A Novel Identity based Secure Distributed Data Storage System in Cloud Computing based on Database –as-a-Service

D.S.Priyanka [#1], Mr. B. Prasad [*2]

[#1] M.TECH, Department of Information Technology, Vignan's Institute of Information Technology,JNTU- KAKINADA, Andhra Pradesh, India
[#1] priyanka.dadi@yahoo.com

[*2] Associate Professor, Department of Information Technology, Vignan's Institute of Information Technology, Vishakhapatnam, Andhra Pradesh, India
[*2] prasad_bode@yahoo.com

Abstract

Storage security is a specialty area of security that is concerned with securing data storage systems and ecosystems and the data that resides on these systems. The same storage security schema if applied on distributed environment may gives more security for the data which is stored on remote servers. Now a day's cloud has become one of the fascinating domain for storing a large number of data on to a server from remote locations, stores them and gives facility for accessing the stored data by using a facility called as "PAUZ". In general the data which is stored in the cloud is encrypted and stored on to the server location with the help of intermediate proxy servers. In general proxy servers are those which can convert encrypted files for the data owner to encrypted files for the data receiver without knowing the original information. For space complexity the data owner will remove the original files from his system. As data was stored on a remote server, we must mainly concentrate on two major issues like confidentiality and integrity of the outsourced data. In this paper, we have proposed two new identity-based secure distributed data storage (IBSDDS) schemes. Our two new schemes can capture the following properties: (1) Firstly whenever the data/file which is uploaded by file owner on remote server he will decide the file access permission independently on his own without the help of any third party private key generator (PKG).(2) For one query, a receiver can only access appropriate one file, instead of all files that are stored by the owner. Our two new schemes are secure against the collusion attacks, namely even if the receiver can compromise the proxy servers; he cannot obtain the owner's secret key. To the best of our knowledge, it is the first IBSDDS schemes where access permissions is made by the owner for an exact file and collusion attacks can be protected in the standard model. We also implemented mailing concept as an extension for this paper in order to send the file name and key to the requested receiver mail id instead of sending directly along with the data. This facility gives high security as all the participating parties will have a proper authentication before request arrives to them, so that there will be no chance of getting the files by unauthorized users.

Keywords

Cloud Computing, Access Control Policy, Data Security, Identity-Based Data Storage Systems, Encryption, Proxy Server

1. Introduction

Cloud computing is a new computing in which large groups of remote servers are networked to allow the centralized data storage, and online access to computer services or resources. Clouds can be classified as public, private or hybrid. Cloud computing relies on restricting sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network which is shown clearly in figure 1. At the foundation of cloud computing is the broader concept of converged infrastructure and services. Cloud computing is a new technology which emerges with a lot of facilities that are really beneficial for the end users to manage their personal files with the convenient notion called database-as-a-service (DAS) [1], [2], [3].

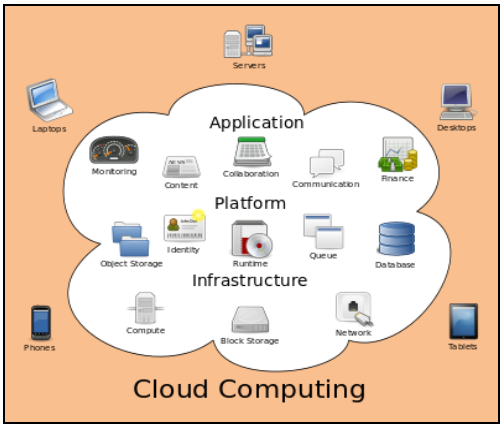


Figure 1. Represents the architecture of Cloud Computing

By using this new DAS schemes, any data user can outsource his/her encrypted data files to untrusted proxy servers. Proxy servers can also perform some functions on the outsourced cipher texts data which was stored by the data owners without knowing anything about the original files. Unfortunately, this technique hasn't been in practice extensively. The main reason which is behind this cause is users are generally concentrating on the main factors like confidentiality, integrity and query of the outsourced files as cloud computing is a lot more complicated than the local data storage systems, as the cloud is managed by an untrusted third party user. As the data owner stores his valuable data on a third party proxy server, he/she will remove the original content in their system in order to reduce the space wastage. Therefore, how to guarantee the outsoured files which can't be accessed by the unauthorized users and not modified by proxy servers is an important problem that has been considered in the data storage research community. Furthermore, we also need to concentrate how efficiently the data user can access his/her files which were stored by them in proxy server. Consequently, a lot of research around these topics grows significantly.

