**CHAPTER 1**

**INTRODUCTION**

* 1. **ABOUT PROJECT**

Homomorphic encryption is a form of encryption that allows computations to be carried out on cipher text, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext.

This is sometimes a desirable feature in modern communication system architectures. Homomorphic encryption would allow the chaining together of different services without exposing the data to each of those services. For example, a chain of different services from different companies could calculate 1) the tax 2) the currency exchange rate 3) shipping, on a transaction without exposing the unencrypted data to each of those services. Homomorphic encryption schemes are malleable by design. This enables their use in cloud computing environment for ensuring the confidentiality of processed data. In addition the homomorphic property of various cryptosystems can be used to create many other secure systems, for example secure voting systems, collision-resistant hash functions, private information retrieval schemes, and many more.

Cloud computing is very much in demand these days. Nowadays computation and storing demands are surging, and instead of deploying single unit for same, alternate solution that can provide low cost storage and processing capabilities. Every organizing is migrating to cloud, in order to use services such as PaaS, IaaS and SaaS. As cloud provides scalability, reliability, resource management, interoperability, portability, virtualization as main theme, but more focus on security, confidentiality and privacy of data being stored on cloud. As the demand for cloud increases, security concern of loss of data, data leakage and breach, distributed denial of service. For example, client is not aware about the practices of cloud provider, how they process and store data. Therefore, client want guarantee about the data processing and data storage [alteration and deletion]. All the above issues are resolved if the data is processed and stored in the encryption form. Data is encrypted by users, using cryptographic algorithms, which is stored in the cloud. The encrypted message using cryptographic techniques is called as cipher text which can be stored in any database. If computations are needed to be performed, then it is necessary to decrypt those data, but the decrypted data are not secure any more, thus, a new cryptosystem was required that permits computation on the encrypted data. This technique is called privacy homomorphism. However, decryption is not performed; the result obtained is same as computations performed on plaintext. While operating encrypted data, additions and multiplications on plaintext can be performed by using homomorphic encryption.

**1.2 AN OVERVIEW ON CRYPTOGRAPHY**

The examples listed above allow homomorphic computation of some operations on cipher texts (e.g., additions, multiplications, quadratic functions, etc.). A cryptosystem that supports arbitrary computation on cipher texts is known as fully homomorphic encryption (FHE) and is far more powerful. Such a scheme enables the construction of programs for any desirable functionality, which can be run on encrypted inputs to produce an encryption of the result. Since such a program need never decrypt its inputs, it can be run by an untrusted party without revealing its inputs and internal state. The existence of an efficient and fully homomorphic cryptosystem would have great practical implications in the outsourcing of private computations, for instance, in the context of cloud computing.

**1.3 TYPES OF** **CRYPTOGRAPHY**

**1.3.1 SYMMETRIC KEY CRYPTOGRAPHY**

In Symmetric Cryptography, for encryption and decryption, the same secret key is used.

**1.3.2 ASYMMETRIC KEY CRYPTOGRAPHY**

In Asymmetric Cryptography, public key is used for encrypting the message. The public key will be available to anyone to send the message, private key is kept confidential, only the authorized person who have the secret decryption key can view the message. A problem with asymmetric encryption is, it is slower than symmetric encryption. It requires more processing power for encryption and decryption process.

**1.3.3 HASH FUNCTION CRYPTOGRAPHY**

In hash function cryptography, it generates a fixed-size blocks of data. The output of the hash function is called Message Digest (MD). Any simple change in hash function after it was generated, it will affect second value of the hash function. Any slight into the original text will reflect some vast differences in hash values. It solves problem of integrity of the messages. The most widely used hash function techniques are: MD5, SHA1.

The mentioned above issue is resolved if the asymmetric key cryptosystem, like in, support addition and multiplication on encrypted data. Many encryption scheme support one of these operation, like RSA is multiplicative, paillier is additive etc. Homomorphic cryptosystems have received considerable attention in the literature. Despite the fact that homomorphic encryption increases the amount of computational complexity, the benefits associated with it are worth. This model preserves the confidentiality and processing from an untrusted third party.

**CHAPTER 2**

**PROJECT**

**2.1. Requirement Analysis**

**2.1.1 Purpose**

Homomorphic encryption is a form of [encryption](https://en.wikipedia.org/wiki/Encryption) that allows computations to be carried out on [cipher text](https://en.wikipedia.org/wiki/Ciphertext), thus generating an encrypted result which, when decrypted, matches the result of operations performed on the [plaintext](https://en.wikipedia.org/wiki/Plaintext).

This is sometimes a desirable feature in modern communication system architectures. Homomorphic encryption would allow the chaining together of different services without exposing the data to each of those services. For example, a chain of different services from different companies could calculate 1) the tax 2) the currency exchange rate 3) shipping, on a transaction without exposing the unencrypted data to each of those services. [Homomorphic](https://en.wikipedia.org/wiki/Homomorphic) encryption schemes are [malleable](https://en.wikipedia.org/wiki/Malleability_(cryptography)) by design. This enables their use in cloud computing environment for ensuring the confidentiality of processed data. In addition, the Homomorphic property of various cryptosystems can be used to create many other secure systems, for example secure voting systems,[[2]](https://en.wikipedia.org/wiki/Homomorphic_encryption#cite_note-2) collision-resistant [hash functions](https://en.wikipedia.org/wiki/Hash_function), [private information retrieval](https://en.wikipedia.org/wiki/Private_information_retrieval) schemes, and many more.

**2.1.2 Use Case**

**PAILLIER CRYPTOSYSTEM**

**2.1.2.1 Key Generation:**

1. Choose two large prime numbers p and q randomly such that gcd(pq, (p −1)(q − 1))=1

2. Compute n = pq and Λ = lcm(p − 1, q − 1)

3. Select random integer g where g ∈ Z\*n2

4. μ = (L(gΛ mod n2))−1 mod n where L(u) = (u-1)/n

5. Public Key = (n, g)

6. Private Key = (Λ, μ)

**2.1.2.2 Encryption:**

1. Let m be a message to be encrypted and m ∈ Zn.

2. Select random r where r ∈ Zn.

3. Compute the cipher text c as c = gm.rn mod n2.

**2.2.1.3 Decryption:**

1. Compute the message as m = L(cΛ mod n2).μ mod n.

**2.2.1.4 Homomorphic Properties:**

A notable feature of the Paillier cryptosystem is its homomorphic properties along with its non-deterministic encryption (see Electronic voting in Applications for usage).

Paillier cryptosystem has the property of additive homomorphism. If m1 and m2 are the message to be encrypted, E() and D() are the encryption and decryption function respectively and n is from the public Key, then the additive property can be expressed as follows.

Homomorphic addition of plaintexts

The product of two cipher texts will decrypt to the sum of their corresponding plaintexts.

D(E(m1, public Key) ∗ E(m2, public Key) mod n2) )= m1 + m2 mod n

The product of a cipher text with a plaintext raising g will decrypt to the sum of the corresponding plaintexts.

D(E(m1, public Key) ∗ gm2 mod n2) )= m1 + m2 mod n 15

**2.2 SOFTWARE AND HARDWARE SPECIFICATION USED FOR DEVELOPMENT**

This is the software specification of the system used for the development of this project and is recommended for a smooth development process of such projects.

