A

Report on

<<Title of the project>>

Thesis submitted in partial fulfillment of the requirement for the award of the degree of

**Master of Technology**

**In**

**<<Branch Name>>**

Submitted By

<< Name of the Student >>

<< Hall Ticket No >>

*Under the Esteemed Guidance of*

<< Guide Name >>

<< Designation >>

<< College Logo >>

Department of << Department name >>,

<< College name (Upper case only) >>

(Affiliated to Jawaharlal Nehru Technological University)

<<Distinct Name >>-Pin code, << State name>> INDIA

2015

**<< College Name >>**

**(Affiliated to Jawaharlal Nehru Technological University)**

**<< Distinct Name >>- << Pin code >>. << State name >>., INDIA**

**CERTIFICATE**

**<< College Logo >>**

This is to certify that the Thesis entitled “**<<**title of the project**>>”** is being submitted by << student name>>bearing **<<**Roll number**>>** in partial fulfilment of the requirement for the award of the degree of M.Tech in << **Branch Name >>** to the <<College Name>>, <<University Name >> is a record of bonafide work carried out by him/her under my guidance. The results presented in this thesis have been verified and are found to be satisfactory. The results embodied in this thesis have not been submitted to any other University for the award of any other degree or diploma.

**Project Guide**  **Head of the Department**

**<<Guide Name>>, <<HOD Name>>**

**<<Designation>> <<Designation>>**

**EXTERNAL PRINCIPAL**

**Next page:** This page is required for the candidates who pursue the project work outside (other

organization), under the supervision of an **EXTERNAL GUIDE.** This page is another certificate

given on the organization letterhead where the project work is pursued, which should be certified

and duly signed by the External guide about work done by the candidate and clearly mentioning the duration of the project work. The format for this page will almost be the same as the previous page with double spacing between the lines.

**(or)**

Include the Declaration page in the below specified Format:

**DECLARATION**

I **<<Student Name>>** bearing **Roll Number <<Roll No>>**, a bonafide student of <<**College Name >>**, would like to declare that the project titled **“<<project title>>”** a partial fulfillment of **M.Tech** Degree course of << **University Name >>** is my original work in the year 2015 under the guidance of **<< Guide Name>>,** <<**Designation>>** of the **<< Department Name >> Department**.

Date: **<<Student Name>>**

**(Roll Number)**

**Acknowledgement**

I would like to express my sincere gratitude to my advisor, **<<Guide Name>>**, whose knowledge and guidance has motivated me to achieve goals I never thought possible. He has consistently been a source of motivation, encouragement, and inspiration. The time I have spent working under his supervision has truly been a pleasure.

I thank H.O.D **<<HOD Name>>** for his effort and guidance and all senior faculty members of CSE Department for their help during my course. Thanks to programmers and non-teaching staff of C.S.E Department of VITS.

I Thank my principal **<<PRINCIPAL NAME>>** and Management for providing excellent facilities to carry out my project work.

Finally Special thanks to my parents for their support and encouragement throughout my life and this course. Thanks to all my friends and well wishers for their constant support.

**Names of Student (roll number)**

**Seventh page:** The seventh page may contain an **ABSTRACT** of the Thesis. The candidates may emphasize here his/her contributions.

**Page 8th, 9th …..:** In these pages candidate must provide a **TABLE OF CONTENTS**, list of tables and list of figures, Screens and notation.

**NOTE:** All the above pages are to be numbered in Roman numerals of lower case.

**Arrangement of chapters:** The following is the suggested format for arranging the Thesis matter into various chapters:

**1) Introduction** ( Do not copy as it ease from Base Paper)

**2) Literature Survey** ( Quantity should be 10 – 15 Pages)

**3) SRS** ( Preferably IEEE Format- Do not include Technology Desciption))

**4) Analysis and Design** (Preferably ER-Diagrams & DFD)

**5) Implementation** ( Architecture, Algorithms used, Module detailed desc)

**6) Testing** (Min. 10 test caes)

**7) Results** (Navigation should be provided)

**8) Conclusion and Future Scope** (In two paragraphs)

**9) Reference / Bibliography** (Format is given below)

**10) Appendices** (if any)

a) Standards (if any)

b) Sample Code(Optional)

**The arrangement of paragraph in a chapter:** Each paragraph in a chapter should be properly numbered for example: 2.1, 2.2, etc., where the first digit represents the chapter number and the second digit the paragraph number. There is no need to indicate the number for the first paragraph in a chapter.

Sub-paragraphs, if any, may be indicated as 1.1.1, 1.1.2, etc., i.e., the first digit representing the chapter, the second representing the paragraph and the third representing the sub-paragraph.

**Photographs and Tables:** The Figures and tables occurring in a chapter may be serially numbered as **Fig: 1.1, 1.2,** etc., where the first digit represents the chapter, the second digit represents the figure number. ( It should be specified below the figure)

The tables may be represented as **Tab 1.1, 1.2,** etc., where the first digit represents the chapter,

the second digit represents the table number. ( It should be specified above the Table)

**Graphs:** The graph should clearly indicate the pinots which are used for drawing the curve or

curves. All the letters in the graph should be written with stencils.

**Bibliography or References:** The following format may be used for writing the Bibliography /References.

` At the end of Thesis where the listing of references is done, the list should be made strictly in alphabetic order of the names of the author. The references/websites have to be listed in the following format:

**[S.No] Author, Paper/Book, Publisher/Magazine/Conference, Volume/Edition, page numbers,Year**

**Examples:**

[1] Bruce, Cryptography, Tata McGraw Hill, 1978

[2] R. R. Duncan, "Remediation of Lead in Water Supplies," IEEE Trans. Microwave Theory Tech.,vol. 99, no. 18, pp. 257-278, Nov. 1986.

[3] <http://www.google.com>

**Guidelines for preparing the Documentation for M.Tech.**

**Size and copies:** The Documentation should be submitted in A4 size and 5 copies (4 Hard copies + 1Soft copy – CD) of the Thesis.

**Paper, Typing and Format:** Bond paper should be used for the preparation of the Thesis. Typing should be done on the 12 point size letters for the running text, 14 point size for the sub-headings and 16 point size for main headings/titles/chapter names/etc. The font type should be preferably TIMES NEW ROMAN.

The layout margin to be provided is 1.5’ on the left, 1’ on the top and bottom, 1’ on the right.

Fresh paragraph should commence after tab space. 1.5 line spacing shall be provided throughout the Thesis.

The page number shall be indicated at the bottom right of the each page. Front Pages Small Roman Numbers (Excluding title page, Certificate page, Acknowledgement page), Body pages 1,2,3 ………., Annexure 1,2,3……….. (Separate for each Annexure).

**Binding**: The Thesis shall be properly bound using resin of **Cement gray color** (Cement Color) for M.Tech. The bound front cover should indicate in golden embossed letter the following:

**Guidelines for the project work of all PG courses:**

1. **Nature of the content**: The project work must be an evidence of the utilization and practical application of the knowledge / techniques learnt in the theory courses pertaining to the faculty / speciality.

2. **Quality and quantity:** Both the extent and the level / standard of the work carried out by the student must be greater than that of an undergraduate project and be commensurate with the depth of coverage and sophistication reflected in the theory courses of the Post Graduate specialization.

3. **Originality:** A mere collection and review, how comprehensive it may be of the previous works or the state-of-art cannot qualify as the project work, though it can form a part of it.

4. The project work must be the result of the theoretical and / or practical investigation actually carried out by the candidate.

5. It must reflect the originality of the student and mere duplication of earlier works does not suffice. However, extensions and modifications of earlier work(s) are acceptable if there is original contribution by the candidate.

6. It should be possible to publish at least one paper in a reviewed journal from the work presented in the thesis.

\* \* \*

**Achieving Secure, Universal, and Fine-Grained Query Results Verification for Secure Search Scheme over Encrypted Cloud Data**

**1. INTRODUCTION**

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Driven by the abundant benefits brought by the cloud computing such as cost saving, quick deployment, flexible resource configuration, etc., more and more enterprises and individual users are taking into account migrating their private data and native applications to the cloud server. A matter of public concern is how to guarantee the security of data that is outsourced to a remote cloud server and breaks away from the direct control of data owners. Encryption on private data before outsourcing is an effective measure to protect data confidentiality. However, encrypted data make effective data retrieval a very challenging task. To address the challenge (i.e., search on encrypted data), Song et al. first introduced the concept of searchable encryption and proposed a practical technique that allows users to search over encrypted data through encrypted query keywords. Later, many searchable encryption schemes were proposed based on symmetric key and public-key setting to strengthen security and improve query efficiency. Recently, with the growing popularity of cloud computing, how to securely and efficiently search over encrypted cloud data becomes a research focus. Some approaches have been proposed based on traditional searchable encryption schemes, which aim to protect data security and query privacies with better query efficient for cloud computing. However, all of these schemes are based on an ideal assumption that the cloud server is an ”honest-but-curious” entity and keeps robust and secure software/hardware environments. As a result, correct and complete query results always are unexceptionally returned from the cloud server when a query ends every time. However, in practical applications, the cloud server may return erroneous or incomplete query results once he behaves dishonestly for illegal profits such as saving computation and communication cost or due to possible software/hardware failure of the server. Therefore, the above fact usually motivates data users to verify the correctness and completeness of query results. Some researchers proposed to integrate the query results verification mechanisms to their secure search schemes, (e.g., embedding verification information into the specified secure indexes or query results). Upon receiving query results, data users use specified verification information to verify their correctess and completeness. There are two limitations in these schemes: 1) These verification mechanisms provide a coarsegrained verification, i.e., if the query result set contains all qualified and correct data files, then these schemes reply yes, otherwise reply no. Thus, if the verification algorithm outputs no, a data user has to abort the decryption for all query results despite only one query result is incorrect. 2) These verification mechanisms are generally tightly coupled to corresponding secure query constructions and have not universality. In a search process, for a returned query results set that contains multiple encrypted data files, a data user may wish to verify the correctness of each encrypted data file (thus, he can remove incorrect results and retain the correct ones as the ultima query results) or wants to check how many or which qualified data files are not returned on earth if the cloud server intentionally omits some query results. These information can be regarded as a hard evidence to punish the cloud server. This is challenging to achieve the fine-grained verifications since the query and verification are enforced in the encrypted environment. We proposed a secure and fine-grained query results verification scheme by constructing the verification object for encrypted outsourced data files. When a query ends, the query results set along with the corresponding verification object are returned together, by which the query user can accurately verify: 1) the correctness of each encrypted data file in the results set; 2) how many qualified data files are not returned and 3) which qualified data files are not returned. Furthermore, our proposed verification scheme is lightweight and loose-coupling to concrete secure query schemes and can be very easily equipped into any secure query scheme for cloud computing. However, some necessary extensions and important works need to be further supplied to perfect our original scheme such as detailed performance evaluation and formal security definition and proof. More importantly, in the dishonest cloud environment, the scheme suffers from the following two important security problems: 1) Just as possibly tampering or deleting query results, the dishonest cloud server may also tamper or forge verification objects themselves to make the data user impossible to perform verification operation. Specially, once the cloud server knows that the query results verification scheme is provided in the secure search system, he may return inveracious verification object to escape responsibilities of misbehavior. 2) When a data user wants to obtain the desired verification object, some important information will be revealed such as which verification objects are being or have been requested before frequently, etc. This information may leak query user’s privacy and expose some useful contents about data files. More importantly, this exposed information may become temptations of misbehavior for the cloud server.

**Objective of the Project**

Secure search techniques over encrypted cloud data allow an authorized user to query data files of interest by submitting encrypted query keywords to the cloud server in a privacy-preserving manner. However, in practice, the returned query results may be incorrect or incomplete in the dishonest cloud environment. For example, the cloud server may intentionally omit some qualified results to save computational resources and communication overhead. Thus, a well-functioning secure query system should provide a query results verification mechanism that allows the data user to verify results. In this paper, we design a secure, easily integrated, and fine-grained query results verification mechanism, by which, given an encrypted query results set, the query user not only can verify the correctness of each data file in the set but also can further check how many or which qualified data files are not returned if the set is incomplete before decryption. The verification scheme is loose-coupling to concrete secure search techniques and can be very easily integrated into any secure query scheme. We achieve the goal by constructing secure verification object for encrypted cloud data. Furthermore, a short signature technique with extremely small storage cost is proposed to guarantee the authenticity of verification object and a verification object request technique is presented to allow the query user to securely obtain the desired verification object. Performance evaluation shows that the proposed schemes are practical and efficient.

**2. LITERATURE SURVEY**

**Security Challenges for the Public Cloud**

In this talk, we will first discuss a number of pressing security challenges in Cloud Computing, including data service outsourcing security and secure computation outsourcing. Then, we will focus on data storage security in Cloud Computing. As one of the primitive services, cloud storage allows data owners to outsource their data to cloud for its appealing benefits. However, the fact that owners no longer have physical possession of the outsourced data raises big security concerns on the storage correctness.

Hence, enabling secure storage auditing in the cloud environment with new approaches becomes imperative and challenging. In this talk, we will present our recent research efforts towards storage outsourcing security in cloud computing and describe both our technical approaches and security & performance evaluations.

**Practical Techniques for Searches on Encrypted Data**

It is desirable to store data on data storage servers such as mail servers and file servers in encrypted form to reduce security and privacy risks. But this usually implies that one has to sacrifice functionality for security. For example, if a client wishes to retrieve only documents containing certain words, it was not previously known how to let the data storage server perform the search and answer the query without loss of data confidentiality. In this paper, we describe our cryptographic schemes for the problem of searching on encrypted data and provide proofs of security for the resulting crypto systems. Our techniques have a number of crucial advantages. They are provably secure: they provide provable secrecy for encryption, in the sense that the untrusted server cannot learn anything about the plaintext when only given the ciphertext; they provide query isolation for searches, meaning that the untrusted server cannot learn anything more about the plaintext than the search result; they provide controlled searching, so that the untrusted server cannot search for an arbitrary word without the user’s authorization; they also support hidden queries, so that the user may ask the untrusted server to search for a secret word without revealing the word to the server.

We have described new techniques for remote searching on encrypted data using an untrusted server and provided proofs of security for the resulting crypto systems. Our techniques have a number of crucial advantages: they are provably secure; they support controlled and hidden search and query isolation; they are simple and fast (More specifically, for a document of length , the encryption and search algorithms only need O(*n*) stream cipher and block cipher operations); and they introduce almost no space and communication overhead. Our scheme is also very flexible, and it can easily be extended to support more advanced search queries. We conclude that this provides a powerful new building block for the construction of secure services in the untrusted infrastructure.

**Public Key Encryption with keyword Search**

We study the problem of searching on data that is encrypted using a public key system. Consider user Bob who sends email to user Alice encrypted under Alice’s public key. An email gateway wants to test whether the email contains the keyword “urgent” so that it could route the email accordingly. Alice, on the other hand does not wish to give the gateway the ability to decrypt all her messages. We define and construct a mechanism that enables Alice to provide a key to the gateway that enables the gateway to test whether the word “urgent” is a keyword in the email without learning anything else about the email. We refer to this mechanism as Public Key Encryption with keyword Search. As another example, consider a mail server that stores various messages publicly encrypted for Alice by others. Using our mechanism Alice can send the mail server a key that will enable the server to identify all messages containing some specific keyword, but learn nothing else. We define the concept of public key encryption with keyword search and give several constructions.

We defined the concept of a public key encryption with keyword search (PEKS) and gave two constructions. Constructing a PEKS is related to Identity Based Encryption (IBE), though PEKS seems to be harder to construct. We showed that PEKS implies Identity Based Encryption, but the converse is currently an open problem. Our constructions for PEKS are based on recent IBE constructions. We are able to prove security by exploiting extra properties of these schemes.

**Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions**

Searchable symmetric encryption (SSE) allows a party to outsource the storage of his data to another party in a private manner, while maintaining the ability to selectively search over it. This problem has been the focus of active research and several security definitions and constructions have been proposed. In this paper we begin by reviewing existing notions of security and propose new and stronger security definitions. We then present two constructions that we show secure under our new definitions. Interestingly, in addition to satisfying stronger security guarantees, our constructions are more efficient than all previous constructions. Further, prior work on SSE only considered the setting where only the owner of the data is capable of submitting search queries. We consider the natural extension where an arbitrary group of parties other than the owner can submit search queries. We formally define SSE in this multi-user setting, and present an efficient construction.

