



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

Securing Bank Details using Elgamal Encryption

*A project submitted
in partial fulfillment of the requirements for the degree of
Int M.Tech. (Software Engineering)*

By

Akash PS (20MIS0085)

Siddharth Kumar (20MIS0087)

Anjali Jha (20MIS0124)

Niharika Sharma (20MIS0131)

Rose Mariya (20MIS0163)

Course Instructor

Dr. Selva Rani B

Associate Professor

Nov 2022

UNDERTAKING

This is to declare that the project entitled “Securing Bank Details using Elgamal Encryption” is an original work done by undersigned, in partial fulfillment of the requirements for the degree *Int. M.Tech. (Software Engineering)* at School of Information Technology and Engineering, VIT, Vellore.

All the analysis, design and system development have been accomplished by the undersigned.

Akash PS (20MIS0085)

Siddharth Kumar (20MIS0087)

Anjali Jha (20MIS0124)

Niharika Sharma (20MIS0131)

Rose Mariya (20MIS0163)

CONTENT

SNo	Topic	PageNo
	Abstract	4
1	Introduction	5
2	Literature Survey	6
3	Modules and Descriptions	12
4	Implementation	13
5	Results	24
6	Conclusion	31
7	Team Members' Contribution	32
	References	32

ABSTRACT

*People now prefer using **Online Banking Portals** to know all related bank information, such as fund transfers, checking the transaction history etc. In the current busy society, people need their money to be safe, and hence they prefer to store it in banks. This brings about a clamour on banking applications to provide the customers with a fast, efficient and secure system. Now moving our attention to security, we will emphasise on the very aspect of a secure banking system which is the locus of this project. To stop intruders from accessing sensitive data, the banking portal needs to be safeguarded. Therefore we will be utilizing some encryption algorithm to ensure that the portal application and the associated data are safe, as there is a possibility of attack by the external sources to access the incorporated into this application system.*

1. Introduction

In the present situation, people utilise online financial services and bank transfers a lot to save their time. Therefore our banking application will enable the user to download the transaction statement in accordance with the account type and other user's requirements. When a user requests it, the bank enables them to do the necessary activities. When a user requests that a bank statement be generated, the bank enables it to be retrieved from the server and offers a download option. During this process of request and approval, user authentication and various security features are processed parallelly with the data transmission. One among this process includes the security of this data transfer so that during the transmission process if the message is being tracked in the transmission channel, they third party members associated with this are not able to see the actual data, but an encrypted data which cannot be understood or analysed easily. In our project, we provide the security for this data transmission using the Elgamal encryption method.

In this project the main theme is to provide the interaction between the bank and the customer to particularly know the bank statement. The user logs into the bank website through the provided Username and MPIN for that customer and the customer is allowed to select the duration of bank statement that he wanted to print for this input the bank responds and provides access of information over the server, then the file in the server will be encrypted using Elgamal algorithm and that encrypted file will be sent to the over the network as it is encrypted un authorized cannot access the data in the file as the key is generated by the user and that helped the server to encrypt the file and on server returning the encrypted file, the returned file from server can be decrypted.

2. Literature Survey

S. No.	Paper Title	Author	Year Publi-shed	Problem Statement	Algorithm Applied	Features
1	SMS Banking Encryption Scheme	Yoso Adi Setyoko and Maman Abdurohm	2017	They proposed a security aspect scheme of data confidentiality for SMS banking using Elgamal algorithm. Randomness feature in this algorithm gives more security support for delivered message. Assures the unreadability of the data by attackers at the communication line with the help of this proposed system. Encryption process consumes less than 2KB data. Messages will also be compressed and to do so they have chosen the compression method to reduce the number of characters of the encrypted message by replacing two pair of characters by one single character. Compression and decompression processes require 0.36 seconds and 0.35 seconds respectively. The memory consumption of the compression and decompression processes is 31.6 KB and 4.97 KB respectively.	Elgamal	Some of the features provided by SMS banking are money transfer, paying electricity bills, purchasing tickets, recharging voucher, changing transaction PIN.
2	3G communication encryption algorithm based on ECC-Elgamal	Dewu Xu and Wei Chen	2010	They have combined the elliptic curve algebra system with ElGamal encryption algorithm. They made use of elliptic curve points over finite fields to constitute a finite group to achieve	Elgamal Encryption	Mostly suitable to applications getting related with smart cards.