**SOFTWARE**

Language used: JAVA

Packages used:

1. Biginteger
2. Mysql
3. Jdbc
4. Swing
5. Awt
6. Util
7. Text
8. Mail
9. Activation

HARDWARE REQUIREMENT

1. Processor: Intel i3
2. Hard Disk 1GB
3. Ram 1GB

**2.3 Database Description**

**2.3.1 DATABASE SCHEMA AND DESCRIPTION OF TABLES**

The database used in our application has following tables:

1. Database used: MySQL
2. Database created- pal
3. Table created- paillier

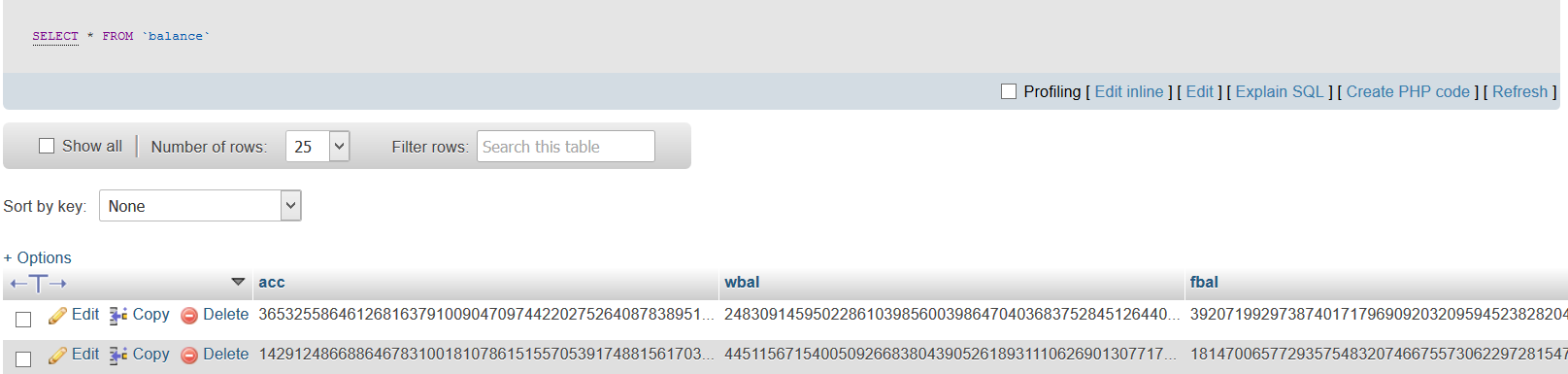


Fig 2.3.1 DATABASE DESCRIPTION OF BALANCE

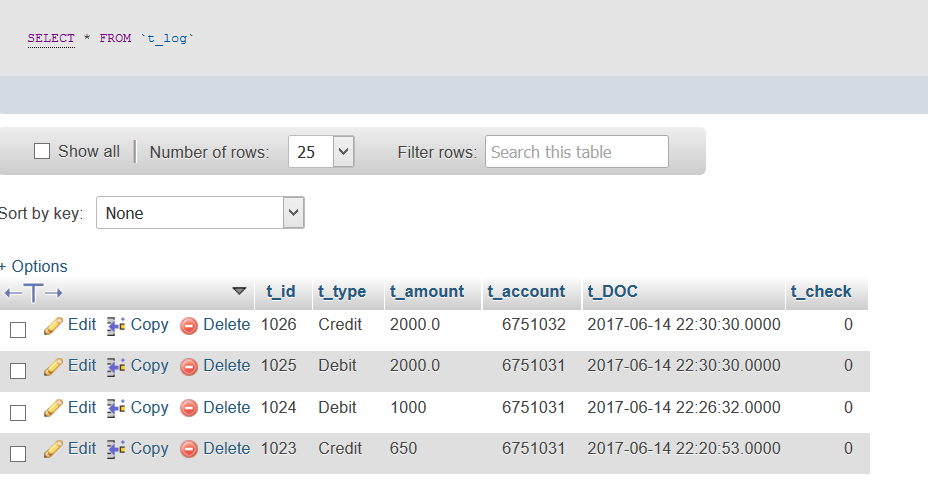


FIG 2.3.2 DATABASE ALL TRANSACTIONS

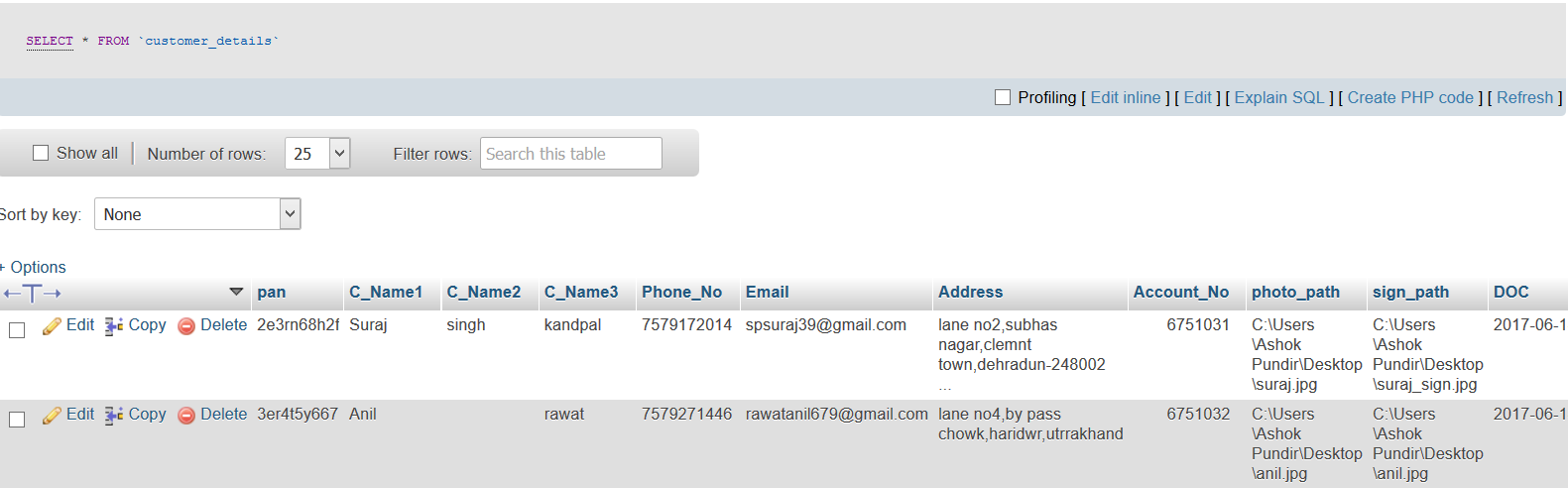


FIG 2.3.3 DATABASE CUSTOMER DETAILS



FIG 2.3.4 DATABASE ENCRYPTED KEY

**CHAPTER 3**

**SOFTWARE / PROJECT DESIGN**

**3.1 ER Diagram**

An Entity Relation(ER) Diagram is a specialized graphics that illustrates the interrelationship between entities in a database. ER diagrams often use symbols to represent 3 different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes.

An Entity Relationship Model (ERM), in software engineering is an abstract and conceptual representation of data. Entity Relationship modelling is a relational schema database modelling method, used to produce a type of conceptual schema or semantic data model of a system, often a relation database, and its requirements in a top-down fashion.

**Entity:**

The thing which we want to store information. It is an elementary basic building block of storing information about business process. An entity represents an object defined within

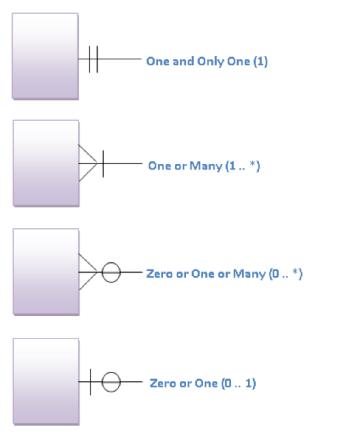
the information system about which you want to store information. Entities are distinct things in the enterprise.

