**CHAPTER-1**

**INTRODUCTION**

**1.1 OVERVIEW**

The phrase Cloud computing is defined as a type of Internet-based computing because the term cloud is metaphor for the internet, where different services are delivered to an organization’s computers and devices through the Internet. Cloud computing is very promising for the Information Technology (IT) applications; however, there are still some issues to be solved for personal users and enterprises to store data and deploy applications in the Cloud computing environment. Data security is one of the most significant barriers to its adoption and it is followed by issues including compliance, privacy, trust, and legal matters. Therefore, one of the important goals is to maintain security and integrity of data stored in the cloud because of the critical nature of Cloud computing and large amounts of complex data it carries. The users concerns for security should be rectified first to make cloud environment trustworthy, so that it helps the users and enterprise to adopt it on large scale.

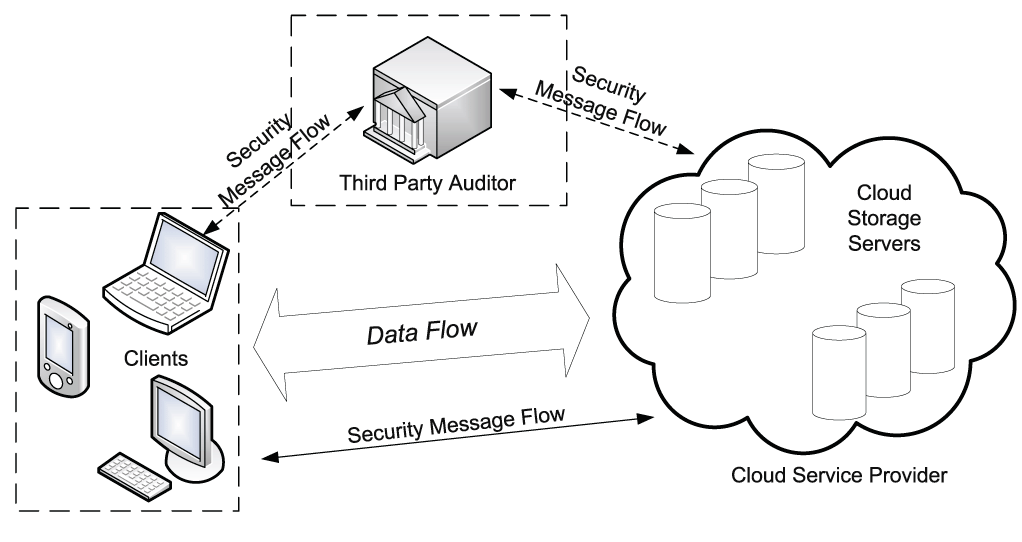
The data privacy, data protection, data availability, data location, and secure transmission are the foremost issues in cloud data security. Threats, data loss, service disruption, outside malicious attacks, and multi tenancy issues are the security challenges included in the cloud. The data should not be lost or modified by unauthorized users. Cloud computing providers are trusted to maintain data integrity and accuracy of data.Data confidentiality is also important aspect from user's point of view because they store their private or confidential data in the cloud.

Authentication and access control strategies are used to ensure data confidentiality. The data confidentiality could be addressed by increasing the cloud reliability and trustworthiness in Cloud computing. Therefore security, integrity, privacy and confidentiality of the stored data on the cloud should be considered and are important requirements from user’s point of view. To achieve all of these requirements, new methods or techniques should be developed and implemented.

**1.2**  **INTRODUCTION TO AUDITING**

Auditing is a process of verification of user data which can be carried out either by the user himself (data owner) or by a TPA. It helps to maintain the integrity of data stored on the cloud. The verifier’s roles are categorized into two: first one is private audit ability, in which only user or data owner is allowed to check the integrity of the stored data. No other person has the authority to question the server regarding the data. But it tends to increases verification overhead of the user. Second is public audit ability, which allows anyone, not just the client, to challenge the server and performs data verification check with the help of TPA. The TPA is an entity which is used so that it can act on behalf of the client. It has all the necessary expertise, capabilities, knowledge and professional skills which are required to handle the work of integrity verification and it also reduces the overhead of the client. It should not introduce any additional on-line burden to the cloud user. The three network entities viz. the client, cloud server and TPA are present in the cloud environment. The client stores data on the storage server provided by the cloud service provider (CSP). TPA keeps a check on client’s data by periodically verifying integrity of data on-demand and notifies client if any variation or fault is found in client’s data. Figure 1 shows the cloud data storage architecture.

To prevent TPA to infer the data using Privacy preserving is essential in the cloud server’s response while auditing. However, the privacy preserving requirement do not achieve the proposed scheme. Though data dynamics is an important feature to facilitate the data owners to insert, modify, and delete on a particular block of data, without changing the meta-data of other blocks, the techniques pro- posed in do not achieve data dynamics requirement. Meanwhile, the schemes like could not achieve batch auditing requirement which ensures that TPA should be capable enough to deal with the multiple numbers of simultaneous veriﬁcation requests from different DUs. This property is to save computation and communication cost between CSP and TPA. Unfortunately, the schemes use pairing based cryptographic operations which are intensive computation and need more time.

In this work, we propose a secure and efficient privacy preserving provable data possession scheme (SEPDP) for cloud storage. It operates in three phases, namely, key generation, signature generation and auditing phase. Most attractive feature of SEPDP is that it does not use any intensive computation like pairing based operation. Further, we extend SEPDP to support multiple data owners, batch auditing, and dynamic data operations. A probabilistic analysis to detect the integrity of the blocks stored at CSP. We evaluated the performance of the proposed scheme and compared with some of the existing popular mechanisms. We observe that the total time for verification carried out by TPA in the proposed scheme is less than that of the existing schemes. This signifies that SEPDP is efficient and suitable to implement the verification at the low powered devices.  **FIGURE 1: CLOUD DATA STORAGE ARCHITECTURE**

Users can encrypt data that is processed or stored within the cloud to prevent unauthorized access. Cloud computing poses privacy concerns because the service provider can access the data that is in the cloud at any time. It could accidentally or deliberately alter or delete information. . If necessary for purposes of law and order without a warrant third party can share information from cloude providers .