				discrete logarithm algorithms. The security of this algorithm depends on difficulty of computing discrete algorithm over finite fields. To combine both the algorithms and make use of advantage of ECC-ElGamal algorithm to pass session keys can completely meet the demand of efficiency of 3G communications. When compared to both algorithms they came to a conclusion that ECC can achieve relatively higher security by means of relatively smaller cost and time delay. In this way they have provided better way of key exchange and management of 3G communications.		
3	A algorithm of fully homomorphic encryption	Guangli Xiang, Ben zhi Yu and Ping Zhu	2012	They calculated data directly and not need to do it after 4 decrypting data by the fully homomorphism, and it has been proved to be secure. They have produced two keys namely public key and private key in order to perform ElGamal algorithm. In their observation they found that this algorithm is a partly secure encryption scheme, and much need to be done meant for development of entire security analysing encryption scheme.	Homomorphi c encryption	This algorithm has many good properties like it ensures that malicious users cannot infer original data through the statistical law.
4	A commutative encryption scheme based on elgamal encryption	Kaibin Huang and Raylin Tso	2012	They co-related with many real life applications such as in secret sharing, database integration and etc. They proposed a new	Commutative encryption based on Elgamal	It can also be applied in protecting horizontal partitioned data

				commutative encryption scheme based on the ElGamal encryption and provide the security proof in the random oracle model. Other related fields it can be imposed on are cryptography applications.		that are stored on different databases.
5	A faster fully homomorphic encryption scheme in Big Data	Wang and Bing Guo, Yan Shen, Shun-Jun Cheng and Yong-Hong Lin	2017	They have found a new improved FHE scheme based on DGHV scheme is proposed. Their proposal is meant for people to encrypt their private data and to connect with the true value of that data at the same time. For the betterment of security, we also need to keep corresponding cipher texts secret during the computational work. They also proved that this scheme is semantically secure under the error-free approximate GCD problem. Future work of them could be to improve the performance of current scheme and develop a practical scheme.	Homomorphic Encryption Scheme in Big Data	Making use of this algorithm, they are allowed to perform unlimited chaining of mathematical operations on encrypted data making it possible for legal companies and suitable instructions to use it.
6	A homomorphic elgamal variant based on BGN's method	Zhiwei. Chen, Ruoqing. Zhang, Yatao. Yang and Zichen. Li	2013	To meet the multiplicative homomorphism they adopted a bilinear pairing map. By setting specific 5 parameter with the assumption of bilinear map, they changed the raw ElGamal to a practical cryptosystems with involvement of both additive and multiplication homeomorphisms.	Elgamal	With the adoption of bilinear pair mapping the scheme is able to implement multiplicative homomorphism.
7	A hybrid encryption scheme based on the satisfiability	RuiSun, QiaoPeng and YouliangTian	2017	They have combined Elgamal scheme with advantages of satisfiability problem to construct a secure, verifiable encryption	Hybrid encryption system	They also proved that the security of the scheme by using the

	problem			scheme. Newly invented thing by them is the private key of the hybrid encryption scheme is shorter and the cipher text satisfies the verifiability, and this scheme is responsible for establishing security against adaptive chosen cipher text attacks. Invention of their scheme is much more superior than the existing one.		provable security theory
8	A new model for Privacy Preserving Multiparty Collaborative Data Mining	S. Bhanumathi and Sakthivel	2013	They proposed a new Binary Integer Programming model for multiparty collaborative data mining, which is meant for providing solutions to the problem of disclosure of sensitive data that is investigated previously. In addition to this, maintaining confidentiality of the newly created pooled data by semantically secured Elgamal Encryption scheme.	semantically secured Elgamal Encryption	Goal is to provide security of data, when it's crossed over the network from data providers to service providers
9	Towards a service-oriented architecture for eVoting	A. S. Sodiya, S. A. Onashoga and D.I. Adelani	2011	They have combined elliptic curve with Elgamal cryptosystem for the enhancement of security to e-voting architecture. A voter is allowed to revoke in a situation arises in such a way that he is encountered to a forced vote at another location. EC-Elgamal cryptosystem is used for encryption of each and every vote to prevent problems and later decrypted in the acceptance phase to ensure fairness.	. Elgamal cryptosystem	Elliptic curve discrete logarithm is the underlying principle which is meant for security of the voting scheme.

