**CONSTRUCTIONS OF EFFECTIVE FUZZY KEYWORD**

**SEARCH**

The key idea behind our secure fuzzy keyword search istwo-fold: 1) building up fuzzy keyword sets that incorporatenot only the exact keywords but also the ones differing slightlydue to minor typos, format inconsistencies, etc.; 2) designingan efficient and secure searching approach for file retrievalbased on the resulted fuzzy keyword sets.*A. Advanced Technique for Constructing Fuzzy Keyword Sets*

To provide more practical and effective fuzzy keyword

search constructions with regard to both storage and search

efficiency, we now propose an advanced technique to improve

the straightforward approach for constructing the fuzzy

keyword set. Without loss of generality, we will focus on

the case of edit distance d = 1 to elaborate the proposed

advanced technique. For larger values of d, the reasoning is

similar. Note that the technique is carefully designed in such

a way that while suppressing the fuzzy keyword set, it will

not affect the search correctness.

**SECURITY ANALYSIS:**

In this section, we analyze the correctness and security ofthe proposed fuzzy keyword search scheme. At first, we showthe correctness of the schemes in terms of two aspects, that

is, completeness and soundness.

**Theorem 1:**The wildcard-based scheme satisfies both

completeness and soundness. Specifically, upon receiving therequest of w, all of the keywords {wi} will be returned.

**Theorem 2*:***The fuzzy keyword search scheme is secure

regarding the search privacy.

*Proof:* In the wildcard-based scheme, the computation of

index and request of the same keyword is identical. Therefore,

we only need to prove the index privacy by using reduction.

Suppose the searchable encryption scheme fails to achieve the

index privacy against the indistinguishability under the chosen

keyword attack, which means there exists an algorithm A who

can get the underlying information of keyword from the index.

Then, we build an algorithm A′ that utilizes A to determine

whether some function f′(·)

**CHAPTER 4 SCHEDULE OF WORK**

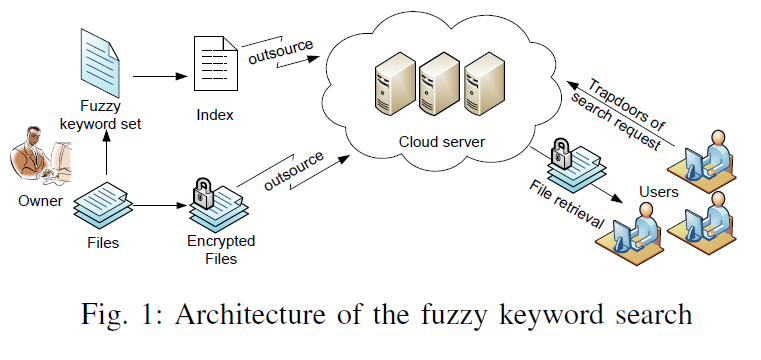
|  |  |  |
| --- | --- | --- |
| **SR.NO** | **NAME OF TASK** | **NO OF DAYS** |
| **1.** | **Information Gathering and problem identification (detailed problem definition of the system to be implemented )** | **10** |
| **2.** | **Requirement Analysis including project plan and literature survey(visiting different websites and studying the reference books)** | **10** |
| **3.** | **Designing Analysis which includes the architecture design between modules ,DFDs and Use-case diagrams, showing all the screens and designing them** | **30** |
| **4.** | **Coding which includes the overall code required to execute the project and implements design details using .NET** | **35** |
| **5.** | **Testing (testing the system for expected result)** | **38** |
| **6.** | **Implementation** | **41** |

**CHAPTER 5 PROJECT DESIGN**

**5.1 SYSTEM REQUIREMENTS**

* **Hardware Requirements**
* RAM : 512MB
* Hard Disk : 40GB
* General : Keyboard,Monitor,Mouse
* Processor : Pentium-IV
* Speed:1.6GHZ
* **Software Requirements:**
* Operating System : Windows XP
* Front End Design : Vb. Net
* Front End Language : C#
* Back End : SQL Server

**SYSTEM ARCHITECTURE:**

****

**TEST CASES:**

Software testing is necessary because we all make mistakes some of these mistakes are unimportant. but some may be dangerous .We need to check everything and anything we produce because things can always go wrong-humans make mistakes all the time.

TESTCASES are set of conditions under which a tester will determine whether a functionality of software is working correctly or not.