That is permitted in their privacy policies, which users must agree to before they start using cloud services. Solutions to privacy include policy and legislation as well as end users' choices for how data is stored.  This delivers great incentive to public cloud computing service providers to prioritize building and maintaining strong management of secure services.

**1.3 PROBLEM STATEMENT**

According to the Cloud Security Alliance, the top three threats in the cloud are Insecure Interfaces and API's, Data Loss & Leakage, and Hardware Failure—which accounted for 29%, 25% and 10% of all cloud security outages respectively.

There is the risk that end users do not understand the issues involved when signing on to a cloud service (persons sometimes don't read the many pages of the terms of service agreement, and just click "Accept" without reading).In a cloud provider platform being shared by different users there may be a possibility that information belonging to different customers resides on same data server. Additionally, [Eugene Schultz](https://en.wikipedia.org/wiki/Eugene_Schultz), chief technology officer at Imagined Security, said that hackers are spending substantial time and effort looking for ways to penetrate the cloud. "There are some real Achilles' heels in the cloud infrastructure that are making big holes for the bad guys to get into". Because data from hundreds or thousands of companies can be stored on large cloud servers , hackers can theoretically gain

"hyper jacking" is called as the control of huge stores of information through a single attack—a process . The Dropbox security breach, and Cloud 2014 leak some examples . Dropbox had been breached in October 2014, having over 7 million of its user’s passwords stolen by hackers in an effort to get monetary value from it by Bitcoins (BTC). By having these passwords, they are able to read private data as well as have this data be indexed by search engines (making the information public).

This is important now that cloud computing is becoming popular and required for some services to work, for example for an [intelligent personal assistant](https://en.wikipedia.org/wiki/Intelligent_personal_assistant) (Apple's [Siri](https://en.wikipedia.org/wiki/Siri) or [Google Now](https://en.wikipedia.org/wiki/Google_Now)). Fundamentally, private cloud is seen as more secure with higher levels of control for the owner, however public cloud is seen to be more flexible and requires less time and money investment from the user.

**1.4 OBJECTIVE**

Multiuser verifiable searchable Symmetric encryption (MVSSE) schema that achieves all the desirable features of a verifiable SSE & allows multiple users sto perform searching allows multiple users to perform searching MVSSE functionality works under the university composable security framework .

**1.5 ORGANIZATION OF THE REPORT**

Chapter 1 overview about the project and introduction to the project along with summary Chapter 2 deals with the literature survey of the related applications along with the summary . Chapter 3 explains the overview of the system , existing system , disadvantages of existing system , along with its summary . Chapter 4 proposes the overview of the project , proposed system, system architecture design and data flow . Chapter 5 explains the system specification , hardware requirement, software requirement and software description . Chapter 6 e the system implementation , list of modules with its description . Chapter 7 gives the experimental results of how the output is obtained and the performance of the project . chapter 8 concludes with the conclusion .

**CHAPTER - 2**

**LITERATURE SURVEY**

**Title**: **Privacy-Preserving and Regular Language Search over Encrypted Cloud Data[1]**

**Authors:** Kaitai Liang, Xinyi Huang, Fuchun Guo, and Joseph K. Liu

Using cloud-based storage service, users can remotely store their data to clouds but also enjoy the high quality data retrieval services, without the tedious and cumbersome local data storage and maintenance. However, the sole storage service cannot satisfy all desirable requirements of users. Over the last decade, privacy-preserving search over encrypted cloud data has been a meaningful and practical research topic for outsourced data security. The fact of remote cloud storage service that users cannot have full physical possession of their data makes the privacy data search a formidable mission. A naive solution is to delegate a trusted party to access the stored data and fulﬁll a search task. This, nevertheless, does not scale well in practice as the fully data access may easily yield harm for user privacy. To securely introduce an effective solution, we should guarantee the privacy of search contents, i.e. what a user wants to search, and return results, i.e. what a server returns to the user. Furthermore, we also need to guarantee privacy for the outsourced data, and bring no additional local search burden to user. In this paper, we design a novel privacy-preserving functional encryption based search mechanism over encrypted cloud data. A major advantage of our new primitive compared to the existing public key based search systems is that it supports an extreme expressive search mode, regular language search. Our security and performance analysis show that the proposed system is provably secure and more efficient than some searchable systems with high expressiveness.

**Title**:  **Dual-Server Public-Key Encryption with Keyword Search for Secure Cloud Storage[2]**

**Authors:** Rongmao Chen, Yi Mu, Guomin Yang, Fuchun Guo and Xiaofen Wang

Searchable encryption is of increasing interest for protecting the data privacy in secure searchable cloud storage. In this work, we investigate the security of a well-known cryptographic primitive, namely Public Key Encryption with Keyword Search (PEKS) which is very useful in many applications of cloud storage. Unfortunately, it has been shown that the traditional PEKS frame work suffers from an inherent insecurity called inside Keyword Guessing Attack (KGA) launched by the malicious server. To address this security vulnerability, we propose a new PEKS framework named Dual-Server Public Key Encryption with Keyword Search (DS-PEKS). As another main contribution, we define a new variant of the Smooth Projective Hash Functions (SPHFs) referred to as linear and homomorphic SPHF (LH-SPHF). We then show a generic construction of secure DS-PEKS from LH-SPHF. To illustrate the feasibility of our new framework, we provide an efficient instantiation of the general framework from a DDH-based LH-SPHF and show that it can achieve the strong security against inside KGA.