10	An accelerated fully homomorphic encryption scheme over the integers	Peng Zhang, Xiaoqiang Sun, Ting Wang, Sizhu Gu, Jianping Yu and Weixin Xie	2016	Homomorphic encryption scheme is obtained by using Gentry's squashing decryption circuit technique, which could resist only chosen plaintext attacks. Efficiency analysis that they have been conducted shows that the proposed scheme's public key size and cipher text size are smaller than that of DGHV. From the observed results they came to a conclusion that the proposed.	Homo morphic encryption	Elliptic curve discrete logarithm is the underlying principle which is meant for security of the voting scheme
11	Automated security proof of the elgamal encryption scheme	Chen Nan, LI Anle, GU Chunxian g and ZHU Yuefei	2010	They have designed a tool which can satisfy much more cryptographic protocols for the automatical proof. Introducing the essential support for structure of attack game. Their aim was to prove that the semantic security of the encryption scheme Elgamal and It's hashed version.	Elgamal	security of some encryption schemes with the automated security instrument under the appropriate attack model
12	A general framework to design secure cloud storage protocol using homomorphic encryption scheme	Jian Zhang, Yang Yang , Yanjiao Chen, Jing Chen, Qian Zhang	2017	In this paper they thought an idea of including homophoric encryption in securing the cloud storage. They constructed three concrete secure cloud storages using three algorithms which are RSA, Paillier and DGHV based homophoric encryption. For these implementations they calculated the storage costs, communication costs and computation costs.	Homo morphic encryption	RSA when used with homophoric encryption reduces all types of costs.
13	Analysis and comparison of fully layered image encryption	Pratibha Chaudhar y, Ritu Gupta, Abhilasha	2019	This paper gives the fundamentals of arrangements for example part of the way homomorphic encryption, to	Homo morphic encryption	Utilizes the hardness of Idealgrid, whole numbers, learning with

	techniques and partial image encryption techniques	Singh, Pramathesh Majumder		some degree homomorphic encryption and completely homomorphic encryption. Fundamentally underscores on full homomorphic encryption and an examination of various full homomorphic encryption plots.		blunder, elliptic bend cryptography based.
14	Visual cryptography and image processing based approach for secure transactions in banking sector	Aaditya Jain and Sourabh Soni	2017	It expalains that good hackers can fetch biometric details of customers from the bank's database and later can use it for fake transactions. To avoid all this catastrophic things Visual cryptographic technique is used. Visual Cryptography is an efficient encryption scheme in which information hide inside the images and decrypted only by human visual system. Here we consider the case of joint account operation.	Visual cryptography	This paper propose a secure XOR operation based visual cryptography and image processing technique to secure banking transaction.
15	Secured privacy data using multi key encryption in cloud storage	P Muthi Reddy, S.H. Manjula and K. R. Venugopa	2018	This paper is about a novel Dynamic Privacy Aggregate Key ReEncryption (DPAK-RE) algorithm. The DPAKRE algorithm provides security to sharing the information in the cloud.they describe a secure data sharing with other users in cloud storage.	DPAK-RE Algorithm	Data corner is authorized to the admittance approach related with individual information that is to be stored within jurisdiction.