In this article, we have revisited the problem of searchable symmetric encryption, which allows a client to store its data on a remote server in such a way that it can search over it in a private manner. We make several contributions including new security definitions and new constructions. Motivated by subtle problems in all previous security definitions for SSE, we propose new definitions and point out that the existing notions have significant practical drawbacks: contrary to the natural use of searchable encryption, they only guarantee security for users that perform all their searches at once. We address this limitation by introducing stronger definitions that guarantee security even when users perform more realistic searches. We also propose two new SSE constructions. Surprisingly, despite being provably secure under our stronger security definitions, these are the most efficient schemes to date and are (asymptotically) optimal (i.e., the work performed by the server per returned document is constant in the size of the data). Finally, we also consider multi-user SSE, which extends the searching ability to parties other than the owner.

**Deterministic and Efficiently Searchable Encryption**

We present as-strong-as-possible definitions of privacy, and constructions achieving them, for public-key encryption schemes where the encryption algorithm is deterministic. We obtain as a consequence database encryption methods that permit fast (i.e. sub-linear, and in fact logarithmic, time) search while provably providing privacy that is as strong as possible subject to this fast search constraint.

One of our constructs, called RSA-DOAEP, has the added feature of being length preserving, so that it is the first example of a public-key cipher. We generalize this to obtain a notion of efficiently-searchable encryption schemes which permit more flexible privacy to search-time trade-offs via a technique called bucketization. Our results answer muchasked questions in the database community and provide foundations for work done there.

**Public-Key Encryption with Fuzzy Keyword Search: A Provably Secure Scheme under Keyword Guessing Attack**

A lot of interest has been drawn recently into public-key encryption with keyword search (PEKS), which keeps publickey encrypted documents amendable to secure keyword search. However, PEKS resist against keyword guessing attack by assuming that the size of the keyword space is beyond the polynomial level. But this assumption is ineffective in practice. PEKS are insecure under keyword guessing attack. As we observe, the key to defend such attack is to avoid the availability of the exact search trapdoor to adversaries. Accordingly, we compromise the exactness of search trapdoor by mapping at least two different keywords into a fuzzy search trapdoor. We propose a novel concept called public-key encryption with fuzzy keyword search (PEFKS), by which the un-trusted server only obtains the fuzzy search trapdoor instead of the exact search trapdoor, and define its semantic security under chosen keyword attack (SS-CKA) and indistinguishability of keywords under non-adaptively chosen keywords and keyword guessing attack (IK-NCK-KGA). For the keyword space with and without uniform distribution, we respectively present two universal transformations from anonymous identity-based encryption to PEFKS, and prove their SSCKA and IK-NCK-KGA securities. To our knowledge, PEFKS is the first scheme to resist against keyword guessing attack on condition that the keyword space is not more than the polynomial level.

In PEKS, a proxy server, who responds the keyword queries of a receiver, can know the content of keywords by implementing KGA. Moreover, it is efficient under the practical condition that the size of the keyword space is not more than the polynomial level. In order to resist against KGA, we novelly defined public-key encryption with fuzzy keyword search (PEFKS) and its IK-NCK-KGA security. And we proposed two universal transformations from IBE to PEFKS under different conditions. Under the condition that the keyword space has uniform distribution, we proposed a SS-CKA and IK-NCK-KGA secure transformation PEFKS-UD, and provided an instance based on BF01 scheme. Under the condition that the keyword space has non-uniform distribution, we proposed another SS-CKA and IK-NCK-KGA secure transformation PEFKS-ND, and provided two methods to sort keywords, which is the key to realize PEFKS-ND. Beyond the perspective of cryptosystem, we discussed the biased advantage of KGA on PEFKSND, which is caused only by the non-uniform distribution of the keyword space. We illuminate that the biased advantage has been decreased as much as possible. So we made PEFKS-ND secure in a broad sense.

**Parallel and Dynamic Searchable Symmetric Encryption**

Searchable symmetric encryption (SSE) enables a client to outsource a collection of encrypted documents in the cloud and retain the ability to perform keyword searches without revealing information about the contents of the documents and queries. Although efficient SSE constructions are known, previous solutions are highly sequential. This is mainly due to the fact that, currently, the only method for achieving sub-linear time search is the inverted index approach which requires the search algorithm to access a sequence of memory locations, each of which is unpredictable and stored at the previous location in the sequence. Motivated by advances in multi-core architectures, we present a new method for constructing sub-linear SSE schemes. Our approach is highly parallelizable and dynamic. With roughly a logarithmic number of cores in place, searches for a keyword w in our scheme execute in o(r) parallel time, where r is the number of documents containing keyword w (with more cores, this bound can go down to O(log n), i.e., independent of the result size r). Such time complexity outperforms the optimal Θ(r) sequential search time—a similar bound holds for the updates. Our scheme also achieves the following important properties: (a) it enjoys a strong notion of security, namely security against adaptive chosen-keyword attacks; (b) compared to existing sub-linear dynamic SSE schemes, updates in our scheme do not leak any information, apart from information that can be inferred from previous search tokens; (c) it can be implemented efficiently in external memory (with logarithmic I/O overhead). Our technique is simple and uses a red-black tree data structure; its security is proven in the random oracle model.

**Dynamic Searchable Encryption via Blind Storage**

Dynamic Searchable Symmetric Encryption allows a client to store a dynamic collection of encrypted documents with a server, and later quickly carry out keyword searches on these encrypted documents, while revealing minimal information to the server. In this paper we present a new dynamic SSE scheme that is simpler and more efficient than existing schemes while revealing less information to the server than prior schemes, achieving fully adaptive security against honest-but-curious servers. We implemented a prototype of our scheme and demonstrated its efficiency on datasets from prior work. Apart from its concrete efficiency, our scheme is also simpler: in particular, it does not require the server to support any operation other than upload and download of data. Thus the server in our scheme can be based solely on a cloud storage service, rather than a cloud computation service as well, as in prior work. In building our dynamic SSE scheme, we introduce a new primitive called Blind Storage, which allows a client to store a set of files on a remote server in such a way that the server does not learn how many files are stored, or the lengths of the individual files; as each file is retrieved, the server learns about its existence (and can notice the same file being downloaded subsequently), but the file’s name and contents are not revealed. This is a primitive with several applications other than SSE, and is of independent interest.

In this work, we introduced a new cryptographic construct called Blind Storage, and implemented it using a novel, yet light-weight protocol SCATTERSTORE. We also showed how a dynamic SSE scheme can be constructed using Blind Storage, in a relatively simple manner. The resulting scheme is more computationally efficient, require simpler infrastructure, and is more secure than the existing schemes.

**Secure Ranked Keyword Search over Encrypted Cloud Data**

As Cloud Computing becomes prevalent, sensitive information are being increasingly centralized into the cloud. For the protection of data privacy, sensitive data has to be encrypted before outsourcing, which makes effective data utilization a very challenging task. Although traditional searchable encryption schemes allow users to securely search over encrypted data through keywords, these techniques support only boolean search, without capturing any relevance of data files. This approach suffers from two main drawbacks when directly applied in the context of Cloud Computing. On the one hand, users, who do not necessarily have pre-knowledge of the encrypted cloud data, have to postprocess every retrieved file in order to find ones most matching their interest; On the other hand, invariably retrieving all files containing the queried keyword further incurs unnecessary network traffic, which is absolutely undesirable in today’s pay-as-you-use cloud paradigm. In this paper, for the first time we define and solve the problem of effective yet secure ranked keyword search over encrypted cloud data. Ranked search greatly enhances system usability by returning the matching files in a ranked order regarding to certain relevance criteria (e.g., keyword frequency), thus making one step closer towards practical deployment of privacy-preserving data hosting services in Cloud Computing. We first give a straightforward yet ideal construction of ranked keyword search under the state-of-the-art searchable symmetric encryption (SSE) security definition, and demonstrate its inefficiency. To achieve more practical performance, we then propose a definition for ranked searchable symmetric encryption, and give an efficient design by properly utilizing the existing cryptographic primitive, order-preserving symmetric encryption (OPSE). Thorough analysis shows that our proposed solution enjoys “as-strong-as-possible” security guarantee compared to previous SSE schemes, while correctly realizing the goal of ranked keyword search. Extensive experimental results demonstrate the efficiency of the proposed solution.

In this paper, as an initial attempt, we motivate and solve the problem of supporting efficient ranked keyword search for achieving effective utilization of remotely stored encrypted data in Cloud Computing. We first give a basic scheme and show that by following the same existing searchable encryption framework, it is very inefficient to achieve ranked search. We then appropriately weaken the security guarantee, resort to the newly developed crypto primitive OPSE, and derive an efficient one-to-many order-preserving mapping function, which allows the effective RSSE to be designed. Through thorough security analysis, we show that our proposed solution is secure and privacy-preserving, while correctly realizing the goal of ranked keyword search.

**Privacy Preserving Multi-Keyword Ranked Search with Anonymous ID Assignment over Encrypted Cloud Data**

The advancement in cloud computing has motivated the data owners to outsource their data management systems from local sites to commercial public cloud for great flexibility and economic savings. But people can enjoy full benefit of cloud computing if we are able to address very real privacy and security concerns that come with storing sensitive personal information. For real privacy, user identity should remain hidden from CSP (Cloud service provider) and to protect privacy of data, data which is sensitive is to be encrypted before outsourcing. Thus, enabling an encrypted cloud data search service is of great importance. By considering the large number of data users, documents in the cloud, it is important for the search service to allow multikeyword query and provide result similarity ranking to meet the effective need of data retrieval search and not often differentiate the search results. In this system, we define and solve the challenging problem of privacy-preserving multikeyword ranked search over encrypted cloud data (MRSE), and establish a set of strict privacy requirements for such a secure cloud data utilization system to be implemented in real. We first propose a basic idea for the Multi-keyword Ranked Search over Encrypted cloud data (MRSE) based on secure inner product computation and efficient similarity measure of coordinate matching, i.e., as many matches as possible, in order to capture the relevance of data documents to the search query, then we give two significantly improved MRSE schemes to achieve various stringent privacy requirements in two different threat models. Assignment of anonymous ID to the user to provide more security to the data on cloud server is done. To improve the search experience of the data search service, further extension of the two schemes to support more search semantics is done.

The previous work mainly focused on providing privacy to the data on cloud in which using multi-keyword ranked search was provided over encrypted cloud data using efficient similarity measure of co-ordinate matching. The previous work also proposed a basic idea of MRSE using secure inner product computation. There was a need to provide more real privacy which this paper presents. In this system, stringent privacy is provided by assigning the cloud user a unique ID. This user ID is kept hidden from the cloud service provider as well as the third party user in order to protect the user’s data on cloud from the CSP and the third party user. Thus, by hiding the user’s identity, the confidentiality of user’s data is maintained.

**3. ANALYSIS**

**Introduction**

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of [software development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies). These methodologies form the framework for planning and controlling the creation of an information system the [software development process](http://en.wikipedia.org/wiki/Software_development_process).

**Existing System**

Recently, with the growing popularity of cloud computing, how to securely and efficiently search over encrypted cloud data becomes a research focus. Some approaches have been proposed based on traditional searchable encryption schemes, which aim to protect data security and query privacies with better query efficient for cloud computing. However, all of these schemes are based on an ideal assumption that the cloud server is an ”honest-but-curious” entity and keeps robust and secure software/hardware environments. As a result, correct and complete query results always are unexceptionally returned from the cloud server when a query ends every time. However, in practical applications, the cloud server may return erroneous or incomplete query results once he behaves dishonestly for illegal profits such as saving computation and communication cost or due to possible software/hardware failure of the server.

**Disadvantages of Existing System:**

1. The verification mechanisms provide a coarse-grained verification, i.e., if the query result set contains all qualified and correct data files, then these schemes reply yes, otherwise reply no. Thus, if the verification algorithm outputs no, a data user has to abort the decryption for all query results despite only one query result is incorrect.
2. The verification mechanisms are generally tightly coupled to corresponding secure query constructions and have not universality

**Proposed System**

We formally propose the verifiable secure search system model and threat model and design a fine-grained query results verification scheme for secure keyword search over encrypted cloud data. We propose a short signature technique based on certificateless public-key cryptography to guarantee the authenticity of the verification objects themselves. We design a novel verification object request technique based on Paillier Encryption, where the cloud server knows nothing about what the data user is requesting for and which verification objects are returned to the user.

**Advantages of Proposed System:**

1. Our scheme can verify the correctness of each encrypted query result or further accurately find out how many or which qualified data files are returned by the dishonest cloud server

**3.1. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term *requirements traceability*.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.2. Software Requirement Specification**

**3.2.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.2.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly Java Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through Java capabilities.

**Software Interfaces**

The required software is JAVA1.6.

**Operating Environment**

Windows XP, Linux.

**HARDWARE REQUIREMENTS:**

# Processor - Pentium –IV

* Speed - 1.1 Ghz
* RAM - 256 MB(min)
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System : Windows XP
* Programming Language : Java
* Database : MySQL

**4. DESIGN**

**UML diagrams**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + **User Model View**
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + **Structural Model view**
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* **Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation Model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

**4.1 Class diagram:-**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. A class with three sections, in the diagram, classes is represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

**Class diagram:**



**4.2 Use case diagram:-**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

**4.2.1 Use case diagram:**



**4.3. Sequence Diagram:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.

**4.3.1 Sequence diagram:**

****

**4.4 Collaboration diagram**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

**4.5.1 Collaboration diagram:**



**4.6 Component Diagram**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

**4.6.1 Component diagram:**



**4.7 Deployment Diagram**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

**4.7.1 Deployment diagram:**



**4.8 Activity diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

**4.8.1 Activity diagram:**

Secure Search System

Cloud Server

Data Owner

Start Server

Register

Save Uploaded files

Is he already registered?

No

Yes

Create verification object

Login

Upload Files

Search Query

Verification

Download & Decrypt file

Logout

View chart

**4.9 Data Flow Diagram:**

[Data flow diagrams](http://www.edrawsoft.com/Data-Flow-Diagrams.php) illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

**Data Flow Diagram:**

5. Download & decrypt the file

3. Upload files

Cloud Server

Data Owner

2. Register & login into system

4. Search queries and verification

6. View chart 1. Run

**5. IMPLEMENTATION**

**5.1. Introduction of technologies used**

**About Java**:

Initially the language was called as “oak” but it was renamed as “java” in 1995.The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

* Java is a programmer’s language
* Java is cohesive and consistent
* Except for those constraint imposed by the Internet environment. Java gives the programmer, full control

Finally Java is to Internet Programming where c was to System Programming.

**Importance of Java to the Internet**

Java has had a profound effect on the Internet. This is because; java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the server and the personal computer. They are passive information and Dynamic active programs. in the areas of Security and probability. But Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

**Applications and applets**

An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++ .Java’s ability to create Applets makes it important. An Applet I san application, designed to be transmitted over the Internet and executed by a Java-compatible web browser. An applet I actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can be react to the user input and dynamically change.

**Java Architecture**

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

**Compilation of code**

When you compile the code, the Java compiler creates machine code (called byte code)for a hypothetical machine called Java Virtual Machine(JVM). The JVM is supposed t executed the byte code. The JVM is created for the overcoming the issue of probability. The code is written and compiled for one machine and interpreted on all machines .This machine is called Java Virtual Machine.

**Compiling and interpreting java source code.**

****

During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be an Intel Pentium windows 95 or sun SPARCstation running Solaris or Apple Macintosh running system and all could receive code from any computer through internet and run the Applets.

**Simple**:

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ Programmer, learning Java will oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

### Object oriented

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank state. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

### Robust

The multi-platform environment of the web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java? Java is strictly typed language; it checks your code at compile time and runtime.

Java virtually eliminates the problems of memory management and deal location, which is completely automatic. In a well-written Java program, all run-time errors can and should be managed by your program.

**AWT and Swings:**

**AWT:**

**Graphical User Interface:**

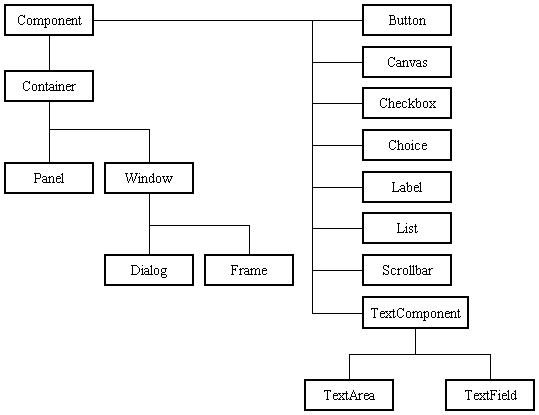
The user interface is that part of a program that interacts with the user of the program. GUI is a type of [user interface](http://en.wikipedia.org/wiki/User_interface) that allows [users](http://en.wikipedia.org/wiki/User_(computing)) to [interact](http://en.wikipedia.org/wiki/Human-computer_interaction) with electronic devices with images rather than text commands. A class library is provided by the Java programming language which is known as Abstract Window Toolkit (AWT) for writing graphical programs. The Abstract Window Toolkit (AWT) contains several graphical widgets which can be added and positioned to the display area with a layout manager.

