1. **INTRODUCTION**
   1. **Introduction about the project**

Cloud computing provides seemingly unlimited “virtualized” resources to users as services across the whole Internet, while hiding platform and implementation de-tails. Today’s cloud service providers offer both highly available storage and massively parallel computing re-sources at relatively low costs. As cloud computing becomes prevalent, an increasing amount of data is being stored in the cloud and shared by users with specified privileges, which define the access rights of the stored data. One critical challenge of cloud storage services is the management of the ever-increasing volume of data. To make data management scalable in cloud computing, deduplication has been a well-known technique and has attracted more and more attention recently.

Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data in storage. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. Instead of keeping multiple data copies with the same content, deduplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy. Deduplication can take place at either the file level or the block level. For file-level deduplication, it eliminates duplicate copies of the same file. Deduplication can also take place at the block level, which eliminates duplicate blocks of data that occur in non-identical files.

Although data deduplication brings a lot of benefits, security and privacy concerns arise as users’ sensitive data are susceptible to both insider and outsider attacks. Traditional encryption, while providing data confidentiality, is incompatible with data deduplication. Specifically, traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different cipher texts, making deduplication impossible. Convergent encryption has been proposed to enforce data confidentiality while making deduplication feasible. It en-crypts/decrypts a data copy with a convergent key, which is obtained by computing the cryptographic hash value of the content of the data copy. After key generation and data encryption, users retain the keys and send the cipher text to the cloud. Since the encryption operation is deterministic and is derived from the data content, identical data copies will generate the same convergent key and hence the same cipher text. To prevent unauthorized access, a secure proof of ownership protocol is also needed to provide the proof that the user indeed owns the same file when a duplicate is found.

After the proof, subsequent users with the same file will be provided a pointer from the server without needing to upload the same file. A user can download the encrypted file with the pointer from the server, which can only be decrypted by the corresponding data owners with their convergent keys. Thus, convergent encryption allows the cloud to perform deduplication on the cipher texts and the proof of ownership prevents the unauthorized user to access the file. However, previous deduplication systems cannot sup-port differential authorization duplicate check, which is important in many applications. In such an authorized deduplication system.

* 1. **Preliminaries**

In this section, we define the notations used in this project, review some secure primitives used in our secure deduplication. Symmetric encryption. Symmetric encryption uses a common secret keyκto encrypt and decrypt information. A symmetric encryption scheme consists of three primitive functions:

KeyGenSE(1λ) *→κ* is the key generation algorithm that generates *κ* using security parameter 1λ;

EncSE(*κ,M*) *→C* is the symmetric encryption algorithm that takes the secret *κ* and message *M* and then outputs the ciphertext*C*; and

DecSE(*κ,C*) *→M* is the symmetric decryption algorithm that takes the secret *κ* and ciphertext*C* and then outputs the original message *M*.

**Convergent encryption.** Convergent encryption [3], [7] provides data confidentiality in deduplication. A user (or data owner) derives a convergent key from each original data copy and encrypts the data copy with the convergent key. In addition, the user also derives a *tag* for the data copy, such that the tag will be used to detect duplicates.

If two data copies are the same, then their tags are the same. To detect duplicates, the user first sends the tag to the server side to check if the identical copy has been already stored. Note that both the convergent key and the tag are independently to

derived, and the tag cannot be used to deduce the convergent key and compromise data confidentiality. Both the encrypted data copy and its corresponding tag will be stored on the server side. Formally, a convergent encryption scheme can be defined with four primitive functions:

KeyGenCE(*M*) *→ K* is the key generation algorithm that maps a data copy *M* to a convergent key *K*;

EncCE(*K,M*) *→C* is the symmetric encryption algorithm that takes both the convergent key *K* and the data copy *M* as inputs and then outputs a ciphertext *C*;

DecCE(*K,C*) *→M* is the decryption algorithm that takes both the ciphertext *C* and the convergent key *K* as inputs and then outputs the original data copy*M*; and

TagGen(*M*) *→T*(*M*) is the tag generation algorithm that maps the original data copy *M* and outputs a tag *T*(*M*).

* 1. **Scope of the project**

In order to implement provide highly secure for confidential data’s And web-based applications with several authentication process. Like text password, image selection, pixel determination, question based on images and mathematical calculation. Some API only require the use of a **client-id**. A client-id simply associates your server, script, or program with a specific application. However, other requests require authentication - specifically requests made on behalf of a user. Authenticated requests require an **access-token**. These tokens are unique to a user and should be stored securely. Access tokens may expire at any time in the future.

### 1.4 Proposed system

In this project, we address the problem of privacy preserving deduplication in cloud computing and propose a new deduplication system supporting for Differential Authorization. Each authorized user is able to get his/her individual token of his file to perform duplicate check based on his privileges. Under this assumption, any user cannot generate a token for duplicate check out of his privileges or without the aid from the private cloud server. Authorized Duplicate Check. Authorized user is able to use his/her individual private keys to generate query for certain file and the privileges he/she owned with the help of private cloud, while the public cloud performs duplicate check directly and tells the user if there is any duplicate. In this project, we are using MD5 algorithm. MD5 algorithm can used as a digital signature mechanism. This algorithm is used to check bit by bit deduplication of uploading data file. The MD5 algorithm is simple to implement and provides a ‘fingerprint’ or message digest of a message of arbitrary length. The difficulty of coming up with two messages with the message digest is on the order of 2∧64 operations.

**1.5 Feasibility study**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are ,

**1.5.1 Economical feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased. The purchase of tools will be like creating a cloud by running servers in online or by renting a cloud vender this is the most expenditure and other tools used to interact with the cloud are open source.

**1.5.2 Technical feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system. Our project is technically feasible for operation system such as windows7 and above with the better web browser such as internet explorer 8.2.5, Mozilla Firefox 23.0 or Google chrome. The internet bandwidth allocated should be 4mbps for easy file sharing and transferring. The cloud service provider should to provide hybrid (public and private cloud).

**1.5.3 Social feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system. This technology used is Socially feasible because it is easy for every user to interact with the cloud because interact with the cloud uses only simple browser it doesn’t need any special training to use a web browser . Bit training id needed for user to transfer a file into the cloud and to handle the error message such as file already exist and to work with it.

**Advantages of proposed system**

Convergent encryption ensures data privacy in deduplication “proofs of ownership” (PoW) for deduplication systems, such that a client can efficiently prove to the cloud storage server that he/she owns a file without uploading the file itself.

**1.6 Organization of chapters**

The chapters are organized as follows: chapter 2 documents Literature survey. In chapter 3, the design of system analysis is covered. The implementation details are described in chapter 4. Chapter 5 describes about Software testing. In chapter 6 conclusions and feature enhancements are discussed.

**2. LITERATURE SURVEY**

**INTRODUCTION**

Literature survey is the documentation of a comprehensive review of the published and unpublished work from secondary sources data in the areas of specific interest to the researcher. The researcher could start the literature survey even as the information from the unstructured and structured interviews is being gathered.

**Related work**

We address the problem of privacy preserving deduplication in cloud computing and propose a new deduplication system supporting for Differential Authorization and Authorized Duplicate Check.

Many enterprises and other organizations need to store and compute on a large amount of data. Cloud computing aims at renting such resources on demand.

Today’s cloud providers offer both, highly available storage and massively parallel computing resources with High Performance Computing (HPC) Clusters at low costs, as they can share resources among multiple clients. Cloud computing promises a cost effective enabling technology to outsource storage and massively parallel computations. However, existing approaches for provably secure outsourcing of data and arbitrary computations are either based on tamper-proof hardware or fully homomorphic encryption. The former approaches are not scaleable, while the latter ones are currently not efficient enough to be used in practice. Sven Bugiel, Stefan Nurnberger [6] propose an architecture and protocols that accumulate slow secure computations over time and provide the possibility to query them in parallel on demand by leveraging the benefits of cloud computing.