Today in cloud computing domain, confidentiality is proposed mainly to prevent unauthorized users from accessing the most sensitive data as it is subject to unauthorized disclose and access after being outsourced. Since the introduction of Database-As-A-Service, the confidentiality of outsourced data has been the primary focus among the research community. To provide confidentiality to the outsourced data, encryption schemes are deployed **[4], [5], [6], [7], [8].**

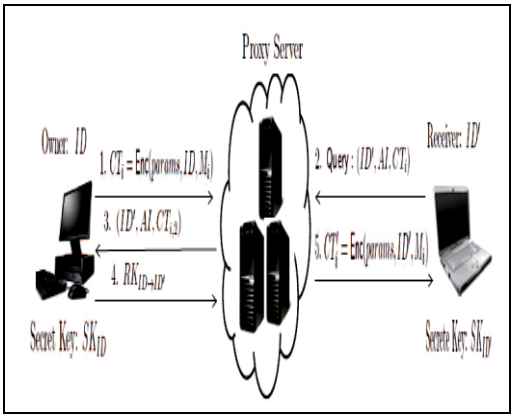


Figure 2. Represents the architecture of the Proposed IBSDDS Schema

In cloud computing domain, integrity can prevent outsourced data from being re- placed and

modified. Some schemes have been proposed to protect the integrity of the outsourced data, such as proof of retrievability [9], [10], [11], [12], [13] and provable data possession [14], [15], [16]. In these schemes, digital signature schemes and message authentication codes (MAC) are deployed. Query in data storage is executed between a receiver and a proxy server. The proxy server can perform some functions on the outsourced ciphertexts and convert them to those for the receiver which is shown clearly in figure 2. As a result, the receiver can obtain the data outsourced by the owner without the proxy server knowing the content of the data [17], [18], [19], [20].

2. Literature Survey

In this section we will describe the assumptions that are used in the proposed paper. This section mainly surveys on the literature of our proposed new identity- based secure distributed data storage (IBSDDS) schemes.

2.1 Cloud Infrastructure Provider

Cloud computing, or in simpler shorthand just "the cloud", also focuses on maximizing the effectiveness of the shared resources. Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand. This can work for allocating resources to users which is shown in Figure 3. For example, a cloud computer facility that serves European users during European business hours with a specific application (e.g., email) may reallocate the same resources to serve North American users during North America's business hours with a different application (e.g., a web server). This approach should maximize the use of computing power thus reducing environmental damage as well since less power, air conditioning, rackspace, etc. are required for a variety of functions. With cloud computing, multiple users can access a single server to retrieve and update their data without purchasing licenses for different applications.The term "moving to cloud" also refers to an organization moving away from a traditional CAPEX model (buy the dedicated hardware and depreciate it over a period of time) to the OPEX model (use a shared cloud infrastructure and pay as one uses it).Proponents claim that cloud computing allows companies to avoid upfront infrastructure costs, and focus on projects that differentiate their businesses instead of on infrastructure. Proponents also claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and enables IT to more rapidly adjust resources to meet fluctuating and unpredictable business demand. Cloud providers typically use a "pay as you go" model. This can lead to unexpectedly high charges if administrators do not adapt to the cloud pricing model. The present availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture, and autonomic and utility computing have led to a growth in cloud computing. Cloud vendors are experiencing growth rates of 50% per annum.

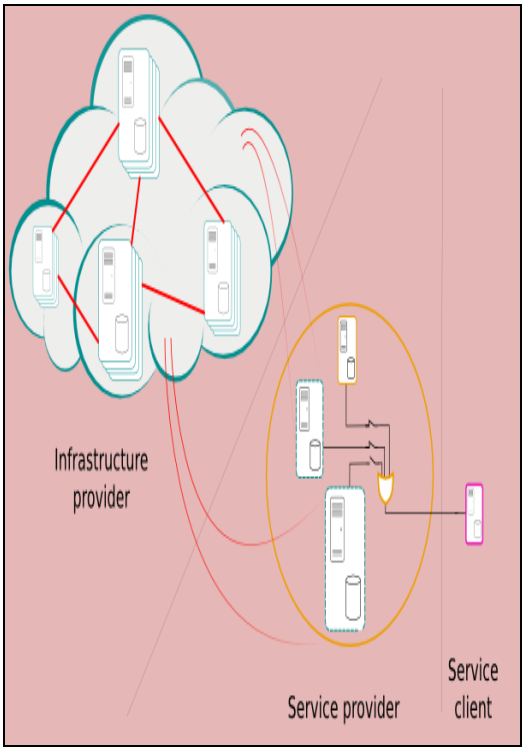


Figure 3.Represents the architecture of Cloud Infrastructure as a provider

2.2 Data Storage Systems

Data storage systems are the new systems which enable users to store their data to external proxy servers to enhance the access and availability, and reduce the maintenance cost. Author Samaritan and Author Vimercati **[21]** addressed the privacy issues in data outsourcing expanding from the data confidentiality to data utility, and pointed out the main research directions in the protection of the externally stored data.