**TESTCASE 1**

If username and password which we type is incorrect and it doesnt match with the expected user name and password ,then it will display as "INVALID" or else if it is correct then it will display as "INSERTED".

**TEST CASE 2**

if u enter the expected filename then only the expected file gets displayed. when the disered file gets displayed it is in encrpyted form.and if we click on the "decrypt the file".it won't display in decrpyted form

**TESTCASE 3:**

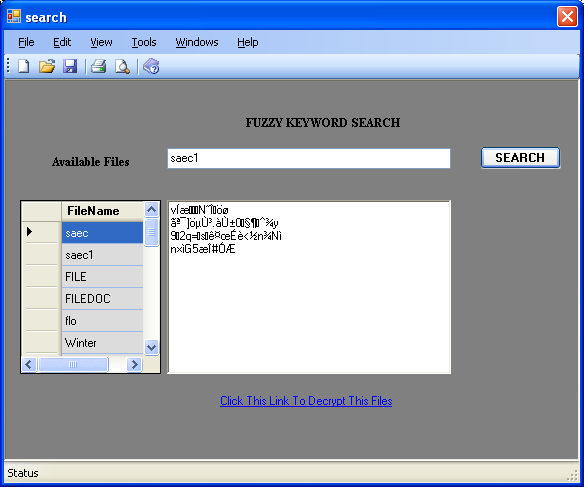
If we type any filename or we search any file that we want as our resultant outcome ..should match according to contents available in directory then only that file will be displayed .if it doesn't matches then it will print the message as"no display of results".

**TEST CASE 4:**

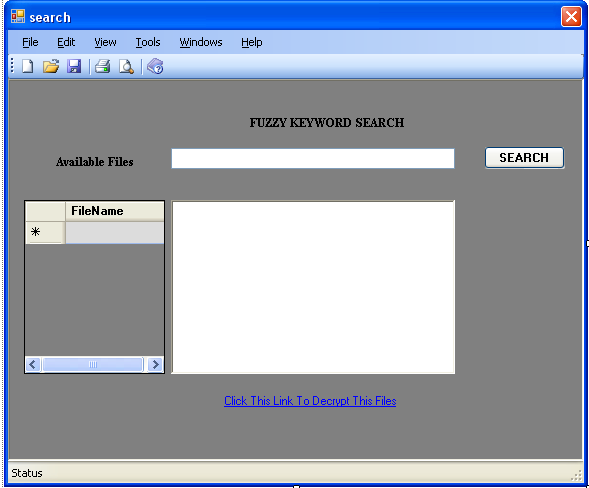
Every file whether it is image or text should have a set of keywords.eg:-flower.rose,flower.chameli .whenever the search algorithm proccesses the search keyword it makes it easier to search in the database of keywords instead of searching in the database of encrypted files.

**SNAPSHOTS:**

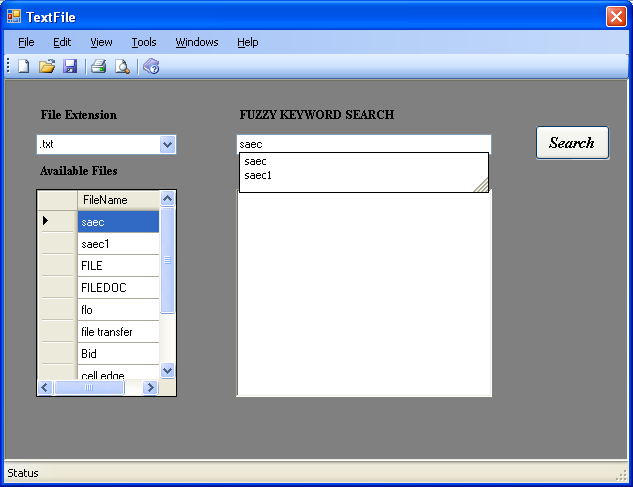
SNAPSHOT 1:

c

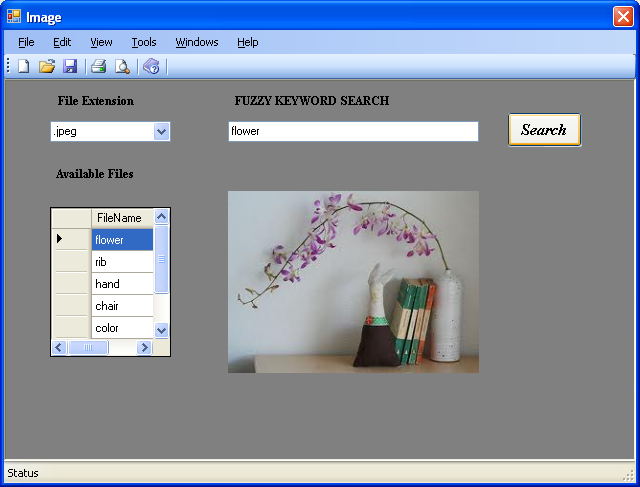
SNAPSHOT 2:



SNAPSHOT 3:



SNAPSHOT 4:



SAMPLE CODE:

namespace fuzzy

{

public partial class File : Form

{

byte[] buffer;

string sKey= "!$3h2i#v@";

public string st, pat;

public string ext, fn;

private static byte[] key = { };

private static byte[] IV = { 131, 52, 204, 248, 78, 164, 21, 13 };

private static string stringKey = "!$3h2i#v@";

//sKey = stringKey;

public File()

{

InitializeComponent();

}

private void ShowNewForm(object sender, EventArgs e)

{

richTextBox1.Visible = true;

//// Create a new instance of the child form.

//Form childForm = new Form();

//// Make it a child of this MDI form before showing it.

//childForm.MdiParent = this;

//childForm.Text = "Window " + childFormNumber++;

//childForm.Show();

}

private void OpenFile(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Personal);

openFileDialog.Filter = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*";

if (openFileDialog.ShowDialog(this) == DialogResult.OK)

{

string FileName = openFileDialog.FileName;

FileStream f = new FileStream(FileName.ToString (),FileMode.Open, FileAccess.Read);

try

{

int length = (int)f.Length;

buffer = new byte[length];

int count = 0;

int sum = 0;

while ((count = f.Read(buffer, sum, length - sum)) > 0)

sum += count;

}

finally

{

f.Close();

}

}

}

private void SaveAsToolStripMenuItem\_Click(object sender, EventArgs e)

{

Stream myStream;

string resultFileName = "";

saveFileDialog1.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

saveFileDialog1.FilterIndex = 1;

saveFileDialog1.RestoreDirectory = true;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

resultFileName = saveFileDialog1.FileName.ToString();

// saveResults

if ((myStream = saveFileDialog1.OpenFile()) != null)

{

try

{

StreamWriter wText = new StreamWriter(myStream);

wText.AutoFlush = true;

string aLine = null;

StringReader strReader = new StringReader(richTextBox1.Text);

while (true)

{

aLine = strReader.ReadLine();

if (aLine == null)

break;

else

wText.WriteLine(aLine);

}

strReader.Close();

strReader = null;

wText.WriteLine("=======================");

myStream.Close();

MessageBox.Show("Results successfully saved!");

richTextBox1.Visible = false;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Warning");

richTextBox1.Visible = false;

}

}

}

//SaveFileDialog saveFileDialog = new SaveFileDialog();

//saveFileDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Personal);

//saveFileDialog.Filter = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*";

//if (saveFileDialog.ShowDialog(this) == DialogResult.OK)

//{

// string FileName = saveFileDialog.FileName;

// // TODO: Add code here to save the current contents of the form to a file.

//}

}

private void ExitToolsStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void CutToolStripMenuItem\_Click(object sender, EventArgs e)

{

// TODO: Use System.Windows.Forms.Clipboard to insert the selected text or images into the clipboard

}

private void CopyToolStripMenuItem\_Click(object sender, EventArgs e)

{

// TODO: Use System.Windows.Forms.Clipboard to insert the selected text or images into the clipboard

}

private void PasteToolStripMenuItem\_Click(object sender, EventArgs e)

{

// TODO: Use System.Windows.Forms.Clipboard.GetText() or System.Windows.Forms.GetData to retrieve information from the clipboard.