**Title**: **Server-Aided Public Key Encryption with Keyword Search[3]**

**Authors:** Rongmao Chen, Yi Mu, Xinyi Huang, Xiaofen Wang, Yongjun Wang

Public Key Encryption with Keyword Search (PEKS) is a well-known cryptographic primitive for secure searchable data encryption in cloud storage. Unfortunately, it is inherently subject to the (inside) off-line keyword guessing attack (KGA), which is against the data privacy of users. Existing countermeasures for dealing with this security issue mainly suffer from low efficiency and are impractical for real applications. In this work, we provide a practical and applicable treatment on this security vulnerability by formalizing a new PEKS system named Server-Aided Public Key Encryption with Keyword Search (SA-PEKS). In SA-PEKS, to generate the keyword cipher text/trapdoor, the user needs to query a semi-trusted third party called Keyword Server (KS) by running an authentication protocol and hence security against the off-line KGA can be obtained. We then introduce a universal transformation from any PEKS scheme to a secure SA-PEKS scheme using the deterministic blind signature. To illustrate its feasibility, we present the ﬁrst instantiation of SA-PEKS scheme by utilizing the FDH-RSA signature and the PEKS scheme proposed by Boneh et al. in Euro crypt 2004. Finally, we describe how to securely implement the client-KS protocol with a rate-limiting mechanism against on-line KGA and evaluate the performance of our solutions in experiments.

**Title**: **Generating Searchable Public-Key Cipher texts with Hidden Structures for Fast Keyword Search [4]**

**Authors:** Peng Xu, Qianhong Wu, Wei Wang, Willy Susilo

Existing semantically secure public-key search- able encryption schemes take search time linear with the total number of the cipher texts. This makes retrieval from large-scale databases prohibitive. To alleviate this problem, this paper proposes Searchable Public-Key Cipher texts with Hidden Structures (SPCHS) for keyword search as fast as possible without sacriﬁcing semantic security of the encrypted keywords. In SPCHS, all keyword-searchable cipher texts are structured by hidden relations, and with the search trapdoor corresponding to a keyword, the minimum information of the relations is disclosed to a search algorithm as the guidance to find all matching cipher texts efficiently. We construct a SPCHS scheme from scratch in which the cipher texts have a hidden star-like structure. We prove our scheme to be semantically secure in the Random Oracle (RO) model. The search complexity of our scheme is dependent on the actual number of the cipher texts containing the queried keyword, rather than the number of all cipher texts. Finally, we present a generic SPCHS construction from anonymous identity-based encryption and collision-free full-identity malleable Identity-Based Key Encapsulation Mechanism (IBKEM) with anonymity. We illustrate two collision-free full-identity malleable IBKEM instances, which are semantically secure and anonymous, respectively, in the RO and standard models. The latter instance enables us to construct an SPCHS scheme with semantic security in the standard model.

**Title**: **Boolean Searchable Symmetric Encryption with Worst-Case Sub-Linear Complexity [5]**

**Authors:** Seny Kamara, Tarik Moataz

Recent work on searchable symmetric encryption (SSE) has focused on increasing its expressiveness. A notable example is the OXT construction (Cash et al., CRYPTO ’13) which is the ﬁrst SSE scheme to support conjunctive keyword queries with sub-linear search complexity. While OXT eﬃciently supports disjunctive and Boolean queries that can be expressed in searchable normal form, it can only handle arbitrary disjunctive and Boolean queries in linear time. This motivates the problem of designing expressive SSE schemes with worst-case sub-linear search; that is, schemes that remain highly eﬃcient for any keyword query. In this work, we address this problem and propose non-interactive highly eﬃcient SSE schemes that handle arbitrary disjunctive and Boolean queries with worst-case sub-linear search and optimal communication complexity.

**2.1 SUMMARY**

As from the above survey , it is understood that the multi user verifiable searchable symmetric encyption is the research and searching . It will be more beneficial to the user as they can create event and also can edit it . the user can make this device as user-friendly to feed the details etc . Hence , the user can also control the devices from single voice command .

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 . EXISTING SYSTEM**

* Existing system uses the Steam cipher and block cipher operations.
* The intuition of keeping privacy of document contents described in security model.
* The content of a document cannot be deduced from the its index and indexes of other documents which have been queried before.
* This scheme realizes the conjunctive queries with negative keyword search as well. Besides, by converting Boolean queries to searchable normal form (SNF for short).