As the Java programming language, the AWT is not platform-independent. AWT uses system peers object for constructing graphical widgets. A common set of tools is provided by the AWT for graphical user interface design. The implementation of the user interface elements provided by the AWT is done using every platform's native GUI toolkit. One of the AWT's significance is that the look and feel of each platform can be preserved.

**Components:**

A graphical user interface is built of graphical elements called components. A *component* is an object having a graphical representation that can be displayed on the screen and that can interact with the user. Components allow the user to interact with the program and provide the input to the program. In the AWT, all user interface components are instances of class Component or one of its subtypes. Typical components include such items as buttons, scrollbars, and text fields.

**Types of Components:**

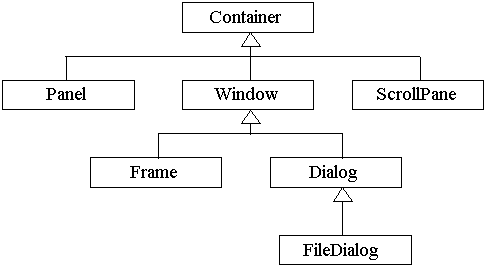
****

Before proceeding ahead, first we need to know what containers are. After learning containers we learn all components in detail.

**Containers:**

Components do not stand alone, but rather are found within containers. In order to make components visible, we need to add all components to the container. Containers contain and control the layout of components. In the AWT, all containers are instances of class Container or one of its subtypes. Components must fit completely within the container that contains them. For adding components to the container we will use add() method.

**Types of containers:**

****

**Basic GUI Logic:**

The GUI application or applet is created in three steps. These are:

* Add components to Container objects to make your GUI.
* Then you need to setup event handlers for the user interaction with GUI.
* Explicitly display the GUI for application.

A new thread is started by the interpreter for user interaction when an AWT GUI is displayed. When any event is received by this new thread such as click of a mouse, pressing of key etc then one of the event handlers is called by the new thread set up for GUI. One important point to note here is that the event handler code is executed within the thread.

**Creating a Frame:**

**Method1:**

In the first method we will be creating frame by extending Frame class which is defined in java.awt package. Following program demonstrate the creation of a frame.

import java.awt.\*;

public class FrameDemo1 extends Frame

{

FrameDemo1()

{

setTitle("Label Frame");

setVisible(true);

setSize(500,500);

}

public static void main(String[] args)

{

new FrameDemo1 ();

}

}

In the above program we are using three methods:

setTitle: For setting the title of the frame we will use this method. It takes String as an argument which will be the title name.

SetVisible: For making our frame visible we will use this method. This method takes Boolean value as an argument. If we are passing true then window will be visible otherwise window will not be visible.

SetSize: For setting the size of the window we will use this method. The first argument is width of the frame and second argument is height of the frame.

**Method 2:**

In this method we will be creating the Frame class instance for creating frame window. Following program demonstrate Method2.

import java.awt.\*;

public class FrameDemo2

{

public static void main(String[] args)

{

Frame f = new Frame();

f.setTitle("My first frame");

f.setVisible(true);

f.setSize(500,500);

}

}

**Types of Components:**

1. **Labels :**

This is the simplest component of Java Abstract Window Toolkit. This component is generally used to show the text or string in your application and label never perform any type of action.

Label l1 = new Label("One");

Label l2 = new Label("Two");

Label l3 = new Label("Three",Label.CENTER);

In the above three lines we have created three labels with the name “one, two, three”. In the third label we are passing two arguments. Second argument is the justification of the label. Now after creating components we will be adding it to the container.

add(l1);

add(l2);

add(l3);

We can set or change the text in a label by using the **setText( )** method. You can obtain the current label by calling **getText( )**. These methods are shown here:

void setText(String *str*)

String getText( )

1. **Buttons :**

This is the component of Java Abstract Window Toolkit and is used to trigger actions and other events required for your application. The syntax of defining the button is as follows :

Button l1 = new Button("One");

Button l2 = new Button("Two");

Button l3 = new Button("Three");

We can change the Button's label or get the label's text by using the Button.setLabel(String) and Button.getLabel() method.

1. **CheckBox:**

A *check box is a control that is used to turn an option on or off. It consists of a small box* that can either contain a check mark or not. There is a label associated with each check box that describes what option the box represents. You change the state of a check box by clicking on it. The syntax of the definition of Checkbox is as follows :

Checkbox Win98 = new Checkbox("Windows 98/XP", null, true);

Checkbox winNT = new Checkbox("Windows NT/2000");

Checkbox solaris = new Checkbox("Solaris");

Checkbox mac = new Checkbox("MacOS");

The first form creates a check box whose label is specified in first argument and whose group is specified in second argument*.* If this check box is not part of a group, then *cbGroup* must be **null**. (Check box groups are described in the next section.) The value truedetermines the initial state of the check box is checked. The second form creates a check box with only one parameter.

To retrieve the current state of a check box, call **getState( )**. To set its state, call **setState( )**. You can obtain the current label associated with a check box by calling **getLabel( )**. To set the label, call **setLabel( )**. These methods are as follows:

boolean getState( )

void setState(boolean *on*)

String getLabel( )

void setLabel(String *str*)

Here, if *on* is **true**, the box is checked. If it is **false**, the box is cleared. The string passed in *str* becomes the new label associated with the invoking check box.

1. **Radio Button:**

This is the special case of the Checkbox component of Java AWT package. This is used as a group of checkboxes which group name is same. Only one Checkbox from a Checkbox Group can be selected at a time. Syntax for creating radio buttons is as follows:

CheckboxGroup cbg = new CheckboxGroup();

Checkbox Win98 = new Checkbox("Windows 98/XP", cbg , true);

Checkbox winNT = new Checkbox("Windows NT/2000",cbg, false);

Checkbox solaris = new Checkbox("Solaris",cbg, false);

Checkbox mac = new Checkbox("MacOS",cbg, false);

For Radio Button we will be using CheckBox class. The only difference in Checkboxes and radio button is in Check boxes we will specify null for checkboxgroup but whereas in radio button we will be specifiying the checkboxgroup object in the second parameter.

1. **Choice:**

The Choice class is used to create a pop-up list of items from which the user may choose. Thus, a Choice control is a form of menu. Syntax for creating choice is as follows:

Choice os = new Choice();

/\* adding items to choice \*/

os.add("Windows 98/XP");

os.add("Windows NT/2000");

os.add("Solaris");

os.add("MacOS");

We will be creating choice with the help of Choice class. Pop up list will be creating with the creation of object, but it will not have any items. For adding items we will be using add() method defined in Choice class.

To determine which item is currently selected, you may call either **getSelectedItem( )** or **getSelectedIndex( )**. These methods are shown here:

String getSelectedItem( )

int getSelectedIndex( )

The **getSelectedItem( )** method returns a string containing the name of the item. **getSelectedIndex( )** returns the index of the item. The first item is at index 0. By default, the first item added to the list is selected.

1. **List:**

List class is also same as choice but the only difference in list and choice is, in choice user can select only one item whereas in List user can select more than one item. Syntax for creating list is as follows:

List os = new List(4, true);

First argument in the List constructor specifies the number of items allowed in the list. Second argument specifies whether multiple selections are allowed or not.

/\* Adding items to the list \*/

os.add("Windows 98/XP");

os.add("Windows NT/2000");

os.add("Solaris");

os.add("MacOS");

In list we can retrieve the items which are selected by the users. In multiple selection user will be selecting multiple values for retrieving all the values we have a method called getSelectedValues() whose return type is string array. For retrieving single value again we can use the method defined in Choice i.e. getSelectedItem().

1. **TextField:**

Text fields allow the user to enter strings and to edit the text using the arrow keys, cut and paste keys. TextField is a subclass of TextComponent. Syntax for creating list is as follows:

TextField tf1 = new TextField(25);

TextField tf2 = new TextField();

In the first text field we are specifying the size of the text field and the second text field is created with the default value. **TextField** (and its superclass **TextComponent**) provides several methods that allow you to utilize a text field. To obtain the string currently contained in the text field, call **getText( )**. To set the text, call **setText( )**. These methods are as follows:

String getText( )

void setText(String *str*)

We can control whether the contents of a text field may be modified by the user by calling **setEditable( )**. You can determine editability by calling **isEditable( )**. These methods are shown here:

boolean isEditable( )

void setEditable(boolean *canEdit*)

**isEditable( )** returns **true** if the text may be changed and **false** if not. In **setEditable( )**, if *canEdit* is **true**, the text may be changed. If it is **false**, the text cannot be altered.

There may be times when we will want the user to enter text that is not displayed, such as a password. We can disable the echoing of the characters as they are typed by calling **setEchoChar( )**.

1. **TextArea:**

TextArea is a multiple line editor. Syntax for creating list is as follows:

TextArea area = new TextArea(20,30);

Above code will create one text area with 20 rows and 30 columns. **TextArea** is a subclass of **TextComponent**. Therefore, it supports the **getText( )**, **setText( )**, **getSelectedText( )**, **select( )**, **isEditable( )**, and **setEditable( )** methods described in the preceding section.

**TextArea** adds the following methods:

void append(String *str*)

void insert(String *str*, int *index*)

void replaceRange(String *str*, int *startIndex*, int *endIndex*)

The **append( )** method appends the string specified by *str* to the end of the current text. **insert( )** inserts the string passed in *str* at the specified index. To replace text, call **replaceRange( )**. It replaces the characters from *startIndex* to *endIndex*–1, with the replacement text passed in *str.*

**Layout Managers:**

A layout manager automatically arranges controls within a window by using some type of algorithm. Each **Container** object has a layout manager associated with it. A layout manager is an instance of any class that implements the **LayoutManager** interface. The layout manager is set by the **setLayout( )** method. If no call to **setLayout( )** is made, then the default layout manager is used. Whenever a container is resized (or sized for the first time), the layout manager is used to position each of the components within it. The **setLayout( )** method has the following general form:

void setLayout(LayoutManager *layoutObj*)

Here, *layoutObj* is a reference to the desired layout manager. If you wish to disable the layout manager and position components manually, pass **null** for *layoutObj.* If we do this, you will need to determine the shape and position of each component manually, using the setBounds( ) method defined by Component.

Void setBounds(int x , int y , int width, int length)

In which first two arguments are the x and y axis. Third argument is width and fourth argument is height of the component.

Java has several predefined **LayoutManager** classes, several of which are described next. You can use the layout manager that best fits your application.

**FlowLayout:**

**FlowLayout** is the default layout manager. This is the layout manager that the preceding examples have used. **FlowLayout** implements a simple layout style, which is similar to how words flow in a text editor. Components are laid out from the upper-left corner, left to right and top to bottom. When no more components fit on a line, the next one appears on the next line. A small space is left between each component, above and below, as well as left and right. Here are the constructors for **FlowLayout**:

FlowLayout( )

FlowLayout(int *how*)

FlowLayout(int *how*, int *horz*, int *vert*)

The first form creates the default layout, which centers components and leaves five pixels of space between each component. The second form lets you specify how each line is aligned. Valid values for *how* are as follows:

FlowLayout.LEFT

FlowLayout.CENTER

FlowLayout.RIGHT

These values specify left, center, and right alignment, respectively. The third form allows you to specify the horizontal and vertical space left between components in *horz* and *vert,* respectively.

**BorderLayout:**

THE JAVA LIBRARYThe **BorderLayout** class implements a common layout style for top-level windows. It has four narrow, fixed-width components at the edges and one large area in the center. The four sides are referred to as north, south, east, and west. The middle area is called the center. Here are the constructors defined by **BorderLayout**:

BorderLayout( )

BorderLayout(int *horz*, int *vert*)

The first form creates a default border layout. The second allows you to specify the horizontal and vertical space left between components in *horz* and *vert,* respectively. **BorderLayout** defines the following constants that specify the regions:

BorderLayout.CENTER BorderLayout.SOUTH

BorderLayout.EAST BorderLayout.WEST

BorderLayout.NORTH

When adding components, you will use these constants with the following form of **add( )**, which is defined by **Container**:

void add(Component *compObj,* Object *region*);

Here, *compObj* is the component to be added, and *region* specifies where the component will be added.

**GridLayout:**

**GridLayout** lays out components in a two-dimensional grid. When you instantiate a **GridLayout**, you define the number of rows and columns. The constructors supported by **GridLayout** are shown here:

GridLayout( )

GridLayout(int *numRows*, int *numColumns* )

GridLayout(int *numRows*, int *numColumns*, int *horz*, int *vert*)

The first form creates a single-column grid layout. The second form creates a grid layout with the specified number of rows and columns. The third form allows you to specify the horizontal and vertical space left between components in *horz* and *vert*, respectively. Either *numRows* or *numColumns* can be zero. Specifying *numRows* as zero allows for unlimited-length columns. Specifying *numColumns* as zero allows for unlimited-length rows.

**Swings:**

**About Swings:**

Swing is important to develop Java programs with a graphical user interface (GUI). There are many components which are used for the building of GUI in Swing. The Swing Toolkit consists of many components for the building of GUI. These components are also helpful in providing interactivity to Java applications. Following are components which are included in Swing toolkit:

* list controls
* buttons
* labels
* tree controls
* table controls

All AWT flexible components can be handled by the Java Swing. Swing toolkit contains far more components than the simple component toolkit. It is unique to any other toolkit in the way that it supports integrated internationalization, a highly customizable text package, rich undo support etc. Not only this you can also create your own look and feel using Swing other than the ones that are supported by it. The customized look and feel can be created using Synth which is specially designed. Not to forget that Swing also contains the basic user interface such as customizable painting, event handling, drag and drop etc.

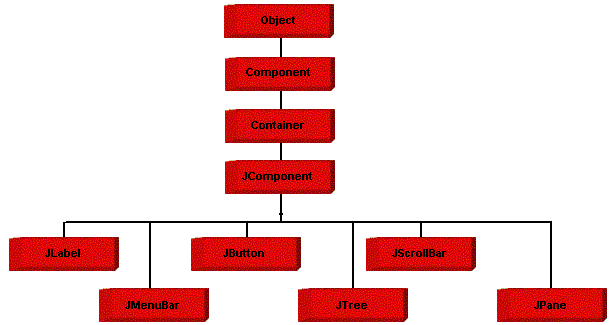
The Java Foundation Classes (JFC) which supports many more features important to a GUI program comprises of Swing as well. The features which are supported by Java Foundation Classes (JFC) are the ability to create a program that can work in different languages, the ability to add rich graphics functionality etc.

There are several components contained in Swing toolkit such as check boxes, buttons, tables, text etc. Some very simple components also provide sophisticated functionality. For instance, text fields provide formatted text input or password field behavior. Furthermore, the file browsers and dialogs can be used according to one's need and can even be customized.

**Difference between Swings and AWT:**

|  |  |
| --- | --- |
| **Swings** | **AWT** |
| Swings are the light weight components. | AWTs are the heavy weight components. |
| Swings are developed by using pure java language. | AWTs are developed by using C and C++. |
| We can have different look and feel in swings. | This feature is not available in awt. |
| Swing has many advanced features like JTabel, JTabbedPane and JTree | This is not available in AWT. |

**Java Swing Class Hierarchy:**



**Swing Components:**

All the components which are supported in AWT same components are also supported in Swings with a slight change in their class name.

|  |  |
| --- | --- |
| **AWT Components** | **Swing Components** |
| Label | JLabel |
| TextField | JTextField |
| TextArea | JTextArea |
| Choice | JComboBox |
| Checkbox | JCheckBox |
| List | JList |
| Button | JButton |
| - | JRadioButton |
| - | JPasswordField |
| - | JTable |
| - | JTree |
| - | JTabbedPane |
| MenuBar | JMenuBar |
| Menu | JMenu |
| MenuItem | JMenuItem |
| - | JFileChooser |
| - | JOptionPane |

We will discuss only those components which are not discussed in AWT chapter.

**JTabbedPane class:**

 The JTabbedPane container allows many panels to occupy the same area of the interface, and the user may select which to show by clicking on a tab.