In our approach, the user communicates with a resource-constrained Trusted Cloud (either a private cloud or built from multiple secure hardware modules) which encrypts algorithms and data to be stored and later on queried in the powerful but untrusted Commodity Cloud. The drawback of this paper, risk of data leakage and user cannot able to share their data.

The late eighties and early nineties saw the proposal of many identity-based identification (IBI) and identity-based signature (IBS) schemes. The main task of analyzing practical IBI and IBS schemes, we pause to consider the following natural design of an IBI scheme, based on any given SI scheme, via the certification paradigm. The verifier needs to know only I and the public key of the authority in order to authenticate the prover secure IBI scheme. An analogous result holds in the IBS case. We believe that this is worth noting because it highlights the fact that, unlike IBE, IBI and IBS are trivial to achieve and enables us to better understand what the practical schemes are trying to do, namely to beat the trivial certification-based schemes in performance. Mihir Bellare, Chanathip Namprempre [4] propose an either security proofs or attacks for a large number of identity-based identification and signature schemes defined either explicitly or implicitly in existing literature. Underlying these are a framework that on the one hand helps explain how these schemes are derived, and on the other hand enables modular security analyses, thereby helping to understand, simplify and unify previous work.

The drawback of this paper, Thought it provides security by identity it don’t provide way to share his own data. But customers may want their data encrypted, for reasons ranging from personal privacy to corporate policy to legal regulations. A client could encrypt its file, under a user’s key, before storing it. But common encryption modes are randomized, making deduplication impossible since the SS (Storage Service) effectively always sees different ciphertexts regardless of the data. If a client’s Appeared at the Security Symposium. encryption is deterministic deduplication is possible, but only for that user. Cross-user deduplication, which allows more storage savings, is not possible because encryptions of different clients, being under different keys, are usually different. Sharing a single key across a group of users makes the system brittle in the face of client compromise. Mihir Bellare, Sriram Keelveedhi [2] propose an architecture that provides secure deduplicated storage resisting brute-force attacks, and realize it in a system called DupLESS. In DupLESS, clients encrypt under message-based keys obtained from a key-server via an oblivious PRF protocol. It enables clients to store encrypted data with an existing service, have the service perform deduplication on their behalf, and yet achieves strong confidentiality guarantees. We show that encryption for deduplicated storage can achieve performance and space savings close to that of using the storage service with plaintext data. The drawback of this paper, It makes encrypted data and key for encryption very vulnerable.

Deduplication improves storage efficiency by eliminating redundant data. Instead of storing multiple copies of data blocks that have identical content, a deduplication system stores only one copy of identical blocks, while other blocks refer to the copy via smaller-size references. Deduplication is mainly studied in content-addressable backup systems. It is also shown to provide space-efficient VM image storage given that VM images have significant content similarities.

Most existing deduplication studies focus on optimizing storage efficiency and write (or backup) performance. However, one drawback of deduplication is that it introduces fragmentation, since some blocks of a file may now refer to other identical blocks of a different file. Hence, accessing a 1file is no longer sequential as in ordinary file systems without deduplication, but instead requires additional disk seeks to the identical blocks being referenced. This significantly degrades read performance. Chun-Ho Ng and Patrick P. C. Lee [9] propose RevDedup, a deduplication system that optimizes reads to latest VM image backups using an idea called reverse deduplication. In contrast with conventional deduplication that removes duplicates from new data, RevDedup removes duplicates from old data, thereby shifting fragmentation to old data while keeping the layout of new data as sequential as possible. We evaluate our RevDedup prototype using micro benchmark and real-world workloads. For a 12-week span of real-world VM images from 160 users, RevDedup achieves high deduplication efficiency with around 97% of saving, and high backup and read throughput on the order of 1GB/s. RevDedup also incurs small metadata overhead in backup/read operations. The drawback of this paper, Cost increases in revdeduplication and it give only limited access foe rev deduplication.

Cloud computing is an emerging service model that provides computation and storage resources on the Internet. One attractive functionality that cloud computing can offer is *cloud storage*.

Individuals and enterprises are often required to remotely archive their data to avoid any information loss in case there are any hardware/software failures or unforeseen disasters. Instead of purchasing the needed storage media to keep data backups, individuals and enterprises can simply outsource their data backup services to the cloud service providers, which provide the necessary storage resources to host the data backups. While cloud storage is attractive, how to provide security guarantees for outsourced data becomes a rising concern. One major security challenge is to provide the property of *assured deletion*, i.e., data files are permanently inaccessible upon requests of deletion. Arthur Rahumed, Henry C. H. Chen [5] present *Fade Version*, a secure cloud backup system that serves as a security layer on top of today’s cloud storage services.

Fade Version follows the standard version-controlled backup design, which eliminates the storage of redundant data across different versions of backups. On top of this, Fade Version applies cryptographic protection to data backups. Specifically, it enables fine-grained assured deletion, that is, cloud clients can assuredly delete particular backup versions or files on the cloud and make them permanently inaccessible to anyone, while other versions that share the common data of the deleted versions or files will remain unaffected. We implement a proof-of-concept prototype of Fade Version and conduct empirical evaluation atop Amazon S3. We show that Fade Version only adds minimal performance overhead over a traditional cloud backup service that does not support assured deletion. The drawback of this paper, It increases storage memory cost and only adds minimal performance.

**3. SYSTEM ANALYSIS AND DESIGN**

At a high level, our setting of interest is an enterprise network, consisting of a group of affiliated clients (for example, employees of a company) who will use the S-CSP and store data with deduplication technique. In this setting, deduplication can be frequently used in these settings for data backup and disaster recovery applications while greatly reducing storage space. Such systems are widespread and are often more suitable to user file backup and synchronization applications than richer storage abstractions. There are three entities defined in our system, that is, *users*, *private cloud* and S-CSP in *public cloud* as shown in Fig. 3.1. The S-CSP performs deduplication by checking if the contents of two files are the same and stores only one of them. The access right to a file is defined based on a set of *privileges*. The exact definition of a privilege varies across applications.

**3.1 Architecture diagram**

***Fig 3.1 Architecture diagram of Project***

Each privilege is represented in the form of a short message called *token*. Each file is associated with some *file tokens*, which denote the tag with specified privileges. A user computes and sends *duplicate-check tokens* to the public cloud for authorized duplicate check. Users have access to the private cloud server, a semitrusted third party which will aid in performing deduplicable encryption by generating file tokens for the requesting users.

**3.2 Module description**

**3.2.1 Login**

This module authorize user in to the system. This adds security to the user data. The login credentials are secured by encryption and they are decrypted back by the server to avoid eavesdropping.

**3.2.2 Granting access**

The user who want the data to be shared need to be authorized by the data owner. This is done by requesting for access token and access token is automatically sent to the user. The authorization token is mandatory to access the file. A user with not access token cannot view the file too.

**3.2.3 Multiple access**

The above module is repeated for many times to grant access to many users. Here the data is not replicated but it is shared and also the data is read only for the users, so every user can read single data at a time.

**3.2.4 Access token generation**

In this module user must log in to the system as data owner and upload the data to the server. This module also have the secure upload facility and request of the user is not recorded to ensure the privacy of the user. The data is transferred from the data owner system to the cloud using http protocol. The access token is generated by the data owner.

**UML diagrams**

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

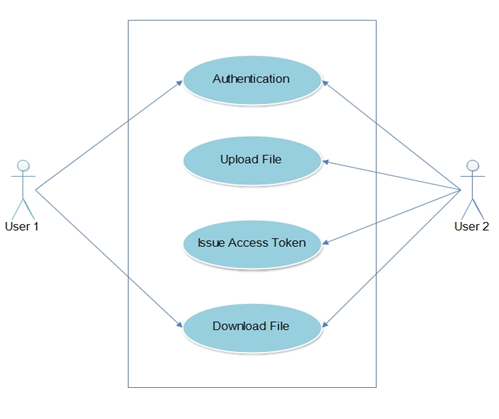
Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer’s requirements into finished software or a system.

Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data. There are various kinds of methods in software design:

**3.3 Use case diagram**

Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what’s called an actor. Use case diagram can be useful for getting an overall view of the system and clarifying that can do and more importantly what they can’t do. Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

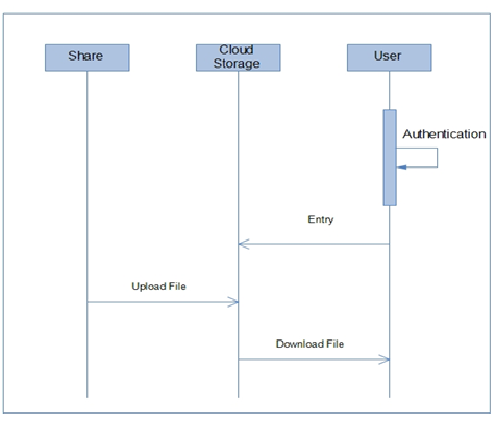


***Fig 3.3 Use case diagrams***

**3.4 Sequence diagram**

Sequence diagram and collaboration diagram are called INTERACTION DIAGRAMS. An interaction diagram shows an interaction, consisting of set of objects and their relationship including the messages that may be dispatched among them.

A sequence diagram is an introduction that empathizes the time ordering of messages. Graphically a sequence diagram is a table that shows objects arranged along the X-axis and messages ordered in increasing time along the Y-axis.

****

***Fig 3.4 Sequence diagram***

**3.5 Dataflow diagrams**

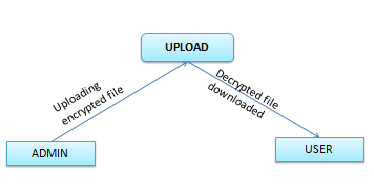
The DFD takes an input-process-output view of a system i.e. data objects flow into the software, are transformed by processing elements, and resultant data objects flow out of the software. Data objects represented by labeled arrows and transformation are represented by circles also called as bubbles. DFD is presented in a hierarchical fashion i.e. the first data flow model represents the system as a whole. Subsequent DFD refine the context diagram (level 0 DFD), providing increasing details with each subsequent level.

The DFD enables the software engineer to develop models of the information domain & functional domain at the same time. As the DFD is refined into greater levels of details, the analyst performs an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of the data as it moves through the process that embodies the applications. A context-level DFD for the system the primary external entities produce information for use by the system and consume information generated by the system. The labeled arrow represents data objects or object hierarchy.

**Rules for DFD**

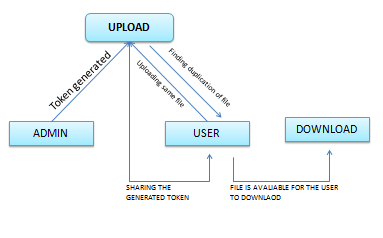
* Fix the scope of the system by means of context diagrams.
* Organize the DFD so that the main sequence of the actions
* Reads left to right and top to bottom.
* Identify all inputs and outputs.
* Identify and label each process internal to the system with Rounded circles.
* A process is required for all the data transformation and Transfers. Therefore, never connect a data store to a data Source or the destinations or another data store with just a Data flow arrow.
* Do not indicate hardware and ignore control information.
* Make sure the names of the processes accurately convey everything the process is done.
* There must not be unnamed process.
* Indicate external sources and destinations of the data, with Squares.
* Number each occurrence of repeated external entities.
* Identify all data flows for each process step, except simple Record retrievals.
* Label data flow on each arrow.
* Use details flow on each arrow.
* Use the details flow arrow to indicate data movements.

**3.5.1 Level 0**



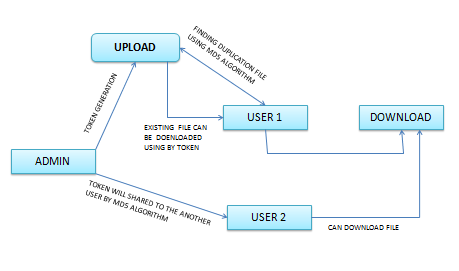
***Fig 3.4 Dataflow Diagram level-0***

**3.5.2 Level 1**



***Fig 3.5 Dataflow Diagram level-1***

**3.5.3 Level 2**



***Fig 3.6 Dataflow Diagram level-2***

**4. IMPLIMENTATION**

**4.1 Secure deduplication system**

To support authorized deduplication, the tag of a file *F* will be determined by the file *F* and the privilege. To show the difference with traditional notation of tag, we call it file token instead. To support authorized access, a secret key *kp* will be bounded with a privilege *p* to generate a file token. Let *ϕ′ F;p* = TagGen(*F, kp*) denote the token of *F* that is only allowed to access by user with privilege *p*. In another word, the token *ϕ′ F;p* could only be computed by the users with privilege *p*. As a result, if a file has been uploaded by a user with a duplicate token *ϕ′ F;p*, then a duplicate check sent from another user will be successful if and only if he also has the file *F* and privilege *p*. Such a token generation function could be easily implemented as *H*(*F, kp*), where *H*(⋅) denotes a cryptographic hash function.

**4.1.1 Token generation**

Before introducing our construction of differential deduplication, we present a straightforward attempt with the technique of token generation TagGen(*F,kp*) above to design such a deduplication system. The main idea of this basic construction is to issue corresponding privilege keys to each user, who will compute the file tokens and perform the duplicate check based on the privilege keys and files. In more details, suppose that there are *N* users in the system and the privileges in the universe is defined as *P* = *{p*1*,..., ps}*. For each privilege *p* in *P*, a private key *kp* will be selected. For a user *U* with a set of privileges *PU*, he will be assigned the set of keys {*kpi}pi∈PU* .

**4.1.2 File upload**

Suppose that a data owner *U* with privilege set *PU* wants to upload and share a file *F* with users who have the privilege set *PF* = *{pj}*. The user computes and sends S-CSP the file token *ϕ′ F;p* = TagGen(*F, kp*) for all *p∈ PF.* If a duplicate is found by the S-CSP, the user proceeds proof of ownership of this file with the S-CSP. If the proof is passed, the user will be assigned a pointer, which allows him to access the file.Otherwise, if no duplicate is found, the user computes the encrypted file *CF* = EncCE(*kF, F*) with the convergent key *kF* = KeyGenCE(*F*) and uploads (*CF*,*{ϕ′F;p}*) to the cloud server. The convergent key *kF* is stored by the user locally.

**4.1.3 File retrieving**

Suppose a user wants to download a file *F*. It first sends a request and the file name to the S-CSP. Upon receiving the request and file name, the S-CSP will check whether the user is eligible to download *F*. If failed, the S-CSP sends back an abort signal to the user to indicate the download failure. Otherwise, the S-CSP returns the corresponding ciphertext *CF*. Upon receiving the encrypted data from the S-CSP, the user uses the key *kF* stored locally to recover the original file *F*.

Problems. Such a construction of authorized deduplication has several serious security problems, which are listed below.

First, each user will be issued private keys {*kpi}pi∈PU* for their corresponding privileges, denoted by *PU* in our above construction. These private keys {*kpi}pi∈PU* can be applied by the user to generate file token for duplicate check. However, during file uploading, the user needs to compute file tokens for sharing with other users with privileges *PF* . To compute these file tokens, the user needs to know the private keys for *PF* , which means *PF* could only be chosen from *PU*. Such a restriction makes the authorized deduplication system unable to be widely used and limited.