**Relationships:**

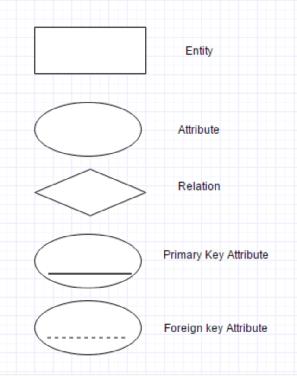
A relationship is a named collection or association between entities or used to relate two or more entities with some common attributes or meaningful interaction between the objects.

**Attributes:**

Attributes are the properties of the entities and relationship, Descriptor of the entity. Attributes are elementary pieces of information attached to an entity.

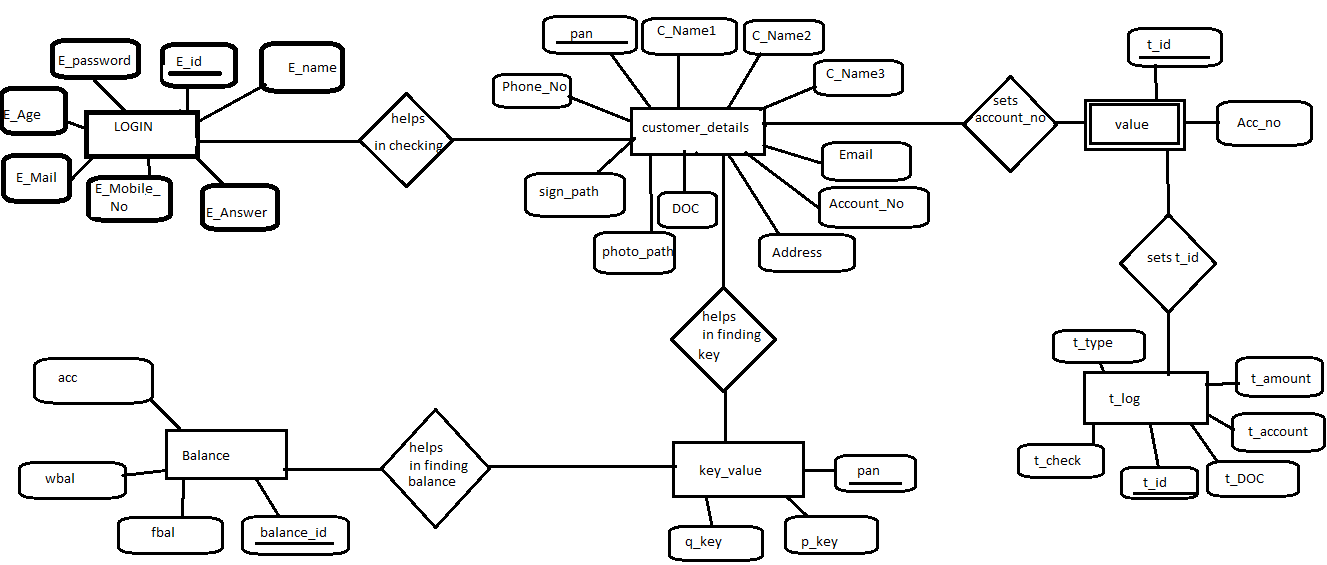


**Figure 1. Cardinality notation used in ER diagram**



**Figure 2. ER diagram symbol**

**ER DIAGRAM**



**Fig: 3**

**CHAPTER 4**

**BLACK BOX TESTING**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case  Description | Test Case  ID | Test Description | Expected Result | Actual Result | Pass/Fail |
| Login | HE001 | Registered Admin | Access to Admin’s Account | Updated Account Details displayed to the Admin. | PASS |
| Unregistered Admin | Deny Access | Access is denied using an appropriate message. | FAIL |
| Signup | HE002 | Open a new account for Admin | Creation of Admin’s account and entry into the database | Admin’s account created and database updated with a welcome message. | PASS |
| Check if Password and Confirm Password are same | Deny Signup | Signup is denied using popup mesage | FAIL |

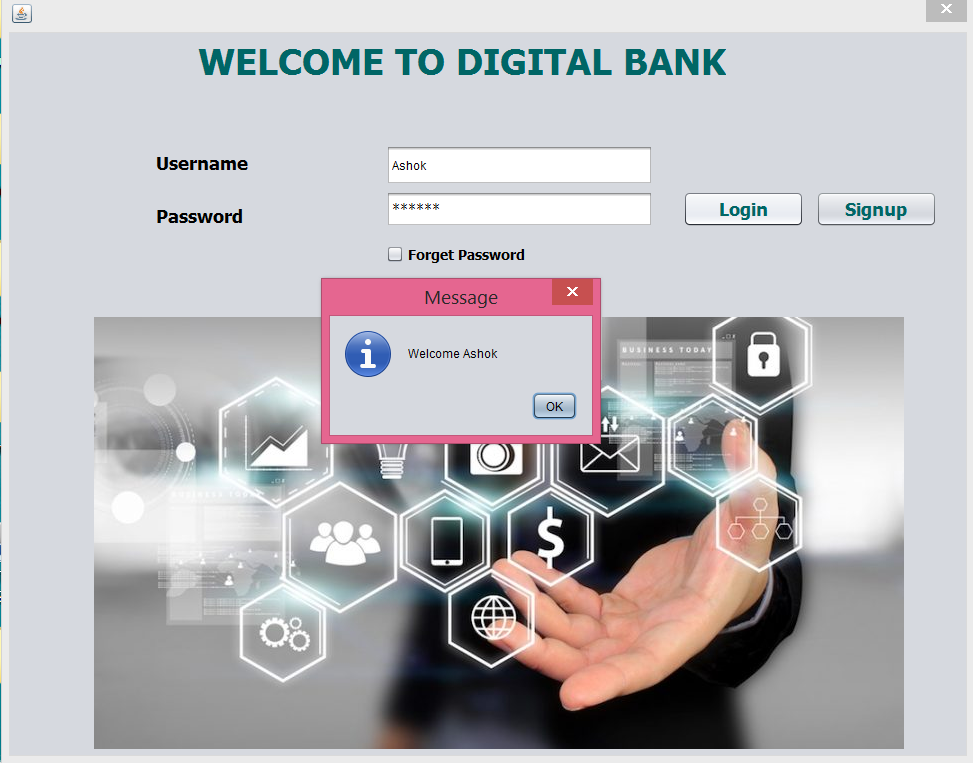


Fig 4.1 After valid credentials a welcome message is displayed.