2.3 Networked File Systems

In these systems, proxy servers are assumed to be trusted. They authenticate receivers and validate access permissions. The interactions be-tween the proxy servers and receivers are executed in a secure channel. Therefore, these systems cannot provide an end-to-end data security, namely they cannot ensure the confidentiality of the data stored at the proxy server [22]. In these schemes, a receiver authenticates himself to the proxy server using his password. Then, the proxy server passes the authentication result to the file owner. The owner will make an access permission according to the received information.

2.4 Identity-based Proxy Re-encryption Schema (IBPRES)

Proxy cryptosystem was introduced by Mambo and Okamoto to delegate the decryption power to a designated decryptor. Then, Blaze, Bleumer, and Strauss proposed an atomic proxy cryptosystem where a semi-trusted proxy server can transfer a cipher text for the original decryptor to a cipher text for the designated decryptor without knowing the plaintext.

Proxy cryptosystem as an efficient primitive has been used in email forwarding, law enforcement and data storage. Identity- based cryptosystem introduced by Shamir is a sys- tem where the public key can be any arbitrary string and the secret key is issued by a trusted party called the private key generator (PKG). Being different from public key infrastructure (PKI), two parties can communicate directly without verifying their public key certificates in identity-based systems. The first secure and practical identity-base encryption (IBE) was proposed by Boneh and Franklin based on pairing.

3. Proposed Algorithm and its Methodology

We are using identity-based secure distributed data storage (IBSDDS) scheme as our proposed scheme. There are mainly four entities in (IBSDDS) scheme:

1. The Private Key Generator (PKG),
2. The Data Owner,
3. The Proxy Server And
4. The Receiver.

The PKG validates the users' identities and issues secret keys to them. The data owner encrypts his data and outsources the ciphertexts to the proxy servers.

Proxy servers store the encrypted data and transfer the cipher text for the owner to the ciphertext for the receiver when they obtain access permission (re-encryption key) from the owner. The receiver authenticates himself to the owner and decrypts the re-encrypted ciphertext to obtain the data. An IBSDDS scheme consists of the following algorithms:

Setup(1^l) → (params,*MSK*)

The setup algorithm takes as input a security parameter 1^l,and outputs the public parameters params and a master secret *MSK*.

KeyGen (params, ID, MSK)→SK_{ID}.

The key generation algorithm takes as input the public parameters params,an identity ID and the master secret key MSK,and outputs a secret

key SK_{ID} for the identity ID.

Encryption (params, ID, MSK) →CT_i.

Suppose that there are k messages $\{M_i\}$.To encrypt the message M_i,the encryption algorithm takes as input the public parameters params,the identity ID and the message M,and outputs the ciphertextCT_i = $(C_{i,1},C_{i,2})$, for i = 1,2,….,k.It sends the ciphertexts CT_i to the proxy servers.

Query (ID', SKID', CTi) → AI.

The query algorithm takes as input the receiver's identity ID',the receiver'ssecret key SK_{ID} and the ciphertextCT_i,and outputs an authentication information AI.It sends (ID',AI,CT_i) to the proxy server. The proxy server redirects (ID', AI, Ci, 2) to the owner with identity ID.

Decryption

There are two algorithms. One is for the owner and the other is for the receiver.

1) **Decryption$_1$ (params, SKID, CTi) → M_i.**

The owner decryption algorithm takes as input the public parameters params, the owner's secret key SK_{ID} and cipher text CT_i, and outputs the message M_i.

2) **Decryption $_2$(params, SKID'CTi') →M_i.**

The receiver decryption algorithm takes as input the public parameters params, the receiver's secret key $SK_{ID'}$ and the re-encrypted cipher text CT_i, and outputs the message M_i.

4. Proposed Algorithm Flow Chart

The below diagram indicates the proposed IBSDDS flow of actions from start to end. This will clearly gives an idea how the IBSDDS algorithm gives security for the cloud data storage.