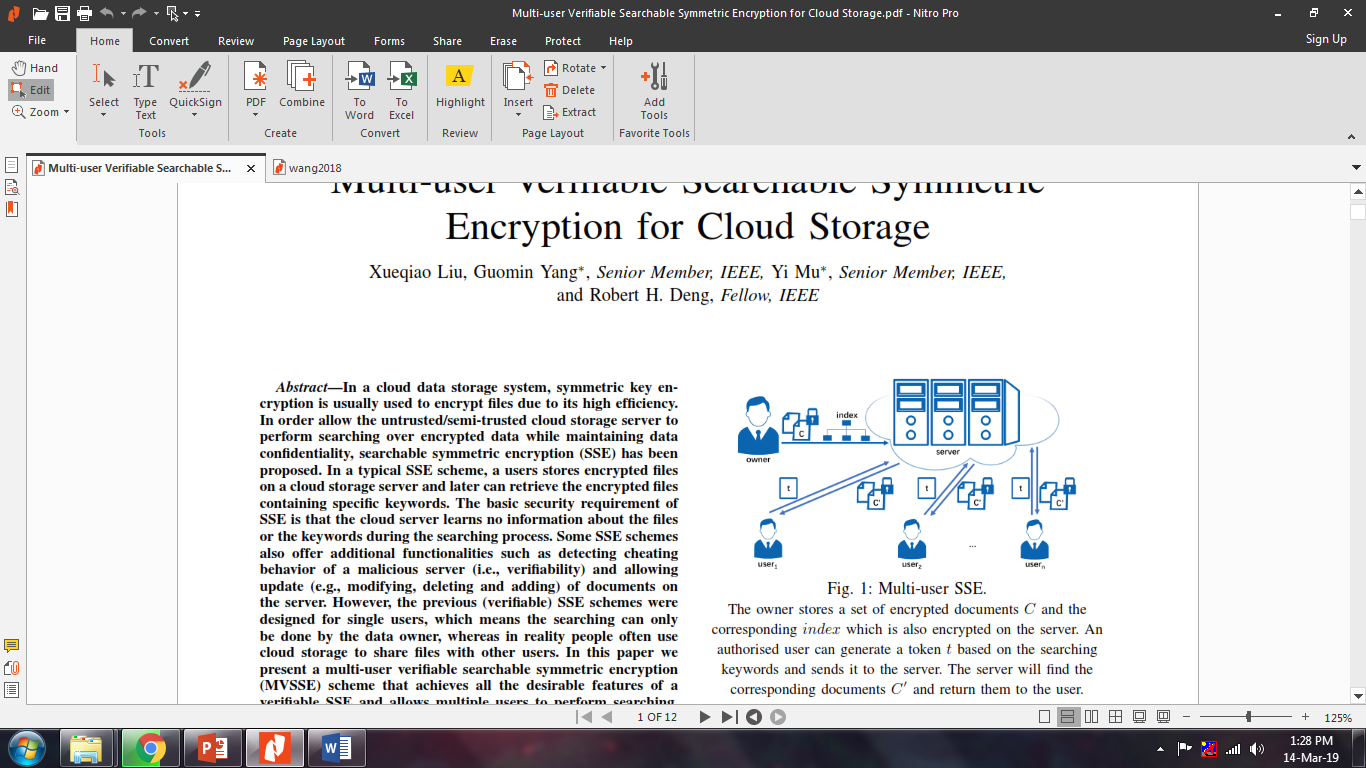
**3.1.1. DRAWBACKS**

* The searching time of the scheme is linear to the length of the document.
* Security is less.

**3.2. PROPOSED SYSTEM**

* To overcome the existing system problems the MVSSE is used.
* In this authorized data users are permitted to directly search documents stored on the cloud storage by preparing a trapdoor associated with some searching keywords.
* The cloud server returns not only searching results but also a proof.
* The data user can test the correctness and completeness of the searching results with the proof.
* The system allows the data owner to add, modify and delete documents efficiently.

**3.2.1 SYSTEM ARCHETECTURE DESIGN**



**FIGURE 2: ARCHITECTURE FOR MULTI USER VERIFIABLE**

**SEARCHABLE SYMMETRIC ENCYPTION**

* These operations should not inﬂuenced the index items related to other documents in order to minimize the cost.

**3.2.2 ADVANTAGES**:

* To process multi-keyword queries efficiently this system is used.
* The cloud server is not be able to learn either the outsourced documents or the related index.
* Other data users should not be inﬂuenced when a new data user is added into the system or an existing data user is revoked .
* Multiple user can use the system at a time and access the file at any time how has authority .

**3.1.3 PROPOSED SYSTEM BLOCK DIAGRAM**

**UPDATA FILE USING POLICY KEY**

**DATA**

**OWNER**

**POLICY KEY**

**UPDATA**

**VIEW**

**FILE**

CCFFFDGYLNVG

**SERVER**

**GET POLICY KEY AND SECRET KEY**

**VIEW**

**FILE**

**REQUEST**

**FILE ACCESS**

**DATA**

**USER**

**CHAPTER 4**

**IMPLEMENTATION**

**4.1. MODULE**

1. Registration
2. Login
3. Data Provider
4. Data User
5. Cloud Server

**4.1.1. REGISTRATION**

* The registration module allow the user to create login username and the password by submitting their information like mail id, phone number, name, etc.
* By registering the network or cloud the user can gain access to the resources stored in the cloud.

**4.1.2. LOGIN**

* In this module the user can login by using their unique username and password.
* The login module verifies the user given username and password with the stored username and password in the cloud.
* If the username and password is matched the user can access the resources.
* If it does not match the user does not allowed accessing the resource.

**4.1.3. DATA PROVIDER**

* The data providers are allowed to upload the file to the cloud server.
* The data providers have the private key. This key is used to perform the encryption operation and also the decryption operation.
* If the user needs to view the uploaded file the data provider need to share their key to the users.

**4.1.4. DATA USER**

* The user can gain access to the cloud to use the cloud resource by registering on the cloud.
* If the user want to view and download the file uploaded by the owner. The users need to gain the two different key decryption key and the policy key to download the file.

**4.1.5. CLOUD SERVER**

* The cloud servers are responsible for the encryption and also the decryption operation.
* They are responsible for updating the data owner policy key.

**4.2 ALGORITHM**

**4.2.1 SYMMETRIC ENCRYPTION**

DES works on bits, or binary numbers--the 0s and 1s common to digital computers. Each group of four bits makes up a hexadecimal, or base 16, number. Binary "0001" is equal to the hexadecimal number "1", binary "1000" is equal to the hexadecimal number "8", "1001" is equal to the hexadecimal number "9", "1010" is equal to the hexadecimal number "A", and "1111" is equal to the hexadecimal number "F".

DES works by encrypting groups of 64 message bits, which is the same as 16 hexadecimal numbers. To do the encryption, DES uses "keys" where apparently 16 hexadecimal numbers long, or apparently 64 bits long are also. However, every 8th key bit is ignored in the DES algorithm, so that the effective key size is 56 bits. But, in any case, 64 bits (16 hexadecimal digits) is the round number upon which DES is organized.

For example, if we take the plaintext message "8787878787878787", and encrypt it with the DES key "0E329232EA6D0D73", we end up with the cipher text "0000000000000000". If the cipher text is decrypted with the same secret DES key "0E329232EA6D0D73", the result is the original plaintext "8787878787878787".

This example is neat and orderly because our plaintext was exactly 64 bits long. The same would be true if the plaintext happened to be a multiple of 64 bits. But most messages will not fall into this category. They will not be an exact multiple of 64 bits (that is, an exact multiple of 16 hexadecimal numbers).