**Constructor**

JTabbedPane tp = new JTabbedPane();

## Adding tabs to the JTabbedPane

Add tabs to a tabbed pane by calling addTab and passing it a String title and an instance of a class which should be called when we pressed a tab. That class should be a subclass of JPanel.

addTab(“String”,instance);

**Example program:**

import javax.swing.\*;

import java.awt.\*;

public class TabbedPaneDemo extends JFrame

{

TabbedPaneDemo()

{

setLayout(new FlowLayout(FlowLayout.LEFT));

setTitle("Tabbed Demo");

setVisible(true);

setSize(500,500);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JTabbedPane pane = new JTabbedPane();

pane.addTab("Countries",new Count());

pane.addTab("Cities",new Cit());

add(pane);

}

public static void main(String a[])

{

new TabbedPaneDemo();

}

}

class Count extends JPanel

{

Count()

{

JButton b1 = new JButton("India");

JButton b2 = new JButton("SriLanka");

JButton b3 = new JButton("Australia");

add(b1);

add(b2);

add(b3);

}

}

class Cit extends JPanel

{

Cit()

{

JCheckBox cb1 = new JCheckBox("Hyderabad");

JCheckBox cb2 = new JCheckBox("Banglore");

JCheckBox cb3 = new JCheckBox("Pune");

add(cb1);

add(cb2);

add(cb3);

}

}

**JMenuBar, JMenu, JMenuItem**

A top-level window can have a menu bar associated with it. A menu bar displays a list of top-level menu choices. Each choice is associated with a drop-down menu. This concept is implemented in Java by the following classes: JMenuBar, JMenu, and JMenuItem. In general, a menu bar contains one or more JMenu objects. Each JMenu object contains a list of JMenuItem objects. Each JMenuItem object represents something that can be selected by the user. To create a menu bar, first create an instance of JMenuBar. This class only defines the default constructor. Next, create instances of JMenu that will define the selections displayed on the bar. Following are the constructors for Menu:

JMenu( )

JMenu(String *optionName*)

Here, *optionName* specifies the name of the menu selection. The first form creates an empty menu. Individual menu items are of type MenuItem. It defines these constructors:

JMenuItem( )

JMenuItem(String *itemName*)

Here, *itemName* is the name shown in the menu.

**5.5 Sample Code**

**UserScreen.java**

package com;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.BorderLayout;

import java.awt.Color;

import javax.swing.JTextArea;

import javax.swing.JScrollPane;

import javax.swing.JOptionPane;

import javax.swing.JFileChooser;

import java.net.Socket;

import java.io.ObjectOutputStream;

import java.io.ObjectInputStream;

import java.io.File;

import java.io.FileInputStream;

import java.util.ArrayList;

import java.math.BigInteger;

import org.jfree.ui.RefineryUtilities;

import java.util.Random;

public class UserScreen extends JFrame{

JButton b1,b2,b3,b4,b5;

JPanel p1,p2;

Font f1;

JTextArea area;

JScrollPane jsp;

Login login;

String user;

JFileChooser chooser;

ArrayList<byte[]> trapdoor = new ArrayList<byte[]>();

ArrayList<VerificationObject> vlist;

long time;

int keyword\_size;

public UserScreen(Login log,String usr){

super("User Screen");

login = log;

user = usr;

p1 = new JPanel();

f1 = new Font("Monospaced",Font.BOLD,16);

chooser = new JFileChooser();

b1 = new JButton("Upload File");

b1.setFont(f1);

p1.add(b1);

b1.addActionListener(new ActionListener(){

public void actionPerformed(java.awt.event.ActionEvent evt) {

int option = chooser.showOpenDialog(UserScreen.this);

if(option == JFileChooser.APPROVE\_OPTION){

File file = chooser.getSelectedFile();

upload(file);

}

}

});

b2 = new JButton("Search Query");

b2.setFont(f1);

p1.add(b2);

b2.addActionListener(new ActionListener(){

public void actionPerformed(java.awt.event.ActionEvent evt) {

search();

}

});

b5 = new JButton("Verification");

b5.setFont(f1);

p1.add(b5);

b5.addActionListener(new ActionListener(){

public void actionPerformed(java.awt.event.ActionEvent evt) {

verification();

}

});

b3 = new JButton("Keywords Vs Verification");

b3.setFont(f1);

p1.add(b3);

b3.addActionListener(new ActionListener(){

public void actionPerformed(java.awt.event.ActionEvent evt) {

Chart chart1 = new Chart("Keywords Vs Verification Chart",time,keyword\_size);

chart1.pack();

RefineryUtilities.centerFrameOnScreen(chart1);

chart1.setVisible(true);

}

});

b4 = new JButton("Logout");

b4.setFont(f1);

p1.add(b4);

b4.addActionListener(new ActionListener(){

public void actionPerformed(java.awt.event.ActionEvent evt) {

setVisible(false);

login.setVisible(true);

}

});

p2 = new JPanel();

p2.setLayout(new BorderLayout());

area = new JTextArea();

area.setFont(f1);

area.setEditable(false);

jsp = new JScrollPane(area);

p2.add(jsp,BorderLayout.CENTER);

getContentPane().add(p1,BorderLayout.NORTH);

getContentPane().add(p2,BorderLayout.CENTER);

PaillierEnc.KeyGeneration();

}

public void verification(){

try{

long start = System.currentTimeMillis();

ArrayList<String> hash = new ArrayList<String>();

ArrayList<String> bloom = new ArrayList<String>();

for(int i=0;i<trapdoor.size();i++){

BloomFilter.generateBloom(50,trapdoor.get(i));

StringBuilder sb = new StringBuilder();

for(int j=0;j<BloomFilter.input.length;j++){

sb.append(BloomFilter.input[j]);

}

System.out.println(sb.toString()+" "+new String(AES.decrypt(trapdoor.get(i))));

bloom.add(sb.toString());

}

for(int i=0;i<bloom.size();i++){

BigInteger input = new BigInteger(bloom.get(i).getBytes());

hash.add(PaillierEnc.Encryption(input).toString());

}

StringBuilder sb = new StringBuilder();

for(int i=0;i<hash.size();i++){

for(int j=0;j<vlist.size();j++){

VerificationObject vo = vlist.get(j);

keyword\_size = keyword\_size + vo.getKeyword().size();

for(int k=0;k<vo.getHash().size();k++){

String h = vo.getHash().get(k);

if(h.equals(hash.get(i))){

File ff = new File(vo.getPath());

File ff1 = new File(ff.getParent());

sb.append(ff1.getName()+"/"+ff.getName()+",");

}

}

}

}

System.out.println("verify "+sb.toString());

if(sb.length() > 0){

sb.deleteCharAt(sb.length()-1);

ViewSearchResult vsr = new ViewSearchResult();

String arr1[] = sb.toString().trim().split(",");

for(int i=0;i<arr1.length;i++){

String row[] = {arr1[i]};

vsr.dtm.addRow(row);

}

vsr.setVisible(true);

vsr.setSize(600,400);

vsr.setTitle("Files obtains from verification object for same query");

}else{

JOptionPane.showMessageDialog(this,"No record found");

}

long end = System.currentTimeMillis();

time = end - start;

}catch(Exception e){

e.printStackTrace();

}

}

public void upload(File file){

try{

FileInputStream fin = new FileInputStream(file);

byte file\_data[] = new byte[fin.available()];

fin.read(file\_data,0,file\_data.length);

fin.close();

byte enc[] = AES.encrypt(file\_data);

String keywords = new String(file\_data);

String arr[] = keywords.trim().toLowerCase().split("\\s+");

ArrayList<byte[]> encrypted\_keywords = new ArrayList<byte[]>();

ArrayList<String> bloom = new ArrayList<String>();

ArrayList<String> hash = new ArrayList<String>();

ArrayList<String> dup = new ArrayList<String>();

for(int i=0;i<arr.length;i++){

if(!dup.contains(arr[i])){

dup.add(arr[i]);

byte data[] = AES.encrypt(arr[i].getBytes());

encrypted\_keywords.add(data);

}

}

for(int i=0;i<encrypted\_keywords.size();i++){

BloomFilter.generateBloom(50,encrypted\_keywords.get(i));

StringBuilder sb = new StringBuilder();

for(int j=0;j<BloomFilter.input.length;j++){

sb.append(BloomFilter.input[j]);

}

System.out.println(sb.toString()+" "+new String(AES.decrypt(encrypted\_keywords.get(i))));

bloom.add(sb.toString());

}

for(int i=0;i<bloom.size();i++){

BigInteger input = new BigInteger(bloom.get(i).getBytes());

hash.add(PaillierEnc.Encryption(input).toString());

}

Socket socket = new Socket("localhost",3333);

ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());

ObjectInputStream in = new ObjectInputStream(socket.getInputStream());

Object req[]={"upload",user,file.getName(),enc,encrypted\_keywords,bloom,hash};

out.writeObject(req);

out.flush();

Object res[]=(Object[])in.readObject();

String response = (String)res[0];

area.append(response+"\n");

out.close();

in.close();

socket.close();

}catch(Exception e){

e.printStackTrace();

}

}

public void search(){

try{

String input = JOptionPane.showInputDialog(this,"Enter input query");

if(input != null){

input = input.trim().toLowerCase();

String arr[] = input.split("\\s+");

trapdoor.clear();

for(int i=0;i<arr.length;i++){

byte enc[] = AES.encrypt(arr[i].trim().getBytes());

trapdoor.add(enc);

area.append("Query = "+arr[i]+" Encrypted trapdoor = "+new String(enc)+"\n");

}

Socket socket = new Socket("localhost",3333);

ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());

ObjectInputStream in = new ObjectInputStream(socket.getInputStream());

Object req[]={"query",user,trapdoor};

out.writeObject(req);

out.flush();

Object res[]=(Object[])in.readObject();

String response = (String)res[0];

System.out.println("==="+response);

vlist = (ArrayList<VerificationObject>)res[1];

if(!response.equals("No record found for given querys")){

ViewSearchResult vsr = new ViewSearchResult();

String arr1[] = response.split(",");

int random = getRandom();

int size = arr1.length;

if(random == 1)

size = size - 1;

for(int i=0;i<size;i++){

String row[] = {arr1[i]};

vsr.dtm.addRow(row);

}

vsr.setVisible(true);

vsr.setSize(600,400);

}else{

JOptionPane.showMessageDialog(this,response);

}

}

}catch(Exception e){

e.printStackTrace();

}

}

public int getRandom(){

Random r = new Random();

return r.nextInt(2);

}

}

**BloomFilter.java**

package com;

import java.util.ArrayList;

public class BloomFilter{

static byte encrypt\_data[];

static int input[] = new int[50];

static ArrayList<Integer> list = new ArrayList<Integer>();

public static void generateBloom(int key,byte enc[]){

encrypt\_data = enc;

list.clear();

for(int i=0;i<encrypt\_data.length;i++){

String data = new String(Byte.toString(encrypt\_data[i]));

int bloom = data.hashCode()%key;

list.add(bloom);

}

for(int i=0;i<50;i++){

if(list.contains(i))

input[i] = 1;

else

input[i] = 0;

}

}

}

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## Implementation

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

**Acceptance Testing**

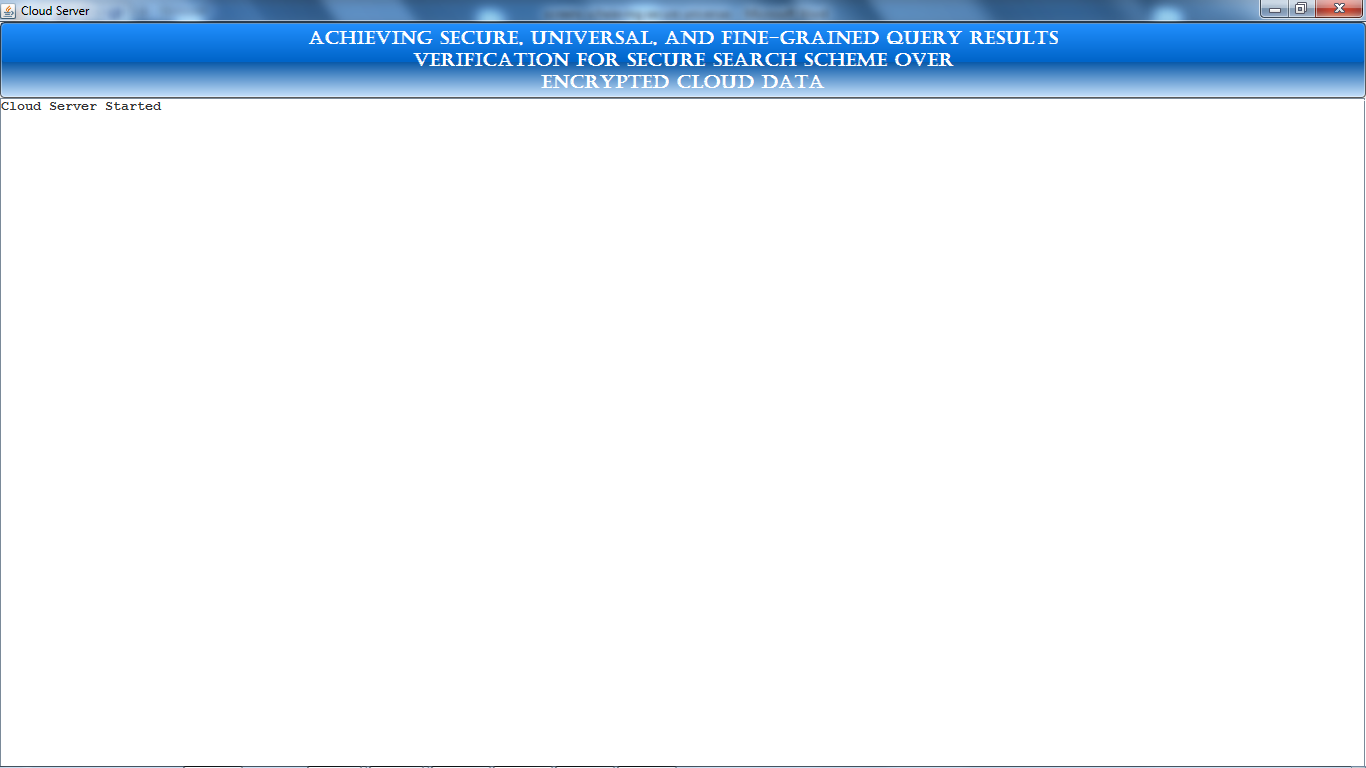
When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

**TEST CASES**

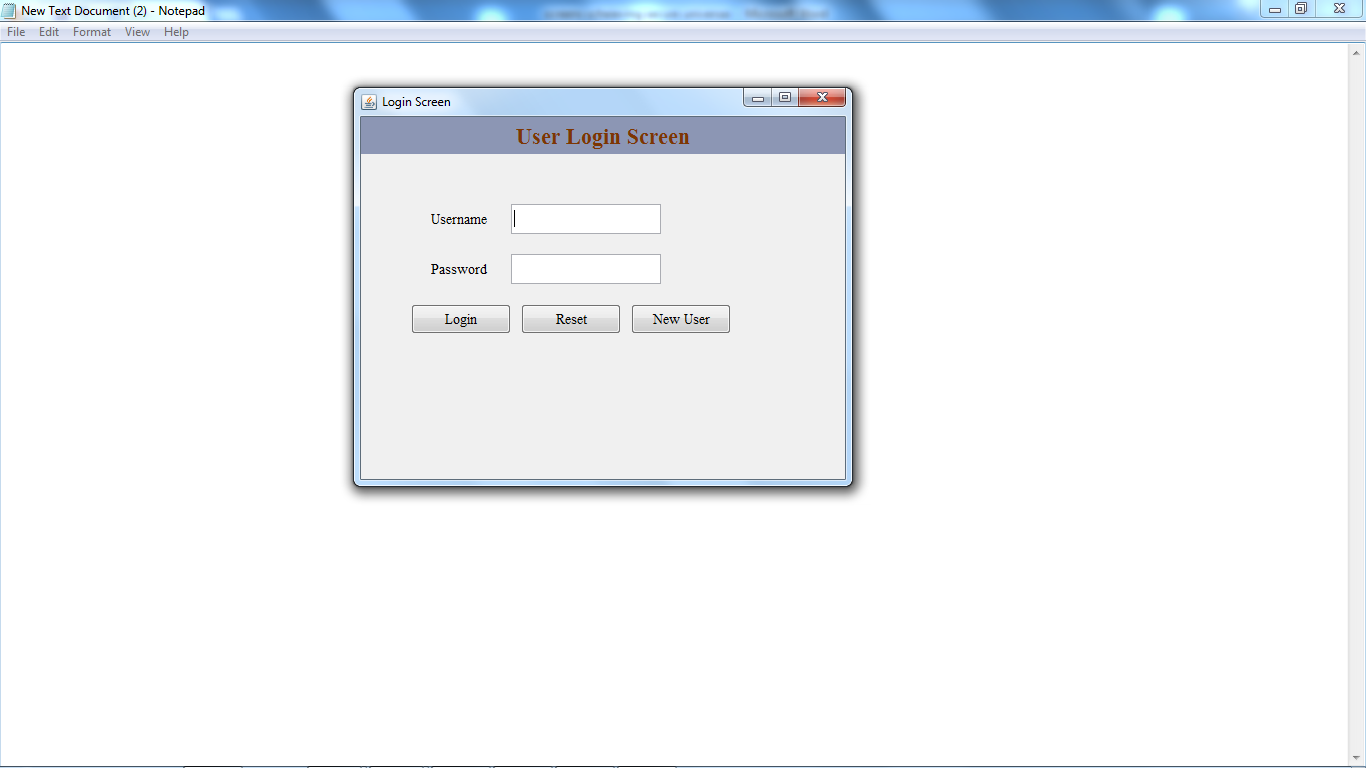
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Start Server | Test whether the cloud server will started or not | If server not started | Then no file will be stored in the cloud server | Server started | High | High |
| 02 | Login | Verify the Data Owner login | Without register | data owner cannot login into the system | Display the data owner home page | High | High |
| 03 | Upload file | Verify the either file uploaded or not | If data owner not upload the file | Then no search results will be displayed | Uploaded successfully and saved in cloud server | High | High |
| 04 | Verification | Test whether the verification will be done or not | Without verification object | Cloud server cannot verify the file | Decrypt & download the file and file saved in system directory | High | High |