Second, the above deduplication system cannot prevent the privilege private key sharing among users. The users will be issued the same private key for the same privilege in the construction. As a result, the users may collude and generate privilege private keys for a new privilege set *P\** that does not belong to any of the colluded user. For example, a user with privilege set *PU*1 may collude with another user with privilege set *PU*2 to get a privilege set *P\**=*PU*1 *∪ PU*2.

The construction is inherently subject to brute-force attacks that can recover files falling into a known set. That is, the deduplication system cannot protect the security of predictable files. One of critical reasons is that the traditional convergent encryption system can only protect the semantic security of unpredictable files.

**4.1.4 File sharing**

Suppose that a data owner wants to upload and share a file *F* with users whose privilege belongs to the set *PF* = *{pj}*. The data owner needs interact with the private cloud before performing duplicate check with the S-CSP. More precisely, the data owner performs an identification to prove its identity with private key

*skU*. If it is passed, the private cloud server will find the corresponding privileges *PU* of the user from its stored table list. The user computes and sends the file

tag *ϕF* = TagGen(*F*) to the private cloud server, who will return {*ϕ′* *F;p€*= TagGen(*ϕF , kp€* )*g* back to the user for all *pᵧ* satisfying R(*p, pᵧ* ) = 1 and *p∈ PU*. Then, the user will interact and send the file token *{ϕ′ F;p€ }* to the S-CSP.

If a file duplicate is found, the user needs to run the PoW protocol POW with the S-CSP to prove the file ownership. If the proof is passed, the user will be provided a pointer for the file. Furthermore, a proof from the S-CSP will be returned, which could be a signature on *{ϕ′ F;p€ }*, *pkU* and a time stamp. The user sends the privilege set *PF* = {*pj}* for the file *F* as well as the proof to the private cloud server. Upon receiving the request, the private cloud server first verifies the proof from the S-CSP. If it is passed, the private cloud server computes *{ϕ′ F;p€*=TagGen(*ϕF , kp€* )*}* for all *pᵧ* satisfying R(*p, pᵧ* ) = 1 for each *p∈ PF* -*PU*, which will be returned to the user. The user also uploads these tokens of the file *F* to the private cloud server. Then, the privilege set of the file is set to be the union of *PF* and the privilege sets defined by the other data owners.

Otherwise, if no duplicate is found, a proof from the S-CSP will be returned, which is also a signature on *{ϕ′ F;p€}*, *pkU* and a time stamp. The user sends the privilege set *PF* = *{pj}* for the file *F* as well as the proof to the private cloud server. Upon receiving the request, the private cloud server first verifies the proof from the S-CSP. If it is passed, the private cloud server computes *{ϕ′ F;p€* = TagGen(*ϕF,kp€*)}for all *pᵧ* satisfying R(*p, pᵧ* ) = 1 and *p∈ PF* . Finally, the user computes the encrypted file *CF* = EncCE(*kF,F*) with the convergent key *kF* = KeyGenCE(*F*) and uploads {*CF , {ϕ′ F;p€ }}* with privilege *PF*.

File Retrieving**.** The user downloads his files in the same way as the deduplication system in Section 4.1.3. That is, the user can recover the original file with the convergent key *kF* after receiving the encrypted data from the S-CSP.

**4.2 Sample coding**

**Login**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"* pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html> <head>

<title>Welcome</title>

<link rel=*"stylesheet"* href=*"style.css"* type=*"text/css"*>

<style type=*"text/css"*>

**fieldset** {

width:*400px*;

border:*1px solid green* }

**legend** {

padding: *0.2em 0.5em*;

border:*1px solid green*;

color:*green*;

font-size:*90%*;

text-align:*left*; }

**li** {

list-style:*none*; }

</style> </head>

<body>

<div class=*"header"*>

<!-- <div class="cname">

<b style="font-family:arial; font-size:23px; color:rgb(250, 96, 16);text-transform:uppercase;position:relative; left:160px;top:15px;">A Hybrid Cloud Approach for Secure Authorized Deduplication</b>

</div> -->

</div>

<div class=*"menu"*>

<li><a href=*"Loginpage.jsp"* style="color:*#fff*;"><b>Login</b></a></li>

</div>

<div class=*"content"*>

<div class=*"left-con"*><br><br><br><br>

<li><a href=*"Loginpage.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">User</a></li>

<li><a href=*"aboutus.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">About Us</a></li>

</div>

<div class=*"right-con"*><br><br>

<form action=*"s1"* method=*"post"*>

<center>

<fieldset><legend><b style="color:*#000*; font-size:*20px*; ">Login</b></legend>

<table> <tr> <td>

<b style="text-transform:*uppercase*;">Username</b>

</td>

<td>

<input type=*"text"* placeholder=*"Enter your username"* name=*"t1"*/ class=*"textbox"*>

</td> </tr>

<tr> <td>

<b style="text-transform:*uppercase*;">Password</b> &nbsp;&nbsp;&nbsp;

</td>

<td>

<input type=*"password"* placeholder=*"Enter your password"* name=*"t2"* /class=*"textbox"*>

</td> </tr> <tr>

<td></td>

<td >

<input class=*"myButton"* type=*"submit"* value=*"Login"* style="position:*relative*; left:*60px*;">

<a href=*'registration.jsp'*><b style="position:*relative*; left:*60px*;">NewUser</b></a>

</td> </tr> </table> </fieldset> </center>

</form><!-- <br> <br> <br><br>

<br> --> </div> </div>

<div class=*"footer"*></div>

</body> </body> </html>

**Granting access**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html> <head>

<title>UploadFile</title>

<style type=*"text/css"*>

**fieldset** {

width:*400px*;

border:*1px solid green* }

**legend** {

padding: *0.2em 0.5em*;

border:*1px solid green*;

color:*green*;

font-size:*90%*;

text-align:*left*;

} **li**  {

list-style:*none*; } **th** {

text-align:*left*; } </style>

<link rel=*"stylesheet"* href=*"style.css"* type=*"text/css"*/>

</head> <body>

<div class=*"header"*>

</div>

<div class=*"menu"*>

<li><a href=*"logout.jsp"*><b>&nbsp;&nbsp;&nbsp;&nbsp;Logout</b></a></li>

<li><a href=*"sharefile.jsp"*><b>&nbsp;&nbsp;&nbsp;&nbsp;ShareFile</b></a></li>

<li><a href=*"downloadfile.jsp"*><b>DownloadFile</b></a></li>

</div> <div class=*"content"*>

<div class=*"left-con"*><br><br><br><br>

<li><a href=*"Loginpage.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">Home</a></li>

<li><a href=*"aboutus.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">About Us</a></li>

<li><a href=*"sharefile.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">ShareFile</a></li>

<li><a href=*"downloadfile.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">DownloadFile</a></li>

<li><a href=*"logout.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">Logout</a></li>

</div>

<div class=*"right-con"*><br><br>

<p style="color:*Green*;position:*relative*; left:*150px*; text-transform:*uppercase*;"><%

HttpSession ses=request.getSession(**false**);

String u1 = (String)ses.getAttribute("uid");

**int** u = Integer.parseInt(u1);

out.println("Your Current UserID ==> "+u);

%> </p> <center><fieldset><legend>

<b>UploadFile</b> </legend>

<form action=*"FileUploadServlet"* method=*"post"* enctype=*"multipart/form-data"*>

<table> <tr> <th>

UploadFile &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

</th> <td>

<input type=*"file"* name=*"t2"* class=*"textbox"*>

<br> <input type=*"submit"* value=*"Submit"* class=*"myButton"* style="position:*relative*; left:*60px*;">