}

private void ToolBarToolStripMenuItem\_Click(object sender, EventArgs e)

{

toolStrip.Visible = toolBarToolStripMenuItem.Checked;

}

private void StatusBarToolStripMenuItem\_Click(object sender, EventArgs e)

{

statusStrip.Visible = statusBarToolStripMenuItem.Checked;

}

private void CascadeToolStripMenuItem\_Click(object sender, EventArgs e)

{

LayoutMdi(MdiLayout.Cascade);

}

private void TileVerticleToolStripMenuItem\_Click(object sender, EventArgs e)

{

LayoutMdi(MdiLayout.TileVertical);

}

private void TileHorizontalToolStripMenuItem\_Click(object sender, EventArgs e)

{

LayoutMdi(MdiLayout.TileHorizontal);

}

private void ArrangeIconsToolStripMenuItem\_Click(object sender, EventArgs e)

{

LayoutMdi(MdiLayout.ArrangeIcons);

}

private void CloseAllToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (Form childForm in MdiChildren)

{

childForm.Close();

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (richTextBox1.Visible==true )

{

string source = richTextBox1.Text;

saveFileDialog1.Filter = "des files |\*.des";

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

string destination = saveFileDialog1.FileName;

EncryptFile(source, destination, sKey);

}

MessageBox.Show("Succesfully Encrypted!");

}

else

{

// sKey = GenerateKey();

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

st = openFileDialog1.FileName;

ext = Path.GetExtension(st);

fn = Path.GetFileNameWithoutExtension(st);

try

{

byte[] imageData = ReadFile2(st);

SqlConnection CN = new SqlConnection("server=.;database=Fuzzy;uid=sa;pwd=Reset123;");

string qry = "insert into Decrypt(FileName,FilePath,Extension,Content)values(@FileName,@FilePath,@Extension,@Content)";

SqlCommand SqlCom = new SqlCommand(qry, CN);

SqlParameter p1 = new SqlParameter("@FileName", fn);

SqlParameter p2 = new SqlParameter("@FilePath", st);

SqlParameter p3 = new SqlParameter("@Extension", ext);

SqlParameter p4 = new SqlParameter("@Content", imageData);

SqlCom.Parameters.Add(p1);

SqlCom.Parameters.Add(p2);

SqlCom.Parameters.Add(p3);

SqlCom.Parameters.Add(p4);

CN.Open();

SqlCom.ExecuteNonQuery();

CN.Close();

}

catch (SqlException ex)

{

MessageBox.Show("error" + ex.ToString());

}

string source = openFileDialog1.FileName;

saveFileDialog1.Filter = "des files |\*.des|All files (\*.\*)|\*.\*";

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

string destination = saveFileDialog1.FileName;

EncryptFile(source, destination, sKey);

MessageBox.Show("Succesfully Encrypted!");

st = saveFileDialog1.FileName;

ext = Path.GetExtension(st);

fn = Path.GetFileNameWithoutExtension(st);

try

{

byte[] imageData = ReadFile1(st);

SqlConnection CN = new SqlConnection("server=.;database=Fuzzy;uid=sa;pwd=Reset123;");

string qry = "insert into Encrypt(FileName,FilePath,Extension,Content)values(@FileName,@FilePath,@Extension,@Content)";

SqlCommand SqlCom = new SqlCommand(qry, CN);

SqlParameter p1 = new SqlParameter("@FileName", fn);

SqlParameter p2 = new SqlParameter("@FilePath", st);

SqlParameter p3 = new SqlParameter("@Extension", ext);

SqlParameter p4 = new SqlParameter("@Content", imageData);

SqlCom.Parameters.Add(p1);

SqlCom.Parameters.Add(p2);

SqlCom.Parameters.Add(p3);

SqlCom.Parameters.Add(p4);

CN.Open();

SqlCom.ExecuteNonQuery();

CN.Close();

}

catch (SqlException ex)

{

MessageBox.Show("error" + ex.ToString());

}

}

}

}

}

private void EncryptFile(string source, string destination, string sKey)

{

key = Encoding.UTF8.GetBytes(stringKey.Substring(0, 8));

FileStream fsInput = new FileStream(source,

FileMode.Open,

FileAccess.Read);

FileStream fsEncrypted = new FileStream(destination,

FileMode.Create,

FileAccess.Write);

DESCryptoServiceProvider DES = new DESCryptoServiceProvider();

//DES.Key = ASCIIEncoding.ASCII.GetBytes(sKey);

//DES.IV = ASCIIEncoding.ASCII.GetBytes(sKey);

ICryptoTransform desencrypt = DES.CreateEncryptor(key,IV);