**7. SCREEN SHOTS**

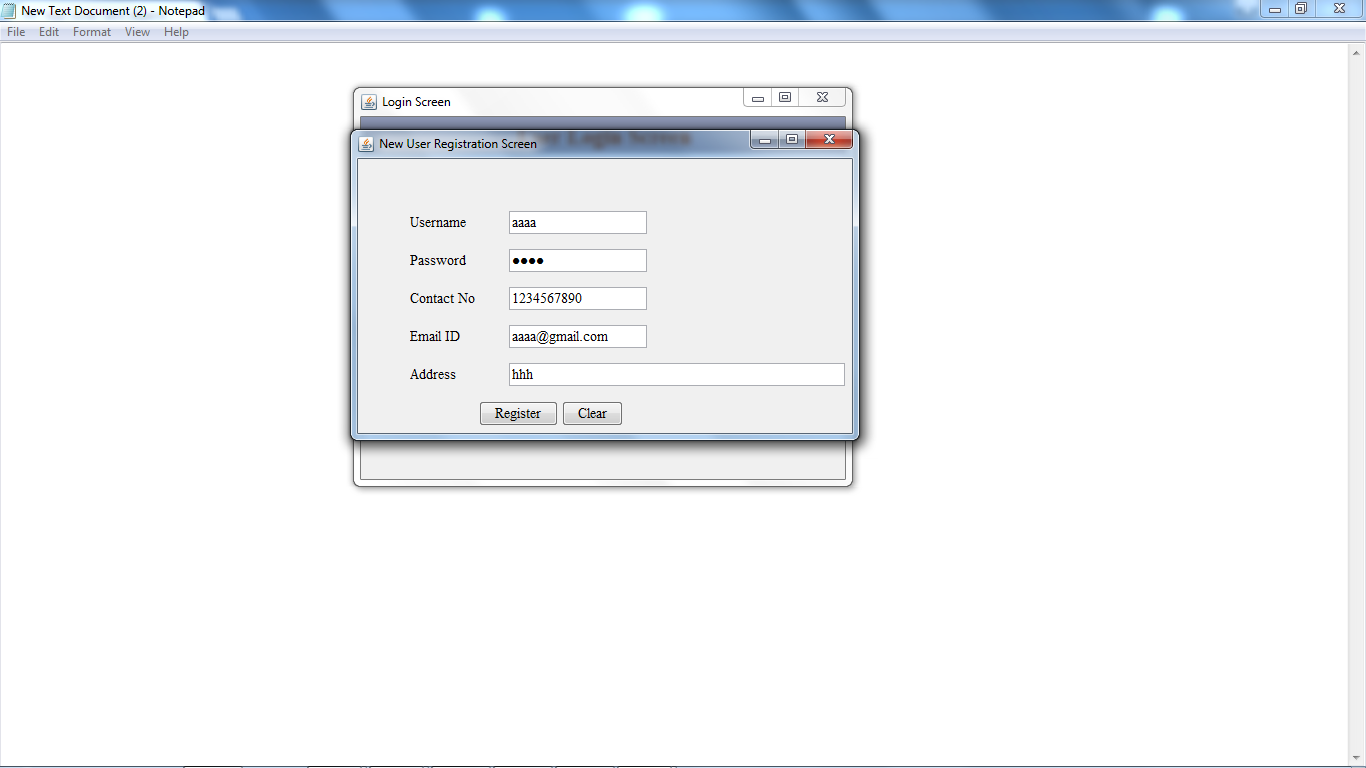
Cloud server:



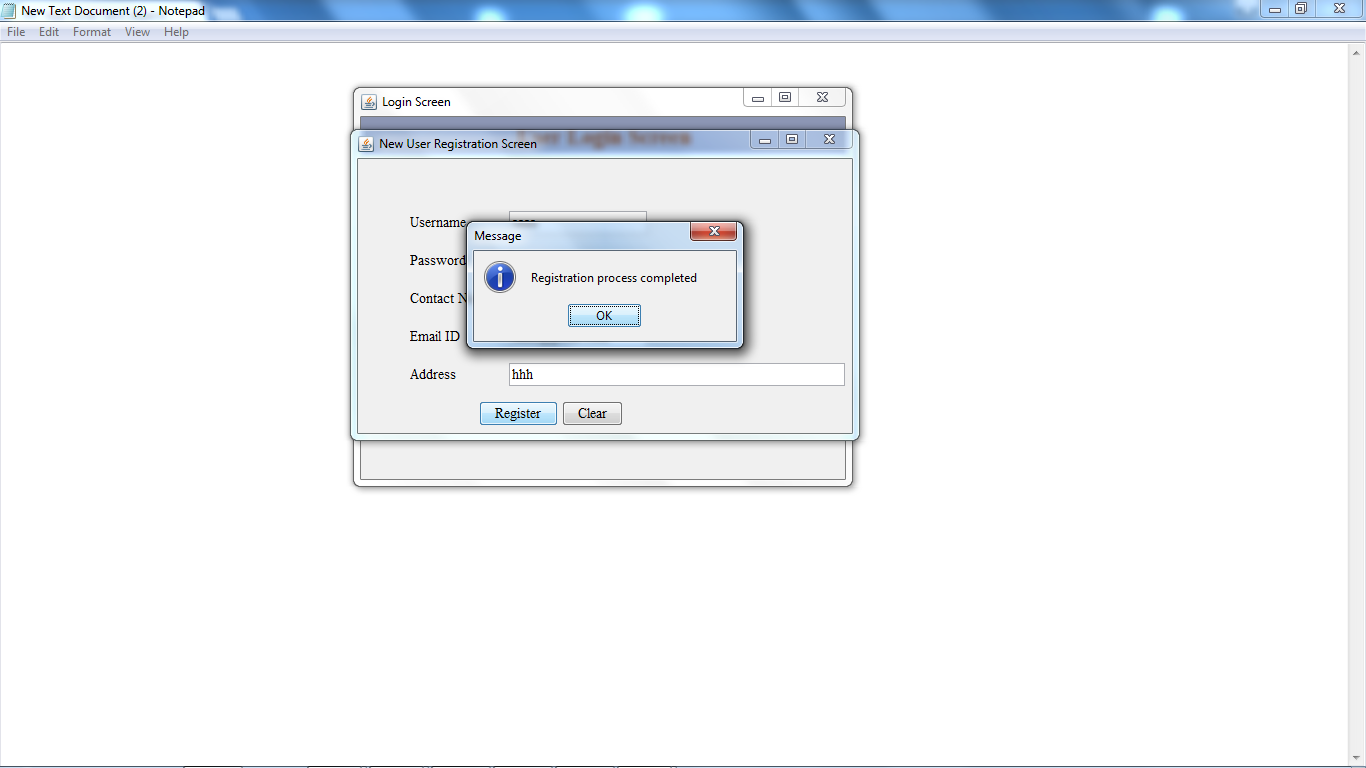
Data owner/ user welcome screen:



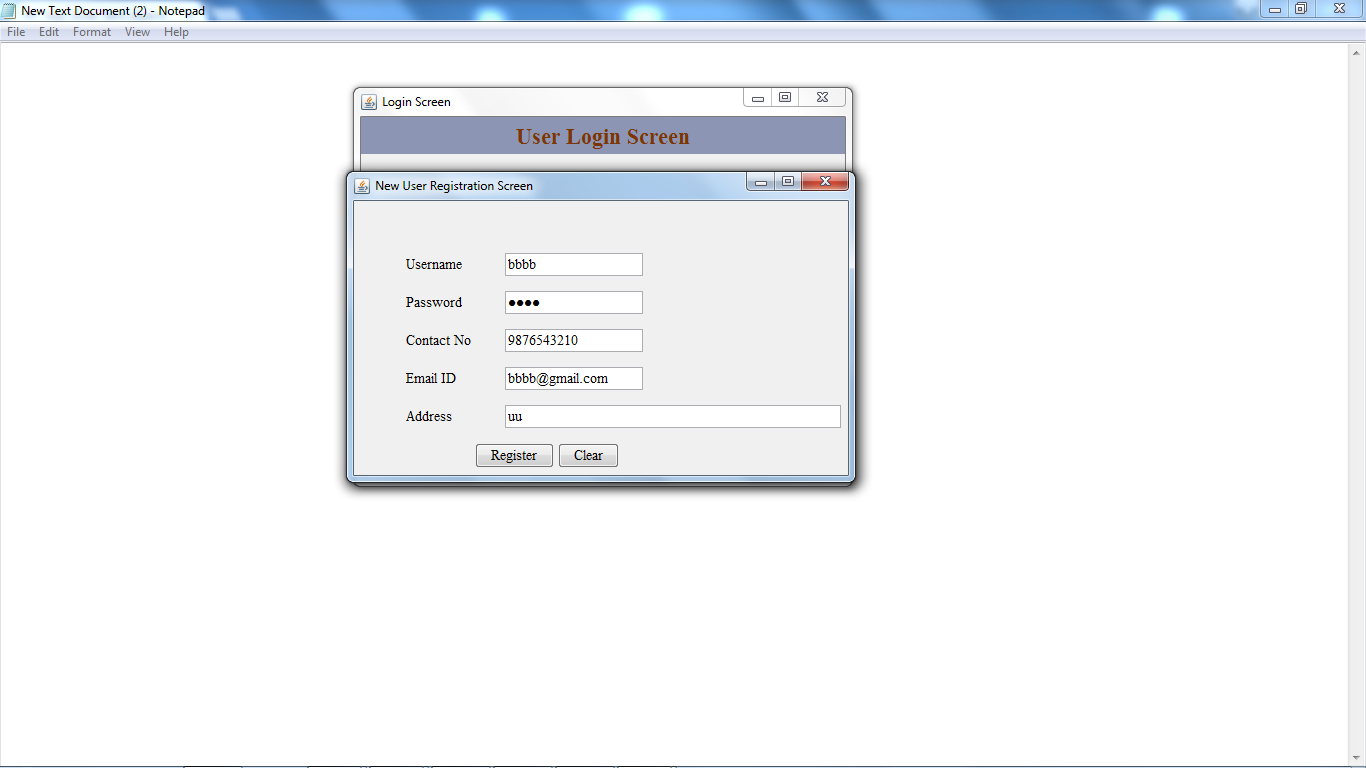
Click on new user to register a data owner:



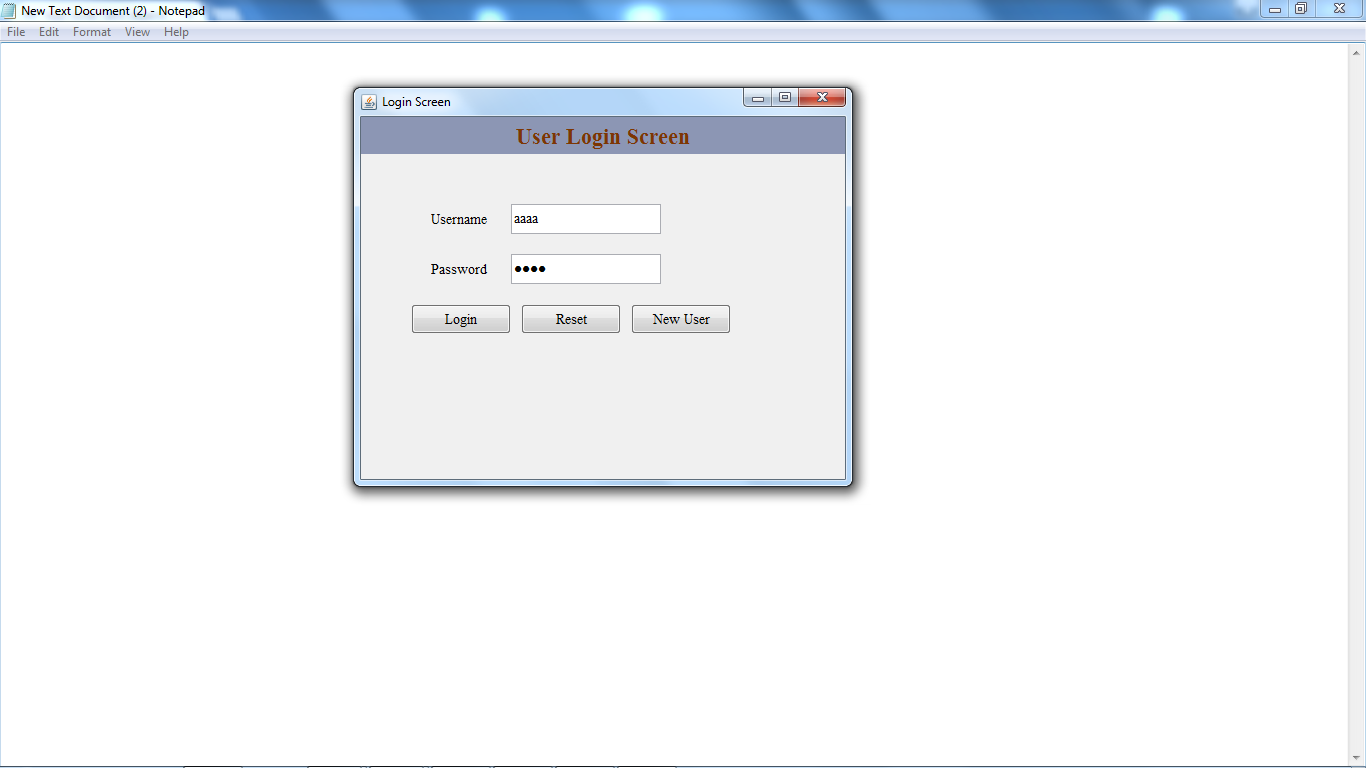
After successfully registering a data owner:



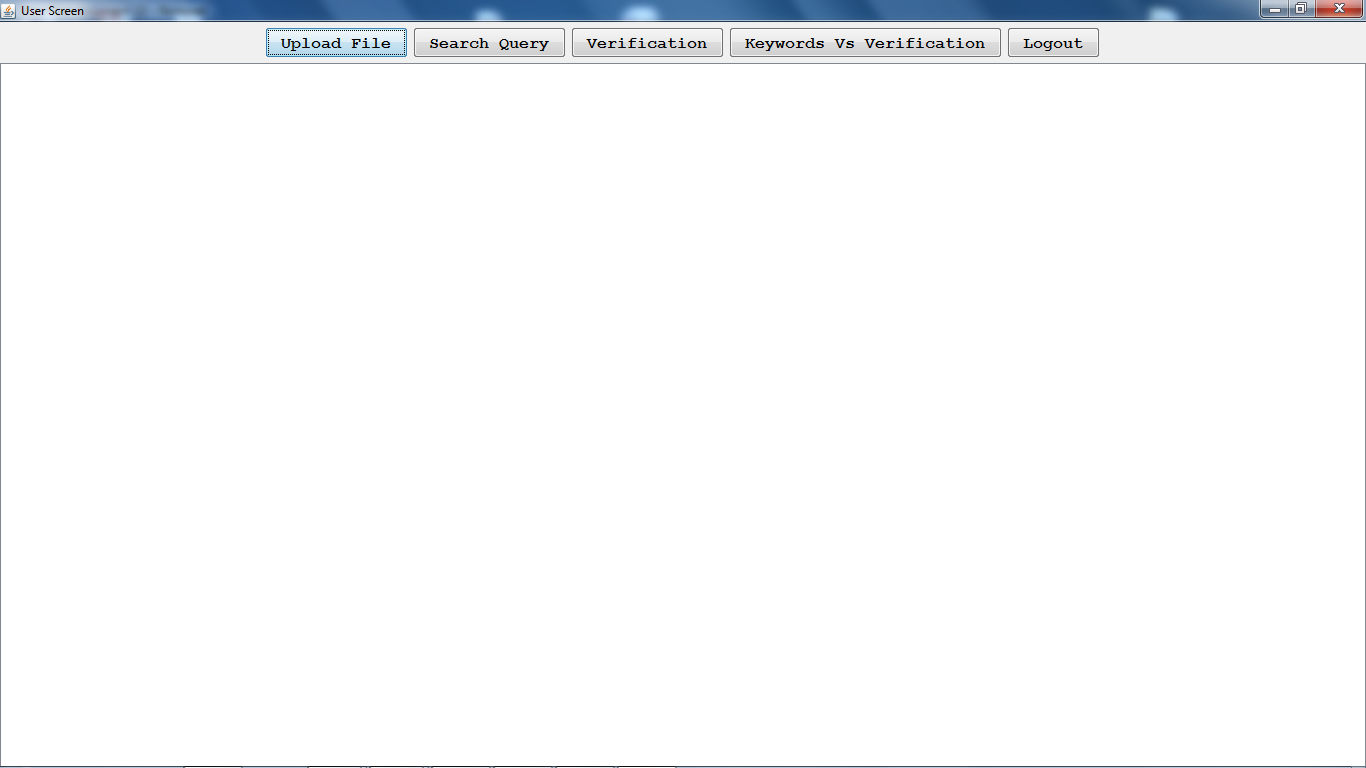
Registering another data owner:



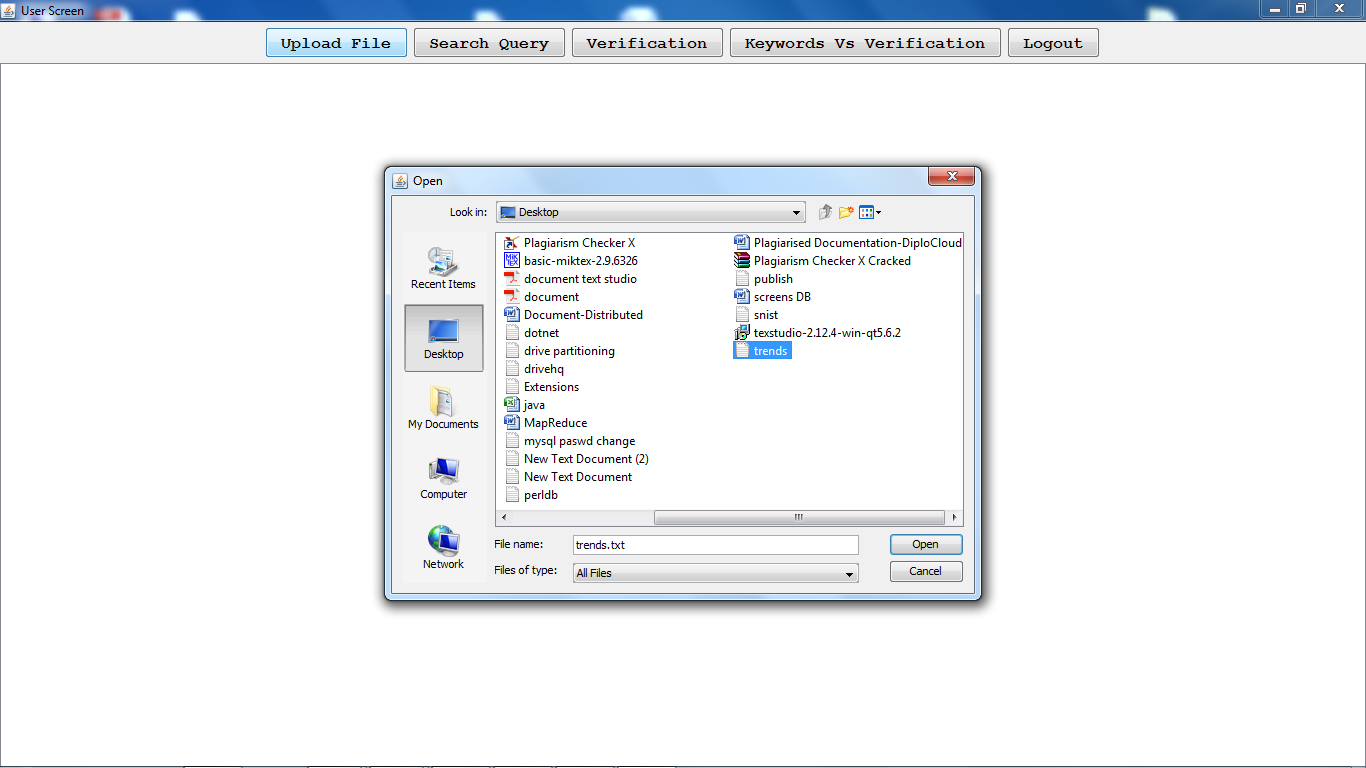
Login as a data owner:



Data owner home screen:

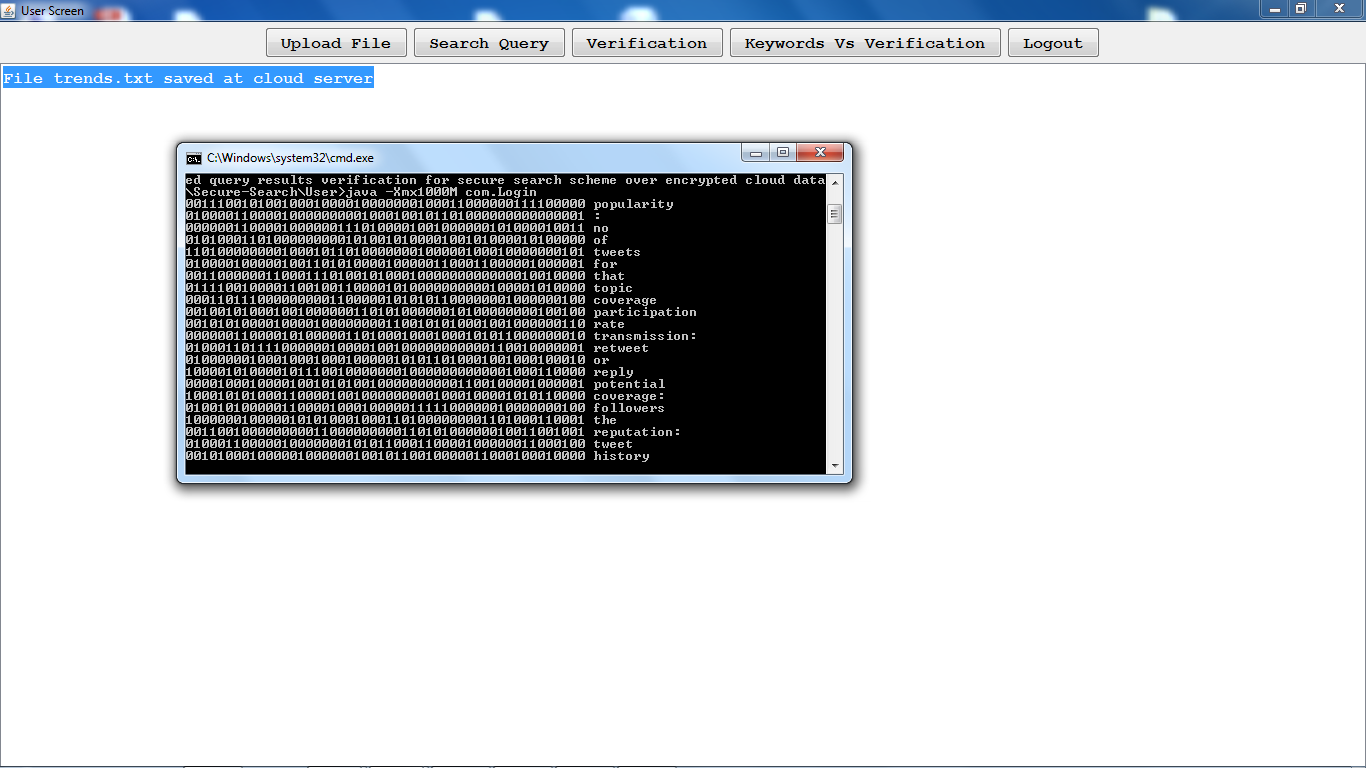


Upload a file onto cloud:

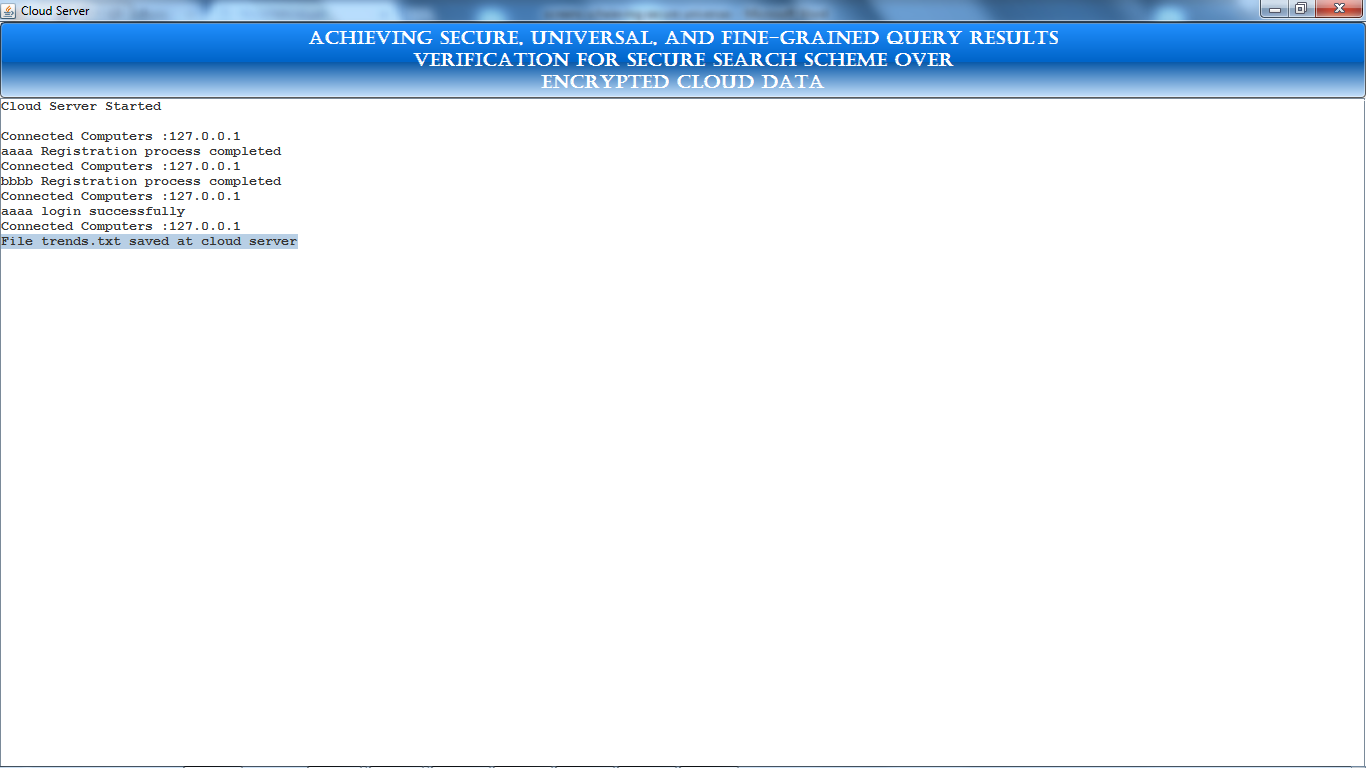


After uploading the file on to cloud

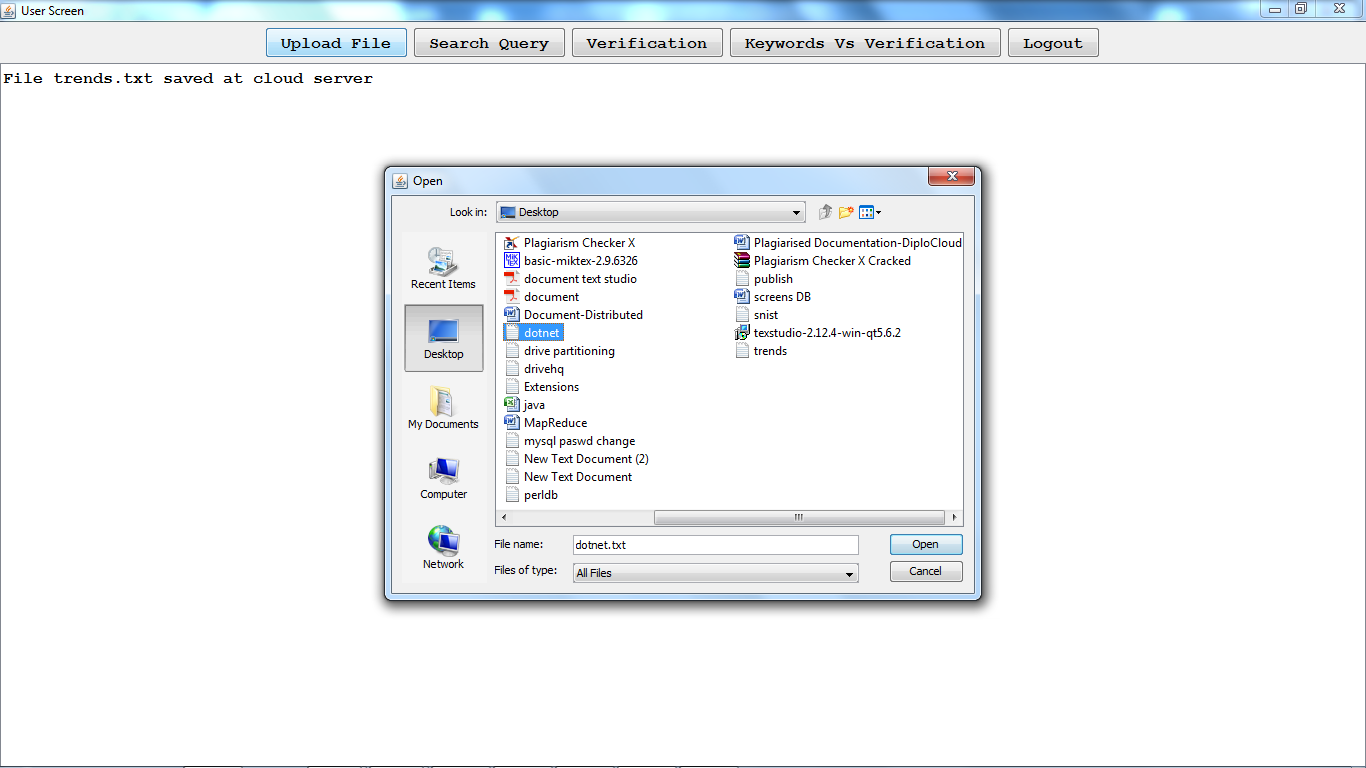
(the generated bloom filter..First for the given word get the bigrams, then generate hash for each bigram by using Paillier Encryption technique then generate some integer value for each hash code (here we are taking the array range up to 50) then generate bloom filter signature)



