1.1 SYSTEM STUDY

1.1.1 Existing System

Recently, the importance of fuzzy search has received attention in the context of plaintext searching in information retrieval community. This problem is addressed 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. However, this trivial construction suffers from the dictionary and statistics attacks and fails to achieve the search privacy. Also, this method does not ensure keyword privacy. Since keywords contain important information related to data files, this is a big demerit of the plain text searching scheme. Goal of encryption also cannot be achieved using plain text searching scheme since encrypted data in cloud computing does not support this scheme. Hence security of data is not guaranteed at all.

1.1.2 Proposed System

Even though data files are encrypted, the cloud server may try to derive sensitive information from users' search requests while performing keyword-based search over Cloud data. 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. Here when designing fuzzy

keyword search scheme, we will follow the security definition deployed in the traditional searchable encryption.

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. 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.

The proposed system was found to feasible in all these different classifications of the study and this led to the design of the system. Feasibility study also includes comparing the existing system and the proposed system. here the proposed system is desired to have some new facilities, which are not in existing system and must overcome at least one disadvantage of the existing one.

2. SYSTEM ANALYSIS & DESIGN

2.1 MODULAR DESIGN

It consists of six modules. They are:

- Cloud configuration

- Database design
- Fuzzy keyword set construction
- Data encryption
- Fuzzy keyword search
- File management

2.1.1 **Cloud Configuration**

Cloud computing can be loosely defined as using scalable computing resources provided as a service from outside our environment on a pay-per-use basis. We can access any of the resources that live in the “cloud” across the Internet and don’t have to worry about computing capacity, bandwidth, storage, security, and reliability. Here we use Linux server and an open source framework named Apache Hadoop, which will be build onto the Linux server to establish the cloud computing framework.

3.1.1 Cloud setup steps

1. Root account is created

Ubuntu is one of the few Linux distributions that will not enable the root account. To enable root login on Ubuntu press Crtl+Alt+T on keyboard to open terminal. When it opens, run the command below to create a root password.

$ sudo password

Run the following command to enable other login.

$ sudo sh –c ‘echo “greeter-show-manual-login=true” >> /etc/lightdm/lightdm.conf’

2. Hadoop Configuration

Copy hadoop to the home folder.

Perform the following steps in root/hadoop/conf/

1.Coresite.xml:

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://hostname:9000</value>

</property>

</configuration>

2.Specify the master and slaves appropriately.

In our system we consider the same system as master and the slave.

Perform the following steps in home/root/etc/

1.Hosts:

Set the hostname and IP address.

2.Hostname:

Set the hostname.

3. SSH Installation

For hadoop to work we have to use SSH to communicate between two machines.

To get installed run the following command:

$sudo apt-get install openssh-server

Issue these commands to generate a key pair and copy the public key part into authorized keys file and private part into known hosts file:

$ssh-keygen –t rsa

$cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys

$cat /root/.ssh/id_rsa.pub >> ~/.ssh/known_hosts

It is the private key, public key pair that makes SSH without passwords possible. Enabling SSH in a single system does not make much sense, but it is needed for testing.

3.1.2 Database design

The database contains the all the information that has to be recorded for future reference. The database of **FUNKEYZZ** consists of five tables namely login table, user details table, file details table, tag table, log table.

The login table records the login details of users which includes user name, password, user id and user type. The user id will be unique for each users.

The user details table stores the registration details such as name, email id, city, mobile number, city. It also records the approve status of the user.

The file details table contains the details of the files uploaded by the users. The fields of the table includes file id, file name, file type, active status of file, user id, tag file.

The tag table stores file id and file tags.

The log table contains the log details of the users which includes file id, user id, file action and action date.

Table 3.1.2.1

FIELD NAME	TYPE
User id	Int
Username	Varchar
Password	Varchar
User type	Varchar

Login table

Table 3.1.2.2

FIELD NAME	TYPE
User id	Int
Name	Varchar
Emailid	Varchar
Mobile Number	Varcher
City	Varchar
Approve Status	Int

User Details Table

Table 3.1.2.3

FIELD NAME	TYPE
File id	Int
Filename	Varchar
File type	Int
Active status	Int
Tag file	Varchar
User id	Int

File Details Table

Table 3.1.2.4

FIELD NAME	TYPE
File id	Int
File tag	Varchar

Tag Table

Table 3.1.2.5

FIELD NAME	TYPE
File id	Int
User id	Int
File action	Int
Action date	Varchar

Log Table

3.1.3 **Fuzzy keyword set construction**

An advanced wildcard based technique is used for fuzzy keyword set construction, which is effective with regard to both storage and search efficiency. In this method we use a wildcard to denote edit distance at the same position. The edit distance ed(w1,w2) between two words w1 and w2 is the number of operations required to transform one of them into the other.

The three primitive operations are:

1) Substitution: changing one character to another in a word.

2) Deletion: deleting one character from a word.

3) Insertion: inserting a single character into a word.

The wildcard-based fuzzy set of wi with edit distance d is denoted as S(wi,d)={S'(wi,0) ,S'(wi,1), ⋯, S'(wi,d)}, where S'(wi,t) denotes the set of words w'i with t wildcards. Here each wildcard represents an edit operation on wi. For example, for the keyword CASTLE with the pre-set edit distance 1, its wildcard-based fuzzy keyword set can be constructed as S(CASTLE,1) = {CASTLE, *CASTLE, *ASTLE, C*ASTLE, C*STLE, ⋯, CASTL*E, CASTL*, CASTLE*}. The total number of variants on CASTLE constructed in this way is only 13 + 1, which is storage efficient.