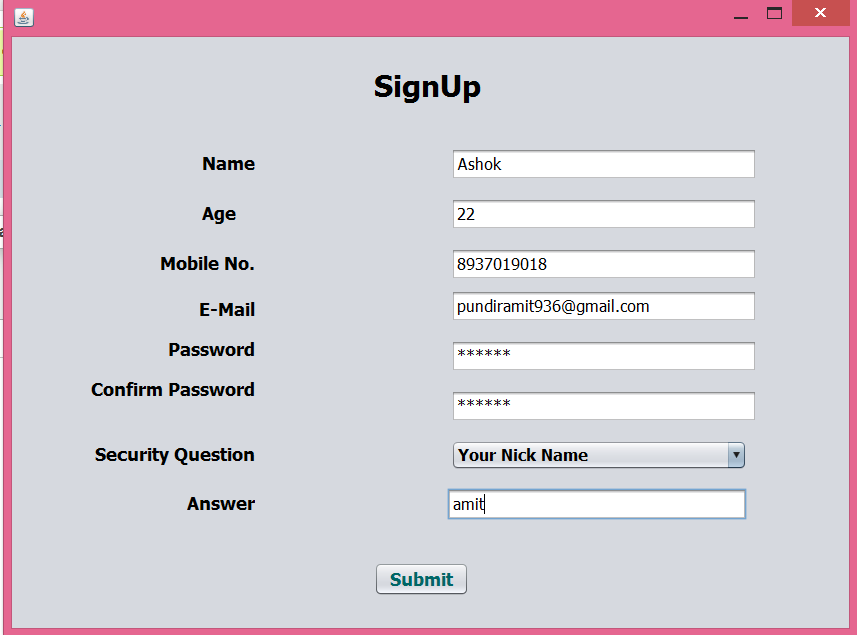
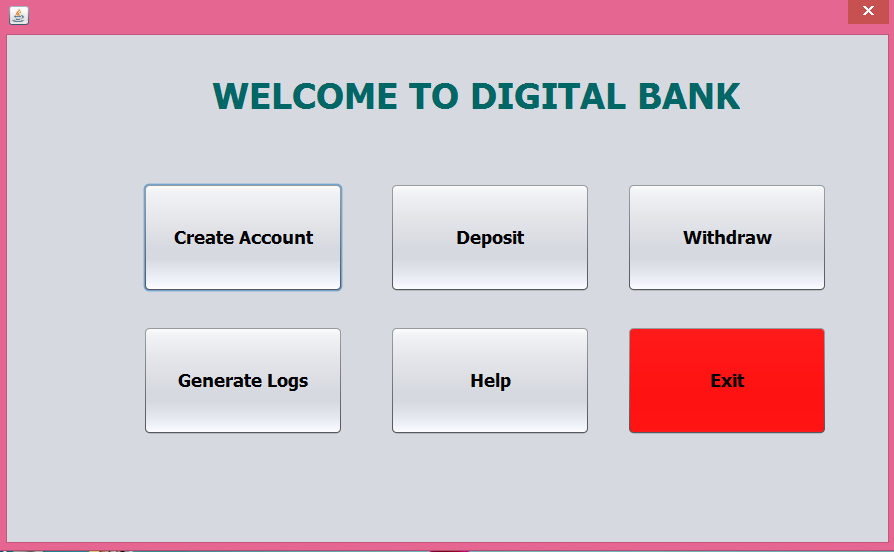


Fig 4.2 Illustration of Admin’s Signup details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test case description | Test Case ID | Test Description | Excepted Result | Actual result | PASS/FAIL |
| Create Account | HE003 | Admin creates a new account for customer | Account is created and an email will be sent to the registered email address. | Account creation successful, database updated, email sent. | PASS |
| Deposit | HE004 | Deposits money into customer’s account. | Amount deposited. | Balance updated and message displayed. | PASS |
| Money Withdrawl | HE005 | Withdrawl from customer’s balance. | Balance and database updated. | Success withdrawl and balance updated. | PASS |
| Insufficient balance | Withdrawl denied | Insufficient balance message displayed. | FAIL |
| Money Transfer | HE006 | Transfer money to payee’s account | Transfer Successful. | Sender’s and Payee’s account updated. | PASS |
| Not enough balance in payee’s account. | Transaction Failed | Money not transferred. | FAIL |