</td> </tr>

</table></form> </fieldset> </center>

<br> <br> <br><br> </div> </div>

<div class=*"footer"*></div>

<script src=*"assets-new/js/jquery-1.8.2.min.js"*></script>

<script src=*"assets-new/bootstrap/js/bootstrap.min.js"*></script>

<script src=*"assets-new/js/jquery.backstretch.min.js"*></script>

<script src=*"assets-new/js/scripts.js"*></script>

</body> </html>

**Multiple access**

<%@page import=*"java.sql.ResultSet"*%>

<%@page import=*"java.sql.PreparedStatement"*%>

<%@page import=*"java.sql.Connection"*%>

<%@page import=*"home.Dbconnection"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html> <head> <title>Home</title>

<link rel=*"stylesheet"* href=*"style.css"* type=*"text/css"*>

<style type=*"text/css"*> **fieldset** {

width:*400px*;

border:*1px solid green* }

**legend** {

padding: *0.2em 0.5em*;

border:*1px solid green*;

color:*green*;

font-size:*90%*;

text-align:*left*;

} **li** {

list-style:*none*; th {

text-align:*left*; } </style> </head>

<body> <div class=*"header"*> </div>

<div class=*"menu"*>

<li><a href=*"logout.jsp"*><b>Log out</b></a></li>

<li><a href=*"uploadFile.jsp"*><b>Home</b></a></li>

</div> <div class=*"content"*>

<div class=*"left-con"*><br><br><br><br>

<li><a href=*"Loginpage.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">Home</a></li>

<li><a href=*"aboutus.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">About Us</a></li>

<li><a href=*"sharefile.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">ShareFile</a></li>

<li><a href=*"downloadfile.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">DownloadFile</a></li>

<li><a href=*"logout.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">Logout</a></li>

</div> <div class=*"right-con"*><br><br> <%

String u1 = (String)session.getAttribute("uid");

**int** fromu = Integer.parseInt(u1);

%> <%

String u2 = request.getParameter("t1");

String p1 = request.getParameter("t2");

System.out.println(u1+" "+p1);

**int** tou=Integer.parseInt(u2);

**int** fid=Integer.parseInt(p1);

Connection c = Dbconnection.conn();

PreparedStatement st;

st=c.prepareStatement("insert into share1 (sfrom\_uid,sto\_uid,fid)values(?,?,?)");

st.setInt(1, fromu);

st.setInt(2, tou);

st.setInt(3,fid);

**int** i =st.executeUpdate();

**if**(i>0) { %>

<h2>File Shared successfully!!!!</h2>

<%} %> </center> <br><br>

</div> </div> </div>

<div class=*"footer"*></div>

<script src=*"assets-new/js/jquery-1.8.2.min.js"*></script>

<script src=*"assets-new/bootstrap/js/bootstrap.min.js"*></script>

<script src=*"assets-new/js/jquery.backstretch.min.js"*></script>

<script src=*"assets-new/js/scripts.js"*></script>

</body> </html>

**Access token generation**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"* import=*"java.sql.\*,home.\*"*%>

<%@page import = *"javax.servlet.http.\*"* %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<head> <script type = *"text/javascript"*>

**function** download(str){

document.getElementById("fd").value = str;

document.getElementById("myForm").submit(); }

</script> <title>Welcome</title>

<link rel=*"stylesheet"* href=*"style.css"* type=*"text/css"*>

<style type=*"text/css"*> **fieldset** {

width:*400px*;

border:*1px solid green* } **legend** {

padding: *0.2em 0.5em*;

border:*1px solid green*;

color:*green*;

font-size:*90%*;

text-align:*left*;

} **li**  {

list-style:*none*;

} **th** {

text-align:*left*; } *.CSSTableGenerator* {

margin:*0px*;padding:*0px*; width:*100%*;

box-shadow: *10px 10px 5px #888888*; border:*1px solid #000000*;

-moz-border-radius-bottomleft:*0px*;

-webkit-border-bottom-left-radius:*0px*; border-bottom-left-radius:*0px*;

-moz-border-radius-bottomright:*0px*;

-webkit-border-bottom-right-radius:*0px*;

border-bottom-right-radius:*0px*;

-moz-border-radius-topright:*0px*;

-webkit-border-top-right-radius:*0px*; border-top-right-radius:*0px*;

-moz-border-radius-topleft:*0px*;

-webkit-border-top-left-radius:*0px*; border-top-left-radius:*0px*;

}*.CSSTableGenerator* **table** { border-collapse: *collapse*;

border-spacing: *0*; width:*100%*; height:*100%*;

margin:*0px*;padding:*0px*;

}*.CSSTableGenerator* **tr***:last-child* **td***:last-child* {

-moz-border-radius-bottomright:*0px*;

-webkit-border-bottom-right-radius:*0px*;

border-bottom-right-radius:*0px*; }

*.CSSTableGenerator* **table** **tr***:first-child* **td***:first-child* {

-moz-border-radius-topleft:*0px*;

-webkit-border-top-left-radius:*0px*;

border-top-left-radius:*0px*; }

*.CSSTableGenerator* **table** **tr***:first-child* **td***:last-child* {

-moz-border-radius-topright:*0px*;

-webkit-border-top-right-radius:*0px*;

border-top-right-radius:*0px*;

}*.CSSTableGenerator* **tr***:last-child* **td***:first-child*{

-moz-border-radius-bottomleft:*0px*;

-webkit-border-bottom-left-radius:*0px*;

border-bottom-left-radius:*0px*;

}*.CSSTableGenerator* **tr***:hover* **td**{

} *.CSSTableGenerator* **tr***:nth-child(odd)*{ background-color:*#ffaa56*; }

*.CSSTableGenerator* **tr***:nth-child(even)* { background-color:*#ffffff*; }*.CSSTableGenerator* **td**{

vertical-align:*middle*; border:*1px solid #000000*;

border-width:*0px 1px 1px 0px*; text-align:*left*;

padding:*7px*; font-size:*10px*;

font-family:*Arial*; font-weight:*bold*;

color:*#000000*; }*.CSSTableGenerator* **tr***:last-child* **td**{

border-width:*0px 1px 0px 0px*;

}*.CSSTableGenerator* **tr** **td***:last-child*{

border-width:*0px 0px 1px 0px*;

}*.CSSTableGenerator* **tr***:last-child* **td***:last-child*{

border-width:*0px 0px 0px 0px*; }

*.CSSTableGenerator* **tr***:first-child* **td**{

background:*-o-linear-gradient(bottom, #ff7f00 5%, #bf5f00 100%)*; background:*-webkit-gradient( linear, left top, left bottom, color-stop(0.05, #ff7f00), color-stop(1, #bf5f00) )*;

background:*-moz-linear-gradient( center top, #ff7f00 5%, #bf5f00 100% )*;

filter:*progid*:*DXImageTransform.Microsoft.gradient*(*startColorstr*=*"#ff7f00", endColorstr*=*"#bf5f00")*; background: *-o-linear-gradient(top,#ff7f00,bf5f00)*;

background-color:*#ff7f00*; border:*0px solid #000000*;

text-align:*center*; border-width:*0px 0px 1px 1px*;

font-size:*14px*; font-family:*Arial*;

font-weight:*bold*; color:*#ffffff*; }

*.CSSTableGenerator* **tr***:first-child:hover* **td**{

background:*-o-linear-gradient(bottom, #ff7f00 5%, #bf5f00 100%)*; background:*-webkit-gradient( linear, left top, left bottom, color-stop(0.05, #ff7f00), color-stop(1, #bf5f00) )*;

background:*-moz-linear-gradient( center top, #ff7f00 5%, #bf5f00 100% )*;

filter:*progid*:*DXImageTransform.Microsoft.gradient*(*startColorstr*=*"#ff7f00", endColorstr*=*"#bf5f00")*; background: *-o-linear-gradient(top,#ff7f00,bf5f00)*; background-color:*#ff7f00*;