3.1.4 **Data encryption**

The cloud server may not be fully trusted. It follows that sensitive data usually should be encrypted prior to outsourcing for data privacy and combating unsolicited accesses. The data encryption is performed using AES(Advanced Encryption

Standard) algorithm. AES algorithm is efficient and secure. Details of AES algorithm is given below:

3.1.4.1 AES algorithm

We use AES algorithm for encrypting the data contained in the files that are uploaded by the registered users. The Advanced Encryption Standard (AES) is a symmetric block cipher whose structure is quite complex when compared to other cryptographic algorithms. The cipher takes a plaintext block size of 128 bits, or 16 bytes. The key length can be 16, 24 or 32 bytes (128, 192 or 256 bits).The algorithm is referred to as AES-128, AES-192, or AES-256, based on the key length.

The input to the encryption and decryption algorithms is a single 128-bit block. This block is depicted as a 4 x 4 matrix of bytes. This block is then copied into the State array, which is modified at the stage of encryption or decryption. After the final stage, State is copied to an output matrix. At the same time, the key provided is depicted in a square matrix of bytes. This key is then expanded into an array of key schedule words. Each word is four bytes.

The encryption process contains N rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key, 12 rounds for a 24-byte key, and 14 rounds for a 32-byte key. The first N-1 rounds consist of four distinct transformation functions: Sub Bytes, Shift Rows, Mix Columns, and Add Round Key. The final round contains only three transformations (Sub Bytes, Shift

Rows, Add Round Key), and there is an initial transformation (Add Round Key) before the first round. Each transformation takes one or more 4 x 4 matrix as input and produces a 4 x 4 matrix as output. Also, the key expansion function generates N+ 1 round key, each of which is a distinct 4 x4 matrix. Each round key serves as one of the inputs to the Add Round Key transformation in each round.

Sub bytes or Substitute bytes is a simple table lookup. In the case of encryption, this uses an S-box to perform a byte – by – byte substitution of the block. S-

box is a 16 x 16 matrix of byte values that contains a permutation of all possible 256 8-bit values. Each individual byte of State is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value. These row and column values serve as indexes into the S-box to select a unique 8-bit output value. Now, in the case of decryption inverse substitute byte transformation is used. This makes use of inverse S-box. The inverse S-box is constructed by applying the inverse of the above quoted transformations.

The next transformation performed is Shift Row transformation. The first row of State is not alerted. For the second row, a byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed. For the fourth row, a 3-byte circular left shift is performed. For decryption, inverse Shift Row transformation is performed. This performs the circular shifts in the opposite direction for each of the last three rows, with a 1-byte right shift for the second row and so on.

Now, mix column transformation is performed. While performing encryption, the forward mix column transformation is performed. This operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in that column. For decrypting, inverse mix column transformation is performed. This performs just opposite operation

performed by forward mix column transformation. The AES document describes another way of characterizing the Mix Column transformation, which in terms of polynomial arithmetic. Each column of state is considered as a four term polynomial and then multiplied by some fixed polynomial say, a[x].

The last transformation that is performed in all the rounds is the Add Round Key transformation. For encryption, forward Add Round Key transformation is used. In this, the 128 bits of State are bitwise XORed with the 128 bits of the round key. The operation can be viewed as a column wise operation between the 4 bytes of a State column and one word of the round key. It can also be viewed as a byte-level operation.

For decryption, inverse Add Round Key transformation is performed. This is identical to forward Add Round Key operation, because the XOR operation is its own reverse.

3.1.5 Fuzzy keyword search

The fuzzy keyword search is implemented inorder to overcome the shortcomings of traditional searching schemes. 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. 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. Here we formalize and solve the problem of effective fuzzy keyword search over encrypted cloud data.

The fuzzy keyword search takes place in accordance with the constructed fuzzy keyword set. The uploaded files can be retrieved back if the

search key matches any of the keyword that was constructed using the wildcard based technique.

3.1.6 File manipulation

3.1.6.1 File uploading

In our system, the file uploaded is stored in the cloud along with the constructed fuzzy keyword set of the same. Data encryption is done before outsourcing which ensures data security.

3.1.6.2 Accessing the file

A fuzzy keyword search mechanism is used to access a file. A fuzzy keyword set for each keywords is constructed. The fuzzy keyword set contains not only the exact keywords but also the ones that slightly differs. The file will be retrieved back if the users' searching input matches a keyword in the fuzzy keyword set.

3.2 SYSTEM ANALYSIS

System analysis emphasis on understanding the details of the proposed system and then deciding whether proposed system is desirable or not and whether the system needs improvements. System analysis is the process of investigating a system, identifying problems, and using the information to recommend improvements to the system. Funkeyzz enables effective fuzzy

keyword search over encrypted cloud data. As cloud computing becomes prevalent more and more data has been centralized into cloud.

Cloud Computing is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility over a network (typically the internet). Cloud computing is a marketing term for technologies that provide computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. Cloud computing providers deliver applications via the internet, which are accessed from web browsers and desktop and mobile apps, while the business software and data are stored on servers at a remote location.
Cloud computing exhibits the following key characteristics:-

- Application programming interface(API):-accessibility to software that enables machines to interact with cloud software in the same way that user interface facilities interaction between humans and computers.

- Cost :- is claimed to be reduced and in a public cloud delivery model capital expenditure is converted to operational expenditure. This as infrastructure is typically provided by a third-party and does not need to be purchased for one-time or infrequent intensive computing tasks.

- Device and location independence:- enable users to access systems using a web browser regardless of their location or what device they are using (e.g., PC, mobile phone).

- Centralization:- of infrastructure in locations with lower costs (such as real estate, electricity, etc.)

- Reliability:-is improved if multiple redundant sites are used, which makes well-designed cloud computing suitable for business continuity and disaster recovery.
- Performance:- is monitored and consistent and loosely coupled architectures are constructed using web services as the system interface.
- Security:-could improve due to centralization of data increased security-focused resources, etc., Security is often as good as or better than other traditional systems, in part because providers are able to devote resources to solving security issues that many customers cannot afford.