1Fig 4.3- Welcome Screen of Admin

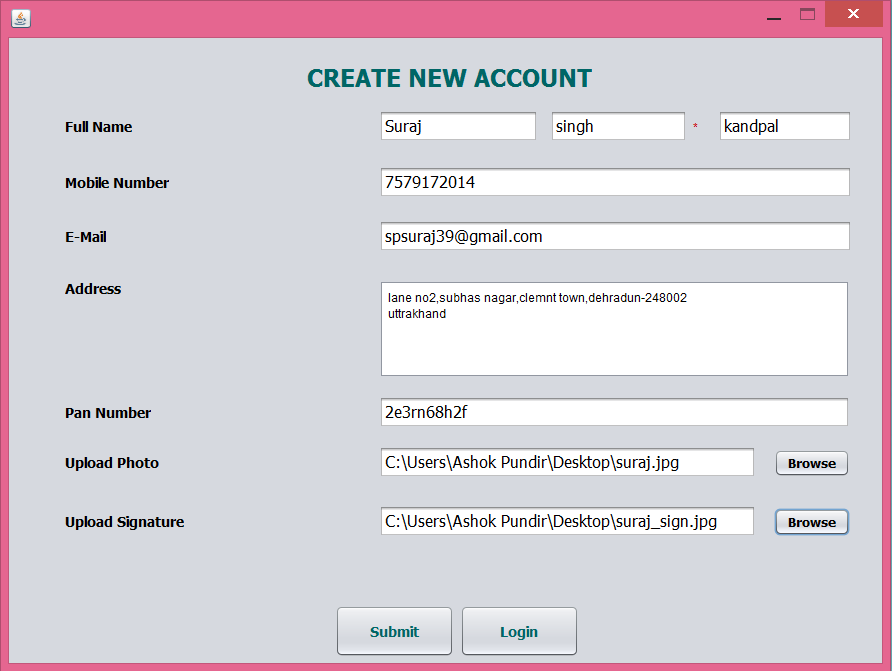


Fig 4.4- Account creation for new customer

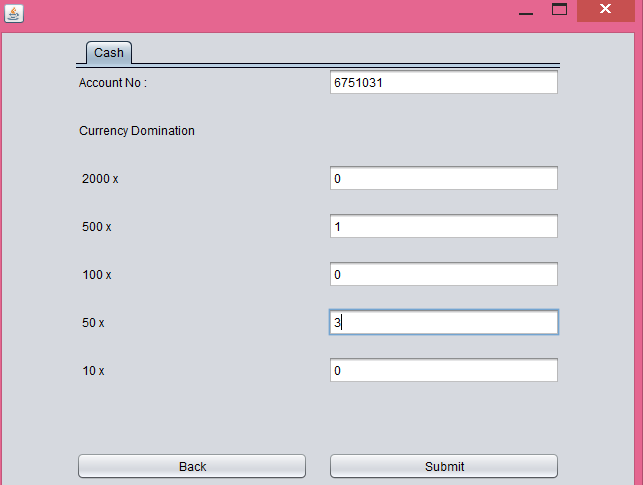


Fig 4.5-Illustration of Money deposition

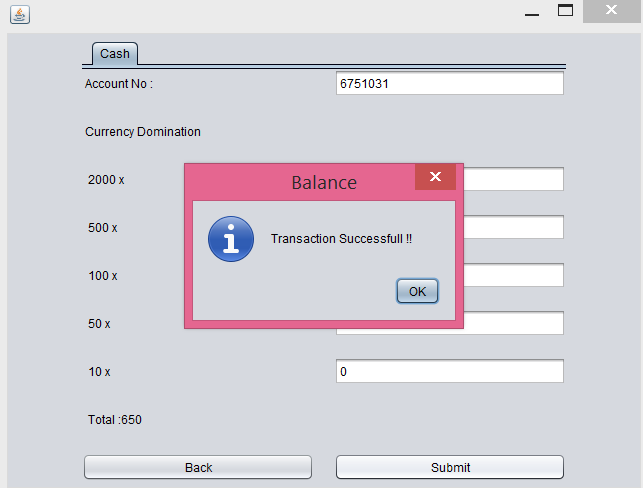


Fig 4.6- Transaction Successful message

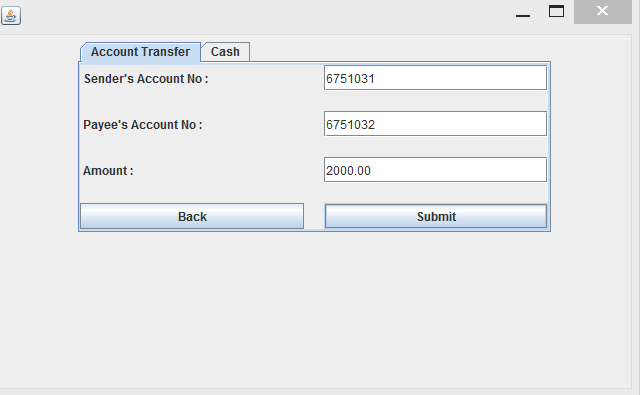


Fig 4.7- Money Transfer

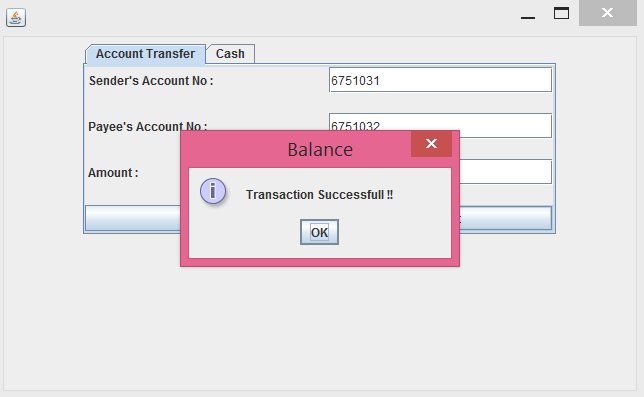


Fig 4.8- Message displayed on successful Transaction

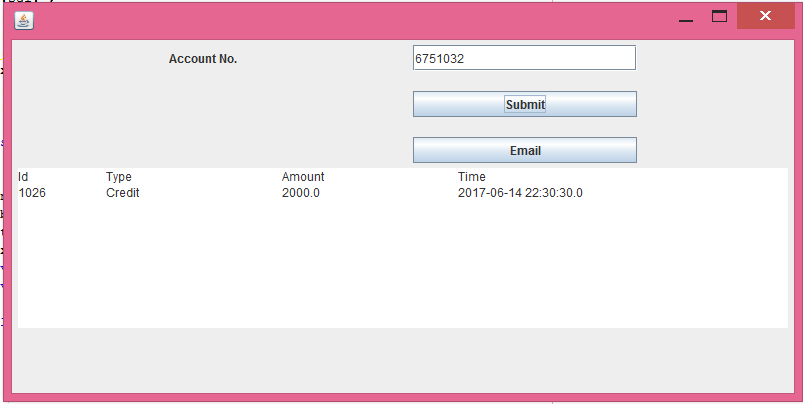


Fig. 4.9- Show log regarding to any account.