}

*.CSSTableGenerator* **tr***:first-child* **td***:first-child*{

border-width:*0px 0px 1px 0px*; }

*.CSSTableGenerator* **tr***:first-child* **td***:last-child*{

border-width:*0px 0px 1px 1px*; }

</style> </head> <body>

<form action = *"DownloadOption.jsp"* id = *"myForm"* method = *"post"*>

<input type = *"hidden"* name= *"fn"* id = *"fd"*/>

</form> <div class=*"header"*> </div> <div class=*"menu"*>

<li><a href=*"logout.jsp"*><b>Log out</b></a></li>

<!-- <li><a href="uploadFile.jsp"><b>Home</b></a></li> -->

</div> <div class=*"content1"*>

<div class=*"left-con1"*><br><br><br><br>

<li><a href=*"Loginpage.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">Home</a></li>

<li><a href=*"aboutus.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">About Us</a></li>

<li><a href=*"sharefile.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">ShareFile</a></li>

<li><a href=*"downloadfile.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">DownloadFile</a></li>

<li><a href=*"logout.jsp"* style="text-decoration:*none*; font-family:*arial*; font-size:*20px*; color:*#000*;margin:*40px*; line-height:*50px*;">Logout</a></li>

</div> <div class=*"right-con1"*><br><br> <center>

<b style="font-family:*arial*; font-size:*23px*; color:*rgb(250, 96, 16)*;text-transform:*uppercase*;position:*relative*;top:*15px*;">Shared File List</b></center>

<br>

<p style="color:*Green*;position:*relative*; left:*150px*; text-transform:*uppercase*;"><%

HttpSession ses=request.getSession(**false**);

String u1 = (String)ses.getAttribute("uid");

**int** u = Integer.parseInt(u1);

//int u=2; out.println("Your Current UserID==> "+u);

%> </p> <% **int** uid,fid;

Connection c= Dbconnection.conn();

PreparedStatement st,st1; ResultSet rs,rs1; %>

<form action=*"DownloadOption.jsp"* method=*"post"*><center>

<table border=*"1"* style="width:*80%*; background-color:*#D7D8BD*;color:*#000*;" class=*"CSSTableGenerator"* >

<tr> <th>Shared From UserID</th>

<th>Access Token Number</th> <th>Download File</th>

</tr> <%

st= c.prepareStatement("select \* from share1 where sto\_uid=?");

st.setInt(1,u); rs =st.executeQuery();

**int** i=1; **while**(rs.next()) {

st1= c.prepareStatement("select \* from owner where fid=?");

st1.setInt(1, rs.getInt(3)); rs1 = st1.executeQuery();

String FileName = ""; **if**(rs1.next()){

FileName = rs1.getString("fname");

} %> <tr> <td> <%=rs.getInt(1)%> <%

HttpSession ss = request.getSession();

ss.setAttribute("fid", Integer.toString(rs.getInt(3)));

%> <td><%=rs.getInt(4)%><td>

<input type = *"button"* value = *'*<%=FileName %>*'* onclick = "download(this.value)"/></td> </tr> <% i++; } %>

</table> </center> </form> <br> <br> <br>

</div> </div> <div class=*"footer"*></div>

<script src=*"assets-new/js/jquery-1.8.2.min.js"*></script>

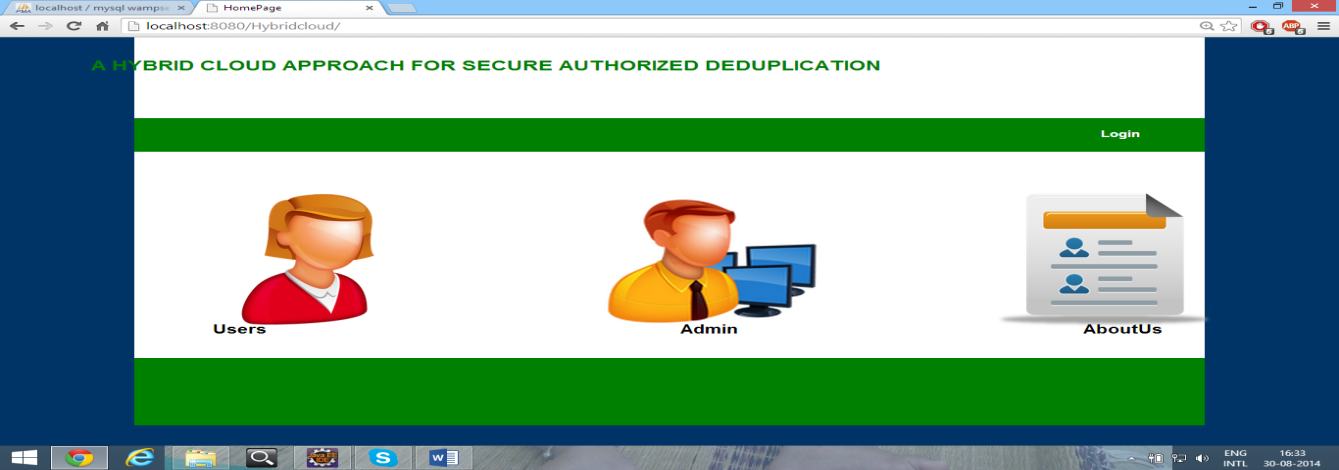
<script src=*"assets-new/bootstrap/js/bootstrap.min.js"*></script>

<script src=*"assets-new/js/jquery.backstretch.min.js"*></script>

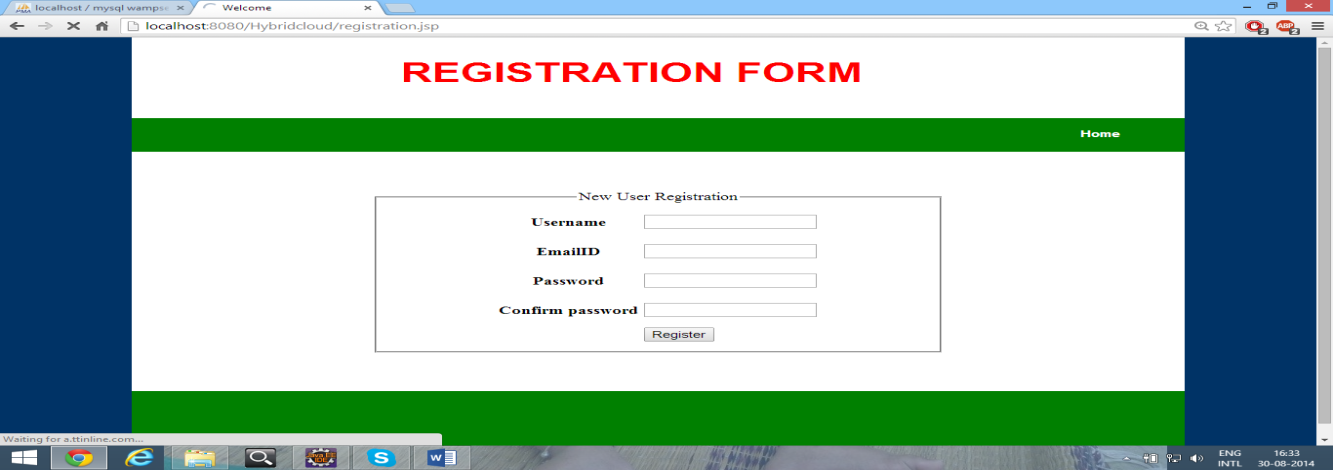
<script src=*"assets-new/js/scripts.js"*></script>

