

ABSTRACT

Provable Data Possession (PDP) enables cloud users to verify the data integrity without retrieving the entire file. all the existing PDP schemes rely on the Public Key Infrastructure (PKI).The scheme is efficient, flexible and supports private verification, delegated verification and public verification.ID-DPDP is flawed since it fails to achieve soundness .Fix the flaw by presenting a generic construction .A new ID-DPDP protocol is obtained by extending the basic ID-PDP to multiple clouds environment. With data storage and sharing services in the cloud, users can easily modify and share data as a group. To ensure shared data integrity can be verified publicly, users in the group need to compute signatures on all the blocks in shared data. Different blocks in shared data are generally signed by different users due to data modifications performed by different users. For security reasons, once a user is revoked from the group, the blocks which were previously signed by this revoked user must be re-signed by an existing user. The straight forward method, which allows an existing user to download the corresponding part of shared data and re-sign it during user revocation, is inefficient due to the large size of shared data in the cloud. In this work, we propose a novel public auditing mechanism for the integrity of shared data with efficient user revocation in mind. By utilizing the idea of proxy re-signatures, we allow the cloud to re-sign blocks on behalf of existing users during user revocation, so that existing users do not need to download and re-sign blocks by themselves. In addition, a public verifier is always able to audit the integrity of shared data without retrieving the entire data from the cloud, even if some part of shared data has been re-signed by the cloud. Moreover, our mechanism is able to support batch auditing by verifying multiple auditing tasks simultaneously. Experimental results show that our mechanism can significantly improve the efficiency of user revocation.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	
	LIST OF FIGURES	
	LIST OF ABBREVIATION	
1	INTRODUCTION	
	1.1 Public Sector Auditing	
	1.1.2 Types Of Public-Sector Audit	
	1.2 User Revocation	
2	LITERATURE REVIEW	
	2.1 A View Of Cloud Computing	
	2.2 Provable Data Possession	
	At Untrusted Stores	
	2.3 Compact Proofs Of	
	Retrievability	
	2.4 Ensuring Data Storage	
	Security In Cloud Computing	
3	SYSTEM ANALYSIS	
	3.1 Existing System	
	3.1.2 Disadvantages	
	3.2 Proposed System	
	3.2.1 Advantages	
4	SYSTEM REQUIREMENTS	
	4.1 Hardware Requirement	
	4.2 Software Requirement	

5

MODULE DESCRIPTION

5.1.1 Registration & Login

5.1.2 Efficient Key Generation

5.1.3 Upload File to Data Multi

Cloud Server

5.1.4 Download File from Data Multi

Cloud Server

5.1.5 Public Auditing with User

Collision Public Verifier

6

SYSTEM DESIGN

6.1 Data Flow Design

6.2 System Flow Diagram

7

SYSTEM TESTING AND

IMPLEMENTATION

7.1 Input Testing

7.2 Output Testing

7.3 System Testing

7.4 Black Box Testing

7.5 System Testing Methodology

7.6 Unit Testing

7.7 Integration Testing

8

CONCLUSION

8.1 Conclusion

8.2 Future Enhancement

9

APPENDIX -1

9.1 Sample Code

APPENDIX-2

9.2 Output Screenshot

10

REFERENCES

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
6.1	DATA FLOW DESIGN	
6.2	SYSTEM FLOW DIAGRAM	

LIST OF ABBREVIATIONS

PDP	- Provable Data Possession
PkI	- Public Key Infrastructure
UL	- User List
ID-DPDP	- Identity- Based Provable Data Possession Revisited
GKC	- Group Key Controller

CHAPTER 1

INTRODUCTION

Cloud Storage is a service where data is remotely maintained, managed, and backed up. Cloud computing, a new kind of Internet-based computing, provides convenient, on-demand network access. Provable Data Possession (PDP) verifies the data integrity by sampling random sets of blocks

1.1 PUBLIC SECTOR AUDITING

Public-sector auditing is essential in that it provides legislative and oversight bodies, those charged with governance and the general public with information and independent and objective assessments concerning the stewardship and performance of government policies, programmes or operations.

1.1.2 TYPES OF PUBLIC-SECTOR AUDIT:

The three main types of public-sector audit are defined as follows:

1. Financial Audit
2. Performance Audit
3. Compliance Audit

1.2 USER REVOCATION

In cloud computing user can easily share and modify data inside group. Here data integrity can be easily ensured by using the public signature of existing users in the cloud group. By using data sharing services in cloud user can able modify data as a group on cloud. Integrity of these services can be improved using signature of public existing users on group. Shared data is divided into blocks. All user are responsible for modifying data on different blocks. When any user shows malicious activity in group then that user must be revoked for security purpose.

As shared Data is exported to the cloud and existing users no longer store it on native devices, simplest method to re-calculate user signature during revocation is to increase associate degree of existing user for preliminary transfer the blocks maliciously signed by the revoked user, It first examines the efficiency of these blocks, and then blocks are re-sign with uploading data on cloud.

CHAPTER 2

LITERATURE REVIEW

2.1 A VIEW OF CLOUD COMPUTING

In this work, M. Armbrust, A. Fox, R. Griffith, et.al[2] has proposed Cloud Computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it.

2.2 PROVABLE DATA POSSESSION AT UNTRUSTED STORES

In this work, G. Ateniese, R. Burns, et.al[3] has proposed provable data possession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs.

2.3 COMPACT PROOFS OF RETRIEVABILITY

In this work, H. Shacham and B. Waters, et.al [4] has proposed In a proof-of-retrievability system, a data storage center must prove to a verifier that he is actually storing all of a client's data. The central challenge is to build systems that are both efficient and provably secure. A proof-of-retrievability protocol in which the client's query and server's response are both extremely short.

2.4 ENSURING DATA STORAGE SECURITY IN CLOUD COMPUTING

In this work, C. Wang, Q. Wang, et.al [5] has proposed Cloud Computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. Cloud data storage security, which has always been an important aspect of quality of service.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the Existing system, Iolus approach proposed the notion of hierarchy subgroup for scalable and secure multi-cloud. In public auditing for shared data revocation, a large communication group is divided into smaller subgroups. When a group member joins or leaves only affect subgroup only while the other subgroup will not be affected. It has the drawback of affecting data path. This occurs in the sense that there is a need for translating the data that goes from one subgroup, and thereby one key, to another. This becomes even more problematic when it takes into account that the PDP has to manage the subgroup and perform the translation needed. The PDP may thus become the bottleneck.