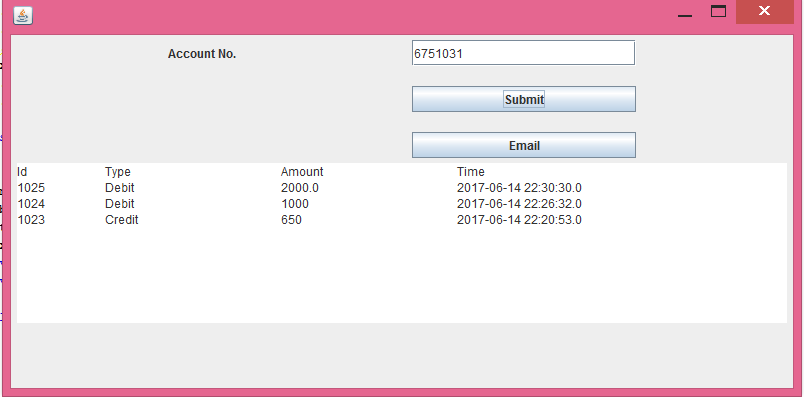


Fig 4.10- Transaction details of customer

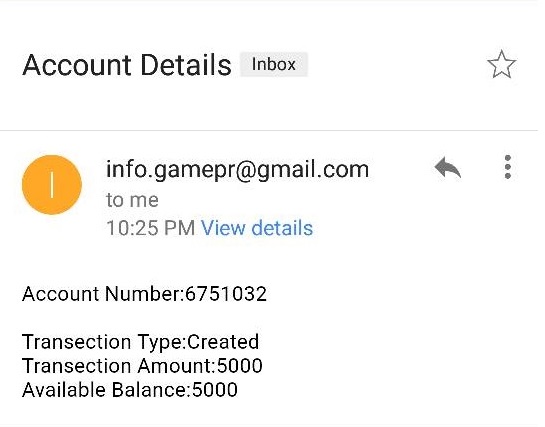


Fig 4.11- Account created email

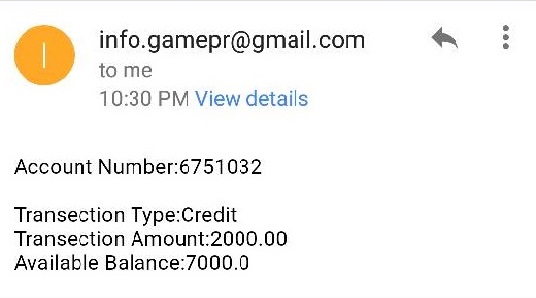


Fig 4.12- Amount credited

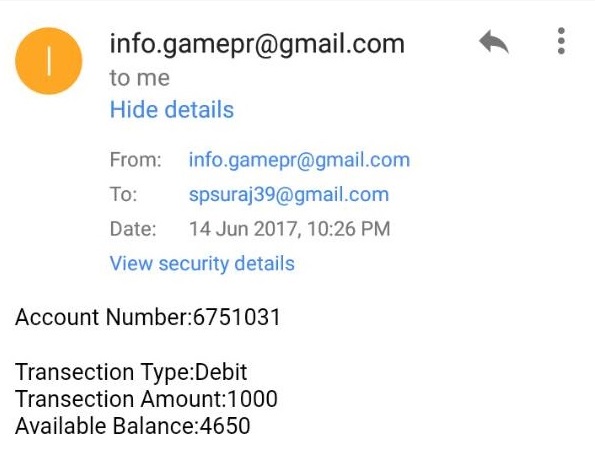


Fig 4.13- Amount debit Success Email

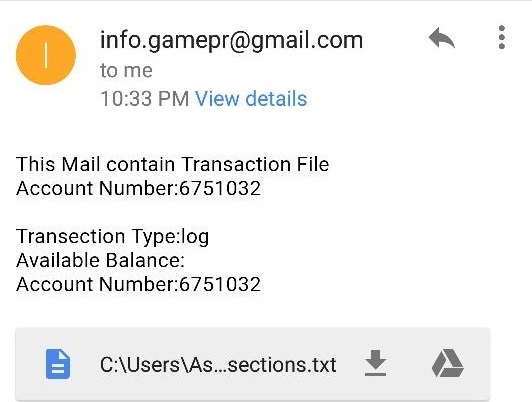


Fig 4.14- Log Sent via email

**VALIDATIONS OF BLACKBOX TESTING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case Id | Test case name | Test case description | Steps | Expected Result |
| TE001 | Login scenario 01 | Validate the system when the username starts with an alphabet. | 1.Open the application.  2.Enter the username with first character as an alphabet.  3.Enter the second character as an integer or a underscore as per your choice | Username accepted |
| TE002 | Login scenario 02 | Validate the system when the username starts with an integer. | 1.Open the application.  2.Enter the username with first character as an integer. | Username not accepted |
| TE003 | Login scenario 03 | Validate the system when the username starts with an underscore. | 1.Open the application.  2.Enter the username with first character as an underscore. | Username not accepted |
| TE004 | Login scenario 04 | Validate the system and enter the password. | 1.Open the application.  2.Enter the password with more than 8 characters. | Password accepted |
| TE005 | Signup scenario 01 | Validate the system and enter the age. | 1.Open the application.  2.Click on signup button.  3.If the age of the customer < 18 | Age will not be accepted |
| TE006 | Signup scenario 02 | Validate the system and enter the mobile number. | 1.Open the application.  2.Click on signup button.  3.If the length of the mobile number exceeds 10 digits. | Mobile number is a not a valid number |
| TE007 | Signup scenario 03 | Validate the system and enter the email address. | 1.Open the application.  2.Click on signup button.  3.Enter the email with first character as an alphabet followed by alphabets,integers and special characters ( dot and underscore only) | Email Address accepted |
| TE008 | Signup scenario 03 | Validate the system and enter the password. | 1.Open the application.  2.Click on signup button.  3.Enter the password with more than 8 characters | Password Successful |
| TE009 | Signup scenario 04 | Validate the system and enter the security question. | 1.Open the application.  2.Click on signup button.  3. Enter the answer of your choice to the selected question | Answer updated |
| TE010 | Create Customer Account Scenario 01 | Validate the system and enter the full name. | 1.Open the application.  2.Click on customer account.  3. Enter the name | Accepted |
| TE011 | Create Customer Account Scenario 02 | Validate the system and enter the mobile number. | 1.Open the application.  2.Click oncustomer account.  3.If the length of the mobile number exceeds 10 digits. | Mobile number is a not a valid number |
| TE012 | Create Customer Account Scenario 03 | Validate the system and enter the email address. | 1.Open the application.  2.Click on customer account.  3.Enter the email with first character as an alphabet followed by alphabets,integers and special characters ( dot and underscore only) | Email Address accepted |
| TE013 | Create Customer Account Scenario 04 | Validate the system and enter the Pan number. | 1.Open the application.  2.Click on customer account.  3.Enter Pan number containing only alphabets and integers | Accepted |
| TE014 | Create Customer Account Scenario 05 | Validate the system and upload customer’s photo and signature. | 1.Open the application.  2.Click on customer account.  3.Click on browse tab and select respective photo and signature.  4.Upload photo with size < 50 KB and signature < 20 KB. | Accepted |

**CHAPTER 5**

**5.1 CONCLUSION AND FUTURE SCOPE**

**Conclusion**

The notion of manipulating encrypted form of data has the capability of revolutionizing the traditional computing. Indeed, this concept has many direct applications in cloud computing environments, banking systems, e-voting, e-cash and many other systems.

We highlighted the application of partial homomorphic encryption and proposed protocol for banking system, though it still requires to overcome the limitations. We found that the present technology is not adequately developed enough to carry out such time-consuming activities. Even the encryption schemes proposed by *Gentry et al.* in [5-7] are unachievable. We believe that there is still a great opening for people to come up with more efficient homomorphic encryption schemes.

As the concept is prompting itself it is capable of revolutionizing the traditional computation. Indeed, this concept has many direct applications in cloud computing environments, banking systems, electronic voting, electronic cash and many other. Right now, this concept is struck with researchers, they are finding more plausible notion to fit in day to day life. The major drawback with this concept is time lagging, as this takes lots of time to compute single operation as mentioned by Gentry Craig in 2009 and 2010.

**5.2 Future scope**

As future work, we are in preparation to work on the optimization of the efficiency of our proposed system and try to overcome the limitations that currently persists. The system needs to be enhanced to perform more complicated bank computations with efficiency. 32

We highlighted the application of partial homomorphic encryption and proposed protocol for banking system, though it still requires to overcome the limitations. Key future scope are electronic voting and electronic cash.

**Electronic voting**

Semantic security is not the only consideration. There are situations under which malleability may be desirable. The above homomorphic properties can be utilized by secure electronic voting systems. Consider a simple binary ("for" or "against") vote. Let *m* voters cast a vote of either *1* (for) or *0* (against). Each voter encrypts their choice before casting their vote. The election official takes the product of the *m* encrypted votes and then decrypts the result and obtains the value *n*, which is the sum of all the votes. The election official then knows that *n* people voted *for* and *m-n* people voted *against*. The role of the random *r* ensures that two equivalent votes will encrypt to the same value only with negligible likelihood, hence ensuring voter privacy.

**Electronic cash**

Another feature named in paper is the notion of self-blinding. This is the ability to change one ciphertext into another without changing the content of its decryption. This has application to the development of ecash, an effort originally spearheaded by David Chaum. Imagine paying for an item online without the vendor needing to know your credit card number, and hence your identity. The goal in both electronic cash and electronic voting, is to ensure the e-coin (likewise e-vote) is valid, while at the same time not disclosing the identity of the person with whom it is currently associated.

**APPENDIX**

**(Code)**

**DEBIT CLASS**

package homomorphicencryption1;

import java.awt.event.\*;

import java.sql.\*;

import java.math.BigInteger;

import java.util.Random;

import java.io.\*;

import javax.swing.\*;

import java.awt.\*;

import java.text.SimpleDateFormat;

public class Debit1 extends JFrame implements ActionListener

{

JPanel p1,p2;

JTabbedPane tp;

JButton sub1,sub2,back1,back2;

JTextField accp,accs,amt1,amt2,acc2;

private String pp;

Debit1()

{

accp=new JTextField(20);

acc2=new JTextField(20);

accs=new JTextField(20);

amt1=new JTextField(20);

amt2=new JTextField(20);

sub1=new JButton("Submit");

sub1.addActionListener(this);

sub2=new JButton("Submit");

back1=new JButton("Back");

back2=new JButton("Back");

sub2.addActionListener(this);

back1.addActionListener(this);

back2.addActionListener(this);

p1=new JPanel(new GridLayout(4,2,20,20));

p1.add(new JLabel(" Sender's Account No :"));

p1.add(accs);

p1.add(new JLabel(" Payee's Account No :"));

p1.add(accp);

p1.add(new JLabel(" Amount :"));

p1.add(amt1);

p1.add(back1);

p1.add(sub1);

p2=new JPanel(new GridLayout(3,2,20,20));

p2.add(new JLabel(" Account No :"));

p2.add(acc2);

p2.add(new JLabel(" Amount :"));

p2.add(amt2);

p2.add(back2);

p2.add(sub2);

tp=new JTabbedPane();

tp.addTab("Account Transfer", p1);

tp.addTab("Cash", p2);

setLayout(new FlowLayout());

add(tp);

setSize(650,400);

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e)

{

BigInteger id=null,wbal=null,fbal=null,newfbal=null,tempbal=null,newwbal=null,fmoney=null;

int wmoney=0,fmoney2=0;

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con= DriverManager.getConnection ("jdbc:mysql://localhost:3306/java\_project","root","root");

Statement stmt=con.createStatement();

HomomorphicFunctions hf=new HomomorphicFunctions();

if(e.getSource()==back1)

{

Main m=new Main(new javax.swing.JFrame(), true);

setVisible(false);

m.setVisible(true);

}

else if(e.getSource()==back2)

{

Main m=new Main(new javax.swing.JFrame(), true);

setVisible(false);

m.setVisible(true);

}

else if(e.getSource()==sub1)

{

ResultSet rs=stmt.executeQuery("select pan from customer\_details where Account\_No="+accs.getText());

rs.first();

if(rs.getRow()!=1)

{

JOptionPane.showMessageDialog(this,"Wrong Sender's Account no.","Balance",JOptionPane. INFORMATION\_MESSAGE);

accs.setText("");

}

else

{

pp=rs.getString("pan");

rs=stmt.executeQuery("select \* from keyval where pan='"+pp+"'");

rs.first();

hf.GeneratingValues(new BigInteger(rs.getString(2)),new BigInteger(rs.getString(3)));

rs=stmt.executeQuery("select pan from customer\_details where Account\_No="+accp.getText());

rs.last();

if(rs.getRow()!=1)

{

JOptionPane.showMessageDialog(this,"Wrong Payee's Account no.","Balance",JOptionPane.INFORMATION\_MESSAGE);

accp.setText("");

}

else

{

pp=rs.getString("pan");

System.out.println(pp);

rs=stmt.executeQuery("select \* from balance");

while(rs.next())

{

id=new BigInteger(rs.getString(1));

BigInteger account=new BigInteger(hf.Decryption(id)+"");

BigInteger tempacc=new BigInteger(accs.getText());

if(account.compareTo(tempacc) ==0)

{

wbal=new BigInteger(rs.getString(2));

fbal=new BigInteger(rs.getString(3));

break;

}

}

String money=amt1.getText();

System.out.println(money);

int strlen=money.length();

char ch1='0';

char ch2='0';

if(money.charAt(strlen-3)=='.')