</body> </html>

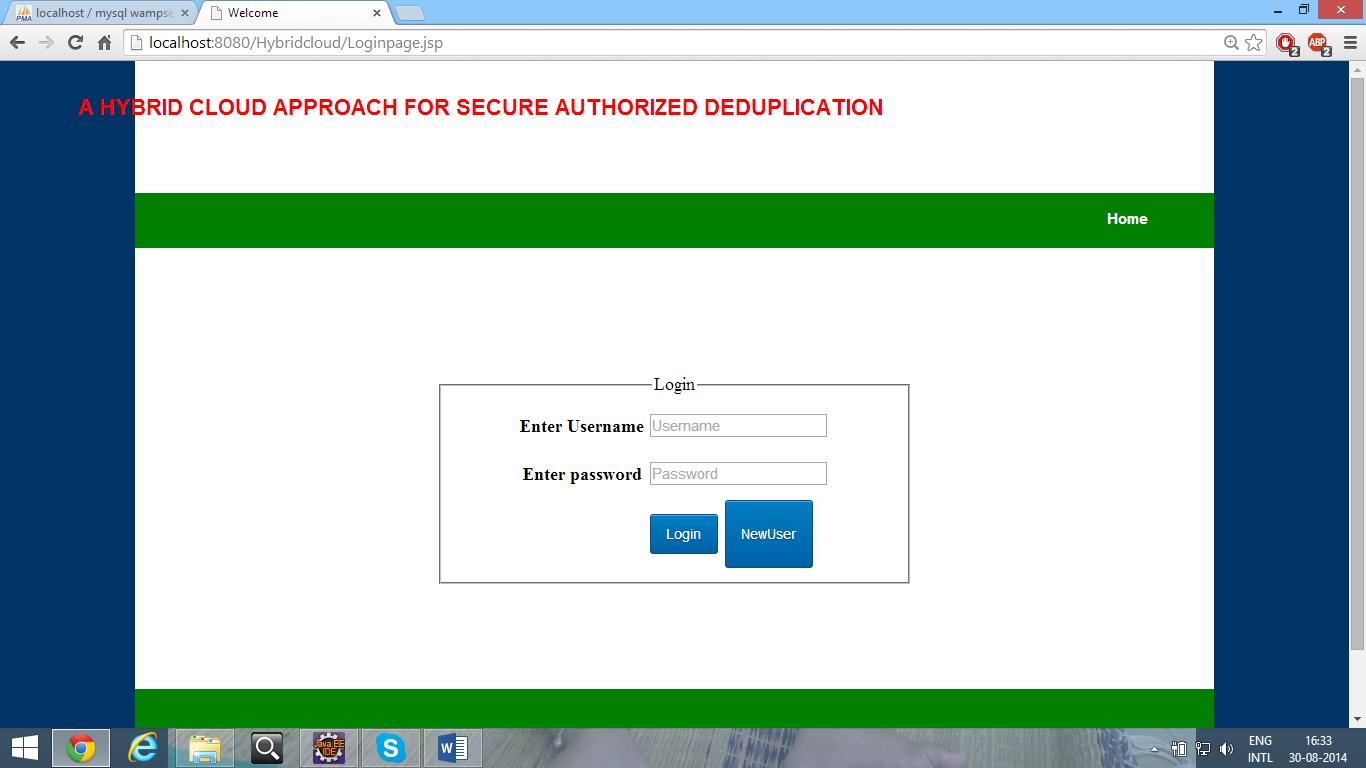
**4.3** **Sample screen shots**



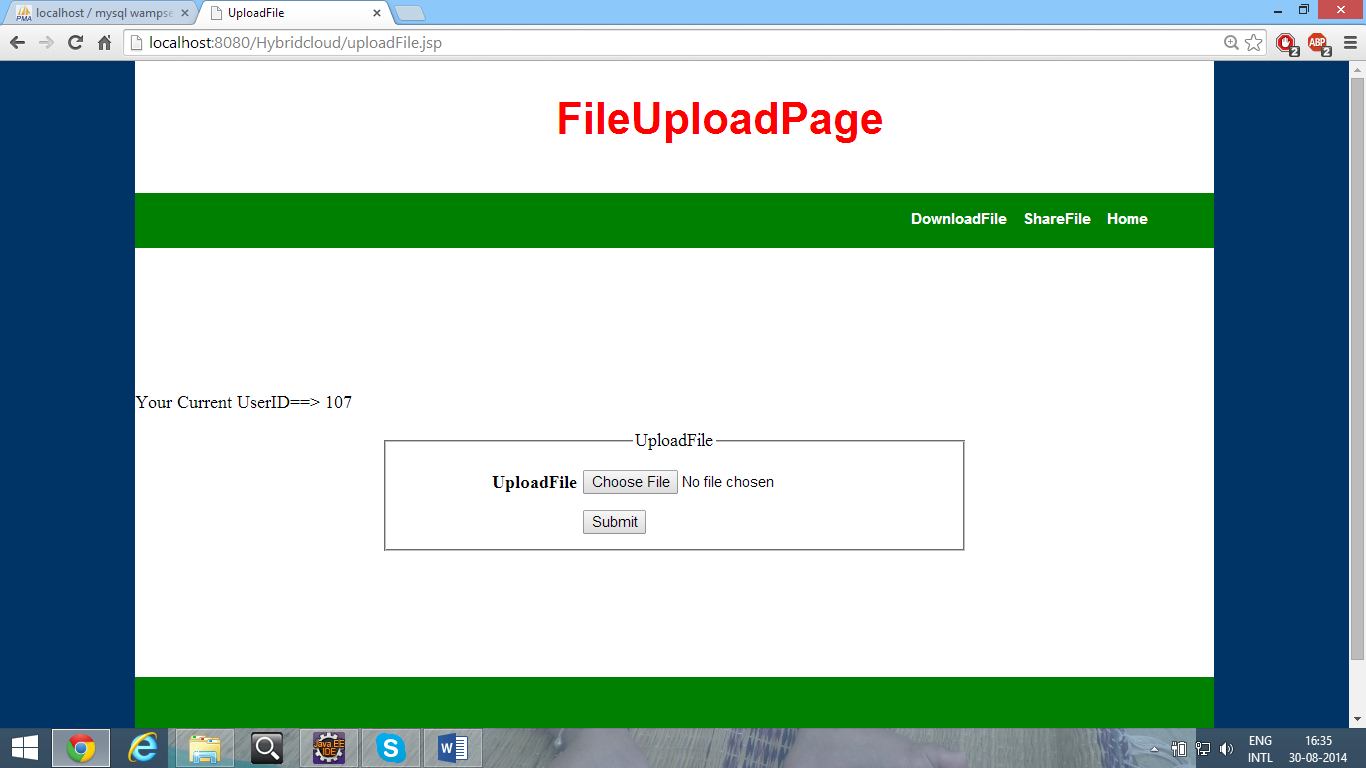
***Fig 4.1 Home page***



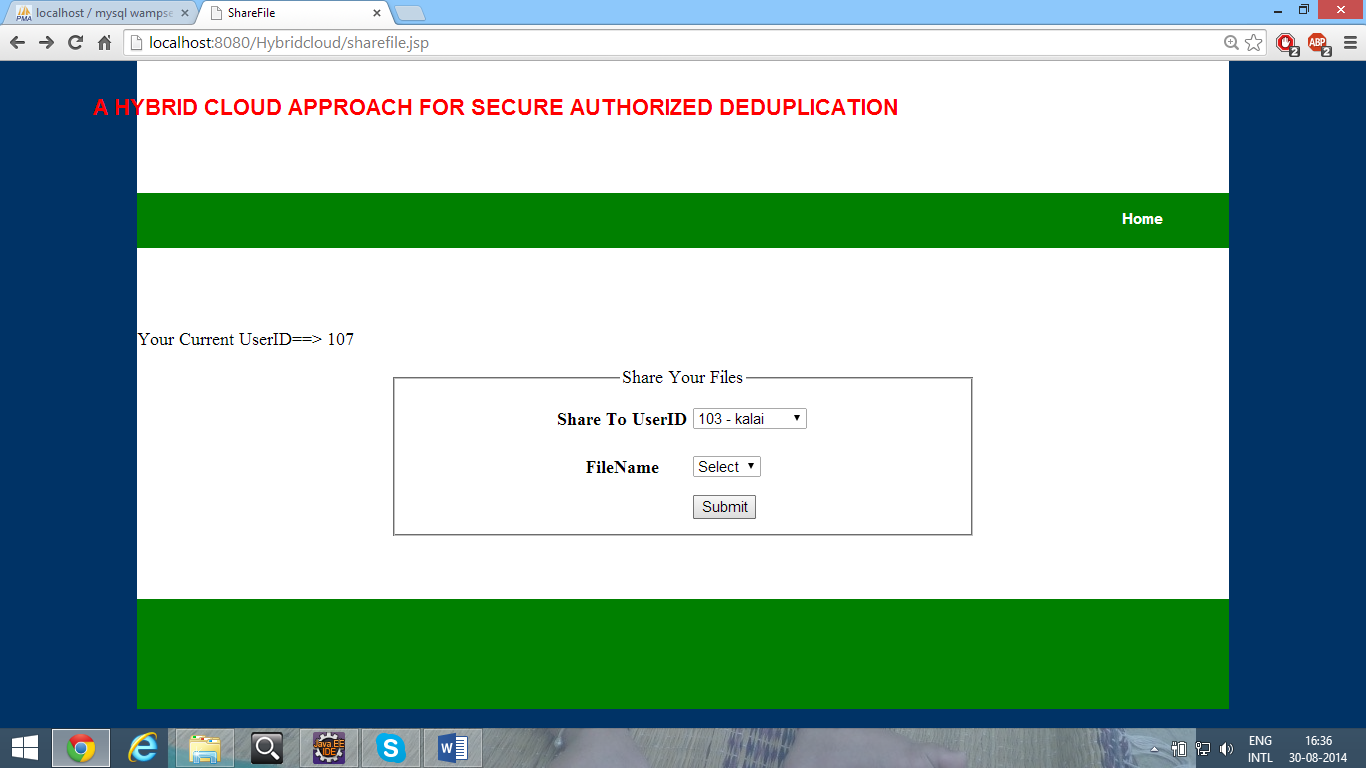
***Fig 4.2 Registration page***



***Fig 4.3 Login page***



***Fig 4.4 File upload page***



***Fig 4.5 Share page***

**5. SOFTWARE TESTING**

**Introduction**

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 thesoftware 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.

**5.1 Types of tests**

**5.1.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**5.1.2 Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields and successfully unit testing, the combination of components is correct and consistent.

**5.1.3 Functional testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

**5.1.4 System testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**5.1.5 White box testing**

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

**5.1.6 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.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test results**

All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test results**

All the test cases mentioned above passed successfully. No defects encountered.

**Table for test cases**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Description | Test Procedure | Test Input | Expected Result | Actual Result | Status |
| TC 001 | To check the valid user details | Type user id and password in login page | Userid:user1  Password:user123 | Should accept | Valued accepted logged in successfully | Passed |
| TC  002 | To check the invalid user details | Type a invalid user id and password in login page | Userid:uxx1  Password:u23 | Login  unsuccessful | Invalid user id or password login not completed | passed |
| TC 003 | To upload a new file | Click upload option and browse a new file | Welcome.txt (to upload) | Should accept | File accepted  Uploaded | Passed |
| TC 004 | To upload a existing file | Click upload option and browse a already uploaded file | We.txt  (to upload) | Should not accept | File not accepted already exist | Passed |
| TC 004 | To download a uploaded file | Click download and choose a file needed to be downloaded | Welcome.txt | Should be downloaded | File downloaded | Passed |
| TC 006 | To share a file | Click the share option and choose the desired user id | User1 | File should be share | File shared | Passed |

**Requirement Specification**

**Hardware requirement**

# Processor - Intel core

* Speed - 1.70 Ghz
* RAM - 256 MB(min)
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

# Software requirement

* Operating System - Windows 8
* Programming Language - JAVA/J2EE
* Java Version - JDK 1.7 & above.
* Database - MYSQL
* IDE - Eclipse

**5.2 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.



***Fig 5.1 Diagram of java programming language***

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.



***Fig 5.2 Diagram of java virtual machine***

### 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) and The Java Application Programming Interface (Java API). 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.



***Fig 5.3 Diagram of java API***

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.

**The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.

**Applets**: The set of conventions used by applets.

**Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.

**Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.

**Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.

**Software components**: Known as JavaBeansTM, can plug into existing component architectures.

**Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).

**Java Database Connectivity (JDBCTM)**: Provides uniform access to a wide range of relational databases.

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.



***Fig 5.4 Diagram of java 2 SDK***

### ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

**JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

### 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.

**SQL level API**

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

**SQL conformance**

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