3.1.2 DISADVANTAGES

- The Group controller takes all responsibilities for the group such as CP-HABE on Multi cloud storage Key Generation.
- Sub work load is high for GKC.
- The group members are not able to communicate with any other groups during the re- keying process
- Collusion may occur.
- Data loss is too high during GKC losses.

3.2 PROPOSED SYSTEM

A novel multi-cloud Authentication protocol, namely CP-HABE, including two schemes. Each subgroup is treated almost like a separate multi-cloud group and is managed by a trusted group security intermediary identity Hierarchal Attribute based distributed provable data possession (CP-HABE). This is a desirable feature especially for the large-scale network systems, because it minimizes the problem of concentrating the workload on a single entity.

3.2.1 ADVANTAGES

- There is no GKC.
- Self verification for each transfer between sender and receiver.
- The group members are not affected by the CP-HABE on Multi cloud storage Key Generation process when they are willing to communicate with any other group members.
- High in security and Improved performance.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENT

This section gives the details and specification of the hardware on which the system is expected to work.

- **Processor Type** : Pentium IV
- **Speed** : 2.4 GHZ
- **RAM** : 256 MB
- **Hard disk** : 20 GB
- **Keyboard** : 101/102 Standard Keys

4.2 SOFTWARE REQUIREMENT

This section gives the details of the software that are used for the development.

- **Operating System** : Windows 10
- **Programming Package** : Net Beans IDE .13
- **Coding Language** : Java

CHAPTER 5

5.1 MODULE DESCRIPTION

- 5.1.1 Registration & Login
- 5.1.2 Efficient Key Generation
- 5.1.3 Upload File to Data Multi Cloud Server
- 5.1.4 Download File from Data Multi Cloud Server
- 5.1.5 Public Auditing with User Collision in Public Verifier

5.1.1 REGISTRATION & LOGIN

The first User entered the username, password, and chooses any one group id then register with Data Cloud Server. This user added in this particular group. Then entered the username, password and choose the user's group id for login.

5.1.2 EFFICIENT KEY GENERATION

In Key Generation module, every user in the group generates public key and private key. User generates a random, and outputs public key and private key. Without loss of generality, In the approach, assume user u_1 is the original user, who is the creator of shared data. The original user also creates a user list (UL), which contains ids of all the users in the group. The user list is public and signed by the original user.

5.1.3 UPLOAD FILE DATA MULTI CLOUD SERVER

The user wants to upload a file. So the user split the files into many blocks. Next encrypt each blocks with the public key. Then, the user generate signature of each blocks for authentication purpose. Then upload each block cipher text with signature, block id and signer id. These metadata and Key Details are stored in Public Verifier for public auditing.

5.1.4 DOWNLOAD FILE FROM DATA MULTI CLOUD SERVER

The next user or group member wants to download a file. So the user gives the filename and gets the secret key. Then entered this secret key. If this secret key is valid then the user able to decrypt this downloaded file. Else, the next user entered wrong secret key then the user1 blocked by Public Verifier. If this secret key is valid then decrypt each block and verify the signature.

If both signatures are equal then combine all blocks then get the original file.

5.1.5 PUBLIC AUDITING WITH USER COLLISION IN PUBLIC VERIFIER

In Public verifier method, the User who entered the wrong secret key then blocked by the public verifier. Next the user added public verifier collision user list. Then the user wants to tries to download any file, the Data Cloud Server replies his blocked information. Then the user wants to un collision, so they ask the public verifier. Finally the public verifier unrevoked this user. Next the user able to download any file with its corresponding secret key. In this approach, by utilizing the idea of proxy re-signatures, once a user in the group is collision, the Data Cloud Server is able to re-sign the blocks, which were signed by the collision user, with a resigning key.