{

ch1= money.charAt(strlen-2);

ch2= money.charAt(strlen-1);

}

else

{

JOptionPane.showMessageDialog(this,"Wrong Format! Only two digits after the decimal.","Balance",JOptionPane.INFORMATION\_MESSAGE);

System.exit(1);

}

String cha=""+ch1+ch2;

System.out.println(cha);

fmoney2=Integer.parseInt(cha);

fmoney=new BigInteger(cha);

float f=Float.parseFloat(money);

wmoney=(int)f;

if(wmoney>Integer.parseInt(hf.Decryption(wbal)+""))

{

JOptionPane.showMessageDialog(this,"Insufficient Balance","Balance", JOptionPane.INFORMATION\_MESSAGE);

}

else

{ newwbal=hf.Subtract(wbal,hf.Encryption(new BigInteger(wmoney+"")));

if(fmoney2>Integer.parseInt(hf.Decryption(fbal)+""))

{

tempbal=hf.add(hf.Encryption(new BigInteger("100")),fbal);

System.out.println(hf.Decryption(tempbal));

newfbal=hf.Subtract(tempbal,hf.Encryption(fmoney));

System.out.println(hf.Decryption(newfbal));

newwbal=hf.Subtract(newwbal,hf.Encryption(new BigInteger("1")));

System.out.println(hf.Decryption(newwbal));

}

else

{

newfbal=hf.Subtract(fbal,hf.Encryption(fmoney));

}

BigInteger snewbal,snewfbal;

snewbal=hf.Decryption(newwbal);

snewfbal=hf.Decryption(newfbal);

String query="update balance set wbal=? where acc=?";

PreparedStatement pstmt1=con.prepareStatement(query);

pstmt1.setString(1, newwbal.toString());

pstmt1.setString(2, id.toString());

query="update balance set fbal=? where acc=?";

PreparedStatement pstmt2=con.prepareStatement(query);

pstmt2.setString(1, newfbal.toString());

pstmt2.setString(2, id.toString());

rs=stmt.executeQuery("select \* from keyval where pan='"+pp+"'");

rs.first();

hf.GeneratingValues(new BigInteger(rs.getString(2)),

new BigInteger(rs.getString(3)));

rs=stmt.executeQuery("select \* from balance");

while(rs.next())

{

id=new BigInteger(rs.getString(1));

BigInteger account=new BigInteger(hf.Decryption(id)+"");

BigInteger tempacc=new BigInteger(accp.getText());

if(account.compareTo(tempacc)==0)

{

wbal=new BigInteger(rs.getString(2));

fbal=new BigInteger(rs.getString(3));

break;

}

}

newwbal=hf.add(wbal,hf.Encryption(new BigInteger(wmoney+"")));

newfbal=hf.add(fbal,hf.Encryption(fmoney));

query="update balance set wbal=? where acc=?";

PreparedStatement pstmt3=con.prepareStatement(query);

pstmt3.setString(1, newwbal.toString());

pstmt3.setString(2, id.toString());

query="update balance set fbal=? where acc=?";

PreparedStatement pstmt4=con.prepareStatement(query);

pstmt4.setString(1, newfbal.toString());

pstmt4.setString(2, id.toString());

rs=stmt.executeQuery("select t\_id from value");

rs.first();

int t\_id=rs.getInt(1);

query="update value set t\_id=? where t\_id=?";

PreparedStatement pstmt5=con.prepareStatement(query);

pstmt5.setInt(1, (t\_id+2));

pstmt5.setInt(2, t\_id);

query="insert into t\_log values(?,?,?,?,?,?)";

PreparedStatement pstmt6=con.prepareStatement(query);

pstmt6.setInt(1,(t\_id));

pstmt6.setString(2,"Debit");

pstmt6.setString(3,wmoney+"."+fmoney2);

pstmt6.setInt(4,Integer.parseInt(accs.getText()));

pstmt6.setString(5,new SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(new java.util.Date()));

pstmt6.setInt(6,0);

query="insert into t\_log values(?,?,?,?,?,?)";

PreparedStatement pstmt7=con.prepareStatement(query);

pstmt7.setInt(1,(t\_id+1));

pstmt7.setString(2,"Credit");

pstmt7.setString(3,wmoney+"."+fmoney2);

pstmt7.setInt(4,Integer.parseInt(accp.getText()));

pstmt7.setString(5,new SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(new java.util.Date()));

pstmt7.setInt(6,0);

rs=stmt.executeQuery("select Email from customer\_details where Account\_No="+accs.getText());

String mail="";

while(rs.next())

{

mail=rs.getString(1);

}

SendEmail sm=new SendEmail();

int i=sm.mail(Integer.parseInt(accs.getText()),mail,"Debit",amt1.getText(),snewbal+"."+snewfbal,"");

rs=stmt.executeQuery("select Email from customer\_details where Account\_No="+accp.getText());

while(rs.next())

{

mail=rs.getString(1);

}

int j= sm.mail(Integer.parseInt(accp.getText()),mail, "Credit",amt1.getText(), hf.Decryption(newwbal)+"."+hf.Decryption(newfbal),"");

if(i==1&&j==1)

{

pstmt1.executeUpdate();

pstmt2.executeUpdate();

pstmt3.executeUpdate();

pstmt4.executeUpdate();

pstmt5.executeUpdate();

pstmt6.executeUpdate();

pstmt7.executeUpdate();

JOptionPane.showMessageDialog(this,"Transaction Successfull !!","Balance",JOptionPane.INFORMATION\_MESSAGE);

Main m=new Main(new javax.swing.JFrame(), true);

setVisible(false);

m.setVisible(true);

}

}

}

}

}

else if(e.getSource()==sub2)

{

ResultSet rs=stmt.executeQuery("select pan from customer\_details where Account\_No="+acc2.getText());

rs.last();

if(rs.getRow()!=1)

{

JOptionPane.showMessageDialog(this,"Wrong Sender's Account no.","Balance",JOptionPane.INFORMATION\_MESSAGE);

accs.setText("");

}

else

{

pp=rs.getString("pan");

System.out.println(pp);

rs=stmt.executeQuery("select \* from keyval where pan='"+pp+"'");

rs.first();

hf.GeneratingValues(new BigInteger(rs.getString(2)),new BigInteger(rs.getString(3)));

rs=stmt.executeQuery("select \* from balance");

while(rs.next())

{

id=new BigInteger(rs.getString(1));

BigInteger account=new BigInteger(hf.Decryption(id)+"");

BigInteger tempacc=new BigInteger(acc2.getText());

if(account.compareTo(tempacc)==0)

{

wbal=new BigInteger(rs.getString(2));

break;

}

}

System.out.println(hf.Decryption(wbal));

int amount=0;

String money=amt2.getText();

try

{

amount=Integer.parseInt(money+"");

if(amount>Integer.parseInt(hf.Decryption(wbal)+""))

{

JOptionPane.showMessageDialog(this,"Insufficient Balance","Balance",JOptionPane.INFORMATION\_MESSAGE);

amt2.setText("");

}

else

{

newwbal=hf.Subtract(wbal,hf.Encryption(new BigInteger(amount+"")));

String query="update balance set wbal=? where acc=?";

PreparedStatement pstmt1=con.prepareStatement(query);

pstmt1.setString(1, newwbal+"");

pstmt1.setString(2, id+"");

rs=stmt.executeQuery("select t\_id from value");

rs.first();

int t\_id=rs.getInt(1);

int nt=t\_id+1;

query="update value set t\_id=? where t\_id=?";