CryptoStream cryptostream = new CryptoStream(fsEncrypted,

desencrypt,

CryptoStreamMode.Write);

byte[] bytearrayinput = new byte[fsInput.Length - 1];

fsInput.Read(bytearrayinput, 0, bytearrayinput.Length);

cryptostream.Write(bytearrayinput, 0, bytearrayinput.Length);

cryptostream.Close();

fsInput.Close();

fsEncrypted.Close();

}

byte[] ReadFile1(string sPath)

{

byte[] data = null;

FileInfo fInfo = new FileInfo(sPath);

long numBytes = fInfo.Length;

FileStream fStream = new FileStream(sPath, FileMode.Open,FileAccess.Read);

BinaryReader br = new BinaryReader(fStream);

data = br.ReadBytes((int)numBytes);

return data;

}

//function to generate a 64 bit key

private string GenerateKey()

{

// Create an instance of Symetric Algorithm. Key and IV is generated automatically.

DESCryptoServiceProvider desCrypto = (DESCryptoServiceProvider)DESCryptoServiceProvider.Create();

// Use the Automatically generated key for Encryption.

return ASCIIEncoding.ASCII.GetString(desCrypto.Key);

}

private void button3\_Click(object sender, EventArgs e)

{

// sKey = GenerateKey();

openFileDialog1.Filter = "DES Files|\*.des|All files (\*.\*)|\*.\*";

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

saveFileDialog1.Filter = "Text Files| \*.txt|Document Files|\*.doc|Destination Files|\*.des|All files (\*.\*)|\*.\*";

string source = openFileDialog1.FileName;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

string destination = saveFileDialog1.FileName;

DecryptFile(source, destination, sKey);

MessageBox.Show("Succesfully Decrypted!");

}

}

}

byte[] ReadFile2(string sPath)

{

byte[] data = null;

FileInfo fInfo = new FileInfo(sPath);

long numBytes = fInfo.Length;

FileStream fStream = new FileStream(sPath, FileMode.Open,FileAccess.Read);

BinaryReader br = new BinaryReader(fStream);

data = br.ReadBytes((int)numBytes);

return data;

}

private void DecryptFile(string source, string destination, string sKey)

{

key = Encoding.UTF8.GetBytes(stringKey.Substring(0, 8));

DESCryptoServiceProvider DES = new DESCryptoServiceProvider();

//DES.Key = ASCIIEncoding.ASCII.GetBytes(sKey);

//DES.IV = ASCIIEncoding.ASCII.GetBytes(sKey);

FileStream fsread = new FileStream(source,FileMode.Open,FileAccess.Read);

ICryptoTransform desdecrypt = DES.CreateDecryptor(key,IV);

CryptoStream cryptostreamDecr = new CryptoStream(fsread,desdecrypt,CryptoStreamMode.Read);

StreamWriter fsDecrypted = new StreamWriter(destination);

fsDecrypted.Write(new StreamReader(cryptostreamDecr).ReadToEnd());

fsDecrypted.Flush();

fsDecrypted.Close();

}

private void saveToolStripMenuItem\_Click(object sender, EventArgs e)

{

Stream myStream;

string resultFileName = "";

saveFileDialog1.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

saveFileDialog1.FilterIndex = 1;

saveFileDialog1.RestoreDirectory = true;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

resultFileName = saveFileDialog1.FileName.ToString();

// saveResults

if ((myStream = saveFileDialog1.OpenFile()) != null)

{

try

{

StreamWriter wText = new StreamWriter(myStream);

wText.AutoFlush = true;

string aLine = null;

StringReader strReader = new StringReader(richTextBox1.Text );

while (true)

{

aLine = strReader.ReadLine();

if (aLine == null)

break;

else

wText.WriteLine(aLine);

}

strReader.Close();

strReader = null;

wText.WriteLine("=======================");

myStream.Close();

MessageBox.Show("Results successfully saved!");

richTextBox1.Visible = false;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Warning");

richTextBox1.Visible = false;

}

}

}

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Home h = new Home();

h.Show();

this.Visible = false;

}

}

}

**CHAPTER 6 DATA FLOW DIAGRAMS**

MODULE DIAGRAM:

SEE DECRYPTED FILES

GIVE KEYWORD

SEARCH FILES

FUZZY KEYWORD SET

ENCRYPT FILES

CREATE FILES

USER LOGIN

SERVER LOGIN

LOGIN

UML DIAGRAM:

SERVER CLIENT

CLASS DIAGRAM:

CLIENT

USERNAME,PASSWORD

SEARCH THE FILE

SEARCH()

ENTER KEY()

VIEW DECRYPTED FILES()

SERVER

USERNAME,PASSWORD

ENCRYPT THE FILE

CREATE FILES()

ENCRYPT()

SET KEYWORD()

OBJECT DIAGRAM:

DATAFLOW DIAGRAM:

CLIENT

SERVER

LOGIN

ENTER KEY & VIEW FILES

SEARCH FILES

FUZZY KEYWORD SET

ENCRYPT FILES

CREATE FILES

PROJECT FLOW DIAGRAM:

OUTPUT

SEARCH, ENTER KEY TO VIEW DECRYPTED FILES

CLIENT

CREATE FILES, SET KET, ENCRYPT

SERVER

**CHAPTER 6 IMPLEMENTATION**

6.1 TESTING

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of A software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

6.1.2 TESTING STRATEGIES:

A **test strategy** is an outline that describes the testing approach of the [software development cycle](http://en.wikipedia.org/wiki/Software_development_process). It is created to inform project managers, testers, and developers about some key issues of the testing process. This includes the testing objective, methods of testing new functions, total time and resources required for the project, and the testing environment.

Test strategies describe how the product risks of the stakeholders are mitigated at the test-level, which types of test are to be performed, and which entry and exit criteria apply. They are created based on development design documents. System design documents are primarily used and occasionally, conceptual design documents may be referred to. Design documents describe the functionality of the software to be enabled in the upcoming release. For every stage of development design, a corresponding test strategy should be created to test the new feature sets.

TYPES OF TESTING:

There are many types of testing like

* Unit Testing
* Integration Testing
* Functional Testing
* System Testing
* Stress Testing
* Performance Testing
* Usability Testing
* Acceptance Testing
* Regression Testing
* Alpha Testing
* Beta Testing
* White-Box Testing
* Black-Box Testing

TYPES OF TESTING IN DETAIL:

1)**UNIT TESTING:**

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

# 2)INTEGRATION TESTING

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

# 3)FUNCTIONAL TESTING

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

# 4)SYSTEM TESTING

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

# 5)STRESS TESTING

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

# 6)PERFORMANCE TESTING:

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

# 7)USABILITY TESTING:

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

# 8)ACCEPTANCE TESTING

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

# 9)REGRESSION TESTING

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing. Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover [software regressions](http://en.wikipedia.org/wiki/Software_regression), as degraded or lost features, including old bugs that have come back. Such regressions occur whenever software functionality that was previously working, correctly, stops working as intended. Typically, regressions occur as an [unintended consequence](http://en.wikipedia.org/wiki/Unintended_consequence) of program changes, when the newly developed part of the software collides with the previously existing code. Common methods of regression testing include re-running previous sets of test-cases and checking whether previously fixed faults have re-emerged. The depth of testing depends on the phase in the release process and the [risk](http://en.wikipedia.org/wiki/Risk_management) of the added features. They can either be complete, for changes added late in the release or deemed to be risky, or be very shallow, consisting of positive tests on each feature, if the changes are early in the release or deemed to be of low risk. Regression testing is typically the largest test effort in commercial software development, due to checking numerous details in prior software features, and even new software can be developed while using some old test-cases to test parts of the new design to ensure prior functionality is still supported

**10) ALPHA TESTING:**

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing

# 11)BETA TESTING:

Beta testing is the testing which is done by end users, a team outside development, or publicly releasing full pre-version of the product which is known as beta version. The aim of beta testing is to cover unexpected errors. It falls under the class of black box testing.

**12)WHITE-BOX TESTING:**

**White-box testing** (also known as **clear box testing**, **glass box testing**, **transparent box testing** and **structural testing**) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. [in-circuit testing](http://en.wikipedia.org/wiki/In-circuit_test) (ICT).