The plaintext message "Your lips are smoother than Vaseline" is, in hexadecimal, "596F7572206C6970 732061726520736D 6F6F746865722074 68616E2076617365 6C696E650D0A".

(Note here that the first 72 hexadecimal digits represent the English message, while "0D" is hexadecimal for Carriage Return, and "0A" is hexadecimal for Line Feed, showing that the message file has terminated.) We then pad this message with some 0s on the end, to get a total of 80 hexadecimal digits:

"596F7572206C6970 732061726520736D 6F6F746865722074 68616E2076617365 6C696E650D0A0000".

"C0999FDDE378D7ED 727DA00BCA5A84EE 47F269A4D6438190 9DD52F78F5358499 828AC9B453E0E653".

This is the secret code that can be transmitted or stored. Decrypting the cipher text restores the original message "Your lips are smoother than Vaseline". (Think how much better off Bill Clinton would be today, if Monica Lewinsky had used encryption on her Pentagon computer!)

**DES WORKING IN DETAIL**

DES is a block cipher--meaning it operates on plaintext blocks of a given size (64-bits) and returns cipher text blocks of the same size. Thus DES results in a permutation among the 2^64 (read this as: "2 to the 64th power") possible arrangements of 64 bits, each of which may be either 0 or 1. Each block of 64 bits is divided into two blocks of 32 bits each, a left half block L and a right half R. (This division is only used in certain operations.)

Example: Let M be the plain text message M = 0123456789ABCDEF, where M is in hexadecimal (base 16) format. Rewriting M in binary format, we get the 64-bit block of text:

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

L = 0000 0001 0010 0011 0100 0101 0110 0111

R = 1000 1001 1010 1011 1100 1101 1110 1111

The first bit of M is "0". The last bit is "1". We read from left to right.

DES operates on the 64-bit blocks using key sizes of 56- bits. The keys are ++actually stored as being 64 bits long, but every 8th bit in the key is not used (i.e. bits numbered 8, 16, 24, 32, 40, 48, 56, and 64). However, we will nevertheless number the bits from 1 to 64, going left to right, in the following calculations. But, as you will see, the eight bits just mentioned get eliminated when we create subkeys.

Example: Let K be the hexadecimal key K = 133457799BBCDFF1. This gives us as the binary key (setting 1 = 0001, 3 = 0011, etc., and grouping together every eight bits, of which the last one in each group will be unused):

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

**CHAPTER 5**

**SOFTWARE ENVIRONMENT**

**Java Technology**

Java technology is both a programming language and a platform.

**The Java Programming Language**

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* Simple
* Architecture neutral
* Object oriented
* Portable
* Distributed
* High performance
* Interpreted
* Multithreaded
* Robust
* Dynamic
* Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



**FIGURE 3 : WORKING OF COMPILER**

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM.

That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



**FIGURE 4 : EXECUTION OF JAVA PROGRAM**

**The Java Platform**

A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

* The Java Virtual Machine (Java VM)
* The Java Application Programming Interface (Java API)

You’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



**FIGURE 5 : DEPICTS A PROGRAM RUN ON JAVA PLATFORM**

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



**FIGURE 6 : DEPICTS WHAT IS INCLUDED IN THE JAVA 2 SDK.**

How Will Java Technology Change My Life?

We can’t promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

* **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it’s easy to learn, especially for programmers already familiar with C or C++.
* **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
* **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people’s tested code and introduce fewer bugs.
* **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
* **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaTM Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
* **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
* **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

**ODBC**

In computing , open Database connectivity (ODBC) is a standard application programming interface (API) for accessing database management systems (DBMS) . The designers of ODBC aimed to make it independent of database systems and operating systems .

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of sODBC is that it isn’t as efficient as talking directly to the native database interface.

**JDBC**

Java Database connectivity (JDBC) is an application programming interface (API) for the programming language java , which defines how a client may access a database . It is a Java-based data access technology used for java database connectivity . It is part of the Java database connectivity . It is part of the Java standard Edition platform , from Oracle corporation .

**JDBC Goals**

Few software packages are designed without goals in mind.

JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java. The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

* SQL Level API
* SQL Conformance
* JDBC must be implemental on top of common database interfaces
* Provide a Java interface that is consistent with the rest of the Java system
* Keep it simple
* Use strong, static typing wherever possible
* Keep the common cases simple

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this work.

**Java Program**

**Compilers**

**Interpreter**

**My Program**

**FIGURE 7 :**  **INTERPRETER OF THE PROGRAM**

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

**NETWORKING**

**TCP/IP stack**

The TCP/IP stack is shorter than the OSI one:



**FIGURE 8 : TCP/IP STACK**

TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

**IP datagram’s**

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The header includes the source and destination addresses.

The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end .

**UDP**

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later

**TCP**

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

**Internet addresses**

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

**Networkaddress**

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

**Subnet address**

Internally , the UNIX network is divided into sub networks . Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

**Host address**

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

**Total address**



The 32 bit address is usually written as 4 integers separated by dots.

**Port addresses**

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

**Sockets**

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

#include <sys/types.h>

#include <sys/socket.h>

int socket(int family, int type, int protocol);

Here "family" will be AF\_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

**JFree Chart**

JFree Chart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFree Chart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

1 . Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XY Plot class in JFree Chart;

Testing, documenting, testing some more, documenting some more.

2 . Time Series Chart Interactivity

Implement a new (to JFree Chart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3 . Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFree Chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4 . Property Editors

The property editor mechanism in JFree Chart only handles a small subset of the properties that can be set for charts. Extend (or re-implement) this mechanism to provide greater end-user control over the appearance of the charts.

**J2ME (Java 2 Micro edition):-**

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the Java One Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

**1. General J2ME architecture**

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.



**FIGURE 9 : GENERAL J2ME ARCHITECTURE**

**2. Developing J2ME applications**

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role pre-verification plays in this process.