3. Modules and Description

List of Modules :

- Login Module (User)
- Enquiry Module (User requests)
- Database Module

Module Description :

Login Module (User)

Login is a process for accessing a system by entering the identity of the user and the password to obtain permissions using the destination computer resources. In banking websites, confidentiality is very important. Login aims to provide security services on the system; here it is the banking system. In the banking system, Elgamal cryptography algorithm is used to secure username and password in web login. The security level of this algorithm is based on the problem of discrete logarithms in the multiplication group of prime modulo primes. This algorithm includes asymmetric cryptography algorithms that use two key types, namely public key and secret key. The data contained in the login is secured by using Elgamal algorithm, so the username and password entered into the database are already in the form of cipher text. When logging in to enter the system, the user logs into the bank website through the provided Username and MPIN, the mobile banking personal identification number specific for each user.

Enquiry Module (User requests)

After the user successfully logging in into the system, the aim of the user will be to gather transactions details based on their requirements. Actually, the portal is designed to access and download the transaction statement. They can customize according to their needs and download the statement like ad-hoc queries. They can able to filter out the transaction details based on the type of account they are using which are savings or current account, and then based on status of the transactions such as successful or failed transactions and

then finally for the time period for which they need transaction details. For the security reason, the user has to enter their MPIN to download the statement. After entering MPIN, portal will verify the MPIN and then only user can able to download the statement.

Database Module

Databse module is the central database repository, which basically contains all the data associated with the banking application. Various entities like user, accounts, employees, transaction history etc are all stored in the central repository, and consists of various call requests and methods to access and manipulate this data. For our project, among various entities, our focus will primarily be on user data and the analogous transaction with that user.

4. Implementation

Elgamal

We will take a look at the ElGamal public key cipher system for a number of reasons:

- To show that RSA is not the only public key system
- To exhibit a public key system based on a different one way function
- ElGamal is the basis for several well-known cryptographic primitives

Setting up Elgamal

*Let p be a large prime

– By “large” we mean here a prime rather typical in length to that of an RSA modulus.

*Select a special number g

– The number g must be a primitive element modulo p .

*Choose a private key x

– This can be any number bigger than 1 and smaller than $p-1$.

*Compute public key y from x , p and g

– The public key y is g raised to the power of the private key x modulo p .

In other words: $y = g^x \bmod p$

Example problem of Elgamal

Step 1: Let $p = 23$

Step 2: Select a primitive element $g = 11$

Step 3: Choose a private key $x = 6$

Step 4: Compute $y = 11^6 \bmod 23$
 $= 9$

Public key is 9 and Private key is 6.

Elgamal Encryption

The first job is to represent the plaintext as a series of numbers modulo p .

Then:

1. Generate a random number k
2. Compute two values $C1$ and $C2$, Where,
 $C1 = g^k \bmod p$ and $C2 = My^k \bmod p$
3. Send the cipher text C , which consists of the two separate values $C1$ and $C2$.

Elgamal Encryption example

To encrypt $M = 10$ using Public key 9

1. Generate a random number $k = 3$
2. Compute $C1 = 11^3 \bmod 23 = 20$
 $C2 = 10 \times 9^3 \bmod 23 = 10 \times 16 = 160 \bmod 23 = 22$
3. Cipher text $C = (20, 22)$

Elgamal Decryption

$C1 = g^k \bmod p$ and $C2 = My^k \bmod p$

1. The receiver begins by using their private key x to transform $C1$ into something more useful:

$$C1^x = (g^k)^x \bmod p$$

- NOTE: $C1^x = (gk)^x = (g^x)^k = (y)^k = y^k \mod p$
2. This is a very useful quantity because if you divide C2 by it you get M.
In other words: $C2 / y^k = (My^k) / y^k = M \mod p$

Elgamal Decryption example

To decrypt $C = (20, 22)$

1. Compute $206 = 16 \mod 23$
2. Compute $22 / 16 = 10 \mod 23$
3. Plaintext = 10

Elgamal Cryptography

Public key cryptosystem related to D-H so uses exponentiation in a finite (Galois) with security based difficulty of computing discrete logarithms, as in D-H each user (eg. A) generates their key

- chooses a secret key (number): $1 < x_A < q-1$
- compute their public key: $y_A = a^{x_A} \mod q$