Funkeyzz established a cloud environment using Apache Hadoop. Apache Hadoop is a software framework that supports data-intensive distributed applications under a free license. It enables applications to work with thousands of nodes and data. Hadoop was inspired by Google's Map Reduce and Google File System(GFS) papers. Hadoop is a top level Apache project being built and used by a global community of contributors, written in the Java programming language. Yahoo! Has been the largest contributor to the project, and uses Hadoop extensively across its businesses. Hadoop was created by Doug Cutting.

The Hadoop framework transparently provides applications both reliability and data motion. Hadoop implements a computational paradigm named Map/Reduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster. In addition, it provides a distributed file system (HDFS) that stores data on the computer nodes, providing very high aggregate bandwidth across the cluster. Both MapReduce and the Hadoop Distributed File System are designed so that node failures are automatically handled by the framework.

Fuzzy keyword search greatly enhances system usability by returning the matching files when users searching input does not match the predefined keywords. Funkeyzz ensures data security through data encryption.

3.3 DATAFLOW DIAGRAM

LEVEL 0

LEVEL 1

LEVEL 2.0: Administrator Login

LEVEL 2.1: User Login

LEVEL 2.2: Owner Login

4. SYSTEM SPECIFICATION

System requirement specification includes what exactly the system may be. It specifies what the system is, what its input must be. It does not specifies what the internal operation is. Here the exact input and output cannot be specified in a single word because it is application software in which users can register, registered users can upload any text files, login and search for files using preassigned keywords, view or delete them at any time whenever needed. Here in our project, the inputs are the keywords that are entered by the registered users and the output is the file that matches the keywords while keyword searching is considered. But if file uploading is considered, inputs are the uploading files and so on.

4.1 SOFTWARE SPECIFICATION

Software selection is an important work in a project development cycle. Software must be selected in accordance with the application and the latest technology available.

Front end: JAVA

Java is a programming language originally developed by James Gosling at Sun Microsystems. The language derives much of its syntax from C and C++. Java applications are typically compiled to byte code(class file) that run on any java Virtual Machine(JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. Java is currently one of the most popular programming languages in use, particularly for client-server web applications.

Back end: MySQL

MySQL is the world's most used open source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software. LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

IDE: Eclipse

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other than programming languages including Ada, C, C++, COBOL etc.

Operating system: Ubuntu 12.10

Ubuntu is a computer operating system based on the Debian Linux distribution and distributed as free and open source software, using its own desktop environment.

Server: Apache Tomcat

Apache Tomcat is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JSP specifications from Oracle Corporation, and provides a “pure Java” HTTP web server environment for Java code to run.

4.2 HARDWARE SPECIFICATION

Processor: Intel Pentium 4

Memory: 2 GB

Hard disk: 320 GB

Networking: 100 Mbps

Monitor: 15” Color Monitor

Mouse: Standard 3 Button Mouse

Keyboard: 104 Keys Standard Keyboard

5. IMPLEMENTATION

5.1IMPLEMENTATION PLAN

Implementation of the system refers to the final installing of the package in its real environment, to the satisfaction of the intended system and the operation of the system. Proper guidance should be imparted to the user so that he is comfortable in using the application. Implementation is done in different sub phases.

5.2 IMPLEMENTATION PROCEDURE

The implementation procedure involves careful planning, investigation of the current system and the constraints on implementation, design of methods to achieve the changeover. Apart from planning, major tasks of preparing the implementation procedures are education and training to the users. The more complex the system being implemented, the more involved are the system's analyst.

The implementation process begins with preparing a plan for the implementation of the system. According to the plan, the activities have been carried out; discussions have been made regarding the equipment and resources. According to the above plan, the activities have been made regarding the equipment and resources. According to the above plan, the necessary equipment has to be acquired to implement the new system.

In our project, the front end is a JSP page for users to register, login, upload files, search for uploaded files using keywords and to view and delete the files while the back end is database used as a mass storage system and a fast index. The

project uses JAVA (Eclipse) for developing the front end and MY SQL for developing the back end.

5.2.1 FRONT END IMPLEMEMENTATION

We made use of Java and JSP in order to implement the front end. The editor used is 'Eclipse'. Eclipse is a multi-language software development environment (IDE). It is written mostly in java. Eclipse supports development for Tomcat, Glass Fish and many other servers. We made use of Apache Tomcat for **FUNKEYZZ**. Apache Tomcat (or simply Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements in the Java Servlet and the Java Server Pages (JSP) specifications.

Java language derives much of its syntax from C and C++. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is currently one of the most popular programming languages in use.

In our project, we made use of JSP with Java. Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML or other document types. Architecturally, JSP may be viewed as a high-level abstraction of JAVA servlets. JSP's are translated into servlets at runtime; each JSP's servlet is cached and re-used until the original JSP is modified.

5.2.2 BACK END IMPLEMENTATION

In our project, we use Hadoop for creating cloud environment. Apache Hadoop is a framework for running applications on large cluster built of commodity hardware. The data uploaded to the cloud are encrypted before outsourcing. It is performed using AES algorithm. The Advanced Encryption Standard (AES) is a symmetric block cipher whose structure is quite complex when compared to other

cryptographic algorithms. We use a database for recording data. The database is implemented using MySQL, an open source relational DBMS. MySQL is the world's most used open source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. Free-software-open source projects that require a full-featured database management system often use MySQL.

6. **TESTING**

The testing phase is the final and important phase of any system to be success. Testing is the stage of implementation, which is aimed at ensuring

that the application works accurately and effectively. Testing ensures that if all components of the application are correct, the product will be successful. Module testing was conducted and modules are tested separately. This test was carried out during programming stage itself. Each module has been found to work satisfactorily as the expected output from the module is obtained. The modules have been tested for robustness. Exceptions have been handled using try/catch blocks in each modules so as to avoid abnormal termination of program and appropriate error messages have been given.