CHAPTER 6

SYSTEM DESIGNS

6.1 DATA FLOW DESIGN

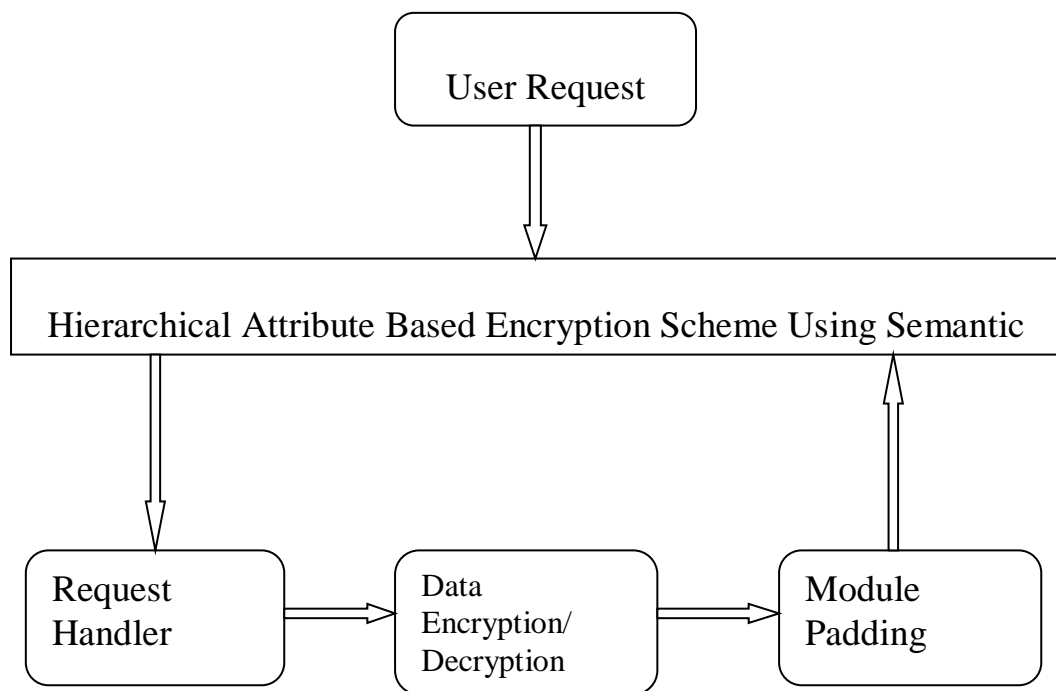


Figure 6.1 Data Flow Design

6.2 SYSTEM FLOW DIAGRAM

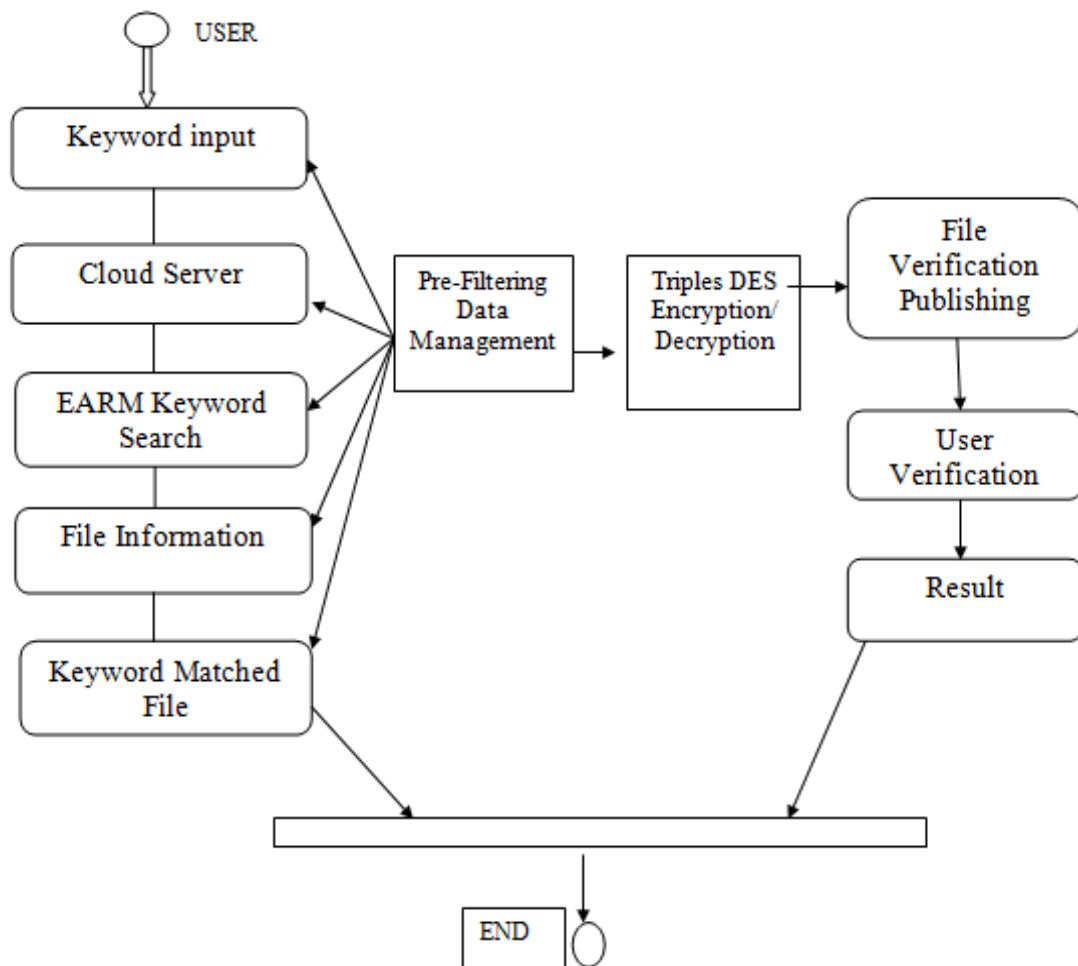


Figure 6.2 System Flow Diagram

CHAPTER 7

SYSTEM TESTING AND IMPLEMENTATION

7.1 INPUT TESTING

Input Testing Of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

7.2 OUTPUT TESTING

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration.

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

7.3 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points. 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 used to test areas that cannot be reached from a black box level.

7.4 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. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.5 SYSTEM TESTING METHODOLOGIES

The following are the Testing Methodologies:

1. Unit Testing.
2. Integration Testing.
3. User Acceptance Testing.
4. Output Testing.
5. Validation Testing.

7.6 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing. During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

7.7 INTEGRATION TESTING

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

Revisited the identity-based distributed provable data possession scheme in multi-cloud storage .The server can still generate a valid proof to prove that the data are stored intact .A generic construction of ID-PDP protocols by using general signature schemes and traditional PDP protocols and proved its security. Built a concrete ID-PDP protocol and an extended version that is suitable for the multi-cloud storage environment we proposed a new public auditing mechanism for shared data with efficient user revocation in the cloud. When a user in the group is revoked, we allow the semi-trusted cloud to re-sign blocks that were signed by the revoked user with proxy re-signatures. Experimental results show that the cloud can improve the efficiency of user revocation, and existing users in the group can save a significant amount of computation and communication resources during user revocation.

8.2 FUTURE ENHANCEMENT

Given the growing threat of both malicious and inadvertent data leaks, As more and more organizations move to the cloud, organizations will move from an identity-as-the-perimeter approach to zero trust frameworks. Future work includes the investigation of agent guilt models that capture the leakage scenarios that are not yet considered .The extension of data allocation strategies so that they can handle agent requests in an online fashion.

CHAPTER 9

APPENDIX -1

9.1 SAMPLE CODE

```
package panda;

public class PandaHome extends javax.swing.JFrame {

    public PandaHome() {

        initComponents();

        @SuppressWarnings("unchecked")

        private void initComponents() {

            jPanel1 = new javax.swing.JPanel(); jLabel1 = new javax.swing.JLabel();

            jLabel2 = new javax.swing.JLabel() jComboBox1 = new javax.swing.JComboBox();

            jButton1 = new javax.swing.JButton();    jButton2 = new javax.swing.JButton();

            setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

            jPanel1.setName(""); // NOI18N

            jLabel1.setFont(new java.awt.Font("Algerian", 0, 24)); // NOI18N

            jLabel1.setText("DATA LEAKAGE PROTECTION");

            jLabel2.setText("select to login");

            jComboBox1.setModel(new javax.swing.DefaultComboBoxModel

            (new String[] { "<--Select-->", "Cloud Login", "User Login",

            "Public Verifier Login" }));

            jButton1.setText("Submit");

            jButton1.addActionListener(new java.awt.event.ActionListener() {

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

                    jButton1ActionPerformed(evt);
```

```

    }

    });

jButton2.setText("Clear");

jButton2.addActionListener(new java.awt.event.ActionListener() {

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

        jButton2ActionPerformed(evt);

    }

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addGap(42, 42, 42)
.addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,365,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING, false)
.addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE, 343,
javax.swing.GroupLayout.PREFERRED_SIZE)
addGroup(jPanel1Layout.createSequentialGroup()
.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 83,