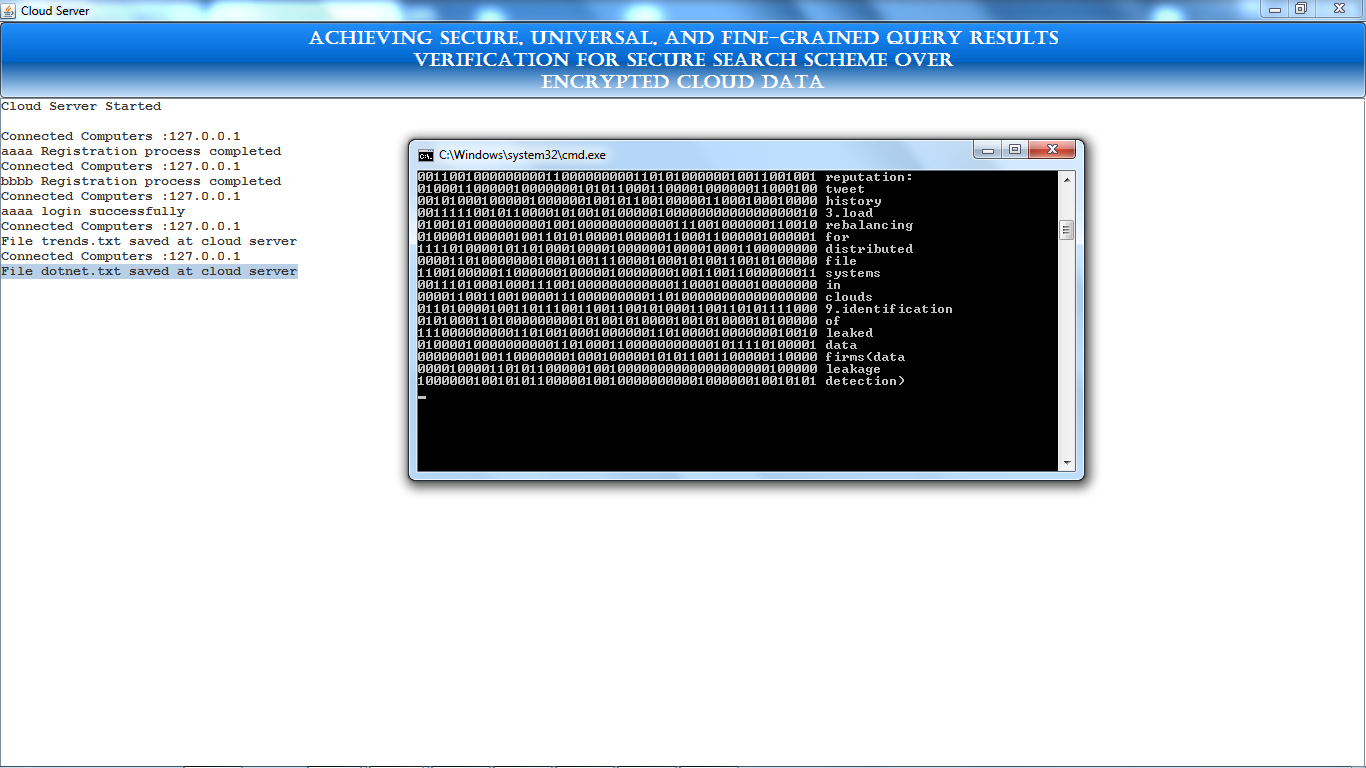
Cloud server after uploading the file: (the file will be saved at cloud in encrypted format by using AES algorithm)



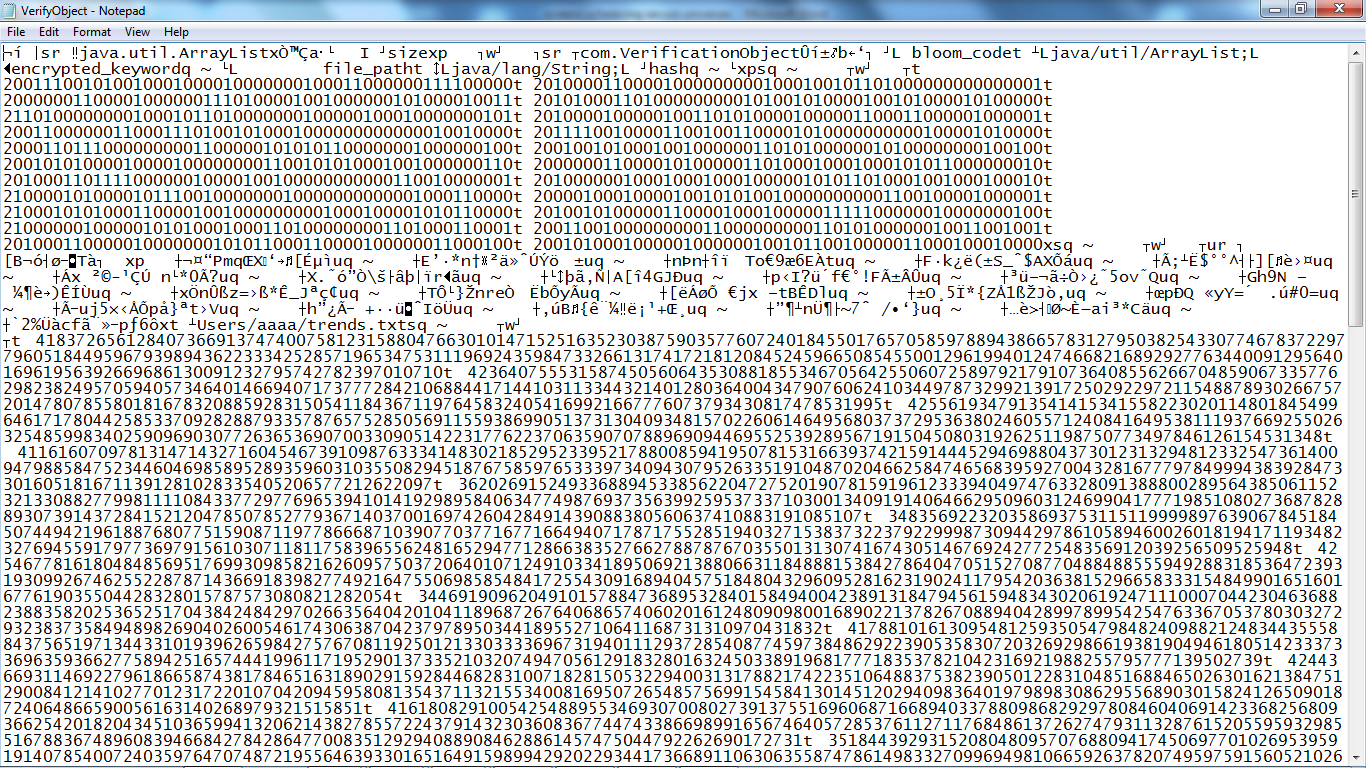
Uploading some other file onto cloud:



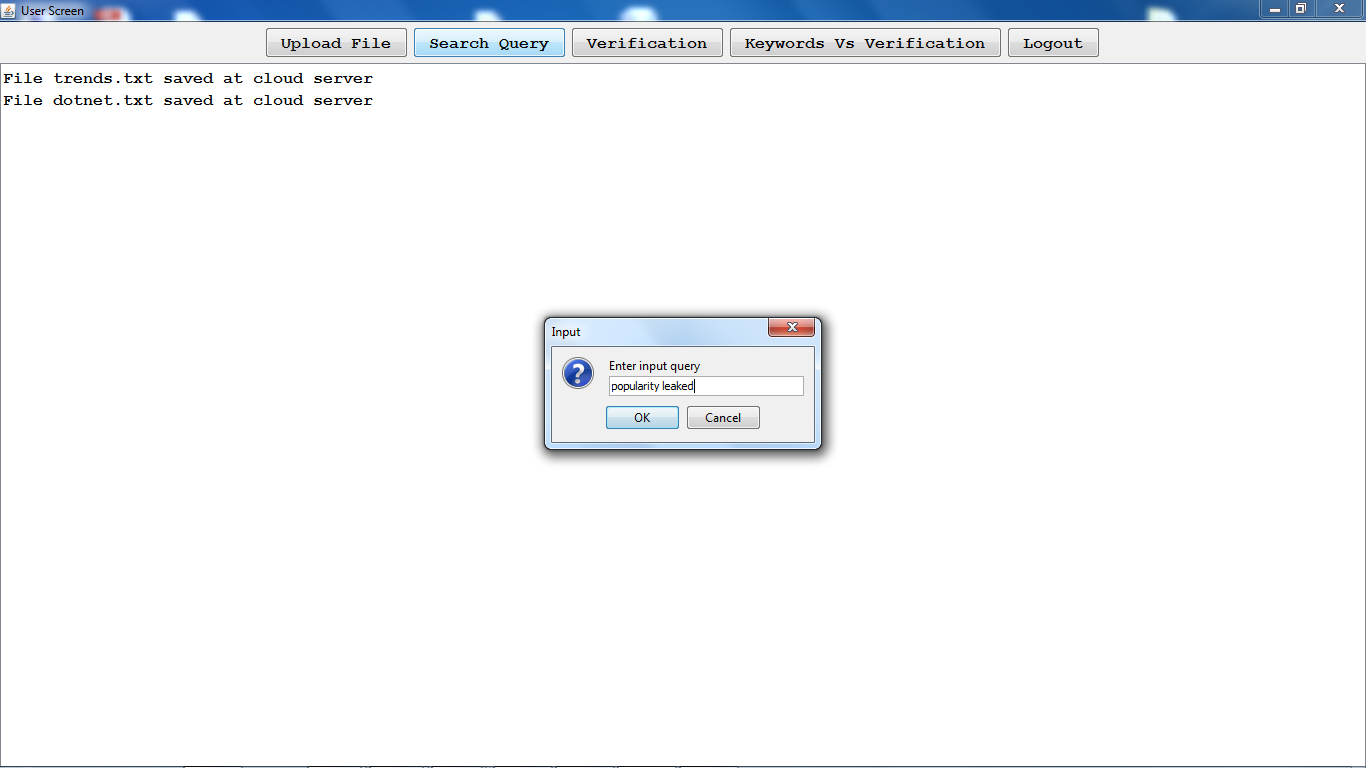
After successfully uploading:



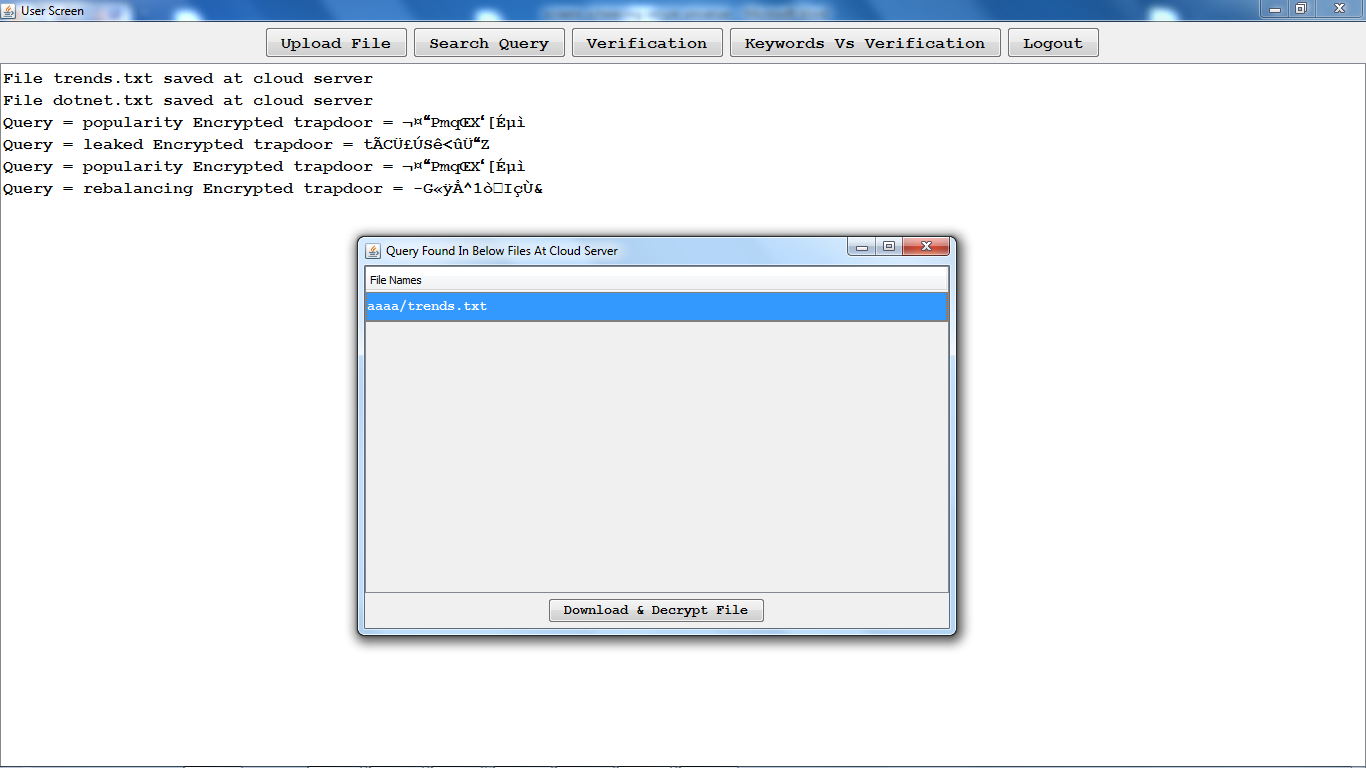
The verification object for the uploaded files will be created and saved at server side:



Search query (enter the keywords from both files) (as per this project to reduce the resources, we can give the search results from few of the uploaded files if the client is satisfied with that then he can download that file otherwise he will keep on search until he will receive the required docs)

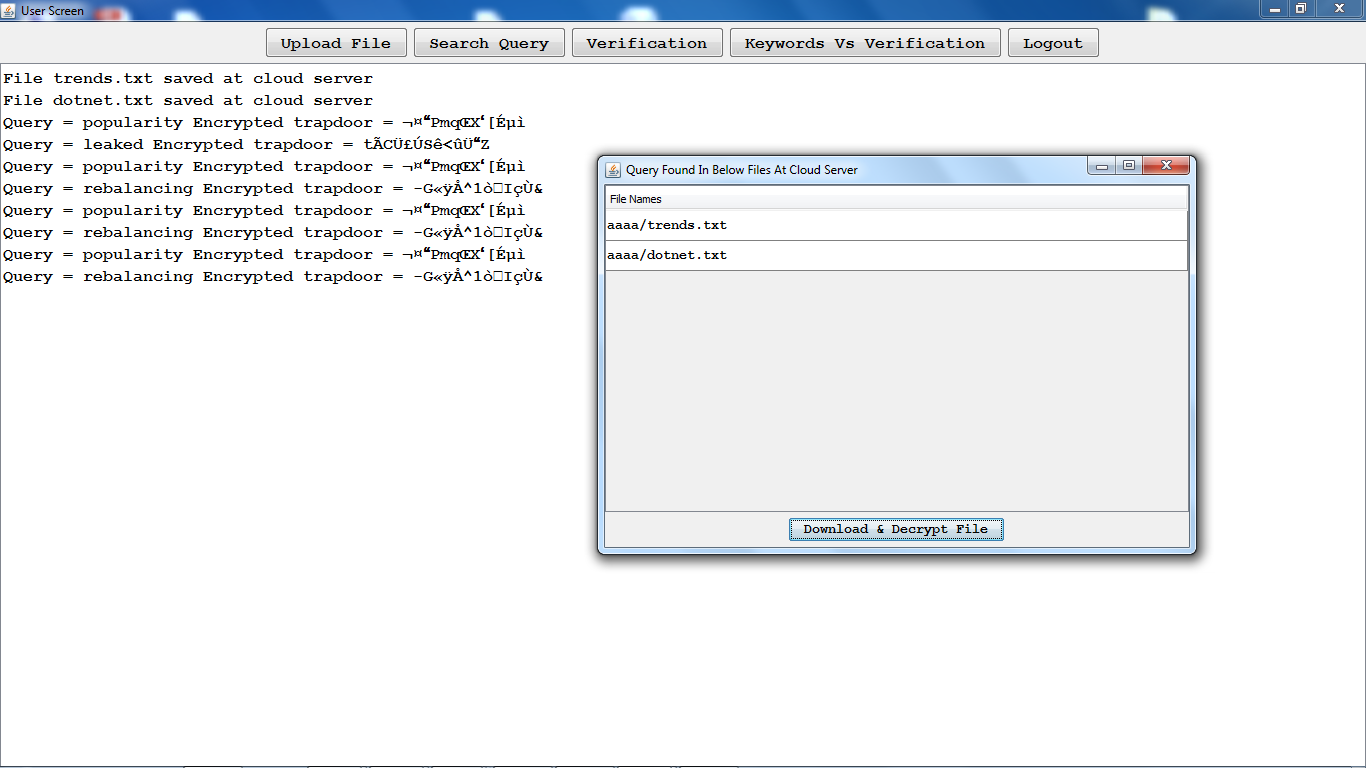


Then for the given query, trapdoor (hashcode) will be generated and searched from verification object for the given hashcode