1.Unit testing

Unit testing of software applications is done during the development (coding) of an application. The objective of unit testing is to isolate a section of code and verify its correctness. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. Here unit testing is done by writing another section of code in the application just to test the function. It was then commented out and finally removed the test code from the code of Funkeyzz.

2.Integration testing

In Integration testing, individual software modules are integrated logically and tested as a group. Although each software module is unit tested, defects may still exist. Some of them can be identified through integration testing. Sample integration test case of Funkeyzz:

Test case 1:

Objective: Check the interface link between the registration and login module.

Description: Enter registration details and click on the register button.

Expected result: To be directed to the login page.

Test result: Pass.

Test case 2:

Objective: Check the interface link between the login and home module.

Description: Enter login details and click on the login button.

Expected result: To be directed to the home page.

Test result: Pass.

3.System testing

Unlike Integration testing, which focuses on data transfer amongst modules, system testing checks complete end to end scenarios, as the way a customer would use the system. Funkeyzz has undergone system testing and found that all the requirements and functionalities that were mentioned during the design phase is met.

Apart from Functional, NON-FUNCTIONAL requirements are also checked during system testing. Non-functional requirements include performance, reliability etc. The performance of Funkeyzz is quite high.

7. CONCLUSION

Here we formalize and solve the problem of supporting efficient yet privacy-preserving fuzzy search for achieving effective utilization of remotely stored encrypted data in Cloud Computing. We designed an advanced technique (i.e., wildcard-based technique) to construct the storage-efficient fuzzy keyword sets by exploiting a significant observation on the similarity metric of edit distance. Based on the constructed fuzzy keyword sets, we implemented an efficient fuzzy keyword search scheme. Our solution is secure and privacy-preserving, while correctly realizing the goal of fuzzy keyword search.

7.1 FUTURE ENHANCEMENTS

Some of the future enhancements includes:

- Search semantics that takes into consideration conjunction of keywords, sequence of keywords, and even the complex natural language semantics to produce highly relevant search results
- Search ranking that sorts the searching results according to the relevance criteria.

8. BIBLIOGRAPHY

[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 search,” in *Proc. of EUROCRYP’04*, 2004.

[6] B. Waters, D. Balfanz, G. Durfee, and D. Smetters, “Building an encrypted and searchable audit 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.

[9] D. Boneh and B. Waters, “Conjunctive, subset, and range queries on encrypted data,” in *Proc. of TCC’07*, 2007, pp. 535–554.

[10] F. Bao, R. Deng, X. Ding, and Y. Yang, “Private query on encrypted data in multi-user settings,” in *Proc. of ISPEC’08*, 2008.

[11] C. Li, J. Lu, and Y. Lu, “Efficient merging and filtering algorithms for approximate string searches,” in *Proc. of ICDE’08*, 2008.

[12] A. Behm, S. Ji, C. Li, , and J. Lu, “Space-constrained gram-based indexing for efficient approximate string search,” in *Proc. of ICDE’09*.