**3. Design considerations for small devices**

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

\* **Keep it simple.** Remove unnecessary features, possibly making those features a separate, secondary application.

\* **Smaller is better.** This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

\* **Minimize run-time memory use.** To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them.

Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

**4. Configurations overview**

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

\* **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

\* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

**5. J2ME profiles**

**What is a J2ME profile?**

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J02ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

**CHAPTER 6**

**SYSTEM DESIGN**

**6.1 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**TYPES OF UML DIAGRAM**

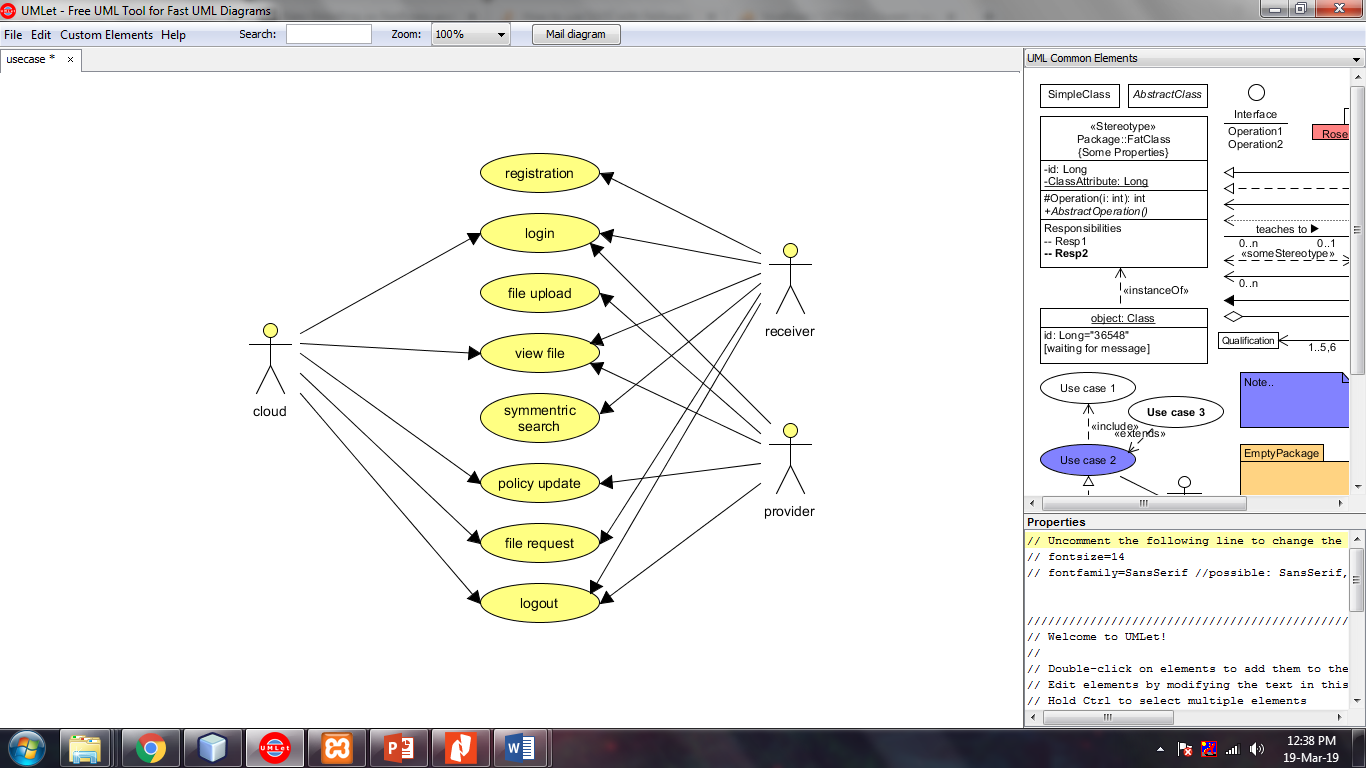
* Use case diagram
* Class diagram
* Sequence diagram
* Deployment diagram
* Data flow diagram

**USES OF UML DIAGRAM**

UML has been used as a general-purpose modeling language in the field of software engineering . however , it has now found its way into the documentation of several business processes or workflows. For example ,activity diagram , can be used as a replacement for flow charts . they provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficency .

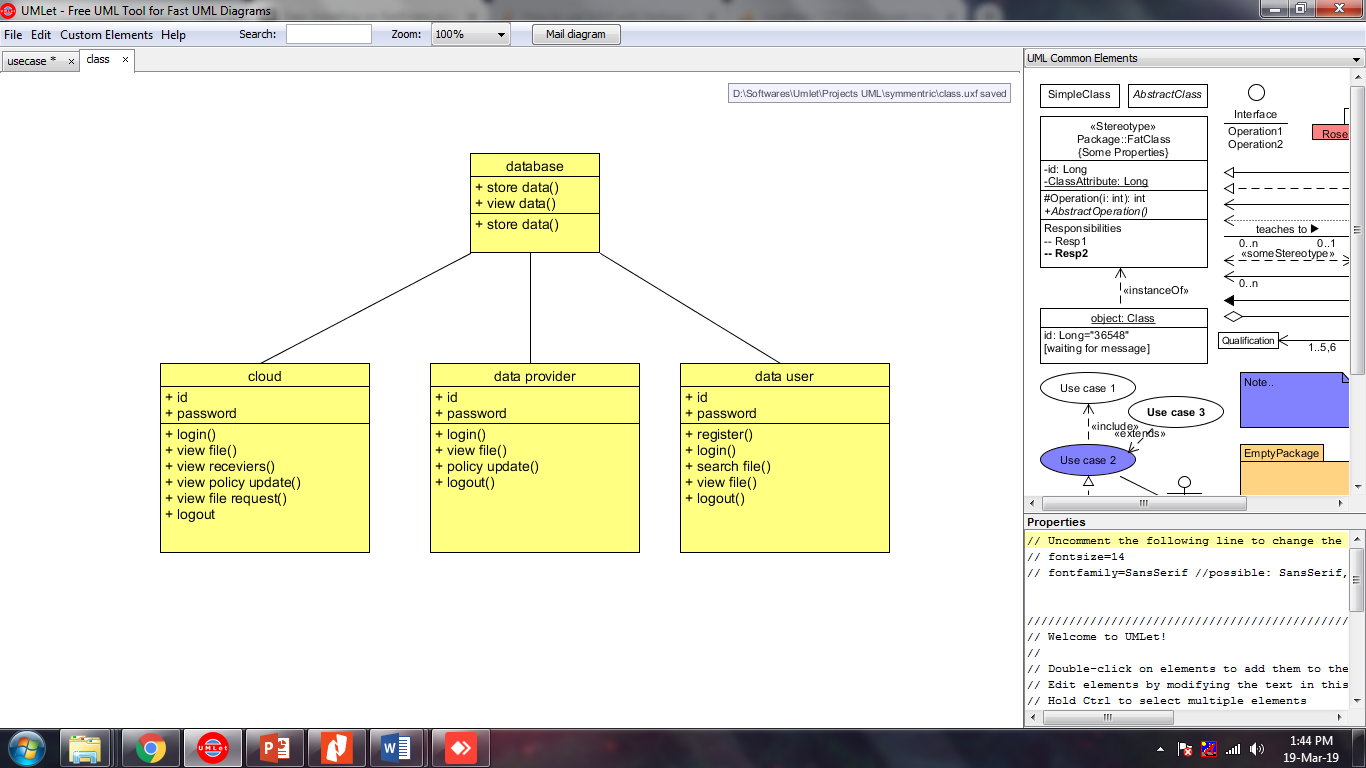
**6.2 USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in termof actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



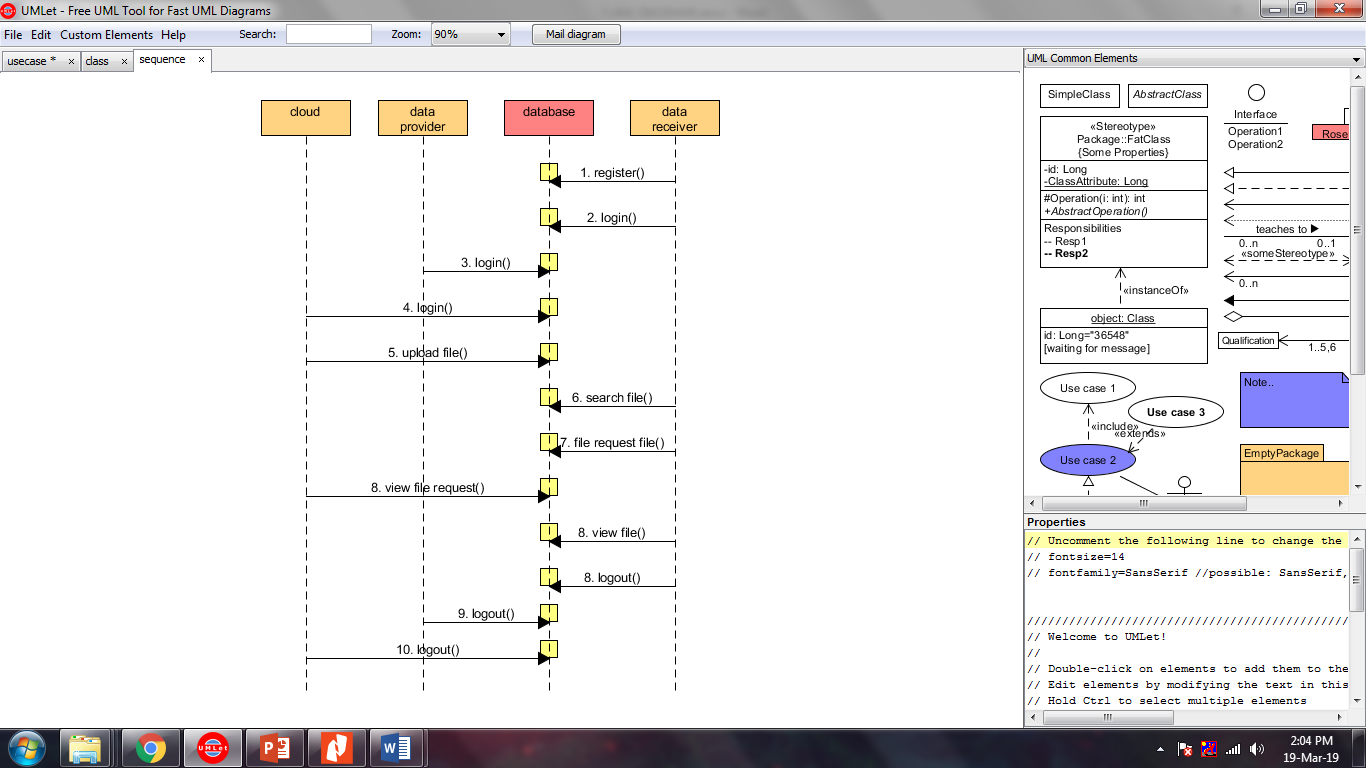
**6.3 CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**6.4 SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) 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. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**6.5 DEPLOYMENT:**

Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware. UML is mainly designed to focus on the software artifacts of a system. However, these two diagrams are special diagrams used to focus on software and hardware components.

**6.6 DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

1. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

**DFD DIAGRAM:**

**LEVEL 0**

**LEVEL 1**

**LEVEL 2**

**CHAPTER 7**

### SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

* Unit testing
* Integration testing
* Functional testing
* System testing
* White box testing
* Black box testing

**Unit testing**

In computer programming , unit testing is a software testing method by which individual units of source code , set of one or more computer program modules together with associated control data , usage procedures , and operating procedures , are tested to determine whether they are fit for use .