PreparedStatement pstmt2=con.prepareStatement(query);

pstmt2.setInt(1, nt);

pstmt2.setInt(2, t\_id);

query="insert into t\_log values(?,?,?,?,?,?)";

PreparedStatement pstmt3=con.prepareStatement(query);

pstmt3.setInt(1,(t\_id));

pstmt3.setString(2,"Debit");

pstmt3.setString(3,amount+"");

pstmt3.setInt(4,Integer.parseInt(acc2.getText()));

pstmt3.setString(5,new SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(new java.util.Date()));

pstmt3.setInt(6,0);

rs=stmt.executeQuery("select Email from customer\_details where Account\_No="+acc2.getText());

String mail="";

while(rs.next())

{

mail=rs.getString(1);

}

SendEmail sm=new SendEmail();

int i=sm.mail(Integer.parseInt(acc2.getText()),mail,"Debit", amt2.getText(),hf.Decryption(newwbal)+"","");

if(i==1)

{

pstmt1.executeUpdate();

pstmt2.executeUpdate();

pstmt3.executeUpdate();

JOptionPane.showMessageDialog(this,"Transaction Successfull !!","Balance", JOptionPane.INFORMATION\_MESSAGE);

Main m=new Main(new javax.swing.JFrame(), true);

setVisible(false);

m.setVisible(true);

}

}

}

catch(NumberFormatException nfe)

{

JOptionPane.showMessageDialog(this,"Wrong Amount! No floating point.","Balance",JOptionPane.INFORMATION\_MESSAGE);

amt2.setText("");

}

}

}

con.close();

}catch(Exception ex)

{

JOptionPane.showMessageDialog(this,"Sorry for Inconvenience.[Database Error]","Balance",JOptionPane.INFORMATION\_MESSAGE);

System.out.println(ex);

}

}

public static void main(String[] args)

{

new Debit1();

}

}

**HOMOMORPHIC FUNCTIONS CLASS**

package homomorphicencryption1;

import java.math.BigInteger;

import java.util.Random;

public class HomomorphicFunctions

{

private BigInteger lambda,n,nsquare,g;

public void GeneratingValues(BigInteger p, BigInteger q)

{

n = p.multiply(q);

nsquare = n.multiply(n);

g = new BigInteger("2");

lambda = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE)).divide(

p.subtract(BigInteger.ONE).gcd(q.subtract(BigInteger.ONE)));

}

public BigInteger Decryption(BigInteger c)

{

BigInteger u = g.modPow(lambda, nsquare).subtract(BigInteger.ONE).divide(n).modInverse(n);

return c.modPow(lambda, nsquare).subtract(BigInteger.ONE).divide(n).multiply(u).mod(n);

}

public BigInteger Encryption(BigInteger m)

{

BigInteger r = new BigInteger(512, new Random());

return g.modPow(m, nsquare).multiply(r.modPow(n, nsquare)).mod(nsquare);

}

public BigInteger add(BigInteger m1,BigInteger m2)

{

BigInteger sum=m1.multiply(m2.mod(nsquare));

return sum;

}

public BigInteger Subtract(BigInteger m1,BigInteger m2)

{

BigInteger m2in=m2.modPow(new BigInteger("-1"),nsquare);

BigInteger sub=m1.multiply(m2in.mod(nsquare));

return sub;

}

}

**SEND EMAIL CLASS**

package homomorphicencryption1;

//package project;

//https://myaccount.google.com/lesssecureapps?pli=1

import javax.activation.DataHandler;

import javax.activation.DataSource;

import javax.activation.FileDataSource;

import javax.mail.\*;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeBodyPart;

import javax.mail.internet.MimeMessage;

import javax.mail.internet.MimeMultipart;

import java.util.\*;

public class SendEmail

{

public int mail(int account,String email,String type,String amount,String bal,String path)

{

final String username="info.gamepr@gmail.com";

final String pass="justgaming";

Properties props = new Properties();

props.put("mail.smtp.auth", "true");

props.put("mail.smtp.starttls.enable", "true");

props.put("mail.smtp.host", "smtp.gmail.com");

props.put("mail.smtp.port", "587");

Session session = Session.getInstance(props,new javax.mail.Authenticator()

{

@Override

protected PasswordAuthentication getPasswordAuthentication()

{

return new PasswordAuthentication(username,pass);

}

});

try

{

Message message = new MimeMessage(session);

message.setFrom(new InternetAddress("info.gamepr@gmail.com"));

message.setRecipients(Message.RecipientType.TO,

InternetAddress.parse(email));

message.setSubject("Account Details");

//message.setContent("<h:body style=background);

if(path=="")

{

Multipart mp=new MimeMultipart();

BodyPart msgbp=new MimeBodyPart();

BodyPart msgbp1=new MimeBodyPart();

BodyPart msgbp2=new MimeBodyPart();

BodyPart msgbp3=new MimeBodyPart();

msgbp.setText("Account Number:"+account);

msgbp1.setText("\nTransection Type:"+type);

msgbp2.setText("Transection Amount:"+amount);

msgbp3.setText("Available Balance:"+bal);

mp.addBodyPart(msgbp);

mp.addBodyPart(msgbp1);

mp.addBodyPart(msgbp2);

mp.addBodyPart(msgbp3);

message.setContent(mp);

}

else

{

Multipart mp=new MimeMultipart();

BodyPart msgbp=new MimeBodyPart();

BodyPart msgbp1=new MimeBodyPart();

BodyPart msgbp3=new MimeBodyPart();

BodyPart msgbp4=new MimeBodyPart();

BodyPart msgbpf=new MimeBodyPart();

msgbp.setText("Account Number:"+account);

msgbp1.setText("\nTransection Type:"+type);

msgbp3.setText("Available Balance:"+bal);

msgbp4.setText("This Mail contain Transaction File");

String fname=path;

DataSource src=new FileDataSource(fname);

msgbpf.setDataHandler(new DataHandler(src));

msgbpf.setFileName(fname);

mp.addBodyPart(msgbp4);

mp.addBodyPart(msgbp);

mp.addBodyPart(msgbp1);

mp.addBodyPart(msgbp3);

mp.addBodyPart(msgbp);

mp.addBodyPart(msgbpf);

message.setContent(mp);

}

Transport.send(message);

System.out.println("DONE!!");

return 1;

}

catch(MessagingException e)

{

System.out.println(e);

return 0;

}

}

}

**REFERENCES**

[1] R. Rivest, A. Shamir, and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM 21 (2): pp. 120–126, 1978.

[2] S. Goldwasser and S. Micali, Probabilistic Encryption. Journal of Computer and System Sciences, 28(2): pp. 270-299, April 1984.

[3] T. ElGamal, A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory, pp. 469–472, 1985.

[4] Benaloh, Josh. "Dense probabilistic encryption." Proceedings of the workshop on selected areas of cryptography. 1994.

[5] P. Paillier, Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, Advances in Cryptology — EUROCRYPT ’99 In Advances in Cryptology — EUROCRYPT ’99 , Vol. 1592 (1999), pp. 223-238, 1999.

[6] C. Gentry, A fully homomorphic encryption scheme. PhD thesis, Stanford University, 2009.

[7] M. V. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, Fully Homomorphic Encryption over the Integers. EUROCRYPT 2010: pp. 24-43, June 2010.

[8] C. Gentry, Computing arbitrary functions of encrypted data, Commun. ACM, Vol. 53, No. 3., pp. 97-105, March 2010.

[9] Wang L., Wang L, Pan Y, Zhang Z, Yang Y. Discrete logarithm based additively homomorphic encryption and security data aggregation. Information and communications security. 2009; 5927:493-502

[10] Cheon JH, Kim J. A Hybrid Scheme of Public-Key Encryption and Somewhat Homomorphic Encryption. IEEE transactions on information forensics and security. 2015 May; 10(5):1052-63. 34