APPENDIX A SCREEN SHOTS

Fig 1. Home Page

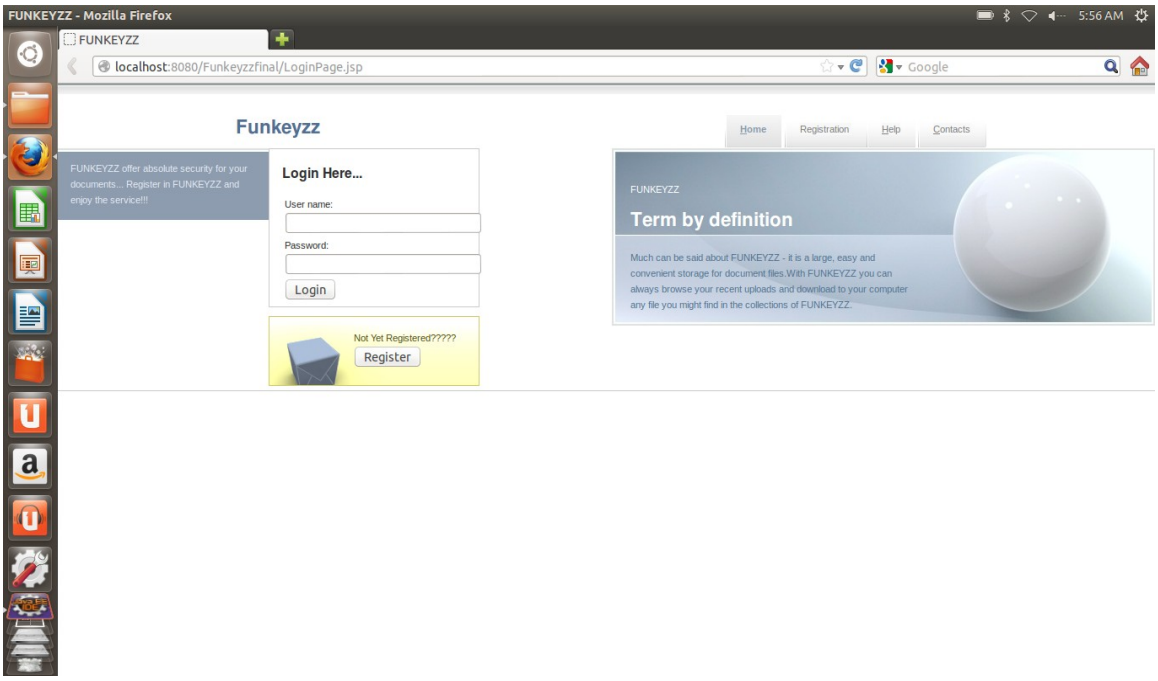


Fig 2. Registration Page

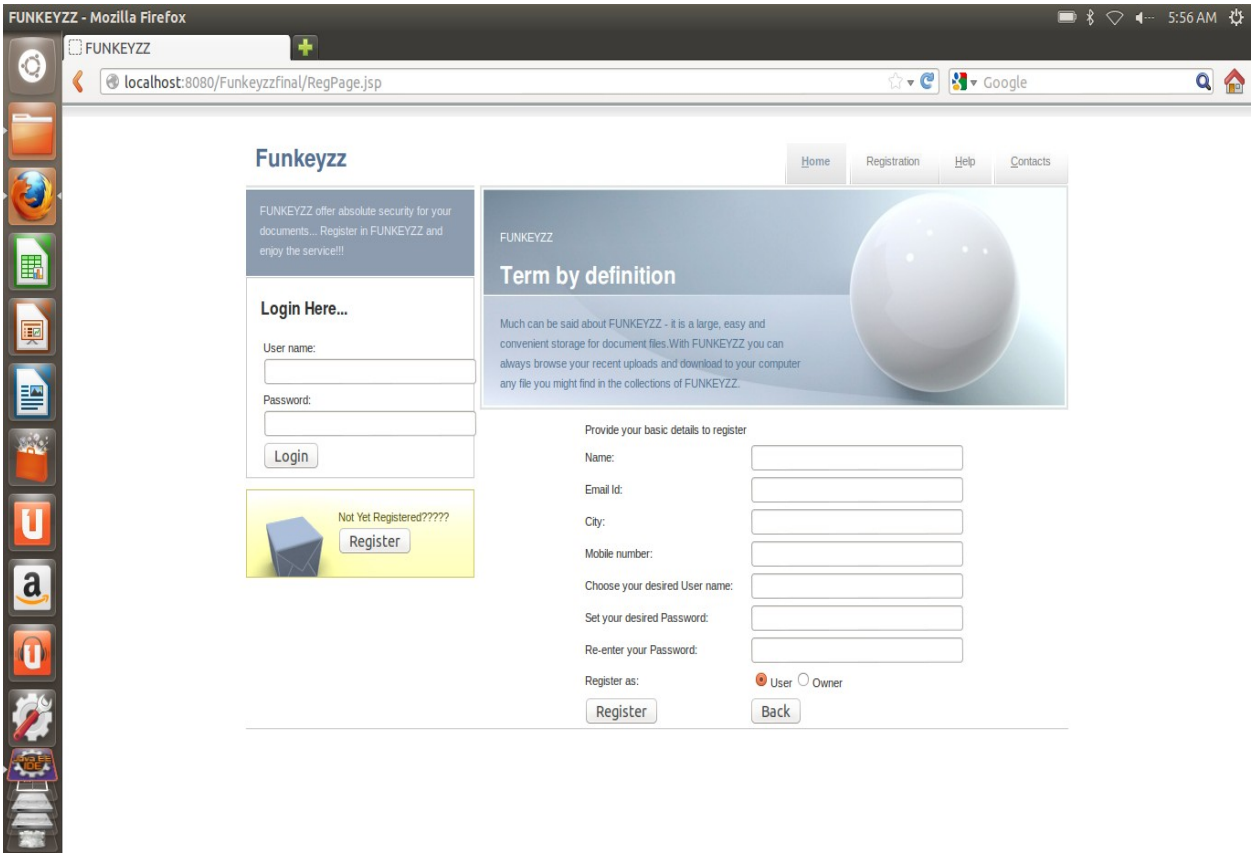


Fig 3. Admin Home Page

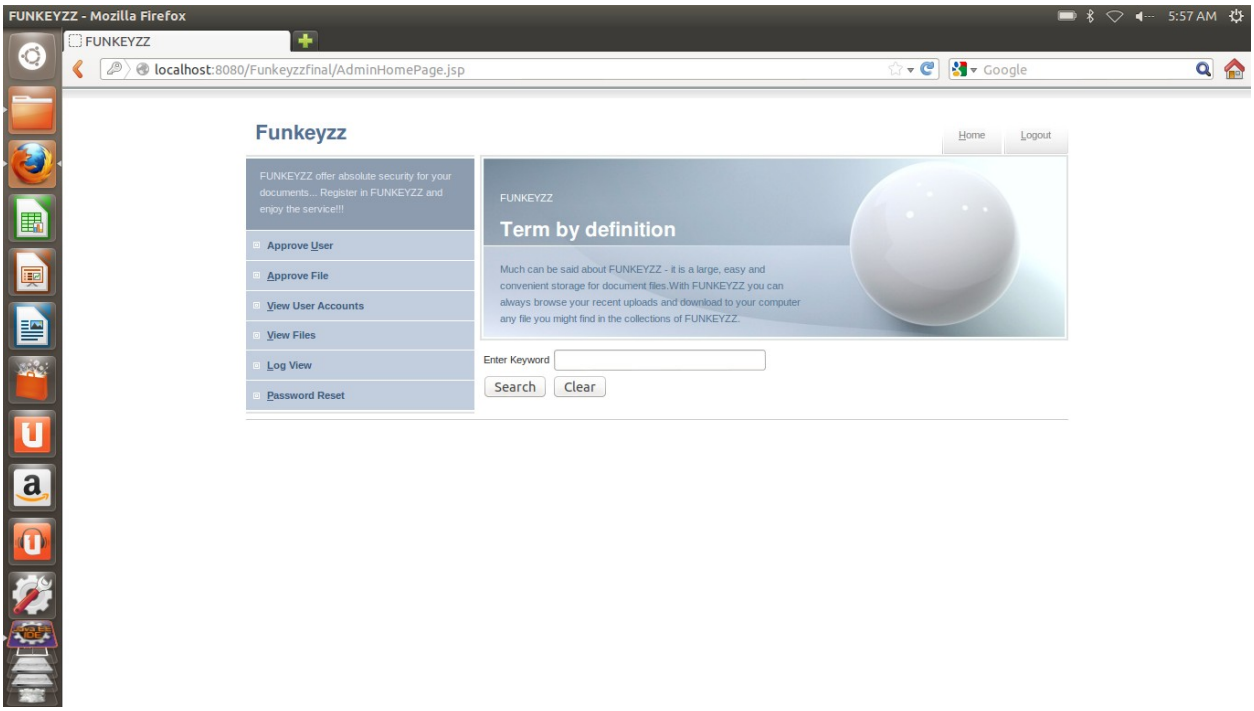


Fig 4.User Home Page

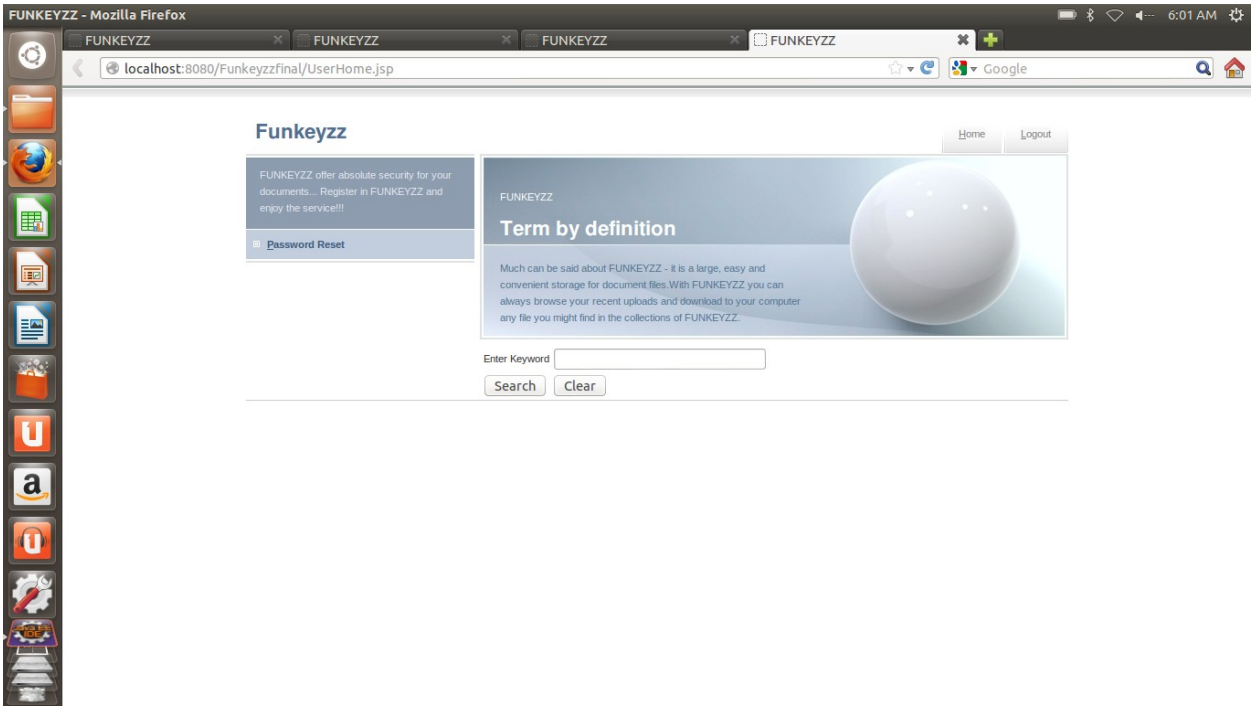


Fig 5 Owner Home Page

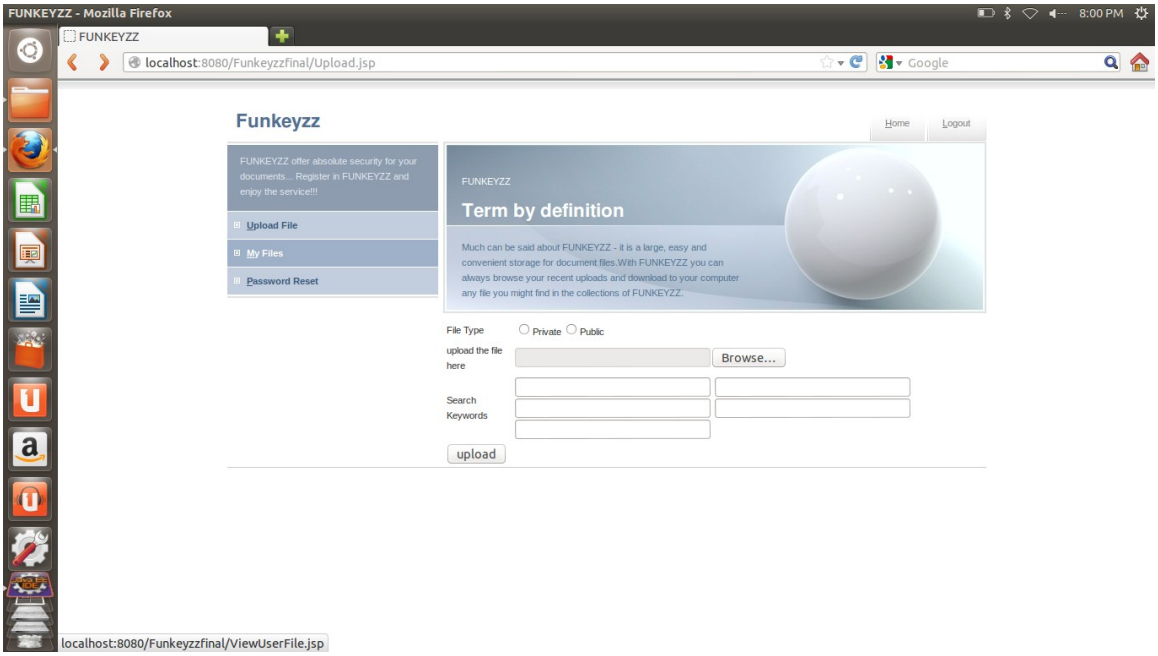
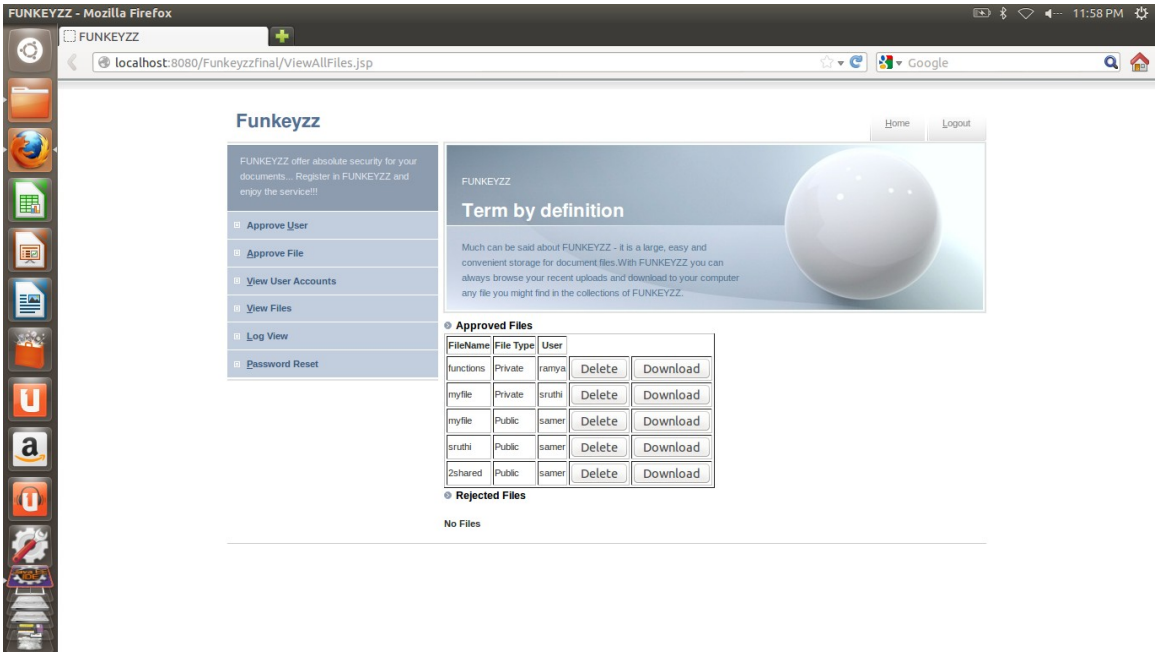


Fig 6 View File



APPENDIX B CONSTRUCTION OF EFFECTIVE FUZZY KEYWORD SET IN CLOUD

The key idea behind secure fuzzy keyword search is two-fold: 1) building up fuzzy keyword sets that incorporate not only the exact keywords but also the ones differing slightly due to minor typos, format inconsistencies, etc.; 2) designing an efficient and secure searching approach for file retrieval based on the resulted fuzzy keyword sets.

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 have 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.

Edit Distance