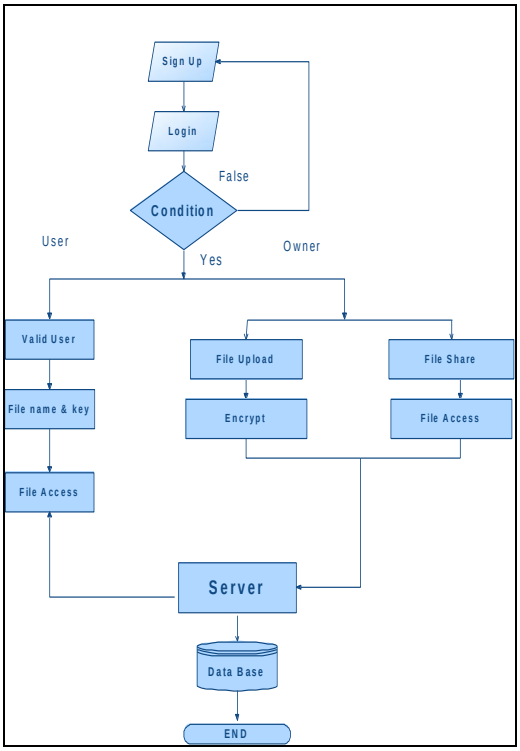


Figure 4.Represents the Flow chart of proposed IBSDDS Algorithm

5. Implementation Modules

Implementation is a stage where the theoretical design is automatically converted into practical form. We have implemented the current application in Java Programming language with Front End as JSP, HTML, and Back End as My-SQL Server data base.

5.1 Main Modules

This current application is mainly divided into following four modules based on the project flow.

1. Data Owner Module
2. Private key Generator Module
3. Proxy Server Module
4. The Receiver Module

1. Data Owner Module

In this module, first the new data owner registers and then gets a valid login credentials. After logged in, the data owner has the permission to upload their file into the Cloud Server. The data owner encrypts his data and outsources the ciphertexts to the proxy servers.

2. Private Key Generator Module

In this module, the private key generator (PKG) validates the users' identities and issues secret keys to them. The key is generated and sent to their respective mail id's with the file name and the corresponding key values.

3. Proxy Server Module

Proxy servers store the encrypted data and transfer the cipher text for the owner to the cipher text for the receiver when they obtain access permission (re-encryption key) from the owner. In these systems, proxy servers are assumed to be trusted. They authenticate receivers and validate access permissions. The interactions between the proxy servers and receivers are executed in a secure channel. Therefore, these systems cannot provide an end-to-end data security, namely they cannot ensure the confidentiality of the data stored at the proxy server. In these schemes, a receiver authenticates himself to the proxy server using his password. Then, the proxy server passes the authentication result to the file owner. The owner will make access permission according to the received information.

4. Receiver Module

The receiver authenticates himself to the owner and decrypts the re-encrypted Ciphertext to obtain the data. In these systems, an end to-end security is provided by cryptographic protocols which are executed by the file owner to prevent proxy servers and unauthorized users from modifying and accessing the sensitive files. These systems can be divided into two types: shared file system and non-shared system. In shared file systems the owner can share his files with a group of users. Cryptographic techniques deployed in these systems are key sharing, key agreement and key revocation. In non-shared file systems in order to share a file with another user, the owner can compute an access key for the user using his secret key. In these two systems, the integrity of the sensitive files is provided by digital signature schemes and message authentication codes (MAC).

5.2 Main Source Code for Proposed Application

Here we will discuss the main source code for the proposed application implementation. Here I will show the data base logic through which the current application is connecting its frond end user interface with the storage data. For this application we are using JSP as front end and My Sql as back end data base. As we implement the application in Java Programming language we are using JDBC as data base connectivity for implementing data base connection with user interface.Aslo we are using type 4 driver for implementing the connection with the data base and front end application.In this section we will be discussing some of the important sample codes for connecting the data base and the main logic for sending key to the registered user mail id's.