Elgamal Message Exchange

Bob encrypt a message to send to A computing

- represent message M in range $0 \leq M \leq q-1$
 - longer messages must be sent as blocks
- chose random integer k with $1 \leq k \leq q-1$
- compute one-time key $K = y_A^k \mod q$
- encrypt M as a pair of integers (C1, C2) where
 - $C1 = a^k \mod q$; $C2 = KM \mod q$

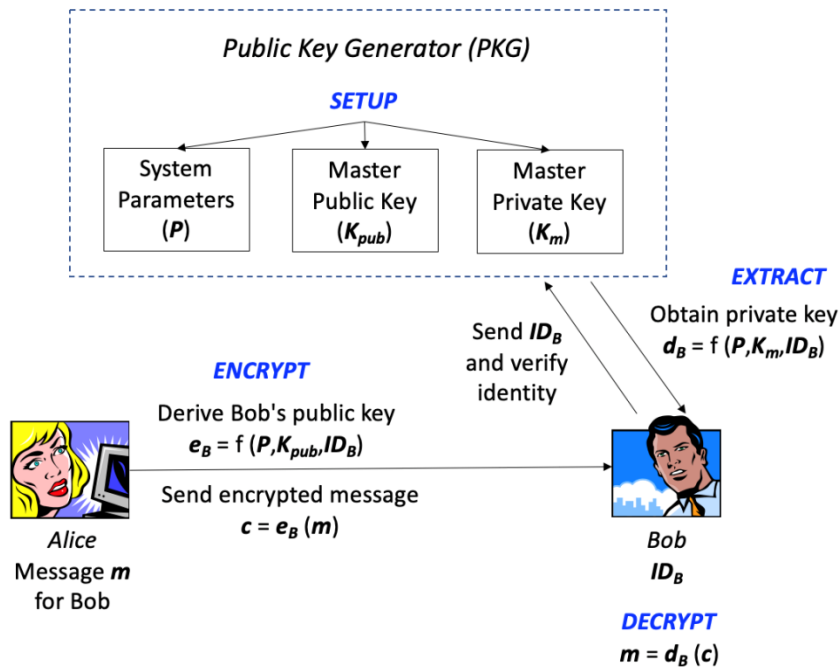
A then recovers message by

- recovering key K as $K = C1^{x_A} \mod q$
- computing M as $M = C2 K^{-1} \mod q$

A unique k must be used each time

- Otherwise result is insecure

Elgamal Example



- Use field $GF(19)$ $q=19$ and $a=10$
- Alice computes her key:
 - A chooses $x_A=5$ & computes $y_A=10^5 \bmod 19 = 3$
- Bob send message $m=17$ as $(11,5)$ by
 - choosing random $k=6$
 - computing $K = y_A^k \bmod q = 3^6 \bmod 19 = 7$
 - computing $C1 = a^k \bmod q = 10^6 \bmod 19 = 11$
 - computing $C2 = K \cdot m \bmod q = 7 \cdot 17 \bmod 19 = 5$
- Alice recovers original message by computing:
 - recover $K = C1^{x_A} \bmod q = 11^5 \bmod 19 = 7$
 - compute inverse $K^{-1} = 7^{-1} = 11$
 - recover $M = C2 \cdot K^{-1} \bmod q = 5 \cdot 11 \bmod 19 = 17$

IMPLEMENTATION CODE:

Key Generation –

```
package elGamal;

import java.lang.*;
import java.io.*;
import java.util.*;
import java.math.BigInteger;
import static java.lang.Math.*;

public class KeyGeneration {
    private BigInteger p;
    private BigInteger g;
    private BigInteger SECRETKEY;
    private BigInteger b;
    private int pval[]={151,157,163,167,179,};
    private int xval[]={45,47,49,51,53,57};

    public void generatekeys()
    {
        Scanner stdin = new Scanner(System.in);
        Random r = new Random();
        double ran= random();
        int ran2=(int) ((ran*100)%5);
        ran2=1;
        p = BigInteger.valueOf(pval[ran2]);
        g = getGenerator(p, r);
        if (g != null)
        {
            SECRETKEY = BigInteger.valueOf(xval[ran2]);
            b = g.modPow(SECRETKEY, p);
            System.out.println("\nPublic keys:: P = "+p+" G = "+g+" B
= "+b +"\nSecret Key :: "+SECRETKEY);
        }
        else
            System.out.println("Sorry, a generator for your prime
couldn't be found.");
    }

    public static BigInteger getNextPrime(String ans)
    {
        BigInteger one = new BigInteger("1");
        BigInteger test = new BigInteger(ans);
        while (!test.isProbablePrime(99))
            test = test.add(one);
        return test;
    }

    public static BigInteger getGenerator(BigInteger p, Random r)
    {
        int numtries = 0;
        while (numtries < 1000)
```

```

        {
            BigInteger rand = new BigInteger(p.bitCount()-1,r);
            BigInteger exp = BigInteger.ONE;
            BigInteger next = rand.mod(p);
            while (!next.equals(BigInteger.ONE))
            {
                next = (next.multiply(rand)).mod(p);
                exp = exp.add(BigInteger.ONE);
            }
            if (exp.equals(p.subtract(BigInteger.ONE)))
                return rand;
        }
        return null;
    }
    public BigInteger getp()
    { return p; }
    public BigInteger getg()
    { return g; }
    public BigInteger getb()
    { return b; }
    public BigInteger getSecretkey()
    { return SECRETKEY; }
}

```

Encryption –

```

package elGamal;

import java.lang.*;
import java.io.*;
import static java.lang.Math.random;
import java.util.*;
import java.math.BigInteger;

public class Encryption {

    FileInputStream fin;
    FileOutputStream fout = null;
    FileWriter f1,f2,f3;
    String s1,s2;
    private int sk;
    double counter=0;

    public void Encrypt(BigInteger p,BigInteger g, BigInteger
b,BigInteger a)
    {
        sk=a.intValue();
        int alert=0;
        try
        {

```

```

        f3=new FileWriter("C:\\Users\\Siddharth Kumar\\eclipse-
workspace\\elGamal\\output of project\\second_encryption.txt");
        f2=new FileWriter("C:\\Users\\Siddharth
Kumar\\eclipse-workspace\\elGamal\\output of project\\Encryption.txt");
        f1=new FileWriter("C:\\Users\\Siddharth
Kumar\\eclipse-workspace\\elGamal\\output of
project\\first_encryptpion.txt");
        Scanner stdin = new Scanner(System.in);
        Random r = new Random();
        BigInteger k = new BigInteger(p.bitCount()-1, r);
        fin = new FileInputStream("C:\\Users\\Siddharth
Kumar\\eclipse-workspace\\elGamal\\output of project\\input.txt");
        int i=0;
        while((i=fin.read())!=-1)
        {
            BigInteger c1 = g.modPow(k, p);
            BigInteger c2 = b.modPow(k, p);
            BigInteger m = BigInteger.valueOf(i);
            c2 = c2.multiply(m);
            c2 = c2.mod(p);
            char ch=(char)i;
            int intc1=c1.intValue();
            int intc2=c2.intValue();
            f1.write("Character=\t" +ch+"\tIts ASCII IS=
"+i+"\tC1 Value is =\t"+intc1+"\tC2 Value is =\t"+intc2+"\n" );
            f1.flush();
            gentryencrytion(intc1, intc2);
            if( intc1 >=100 ||intc2>=100)
                alert=1;
            String s1=Integer.toString(intc1);
            String s2=Integer.toString(intc2);
            byte b1[]=s1.getBytes();
            byte b2[]=s2.getBytes();
            BigInteger temp = c1.modPow(a,p);
            temp = temp.modInverse(p);
            BigInteger recover = temp.multiply(c2);
            recover = recover.mod(p);
            counter++;
        }
    }
    catch(Exception e)
    {
        System.out.println(e+"\n--Create an Output Project Folder--");
    }
}

public void gentryencrytion(int a, int b)
{
    s1=Integer.toBinaryString(a);
    s2=Integer.toBinaryString(b);
    switch(s1.length())
    {
        case 1:
            s1="0000000"+s1;
            break;
        case 2:
            s1="000000"+s1;