There are several methods to quantitatively measure the string similarity. Edit distance is important among them. The edit distance ed(w1, w2) between two words w1 and w2 is the number of operations required to transform one of them into the other. The three primitive operations are 1) Substitution: changing one character to another in a word; 2) Deletion: deleting one character from a word; 3) Insertion: inserting a single

character into a word. Given a keyword w, we let Sw,d denote the set of words w_ satisfying ed(w,w_) ≤ d for a certain integer d.

Wildcard-based Fuzzy Set Construction

In the straightforward approach, all the variants of the keywords have to be listed even if an operation is performed at the same position. Based on the above observation, we use a wildcard to denote edit operations at the same position. The wildcard-based fuzzy set of wiith edit distance d is denoted as S(wi,d)={S'(wi,0), S'(wi,1), · · · , S'(wi,d)}, where S'(wi,t) denotes the set of words w'I with t wildcards. Note each wildcard represents an edit operation on wi. For example, for the keyword CASTLE with the pre-set edit distance 1, its wildcard-based fuzzy keyword set can be constructed as SCASTLE,1 = {CASTLE, *CASTLE, *ASTLE, C*ASTLE, C*STLE, · · · , CASTL*E, CASTL*, CASTLE*}. The total number of variants on CASTLE constructed in this way is only 13 + 1, instead of 13 × 26 + 1 as in the above exhaustive enumeration approach when the edit distance is set to be 1. Generally, for a given keyword wiwith length _, the size of Swi,1 will be only 2_ + 1 + 1, as compared to (2_ + 1) × 26 + 1 obtained in the straightforward approach. The larger the pre-set edit distance, the more storage overhead can be reduced: with the same setting of the example in the straightforward approach, the proposed technique can help reduce the storage of the index from 30GB to approximately 40MB. In case the edit distance is set to be 2 and 3, the size of Swi,2 and Swi,3 will be C1 _+1+C1 _· C1 _ +2C2 _+2 and C1 _ + C3 _ + 2C2 _ + 2C2 _· C1 _ . In other words, the number is only O(_d) for the keyword with length _ and edit distance d.