Sample Code for Databasecon

```
/*
 *  Designed and Developed by Priyanka @ Vignan M.Tech 2012 -2014.
 */
package design;
import java.util.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
```



```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.commons.fileupload.FileItem;
import
org.apache.commons.fileupload.FileUploadExceptio
n;
import
org.apache.commons.fileupload.disk.DiskFileItemF
actory;
import
org.apache.commons.fileupload.servlet.ServletFileU
pload;

public class upload extends HttpServlet {

   SimpleFTPClient client;
   File file;

  protected void processRequest(HttpServletRequest
request, HttpServletResponse response)
        throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-
8");
      PrintWriter out = response.getWriter();
      try {

         DiskFileItemFactory diskFileItemFactory =
new DiskFileItemFactory();
         diskFileItemFactory.setRepository(file);
         diskFileItemFactory.setSizeThreshold(1 *
1024 * 1024);

         ServletFileUpload newHUpload = new
ServletFileUpload(diskFileItemFactory);
         List items =
newHUpload.parseRequest(request);
         Iterator iterator = items.iterator();
         FileItem fileItem = (FileItem) iterator.next();
         Connection con = null;
      PreparedStatement pstm = null;
      FileInputStream fis = null;
      long size = 0;

         client = new SimpleFTPClient();
         client.setHost("ftp.drivehq.com");
         client.setUser("VIGNAN");
         client.setPassword("VIGNAN");
         client.setRemoteFile(fileItem.getName());

         boolean log = client.connect();

Class.forName("com.mysql.jdbc.Driver");
         con =
DriverManager.getConnection("jdbc:mysql://loc
alhost:3306/idbased", "root", "root");
         // List items =
uploadHandler.parseRequest(request);
         Iterator itr = items.iterator();
         String sql = "insert into files
(file,name,user,skey)values(?,?,?,?)";
         pstm = con.prepareStatement(sql);
        //  while (itr.hasNext()) {

            FileItem item = (FileItem) itr.next();
                  System.out.println("getD
"+item.getName());

pstm.setBinaryStream(1,item.getInputStream());
            pstm.setString(2, item.getName());
                                    HttpSession
session = request.getSession(true);

            //HttpSession session =
request.getSession(false);

            pstm.setString(3,
(String)session.getAttribute("us"));

                                    Random r =
new Random();
       int tt = r.nextInt(1000-500)+500;


            pstm.setString(4, Integer.toString(tt));
             pstm.execute();

            session.setAttribute("nn",
item.getName());
          System.out.println("Values inserted");

         if (log) {
            if
(client.uploadFile(fileItem.getInputStream())) {

response.sendRedirect("owneruserpage.jsp?msg=
sucess..!");
```



```
            } else {

response.sendRedirect("owneruserpage.jsp?msgg=
NOT sucess..!");
             }
          } else {
            out.println("not connected");
          }



      } catch (SQLException ex) {

Logger.getLogger(upload.class.getName()).log(Leve
l.SEVERE, null, ex);
      } catch (ClassNotFoundException ex) {

Logger.getLogger(upload.class.getName()).log(Leve
l.SEVERE, null, ex);
      } catch (FileUploadException ex) {

Logger.getLogger(upload.class.getName()).log(Leve
l.SEVERE, null, ex);
      } finally {
         out.close();
      }
   }

   // <editor-fold defaultstate="collapsed"
desc="HttpServlet methods. Click on the + sign on
the left to edit the code.">
   /**
    * Handles the HTTP
    * <code>GET</code> method.
    *
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific
error occurs
    * @throws IOException if an I/O error occurs
    */
   @Override
   protected void doGet(HttpServletRequest request,
HttpServletResponse response)
         throws ServletException, IOException {
      processRequest(request, response);
   }

   /**
    * Handles the HTTP
    * <code>POST</code> method.
    *
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific
error occurs
    * @throws IOException if an I/O error occurs
    */
   @Override
   protected void doPost(HttpServletRequest
request, HttpServletResponse response)
         throws ServletException, IOException {
      processRequest(request, response);
   }

   /**
    * Returns a short description of the servlet.
    *
    * @return a String containing servlet description
    */
   @Override
   public String getServletInfo() {
      return "Short description";
   }// </editor-fold>
}
```

Sample code for Mail Atatchments

```
%@page import="java.io.*,java.util.*,javax.mail.*"%>
<%@ page
import="javax.mail.internet.*,javax.activation.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*"
%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.util.Random"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%
String mail = null;
                System.out.println(".....1.....");
        String me = session.getAttribute("us").toString();
                System.out.println(".....2.....");
        String fm = session.getAttribute("nn").toString();
                System.out.println(".....3....."+fm);
        //String mail
=session.getAttribute("mm").toString();
        String frnd = request.getParameter("user");
                System.out.println(".....4.....");
```

```
        Random r = new Random();
        int ii = r.nextInt(100000 - 50000) + 50000;
        String k = Integer.toString(ii);
        // String k1 = ii+"";
                        System.out.println(".....5.....");
        System.out.println(" secert key is:" + ii);
                        System.out.println(".....6.....");
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:330
6/idbased", "root", "root");

          Statement st = con.createStatement();
        Statement st1 = con.createStatement();
        ResultSet rs = st1.executeQuery("select * from
regpage_user where user='" + frnd + "'");
        if (rs.next())
           mail = rs.getString("mail");
          // System.out.println(" friend name" + mail);

        int i = st.executeUpdate("update files set user ='" +
me + "',friend ='" + frnd + "', skey='" + ii + "' where name
= '" + session.getAttribute("nn").toString() + "' ");
String host="", user="", pass="";

host ="smtp.gmail.com"; //"smtp.gmail.com";

user ="satyapriyanka.tuni@gmail.com";
//"YourEmailId@gmail.com" // email id to send the emails

pass ="8886231618"; //Your gmail password

String SSL_FACTORY =
"javax.net.ssl.SSLSocketFactory";

String to =user; // out going email id

String from ="customerservice404@gmail.com"; //Email
id of the recipient
String subject=(String)session.getAttribute("subject");
//String subject ="welcome";

String messageText ="<b>"+user+"</b><br>password
is=<b>"+ii+"</b><br><b>File Name=</b>"+fm+"";

//session.setAttribute("userid",userid);
//session.setAttribute("password",c);
session.setAttribute("emailid",user);
boolean sessionDebug = true;
if (i != 0) {
Properties props = System.getProperties();
props.put("mail.host", host);
props.put("mail.transport.protocol.", "smtp");
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.", "true");
props.put("mail.smtp.port", "465");
props.put("mail.smtp.socketFactory.fallback", "false");
props.put("mail.smtp.socketFactory.class",
SSL_FACTORY);
Session mailSession = Session.getDefaultInstance(props,
null);
mailSession.setDebug(sessionDebug);
Message msg = new MimeMessage(mailSession);
msg.setFrom(new InternetAddress(from));
InternetAddress[] address = {new InternetAddress(mail)};
msg.setRecipients(Message.RecipientType.TO, address);
msg.setSubject(subject);
msg.setContent(messageText, "text/html"); // use setText if
you want to send text
Transport transport = mailSession.getTransport("smtp");
transport.connect(host, user, pass);
try {
transport.sendMessage(msg, msg.getAllRecipients());
//out.println("message successfully sended"); // assume it
was sent
//response.sendRedirect("key.jsp");
response.sendRedirect("share.jsp?sh=Secert key Send
success");
}
catch (Exception err) {

out.println("message not successfully
sended"+err.getMessage()); // assume it's a fail
}



        }

else {
            response.sendRedirect("share.jsp?shr=Action
fails");
        }

%>
```