```

```

javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(60, 60, 60)

.addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))

.addGroup(jPanel1Layout.createSequentialGroup())

.addGap(103, 103, 103)

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 190,
    Short.MAX_VALUE)

.addGap(124, 124, 124)))

    .addContainerGap()

);

jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup())

.addGap(32, 32, 32)

.addComponent(jLabel1)

.addGap(26, 26, 26)

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGap(26, 26, 26)

.addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(32, 32, 32)

.addGroup(jPanel1Layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGap(39, 39, 39))

);

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    jComboBox1.setSelectedIndex(0);

}

java.awt.EventQueue.invokeLater(new Runnable() {

    public void run() {

        new PandaHome().setVisible(true);

    }

});

}

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JComboBox jComboBox1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JPanel jPanel1;

// End of variables declaration

}

```

APPENDIX-2

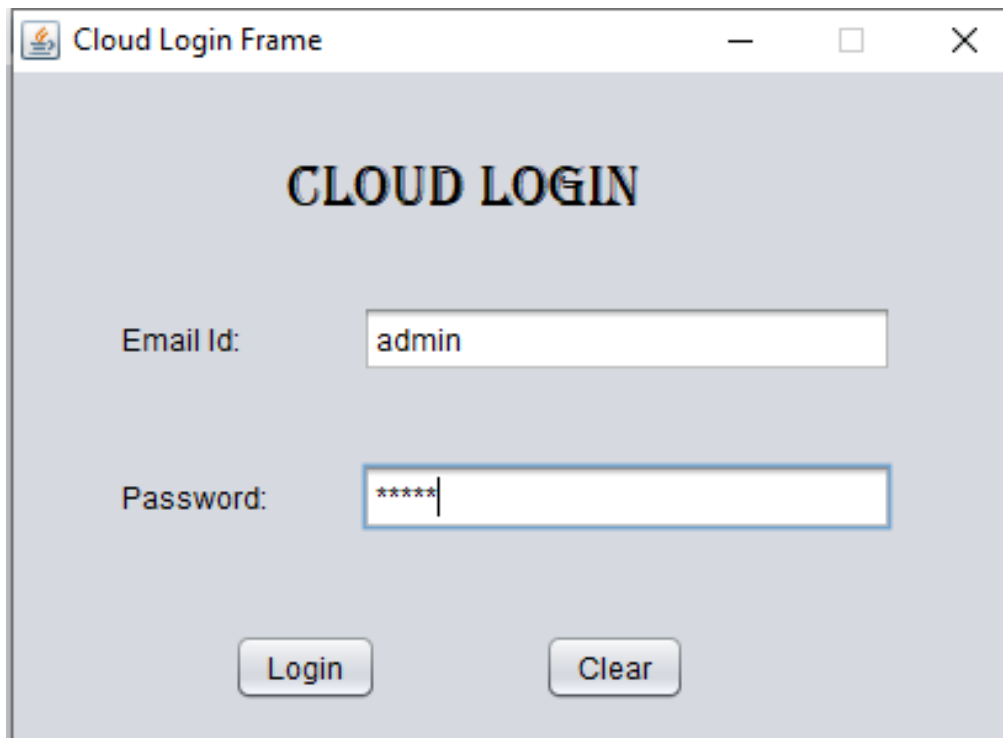
9.2 OUTPUT SCREENSHOT

CLOUD GROUP KEY



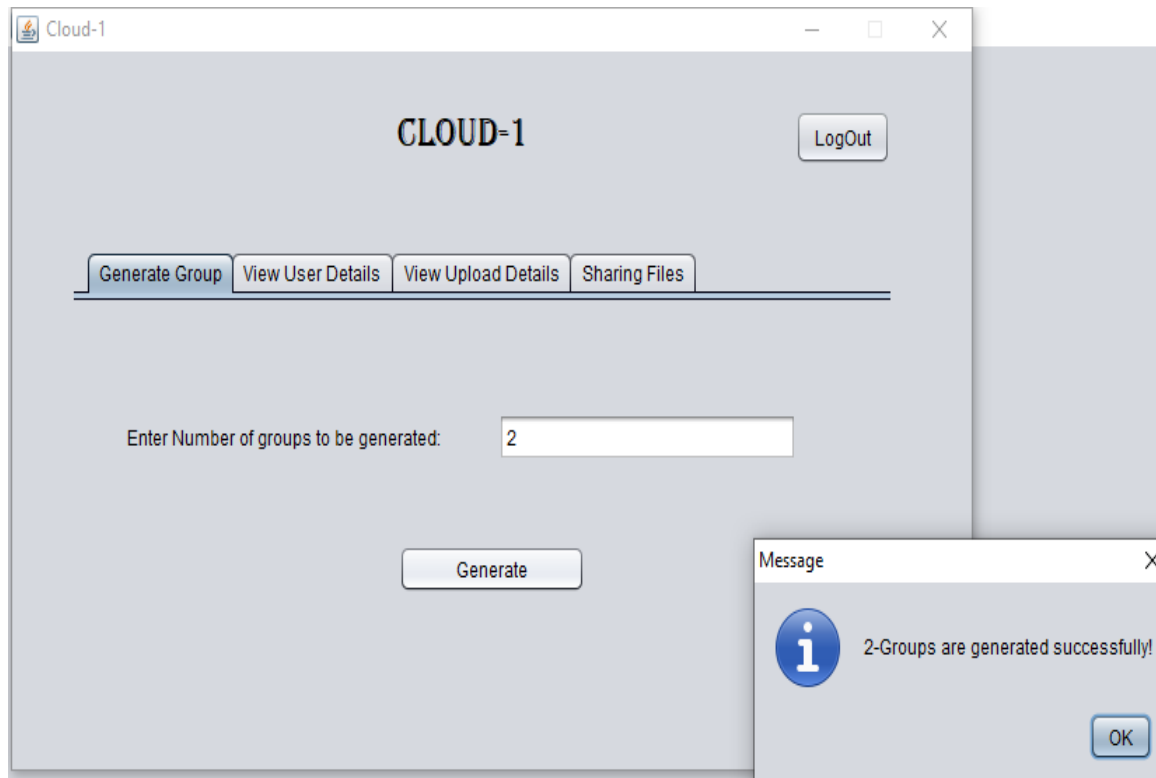
The screenshot shows a web application window titled "DATA LEAKAGE PROTECTION". Below the title, there is a label "select to login". A dropdown menu is displayed with "Cloud Login" selected. Below the dropdown, there are two buttons: "Submit" and "Clear".