The Efficient Fuzzy Keyword Search Scheme

Based on the storage-efficient fuzzy keyword sets, an efficient and effective fuzzy keyword search scheme can be constructed. The scheme of the fuzzy keyword search goes as follows: 1) To build an index for wi with edit distance d, the data owner first constructs a fuzzy keyword set S(wi,d) using the wildcard based technique. Then he computes trapdoor set {Tw l} for each w l

∈S(wi,d) with a secret key sk shared between data owner and authorized users.

The data owner encrypts FIDwi as Enc(sk, FIDwiwi). The index table {({Twl }w l

∈S(wi,d) , Enc(sk, FIDwiwi))}wi∈W and encrypted data files are outsourced to the

cloud server for storage; 2) To search with (w, k), the authorized user computes

the trapdoor set {Tw}w∈S(w,k) , where S(w,k) is also derived from the wildcard-

based fuzzy set construction. He then sends {Tw}w∈S(w,k) to the server; 3) Upon

receiving the search request {Tw}w∈S(w,k), the server compares them with the

index table and returns all the possible encrypted file identifiers {Enc(sk, FIDwiwi)} according to the fuzzy keyword definition in section III-D. The user decrypts the returned results and retrieves relevant files of interest. In this construction, the technique of constructing search request for w is the same as the construction of index for a keyword. As a result, the search request is a trapdoor set based on Sw,k, instead of a single trapdoor as in the straightforward approach. In this way, the searching result correctness can be ensured.

APPENDIX C SECURITY ANALYSIS

In this section, we analyze the correctness and security of the proposed fuzzy keyword search scheme. At first, we show the 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 the request of w, all of the keywords {wi} will be returned if and only if ed(w,wi) ≤ k.

The proof of this Theorem can be reduced to the following Lemma:

Lemma 1:

The intersection of the fuzzy sets S(wi,d) and S(w,k) for wi and w is not empty if and only if ed(w,wi) ≤ k. Proof: First, we show that S(wi,d) ∩ S(w,k) is not empty when ed(w,wi) ≤ k. To prove this, it is enough to find an element in S(wi,d) ∩ S(w,k). Let w = a1a2 · · · as and wi = b1b2 · ·· bt, where all these ai and bj are single characters. After ed(w,wi) edit operations, w can be changed

towi according to the definition of edit distance. Let w∗ = a∗1a∗ 2 · · · a∗m, where

a∗i = aj or a∗i = ∗ if any operation is performed at this position. Since the edit

operation is inverted, from wi, the same positions containing wildcard at $w*$ will be performed. Because ed(w,wi) ≤ k, $w*$ is included in both S(wi,d) and S(w,k), we get the result that S(wi,d) ∩ S(w,k) is not empty.

Next, we prove that S(wi,d) ∩ S(w,k) is empty if ed(w,wi) > k. The proof is given by reduction. Assume there exists an $w*$ belonging to S(wi,d) ∩ S(w,k). We will show that ed(w,wi) ≤ k, which reaches a contradiction. First, from the assumption that $w* \in S(wi,d) \cap S(w,k)$, we can get the number of wildcard in $w*$, which is denoted by $n*$, is not greater than k. Next, we prove that ed(w,wi) ≤ $n*$. We will prove the

inequality with induction method. First, we prove it holds when $n* = 1$. There are nine cases should be considered: If $w*$ is derived from the operation of deletion from both wi and w, then, $ed(wi,w) \leq 1$ because the other characters are the same except the character at the same position. If the operation is deletion from wi and substitution from w, we have $ed(wi,w) \leq 1$ because they will be the same after at most one substitution from wi. The other cases can be analyzed in a similar way and are omitted. Now, assuming that it holds when $n* = \gamma$, we need to prove it also holds when $n* = \gamma + 1$. If ˆ $w* = a*1a*2 \cdots a*n \in S(wi,d) \cap S(w,k)$, where $a*i = aj$ or $a*i = *$. For a wildcard at position t, cancel the underlying operations and revert it to the original characters in wi and w at this position.

Assume two new elements $w*i$ and $w*$ are derived from them respectively. Then perform one operation at position t of $w*i$ to make the character of wi at this position be the same with w, which is denoted by w i. After this operation, $w*i$ will be changed to $w*$, which has only k wildcards. Therefore, we have $ed(wi,w) \leq \gamma$ from the assumption. We know that $ed(wi,w) \leq \gamma$ and $ed(wi,wi) = 1$, based on which we know that $ed(wi, w) \leq \gamma + 1$. Thus, we can get $ed(w,wi) \leq n*$. It renders the contradiction $ed(w,wi) \leq k$ because $n* \leq k$. Therefore, $S(wi,d) \cap S(w,k)$ is empty if $ed(w,wi) > k$.

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 search-able encryption scheme fails to achieve the index privacy against the indistinguishably 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(·) is a

pseudo-random function such that f(·) is equal to f(sk, ·) or a random function. A has an access to an oracle O f(·) that takes as input secret value x and returns f(x). Upon receiving any request of the index computation, A answers it with request to the oracle O f(·). After making these trapdoor queries, the adversary

outputs two challenge keywords w∗0 and w∗ 1 with the same length and edit

distance, which can be relaxed by adding some redundant trapdoors. A picks one

random b ∈ {0, 1} and sends w∗ b to the challenger. Then, A is given a challenge

value y, which is either computed from a pseudo-random function f(sk, ·)or a

random function. A sends y back to A, who answers with b $\in$ {0, 1}. Suppose A guesses b correctly with non negligible probability, which indicates that the value is not randomly computed. Then, A makes a decision that f(·) is a pseudo-random function. As a result, based on the assumption of the indistinguishably of the pseudo-random function from some real random function, A at most guesses b correctly with approximate probability 1/2. Thus, the search privacy is obtained.