6. Enhanced System Architecture

The following diagram represents the enhanced system architecture /flow of IBSDDS model which is clearly shown in figure.2. In the proposed paper, we have extended the same IBSDDS schema with more enhanced security by including a mailing concept like each and every user whenever registers in his/her account ,they should provide a valid email id, where the data owner uploads any data can select any of the user from the set of user list .The data owner will now try to give

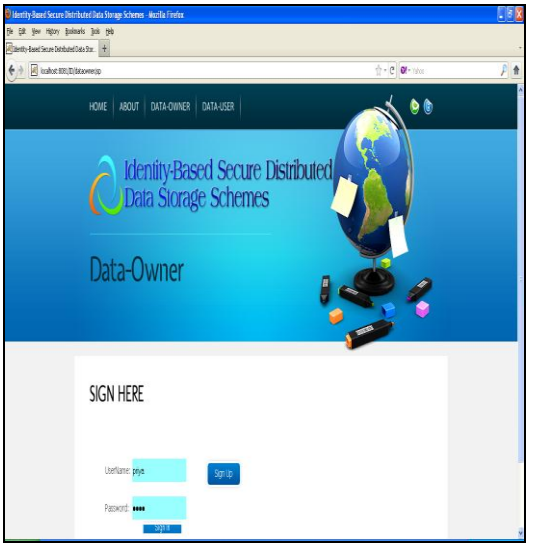
key for accessing the uploaded file for the selected user, the key is sent to the selected user registered mail id and the end user should verify his valid mail id and with that key only he can decrypt the data if not he cannot access his data files.In the above section we will be discusiing about the mail concept and its logic part.

7. Experimental Results

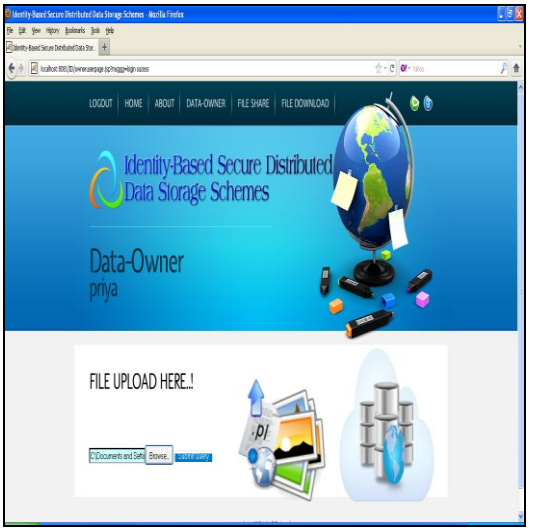
In this paper, we have proposed a new secure algorithm called as **IBSDDS** Algorithm for providing security for the data which is used for distributed data storage in the cloud. By designing this application we gave a new scope for the present cloud vendors a proof of how to give security for their data there large complex data storage under the remote systems.We have shown the application in a web interface with JEE 6.0 edition .In this JEE we are using front end as Java Server pages (JSP) and HTML pages. As we are deploying the application in web interface, we are using tomcat server for deploying the application .Hence we use tomcat 7.0 as the deployment web server. We are using My Sql data base for storing the data temporarily on to our system and then retrieve the same data whenever needed. This MY-SQL data base is chosen on highest priority as it has GUI support and it is always Auto Commit by its nature.