CLOUD LOGIN

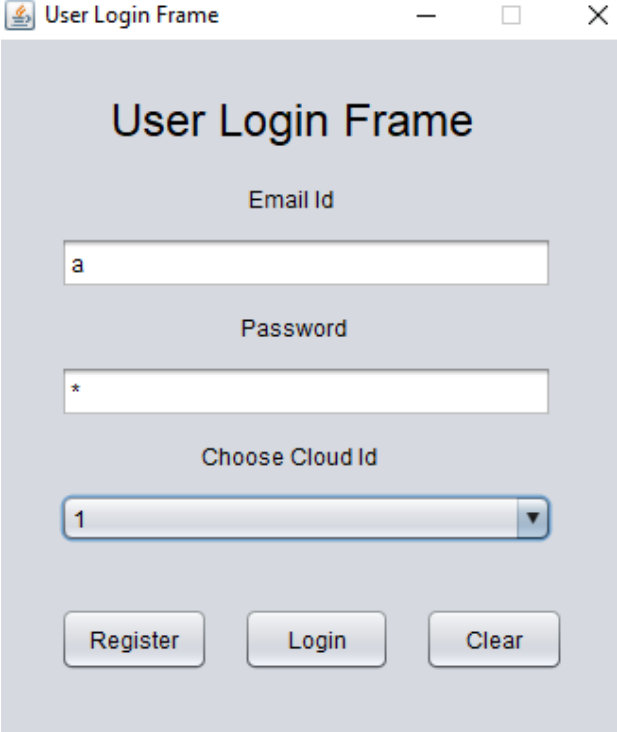


The screenshot shows a web application window titled "Cloud Login Frame". The main heading is "CLOUD LOGIN". Below the heading, there are two input fields: "Email Id:" with the value "admin" and "Password:" with the value "*****". Below the input fields, there are two buttons: "Login" and "Clear".

CLOUD 1

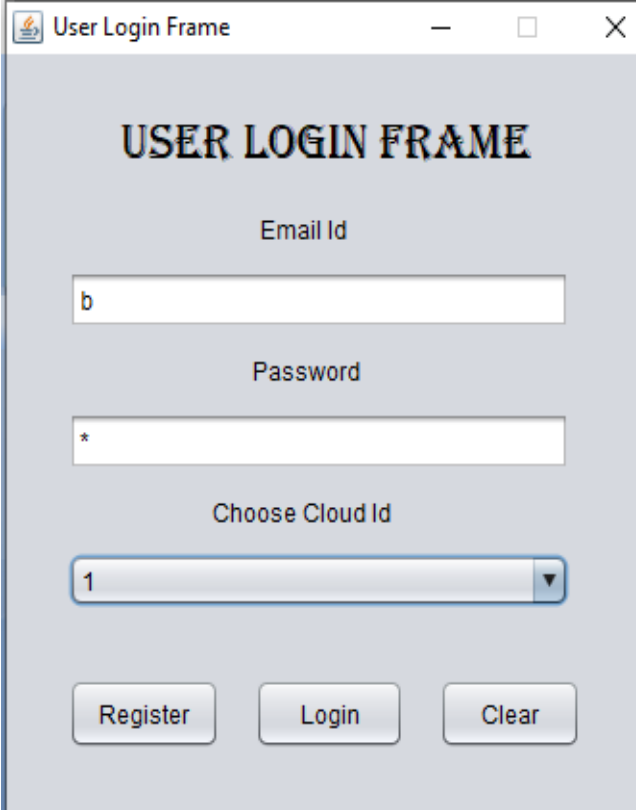


USE LOGIN



A screenshot of a Java Swing window titled "User Login Frame". The window has a light gray background and a standard title bar with a minimize button, a maximize button (disabled), and a close button. The main content area contains the following elements:

- A title "User Login Frame" in a large, bold, black font.
- A label "Email Id" above a text input field containing the letter "a".
- A label "Password" above a text input field containing an asterisk (*).
- A label "Choose Cloud Id" above a dropdown menu showing the value "1".
- Three buttons at the bottom: "Register", "Login", and "Clear", each with a light gray background and a thin border.



A screenshot of a Java Swing window titled "User Login Frame". The window has a light gray background and a standard title bar with a minimize button, a maximize button (disabled), and a close button. The main content area contains the following elements:

- A title "USER LOGIN FRAME" in a large, bold, black font.
- A label "Email Id" above a text input field containing the letter "b".
- A label "Password" above a text input field containing an asterisk (*).
- A label "Choose Cloud Id" above a dropdown menu showing the value "1".
- Three buttons at the bottom: "Register", "Login", and "Clear", each with a light gray background and a thin border.

USER REGISTER FORM



A screenshot of a Java Swing window titled "User Register Frame". The window has a light gray background and a title bar with standard Windows controls. The form contains the following elements: a title "User register frame", an "Email Id" label above a text field containing "a", a "Password" label above a text field containing "*", a "Name" label above a text field containing "a", a "Choose Cloud Id" label above a dropdown menu showing "1", and a "Choose Group" label above a dropdown menu showing "1". At the bottom are two buttons: "Register" and "Clear".

User register frame

Email Id

a

Password

*

Name

a

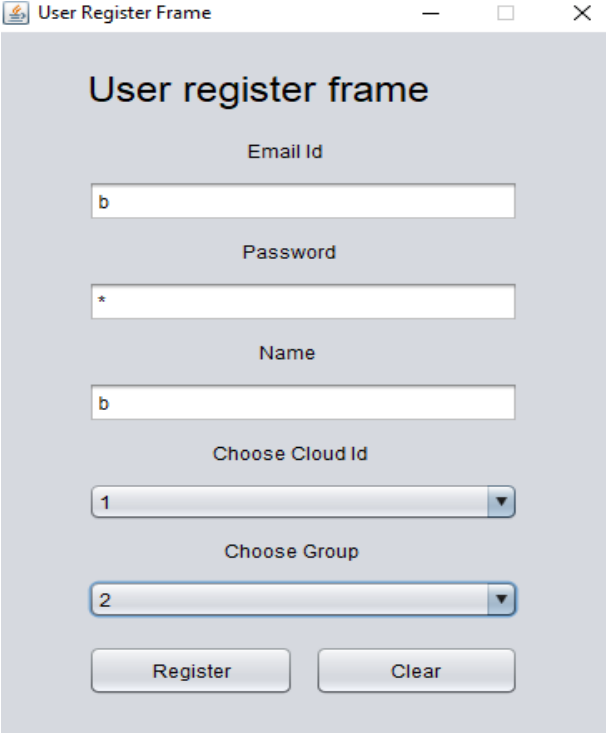
Choose Cloud Id

1

Choose Group

1

Register Clear



A second screenshot of the "User Register Frame" window. The form fields are updated: "Email Id" is "b", "Name" is "b", "Choose Cloud Id" is "1", and "Choose Group" is "2". The "Password" field still contains "*". The "Register" and "Clear" buttons remain at the bottom.