Finally we decided to proceed the implementation using Java Networking and for dynamically updating the cache table we go for MS Access database. Java ha two things: a programming language and a platform. Java is a high-level programming language that is all of the following, Simple**,** Architecture-neutral**,** Object-oriented**,** Portable**,** Distributed**,** High**-**performance, Interpreted, Multithreaded, Robust, Dynamic and Secure. You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM).

## 5.3 Networking

### TCP/IP stack

The TCP/IP stack is shorter than the OSI one. 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 IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet.

### 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.

### Network address

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



***Fig 5.5 Diagram of 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**

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

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);

JFreeChart is "open source" or, more specifically, [free software](http://www.gnu.org/philosophy/free-sw.html). It is distributed under the terms of the [GNU Lesser General Public Licence](http://www.gnu.org/licenses/lgpl.html) (LGPL), which permits use in proprietary applications.

## 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.

## Time series chart interactivity

Implement a new (to JFreeChart) 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.

## Dashboards

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

## Property Editors

The property editor mechanism in JFreeChart 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 JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications.

**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.

**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 preverification plays in this process.

**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.

**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) and Connected Device Configuration (CDC).

**J2ME profiles**

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 J2ME 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.

**Profile 1: KJava**

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

**Profile 2: MIDP**

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application Development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

\* java.lang \* java.io \* java.util \* javax.microedition.io

\* javax.microedition.lcdui \* javax.microedition.midlet

\* javax.microedition.rms

**6 CONCLUSION**

In this project, the notion of authorized data deduplication was proposed to protect the data security by including differential privileges of users in the duplicate check. We also presented several new deduplication constructions supporting authorized duplicate check in hybrid cloud architecture, in which the duplicate-check tokens of files are generated by the private cloud server with private keys. Security analysis demonstrates that our schemes are secure in terms of insider and outsider attacks specified in the proposed security model. As a proof of concept, we implemented a prototype of our proposed authorized duplicate check scheme and con-duct test bed experiments on our prototype. We showed that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer.

**REFERENCES**

[1] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In Proc. of USENIX LISA, 2010.

[2] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Server-aided encryption for deduplicated storage. In USENIX Security Symposium, 2013.

[3] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In EUROCRYPT, pages 296–312, 2013.

[4] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. J. Cryptology, 22(1):1–61, 2009.

[5] A.Rahumed, H. C. H. Chen, Y. Tang, P. P. C. Lee, and J. C. S. Lui. A secure cloud backup system with assured deletion and version control. In Security in Cloud Computing, 2011.

[6] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider. Twin clouds: Secure cloud computing with Low Latency. In Workshop on Cryptography and Security in Clouds (WCSC 2011), 2011.

[7] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In ICDCS, pages 617–624, 2002.

[8] D. Ferraiolo and R. Kuhn. Role-based access controls. In 15th NIST-NCSC National Computer Security Conf., 1992.

[9] C. Ng and P. Lee. Revdedup: A reverse deduplication storage system optimized for reads to latest backups. In Proc. of APSYS, Apr 2013.

[10] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou. Secure deduplication with efficient and reliable convergent key management. In IEEE Transactions on Parallel and Distributed Systems, 2013.