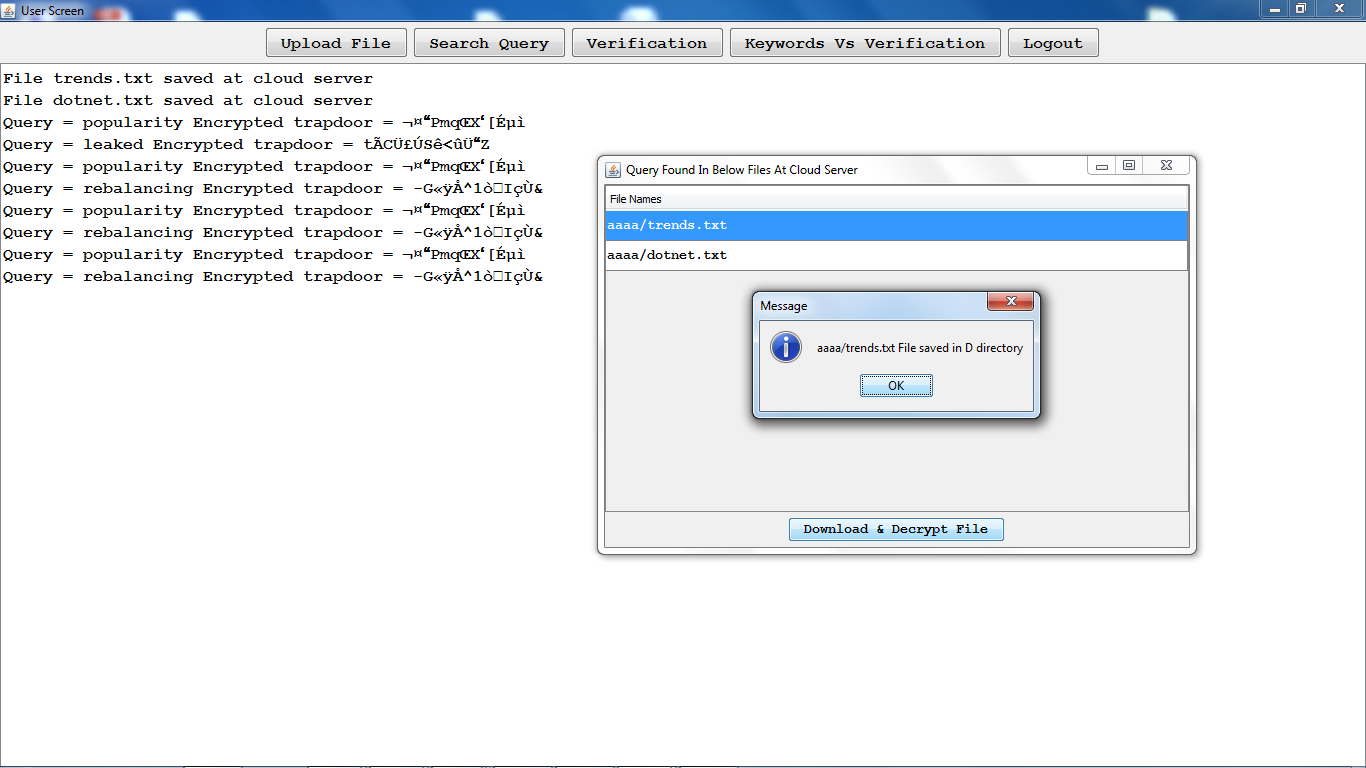


Here it shows the results from one file.

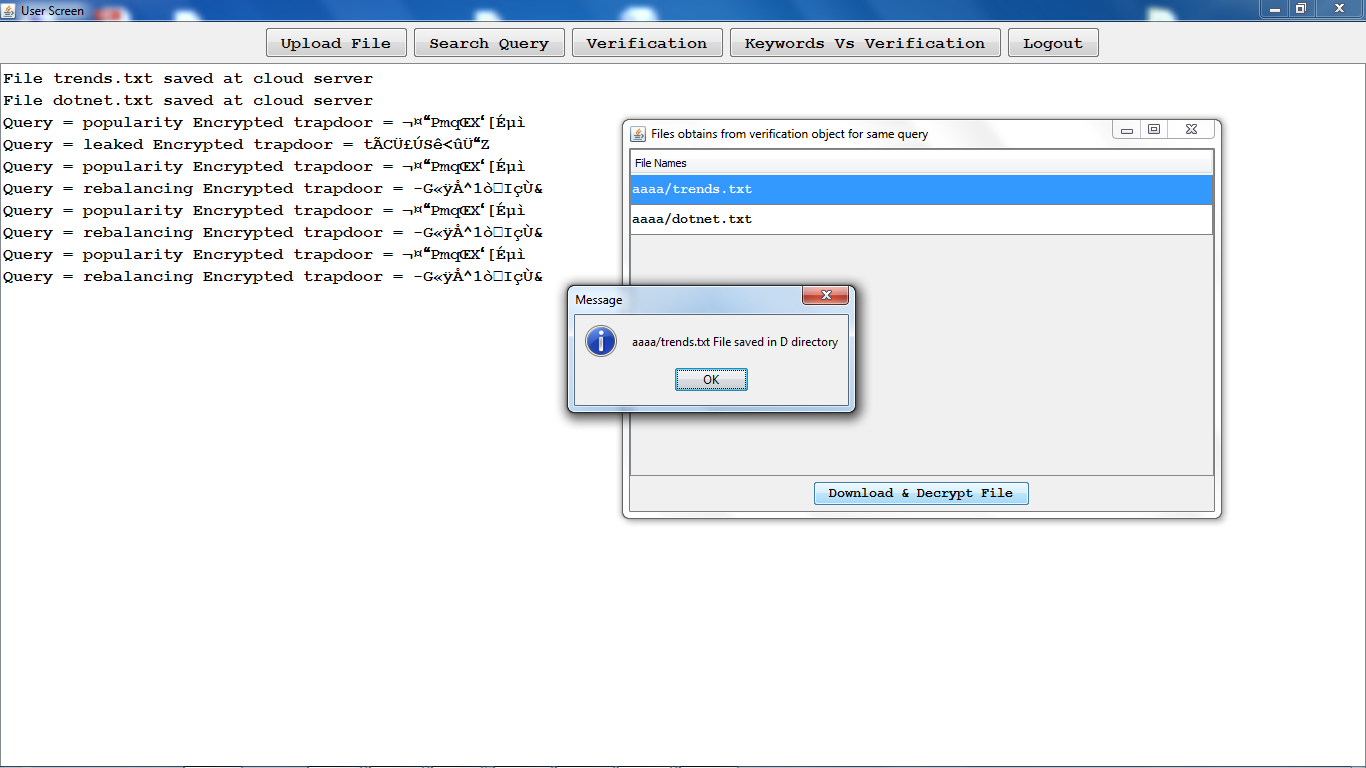
So if the user satisfied then he can download or else he will keep on search until he will get the desire output:



Select the required file and decrypt and download:

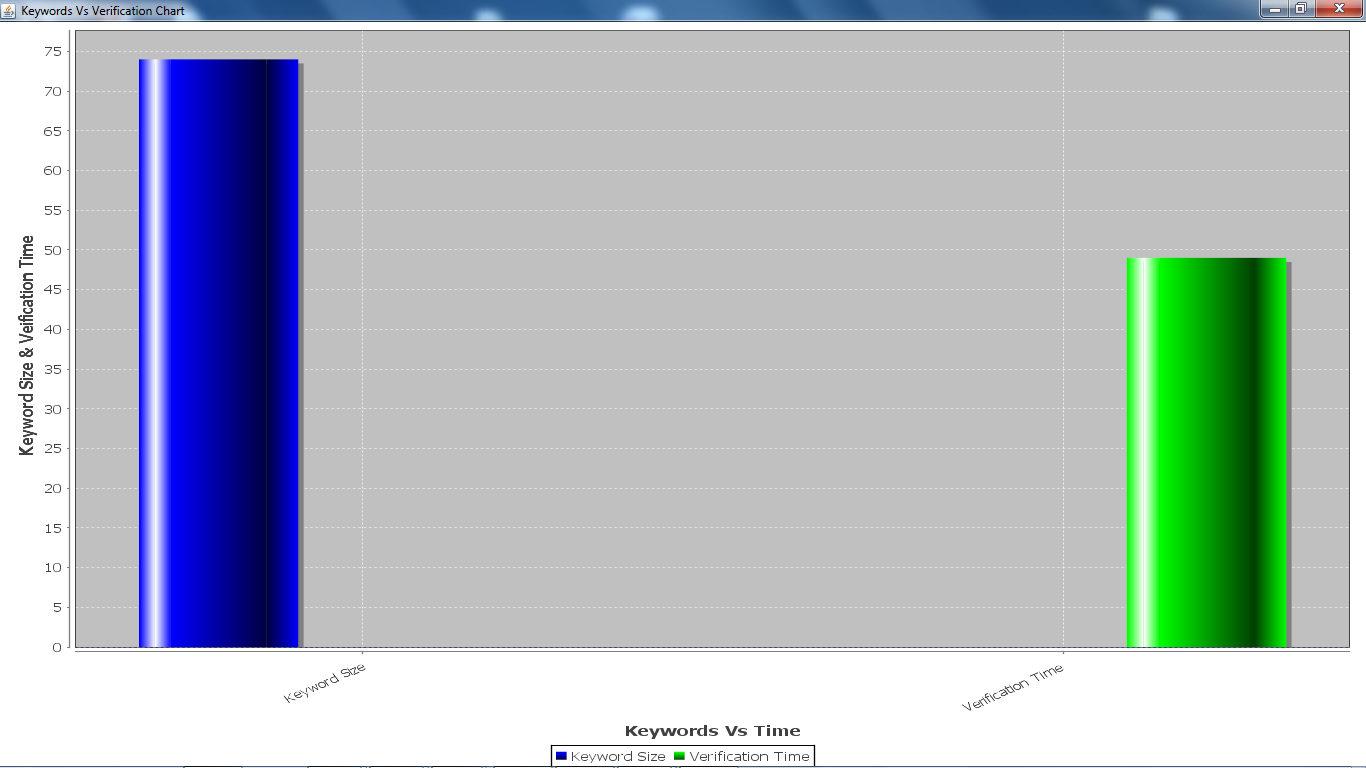


Verification:

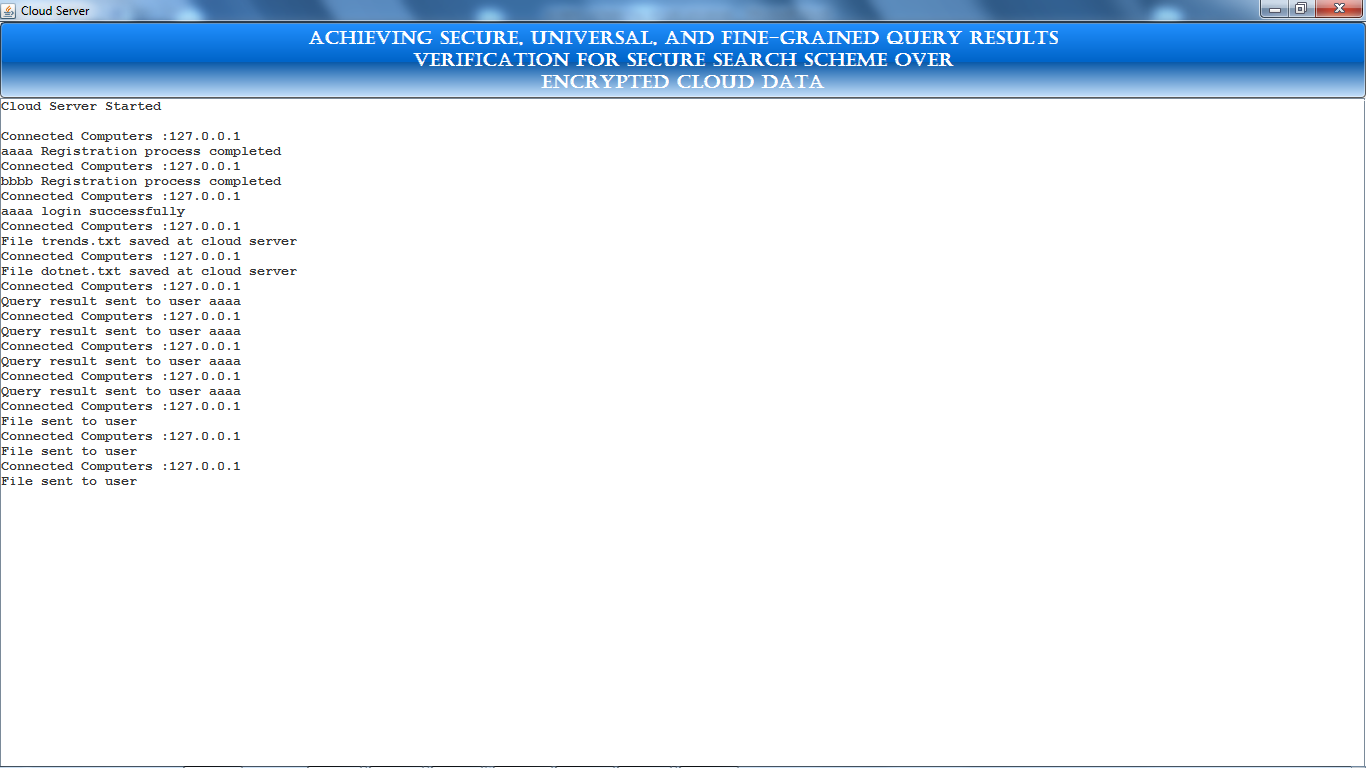


Keywords Vs verification chart:

(how many keywords are there in the uploaded files and how much time it has taken to verify)



The cloud server:



**8. CONCLUSION**

In this paper, we propose a secure, easily integrated, and fine-grained query results verification scheme for secure search over encrypted cloud data. Different from previous works, our scheme can verify the correctness of each encrypted query result or further accurately find out how many or which qualified data files are returned by the dishonest cloud server. A short signature technique is designed to guarantee the authenticity of verification object itself. Moreover, we design a secure verification object request technique, by which the cloud server knows nothing about which verification object is requested by the data user and actually returned by the cloud server. Performance and accuracy experiments demonstrate the validity and efficiency of our proposed scheme.

**9. BIBLIOGRAPHY**

1. K. Ren, C. Wang, and Q. Wang, “Security challenges for the public cloud,” IEEE Internet Computing, vol. 16, no. 1, pp. 69–73, 2012
2. S. Kamara and K. Lauter, “Cryptographic cloud storage,” in Springer RLCPS, January 2010
3. D. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in IEEE Symposiumon Security and Privacy, vol. 8, 2000, pp. 44–55.
4. E.-J.Goh, “Secure indexes,” IACR ePrint Cryptography Archive, http://eprint.iacr.org/2003/216, Tech. Rep., 2003.
5. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, “Public-key encryption with keyword search,” in EUROCRYPR, 2004, pp. 506–522.
6. R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption: improved deinitions and efficient constructions,” in ACM CCS, vol. 19, 2006, pp. 79–88.
7. M. Bellare, A. Boldyreva, and A. O’Neill, “Deterministic and efficiently searchable encryption,” in Springer CRYPTO, 2007.
8. K. Kurosawa and Y. Ohtaki, “Uc-secure searchable symmetric encryption,” Lecture Notes in Computer Science, vol. 7397, pp. 258– 274, 2012.
9. P. Xu, H. Jin, Q. Wu, and W. Wang, “Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack,” IEEE Transactions on Computers, vol. 62, no. 11, pp. 2266–2277, 2013.
10. S. Kamara and C. Papamanthou, “Parallel and dynamic searchable symmetric encryption,” in Financial Cryptography and Data Security. Springer Berlin Heidelberg, 2013, pp. 258–274
11. M. Naveed, M. Prabhakaran, and C. A. Gunter, “Dynamic searchable encryption via blind storage,” in IEEE S&P, May 2014, pp. 639–654.