User register frame

Email Id

b

Password

*

Name

b

Choose Cloud Id

1

Choose Group

2

Register Clear

USER FRAME

User Frame

— □ ×

USER FRAME

LogOut

Welcome abc

Group No: 1

User Type: Original User

View Group Members

Upload

Download

View Revocked Users

Data Sharing

Receive

Cloud Id: 1

EmailId	Password	Name	Group No	UserType
a@gmail.com	a	abc	1	Original User

User Frame

— □ ×

USER FRAME

LogOut

Welcome a

Group No: 1

User Type: Original User

View Group Members

Upload

Download

View Revocked Users

Data Sharing

Receive

DATA SHARING FOR ANOTHER GROUP MEMBER

Choose Cloud Id: 1

Choose Group Id: 2

Choose Member Name: b

Enter the File Name: sen1.txt

Browse

Key Generation

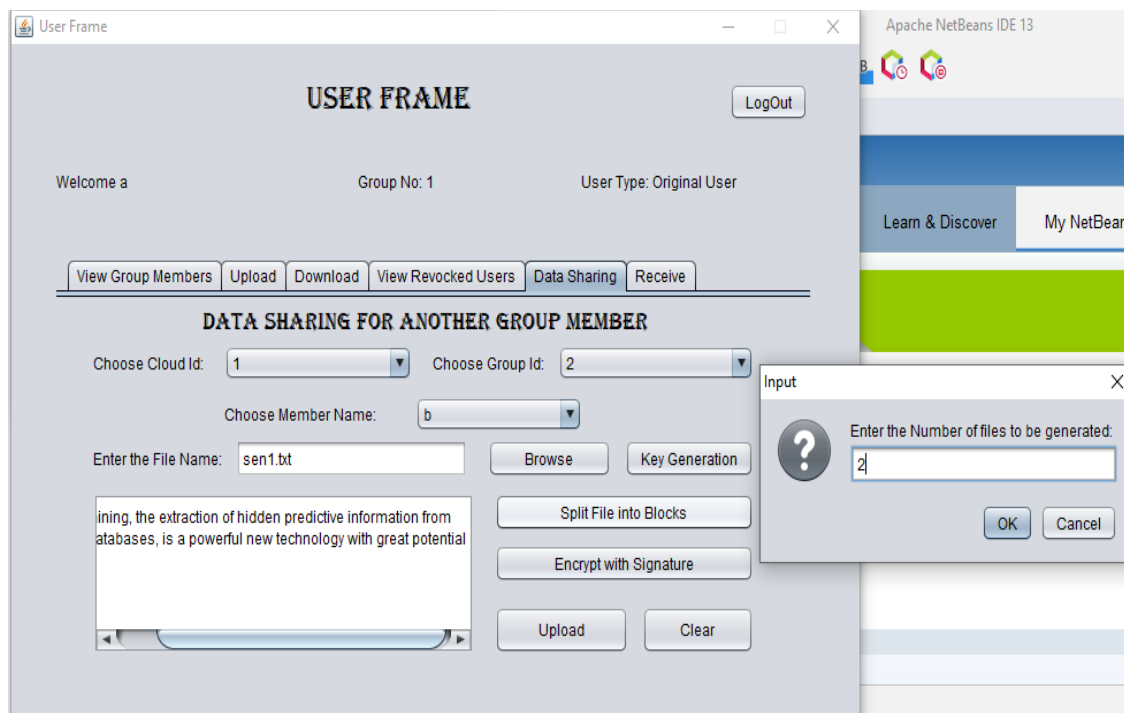
ining, the extraction of hidden predictive information from
atabases, is a powerful new technology with great potential