Sample Screen of Home page

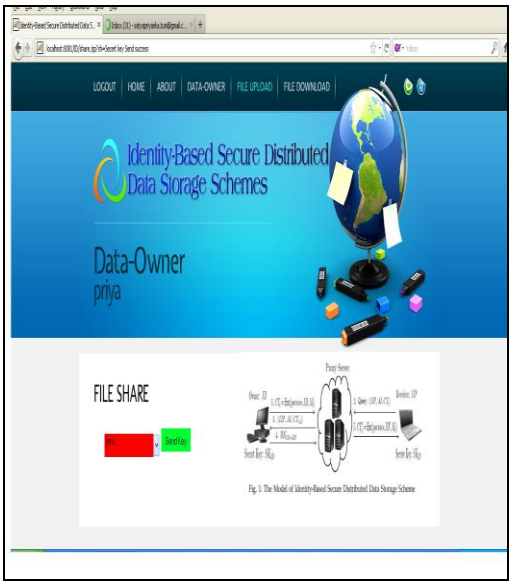
Sample Screen for Data Owner Login



Sample Screen for Uploading a File by Data Owner

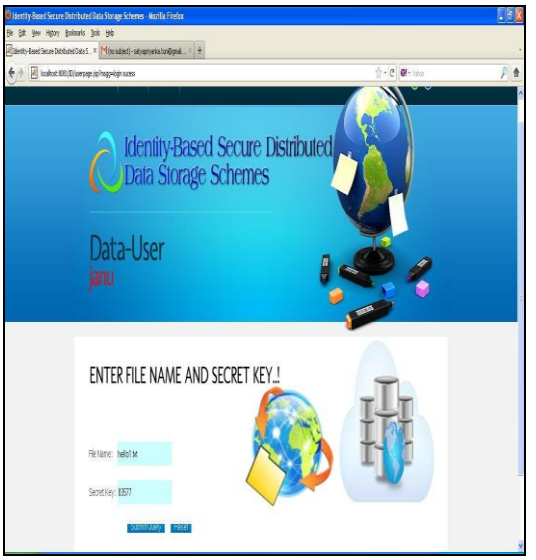


Sample Screen for giving file share permission

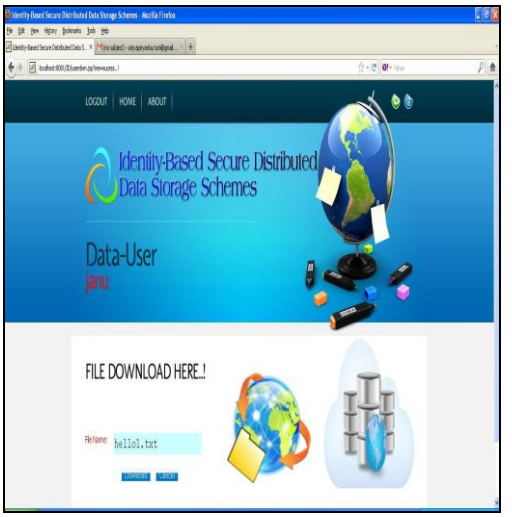


Sample Screen for User Registration

Sample Screen for User downloading the file with specifying File Name and Secret Key



Sample screen for Download the File by Data User



8. Conclusion

Now a day's by using secure distributed data storage schema, we can reduce a lot of burden of maintaining a large number of files from the data owner to proxy servers. Identity-based secure distributed data storage (IBSDDS) schemes are a special kind of distributed data storage schemes where users are identified by their identities and can communicate without the need of verifying the public key certificates. In this paper, we proposed two new IBSDDS schemes in standard model where, for one query, the receiver can only access one file, instead of all files. Furthermore, the access permission can be made by the owner, instead of the trusted party. Notably, our schemes are secure against the collusion attacks. The first scheme is CPA secure, while the second one is CCA secure.

9. Future scope

As a future work the same application should be extended for giving security for the files under transfer from one system to other. If there was any file been modified un-authorized then it should report the error or alert for the cloud server.

10. References

[1] H. Hacig¨um¨us, B. R. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proceedings: SIGMOD Conference - SIGMOD'02* (M. J. Franklin, B. Moon, and A. Ailamaki, eds.), vol. 2002, (Madison, Wisconsin, USA), pp. 216–227, ACM, Jun. 2002.

[2] L. Bouganim and P. Pucheral, "Chip-secured data access: Confidential data on untrusted servers," in *Proc. International Conference on Very Large Data Bases - VLDB'02*, (Hong Kong, China), pp. 131–142, Morgan Kaufmann, Aug. 2002.