**Integration testing**

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group . integration testing is conducted to evaluate the compliance of a system or component which specified functional requirements . it occur after unit testing and before validation testing .

**Functional testing**

Functional testing is a quality assurance process and a type of black box testing that bases its test cases on the specifications of the software component under test . functions are tested by feeding them input and examining the output , and internal program structure is rarely considered .

**System testing**

System testing is testing conducted on a complete integrated system to evaluate the system’s compliance with its specified requirements . system testing takes , as its input , all of the integrated components that have passed integration testing.

**White Box Testing**

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

**CHAPTER 8**

**CONCLUSION**

In this paper , cloud storage systems has multi - user veriﬁable searchable symmetric encryption (MVSSE) is more practical than single-user SSE . We presented the security deﬁnitions , i.e. privacy and reliability, for MVSSE and an efﬁcient construction. We also introduced a Universally Composable (UC-) security framework for proving the security of MVSSE, and proved that our proposed scheme is UC-secure. We also presented several extensions of the proposed scheme and illustrated that the scheme could achieve better performance on conjunctive and boolean queries for both server and client. Our scheme has a limitation that the maximum number of users is determined in the system setup and we leave the construction of a scheme without such a limitation as future work. Another interesting open problem is to explore other keyword index structures that can improve the searching efﬁciency for multi-keyword boolean queries.

**REFERENCES**

[1] J. Katz and Y. Lindell, Introduction to Modern Cryptography, 2nd ed. CRC Press, 2014.

[2] D. Song, D. Wagner, and A. Perrig, “Practical Techniques for Searches on Encrypted Data,” in Proc. of IEEE S&P’00, 2000.

[3] E.-J. Goh, “Secure Indexes,” <https://eprint.iacr.org/2003/216>.

[4] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, “Searchable Symmetric Encryption: Improved Deﬁnitions and Efﬁcient Constructions,” in Proc. of ACM CCS’06, 2006.

[5] S. Kamara, C. Papamanthou, and T. Roeder, “Dynamic Searchable Symmetric Encryption,” in Proc. of ACM CCS’12, 2012, pp. 965–976.

[6] S. Kamara and C. Papamanthou, “Parallel and Dynamic Searchable Symmetric Encryption,” in Proc. of FC, 2013.

[7] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, “Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries,” in Proc. of CRYPTO’13, 2013.

[8] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, “Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation,” in Proc. of NDSS’14, 2014.

[9] K. Kurosawa and Y. Othtaki, “UC-Secure Searchable Sym- metric Encryption,” in Proc. of FC’12, 2012.

[10] ——, “How to Update Documents Veriﬁably in Searchable Symmetric Encryption,” in Proc. of CANS’13, 2013.

[11] R. Cheng, J. Yan, C. Guan, F. Zhang, and K. Ren, “Veriﬁable Searchable Symmetric Encryption from Indistinguishability Obfuscation,” in Proc. of ACM AISACCS’15, 2015.

[12] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, “Catch You If You Lie to Me: Efﬁcient Veriﬁable Conjunctive Keyword Search over Large Dynamic Encrypted Cloud Data,” in Proc. of IEEE INFOCOM’15, 2015.

[13] J. Benaloh and M. de Mare, “One-way Accumulators: A Decentralized Alternative to Disgital Signatures,” in Proc. of EUROCRYP’93, 1993.

[14] L. Nguyen, “Accumulators from Bilinear Pairings and Applications,” in Proc. of CT-RSA’05, 2005.

[15] J. Katz and Y. Lindell, “Aggregate Message Authentication Codes,” in Proc. of CT-RSA’08, 2008.

**APPENDIX 1**

**SCREENSHOTS AND SOURCECODE**

**SOURCECODE**

import static DbConnect.DB.Con;

String uname,pass,type;

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

{

HttpSession session = request.getSession();

uname=request.getParameter("uname");

pass=request.getParameter("pass");

type=request.getParameter("type");

Connection con = Con();

if(type.equals("user"))

{

try

{

PreparedStatement query=Con().prepareStatement("select \* from ureg where uname='"+uname+"' and pass='"+pass+"' ");

ResultSet rs = query.executeQuery();

if(rs.next())

{

session.setAttribute("uname", uname);

out.println("<script type=\"text/javascript\">");

out.println("alert(\"Welcome "+uname+"\")");

out.println("</script>");

RequestDispatcher rd=request.getRequestDispatcher("user.jsp");

rd.include(request, response);

}

else

{

out.println("<script type=\"text/javascript\">");

out.println("alert(\"Please Check Your Username and Password.\")");

out.println("</script>");

RequestDispatcher rd=request.getRequestDispatcher("index.jsp");

rd.include(request, response);

}

} catch (SQLException ex)

{

System.out.println(ex);

}

finally

{

try {

con.close();

} catch (SQLException ex) {

Logger.getLogger(login.class.getName()).log(Level.SEVERE, null, ex);

}}}

else if(type.equals("owner")&&uname.equals("owner")&&pass.equals("owner"))

{

out.println("<script type=\"text/javascript\">");

out.println("alert(\"Welcome Owner\")");

out.println("</script>");

RequestDispatcher rd=request.getRequestDispatcher("owner.jsp");

rd.include(request, response);

}

else if(type.equals("server")&&uname.equals("server")&&pass.equals("server"))

{

out.println("<script type=\"text/javascript\">");

out.println("alert(\"Welcome Cloud Server\")");

out.println("</script>");

RequestDispatcher rd=request.getRequestDispatcher("server.jsp");

rd.include(request, response);

}

else{

out.println("<script type=\"text/javascript\">");

out.println("alert(\"Invalid Credentials\")");

out.println("</script>");

RequestDispatcher rd=request.getRequestDispatcher("index.jsp");

rd.include(request, response);

} }

}@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

} @Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

} @Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}