Split File into Blocks

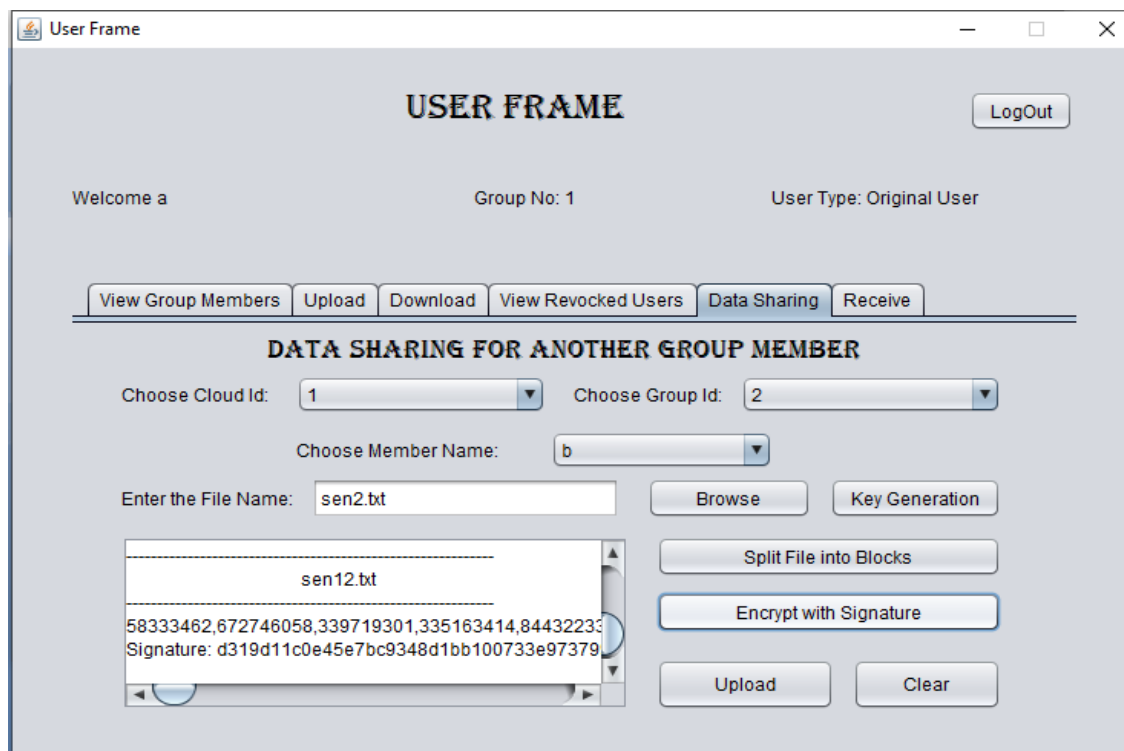
Encrypt with Signature

Upload

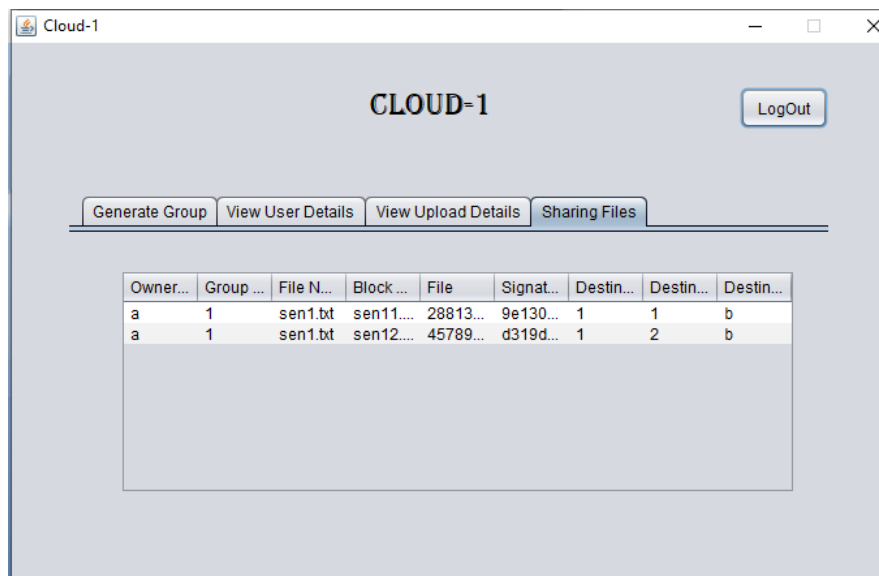
Clear



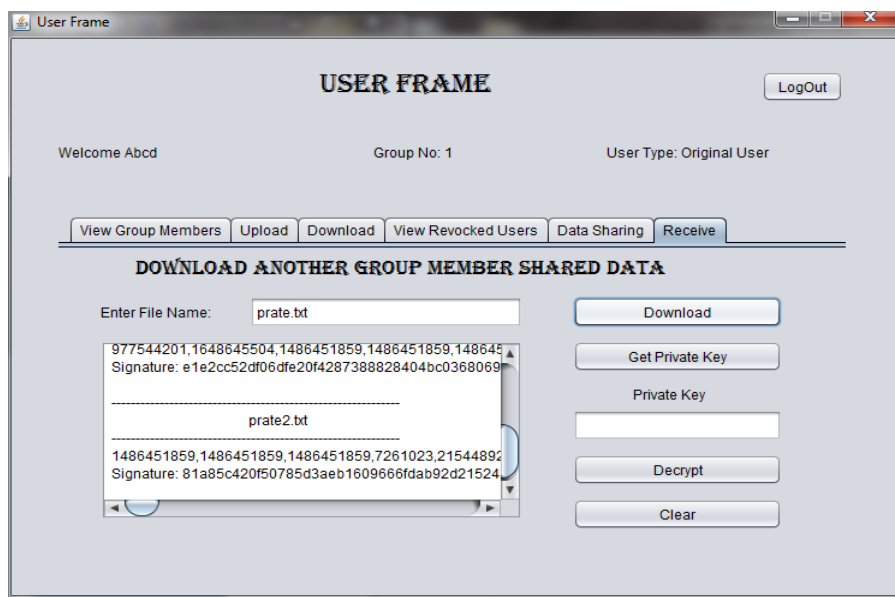
ENCRYPTED FORM



CLOUD 1 SHARED A FILE



USER FRAME RECEIVER SIDE



User Frame

USER FRAME

LogOut

Welcome Abcd

Group No: 1

User Type: Original User

View Group Members

Upload

Download

View Revoked Users

Data Sharing

Receive

DOWNLOAD ANOTHER GROUP MEMBER SHARED DATA

Enter File Name: prate.txt

Download

prate1.txt

1114386616,603956904,1377461755,171166141,1071755

Signature: 82831ce243586fd1836bb07de7cfcbd1c2aa5de1

prate3.txt

Get Private Key

Private Key

1280854237,2413855097

Decrypt

Clear

RECEIVER SIDE DECRYPTON FORM

User Frame

USER FRAME

LogOut

Welcome Efgh

Group No: 1

User Type: Group User

View Group Members

Upload

Download

View Revoked Users

Data Sharing

Receive

DOWNLOAD ANOTHER GROUP MEMBER SHARED DATA

Enter File Name: prate.txt

Download

INTRODUCTION

MOBILE ADHOC NETWORKS ?

Mobile ad hoc networks (i.e., decentralized networks create

In August of 2015, researchers at the National Institute of Unmanned vehicles (aerial, terrestrial, and aquatic) with a