[3] U. Maheshwari, R. Vingralek, and W. Shapiro, "How to build a trusted database system on untrusted storage," in *Proc. Symposium on Operating System Design and Implementation - OSDI'00*, (San Diego, California, USA), pp. 135–150, USENIX, Oct. 2000.

[4] A. Ivan and Y. Dodis, "Proxy cryptography revisited," in *Proc.Network and Distributed System Security Symposium - NDSS'03*, (San Diego, California, USA), pp. 1–20, The Internet Society, Feb. 2003.

[5] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," in *Proc. Network and Distributed System Security Symposium - NDSS'05*, (San Diego, California, USA), pp. 1–15, The Internet Society, Feb. 2005.

[6] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security*, vol. 9, no. 1, pp. 1–30, 2006.

[7] S. D. C. di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, and P. Samarati, "Effficient and private access to outsourced data," in *Proc. International Conference on Distributed Computing Systems - ICDCS'11*, (Minneapolis, Minnesota, USA), pp. 710–719, IEEE, Jun. 2011.

[8] H.-Y. Lin and W.-G. Tzeng, "A secure erasure code-based cloud storage system with secure data forwarding," *IEEE Transactions on Parallel and Distributed Systems*, Digital Object Indentifier 10.1109/TPDS.2011.252 2012.

[9] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Advances in Cryptology - ASIACRYPT'08* (J. Pieprzyk, ed.), vol. 5350 of *Lecture Notes in Computer Science*, (Melbourne, Australia), pp. 90–107, Springer, Dec. 2008.

[10] A. Juels and B. S. K. Jr., "PORs: Proofs of retrievability for large files," in *Proceedings: ACM Conference on Computer and Communications Security - CCS'07* (P. Ning, S. D. C. di Vimercati, and P. F.Syverson, eds.), (Alexandria, Virginia, USA), pp. 584–597, ACM,Oct. 2007.

[11] Y. Dodis1, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *Proc. Theory of Cryptography Conference - TCC'09* (O. Reingold, ed.), vol. 5444 of *Lecture Notes in Computer Science*, (San Francisco, CA, USA), pp. 109–127, Springer, Mar.2009.

[12] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in *Proc. ACM Cloud Computing Security Workshop - CCSW'09*, (Chicago, Illinois, USA), pp. 43–53, ACM, Nov. 13 2009.

[13] G. Ateniese, S. Kamara, and J. Katz, "Proofs of storage from homomorphic identification protocols," in *Proc. Advances in Cryptology - ASIACRYPT'09* (M. Matsui, ed.), vol. 5912 of *Lecture Notes in Computer Science*, (Tokyo, Japan), pp. 319–333, Springer, Dec. 2009.

[14] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson,and D. Song, "Provable data possession at untrusted stores,"in *Proc. ACM Conference on Computer and Communications Security- CCS'07* (P. Ning, S. D. C. di Vimercati, and P. F. Syverson, eds.),(Alexandria, Virginia, USA), pp. 598–610, ACM, Oct. 2007.

[15] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data

possession," in *Proc. International conference on Security and privacy in communication netowrks - SecureComm'08*, (Istanbul, Turkey), Sep., ACM, 2008.

[16] C. C. Erway, A. K¨upc¨u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. ACM Conference on Computer and Communications Security - CCS'09* (E. Al-Shaer, S. Jha, and A. D. Keromytis, eds.), (Chicago, Illinois, USA), pp. 213–222, ACM, Nov. 2009.

[17] B. Carbunar and R. Sion, "Toward private joins on outsourced data," *IEEE Transactions on Knowlege and Data Engineering*, vol. 9, no. 24, pp. 1699–1710, 2012.

[18] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE Transactions on Knowlege and Data Engineering*, p. Digital Object Indentifier 10.1109/TKDE.2011.78.

[19] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1214–1221, 2011.

[20] M. L. Yiu, G. Ghinita, C. S. Jensen, and P. Kalnis, "Enabling search services on outsourced private spatial data," *The VLDB Journal*, vol. 19, no. 3, pp. 363–384, 2010.

[21] P. Samarati and S. D. C. di Vimercati, "Data protection in outsourcing scenarios: Issues and directions," in *Proc. ACM Symposium on Information, Computer and Communications Security - ASIACCS'10* (D. Feng, D. A. Basin, and P. Liu, eds.), (Beijing, China), pp. 1–14, ACM, Apr. 2010.

[22] V. Kher and Y. Kim, "Securing distributed storage: Challenges, techniques, and systems," in *Proc. ACM Workshop On Storage Security And Survivability - StorageSS'05* (V. Atluri, P. Samarati, W. Yurcik, L. Brumbaugh, and Y. Zhou, eds.), (Fairfax, VA, USA), pp. 9–25, ACM, Nov. 2005.