While white-box testing can be applied at the [unit](http://en.wikipedia.org/wiki/Unit_testing), [integration](http://en.wikipedia.org/wiki/Integration_testing) and [system](http://en.wikipedia.org/wiki/System_testing) levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include:

* [API](http://en.wikipedia.org/wiki/Application_programming_interface) testing (application programming interface) – testing of the application using public and private APIs
* [Code coverage](http://en.wikipedia.org/wiki/Code_coverage) – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
* [Fault injection](http://en.wikipedia.org/wiki/Fault_injection) methods – intentionally introducing faults to gauge the efficacy of testing strategies
* [Mutation testing](http://en.wikipedia.org/wiki/Mutation_testing) methods
* [Static testing](http://en.wikipedia.org/wiki/Static_testing) methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important [function points](http://en.wikipedia.org/wiki/Function_points) have been tested.[[22]](http://en.wikipedia.org/wiki/Software_testing#cite_note-22) Code coverage as a [software metric](http://en.wikipedia.org/wiki/Software_metric) can be reported as a percentage for:

* *Function coverage*, which reports on functions executed
* *Statement coverage*, which reports on the number of lines executed to complete the test

**13) BLACK-BOX TESTING:**

[http://upload.wikimedia.org/wikipedia/commons/thumb/f/f6/Blackbox.svg/200px-Blackbox.svg.png](http://en.wikipedia.org/wiki/File:Blackbox.svg)

Black box diagram

**Black-box testing** treats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it.[[23]](http://en.wikipedia.org/wiki/Software_testing#cite_note-REF1-23) Black-box testing methods include: [equivalence partitioning](http://en.wikipedia.org/wiki/Equivalence_partitioning), [boundary value analysis](http://en.wikipedia.org/wiki/Boundary_value_analysis), [all-pairs testing](http://en.wikipedia.org/wiki/All-pairs_testing), [state transition tables](http://en.wikipedia.org/wiki/State_transition_table), [decision table](http://en.wikipedia.org/wiki/Decision_table) testing, [fuzz testing](http://en.wikipedia.org/wiki/Fuzz_testing), [model-based testing](http://en.wikipedia.org/wiki/Model-based_testing), [use case](http://en.wikipedia.org/wiki/Use_case) testing, [exploratory testing](http://en.wikipedia.org/wiki/Exploratory_testing) and specification-based testing.

**Specification-based testing** aims to test the functionality of software according to the applicable requirements.[[24]](http://en.wikipedia.org/wiki/Software_testing#cite_note-24) This level of testing usually requires thorough [test cases](http://en.wikipedia.org/wiki/Test_case) to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be [functional](http://en.wikipedia.org/wiki/Functional_testing) or [non-functional](http://en.wikipedia.org/wiki/Non-functional_testing), though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.[[25]](http://en.wikipedia.org/wiki/Software_testing#cite_note-25)

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight."[[26]](http://en.wikipedia.org/wiki/Software_testing#cite_note-26) Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested.

SYSTEM ARCHITECTURE:

Fuzzy keyword set cloud server

C:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0299125.wmfC:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0299125.wmf

C:\Program Files\Microsoft Office\MEDIA\OFFICE12\AutoShap\BD18252_.wmfC:\Program Files\Microsoft Office\MEDIA\OFFICE12\AutoShap\BD18252_.wmfC:\Program Files\Microsoft Office\MEDIA\OFFICE12\AutoShap\BD18252_.wmf

Index sending request

C:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0292020.wmfC:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0195384.wmf

owner  
 files C:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0299125.wmfC:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0299125.wmfencrypted files fileC:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0299125.wmf retrieval C:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0195384.wmf

REFERENCES:

[1] Google, “Britney spears spelling correction,” Referenced online at http:

//www.google.com/jobs/britney.html, June 2009.

[2] M. Bellare, A. Boldyreva, and A. O’Neill, “Deterministic and efficiently

searchable encryption,” in *Proceedings of Crypto 2007, volume 4622 of*

*LNCS*. Springer-Verlag, 2007.

[3] D. Song, D. Wagner, and A. Perrig, “Practical techniques for searches

on encrypted data,” in *Proc. of IEEE Symposium on Security and*

*Privacy’00*, 2000.

[4] E.-J. Goh, “Secure indexes,” Cryptology ePrint Archive, Report

2003/216, 2003, http://eprint.iacr.org/.

[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, “Public key

encryption with keyword searcht log,” in *Proc. of 11th Annual Network*

*and Distributed System*, 2004.

[7] Y.-C. Chang and M. Mitzenmacher, “Privacy preserving keyword

searches on remote encrypted data,” in *Proc. of ACNS’05*, 2005.

[8] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, “Searchable

symmetric encryption: improved definitions and efficient constructions,”

in *Proc. of ACM CCS’06*, 2006.