Get Private Key

Private Key

2048409499,2654207357

Decrypt

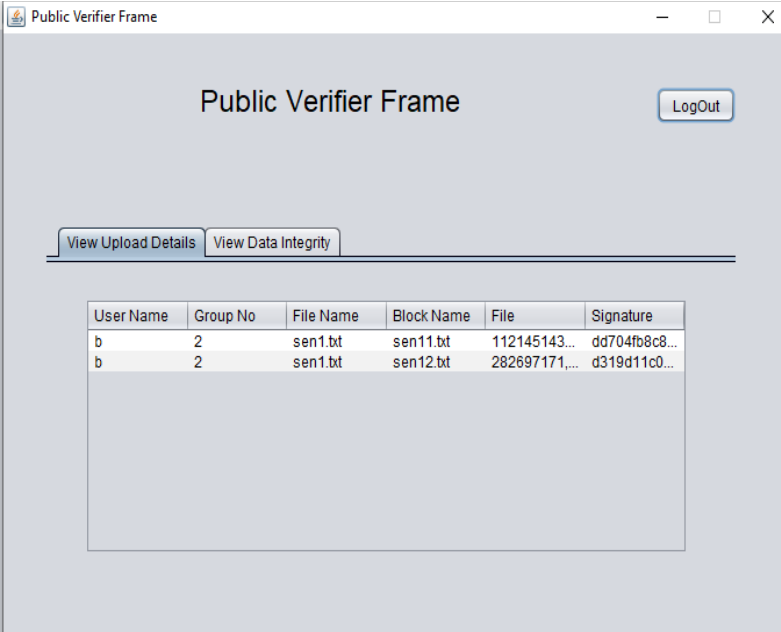
Clear

PUBLIC VERIFIER LOGIN



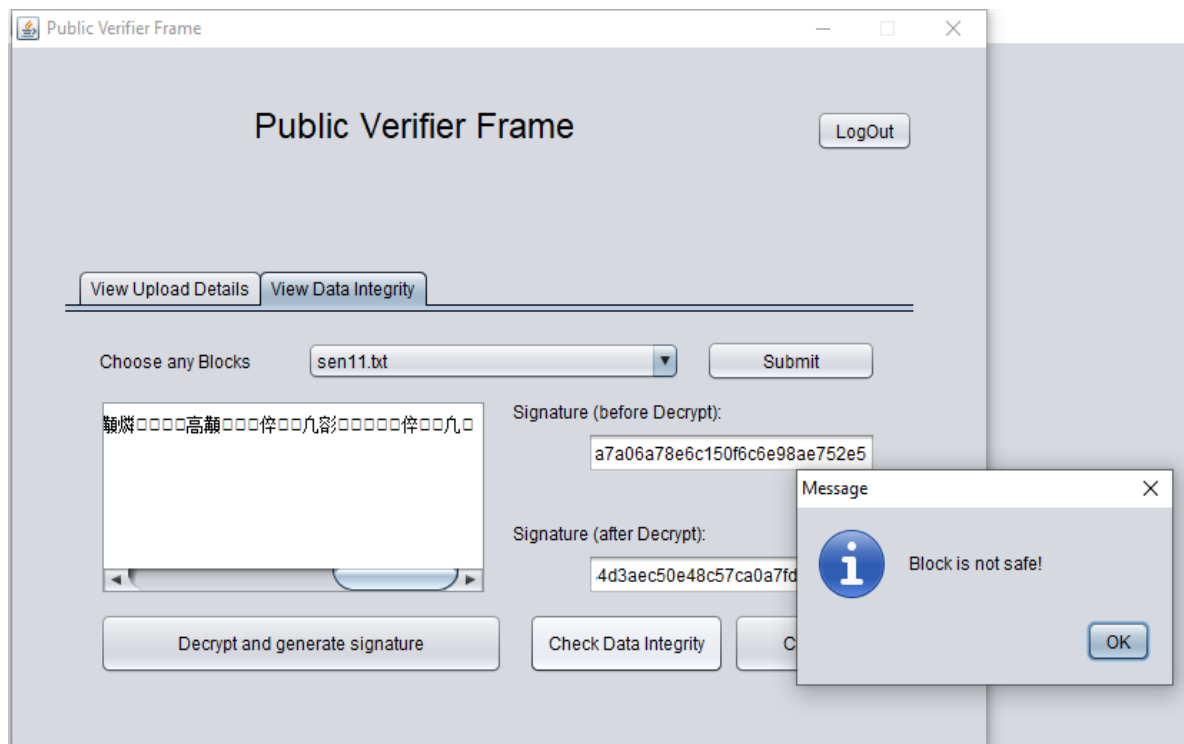
A screenshot of a web application window titled "Public Verifier Login Frame". The window has a light gray background. At the top, the title "Public Verifier Login" is displayed in a large, bold, black font. Below the title, the text "Email Id" is centered. Underneath, there is a text input field containing the text "pv". Below the email field, the text "Password" is centered. Underneath, there is a password input field with two asterisks "**" and a cursor. At the bottom of the form, there are two buttons: "Login" and "Clear", both with a light blue gradient and rounded corners.

PUBLIC VERIFIER FRAME



A screenshot of a web application window titled "Public Verifier Frame". The window has a light gray background. At the top, the text "Public Verifier Frame" is displayed in a large, bold, black font. To the right of this text is a "LogOut" button with a light blue gradient and rounded corners. Below the title, there are two tabs: "View Upload Details" and "View Data Integrity". The "View Upload Details" tab is selected. Below the tabs, there is a table with the following data:

User Name	Group No	File Name	Block Name	File	Signature
b	2	sen1.txt	sen11.txt	112145143...	dd704fb8c8...
b	2	sen1.txt	sen12.txt	282697171,....	d319d11c0...



REFERENCES

1. N. Cao, S. Yu, Z. Yang, W. Lou, and Y. T. Hou, "LT Codes-based Secure and Reliable Cloud Storage Service," in the Proceedings of IEEE INFOCOM 2012, 2012, pp. 693–701.
2. Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and S. Chen, "Dynamic Audit Services for Outsourced Storage in Clouds," IEEE Transactions on Services Computing, accepted.
3. C. Wang, Q. Wang, K. Ren, and W. Lou, "Towards Secure and Dependable Storage Services in Cloud Computing," IEEE Transactions on Services Computing, vol. 5, no. 2, pp. 220–232, 2011.
4. C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," in the Proceedings of IEEE INFOCOM 2010, 2010, pp. 525–533.
5. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamic for Storage Security in Cloud Computing," in the Proceedings of ESORICS 2009. Springer-Verlag, 2009, pp. 355–370.
6. C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," in the Proceedings of ACM/IEEE IWQoS 2009, 2009, pp. 1–9.
7. H. Shacham and B. Waters, "Compact Proofs of Retrievability," in the Proceedings of ASIACRYPT 2008. Springer-Verlag, 2008, pp. 90–107.

8. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable Data Possession at Untrusted Stores,” in the Proceedings of ACM CCS 2007, 2007, pp. 598–610.
9. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A View of Cloud Computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, April 2010.
10. B. Wang, B. Li, and H. Li, “Public Auditing for Shared Data with Efficient User Revocation in the Cloud,” in the Proceedings of IEEE INFOCOM 2013, 2013, pp. 2904–2912.