```

```

        break;
    case 3:
        s1="00000"+s1;
        break;
    case 4:
        s1="0000"+s1;
        break;
    case 5:
        s1="000"+s1;
        break;
    case 6:
        s1="00"+s1;
        break;
    case 7:
        s1="0"+s1;
        break;
}

switch(s2.length())
{
    case 1:
        s2="00000000"+s2;
        break;
    case 2:
        s2="0000000"+s2;
        break;
    case 3:
        s2="000000"+s2;
        break;
    case 4:
        s2="00000"+s2;
        break;
    case 5:
        s2="0000"+s2;
        break;
    case 6:
        s2="000"+s2;
        break;
    case 7:
        s2="00"+s2;
        break;
}

int gentrypubkey[]=new int[4];int r1=5,r2=3,q1=9,q2=11;
gentrypubkey[0]=(sk*q1)+2*r1;
gentrypubkey[1]=(sk*q1)+2*r2;
gentrypubkey[2]=(sk*q2)+2*r1;
gentrypubkey[3]=(sk*q2)+2*r2;
String s3=s1+s2;
char arr[]=s3.toCharArray();
String s4=""; int gencl;

for(int i=0;i<arr.length;i++)
{
    if(arr[i]=='1')
    {
        double ran= random();

```

```

        int ran2=(int) ((ran*100)%4);
        genc1= gentrypubkey[ran2]+1+(2*r1);
        s4=s4+Integer.toString(genc1);
    }
    else
    {
        double ran= random();
        int ran2=(int) ((ran*100)%4);
        genc1= gentrypubkey[ran2]+0+(2*r1);
        s4=s4+Integer.toString(genc1);
    }
}

try
{
    f3.write("C1="+a+"\tC2="+b+"\tBinary C1 "+s1+"\tBinary C2 "+s2+"\n");
    f3.flush();
}
catch(Exception e){e.printStackTrace();}

try
{
    f2.write(s4+"\n");
    f2.flush();
}
catch(Exception e){e.printStackTrace();}
}
}

```

Decryption –

```

package elGamal;

import java.io.*;
import java.lang.*;
import java.util.*;
import java.math.BigInteger;

public class Decryption {

    String s1="",s2="";
    FileWriter f1,f2,f3;

    public void decrypt(BigInteger p,BigInteger secretkey,String file)
    {
        try
        {
            f2=new FileWriter("C:\\Users\\Siddharth Kumar\\eclipse-
workspace\\elGamal\\output of project\\second decryption.txt");
            f3=new FileWriter("C:\\Users\\Siddharth Kumar\\eclipse-
workspace\\elGamal\\output of project\\decrypted.txt");

```

```

        f1=new FileWriter("C:\\Users\\Siddharth Kumar\\eclipse-
workspace\\elGamal\\output of project\\first_decryptpion.txt");
        String substring="";
        int j=0,sk=secretkey.intValue();
        try
        {
            BufferedReader br = new BufferedReader(new
FileReader("C:\\Users\\Siddharth Kumar\\eclipse-
workspace\\elGamal\\output of project\\Encryption.txt"));
            String line;
            while ((line = br.readLine()) != null)
            {
                int i=0;j=0;
                for(i=0;i<48;i=i+3)
                {
                    if(j<8)
                    {
                        substring=line.substring(i,i+3);
                        int a=Integer.parseInt(substring);
                        int bit=(a%sk)%2;
                        s1=s1+Integer.toString(bit);
                        j++;
                    }
                    else
                    {
                        substring=line.substring(i,i+3);
                        int a=Integer.parseInt(substring);
                        int bit=(a%sk)%2;
                        s2=s2+Integer.toString(bit);
                        j++;
                    }
                    if(j==8);
                }
                BigInteger c1,c2;
                f1.write("INPUT IS = "+line+" BINARY - "+s1+"
"+s2+"\n");

                f1.flush();
                int a1=binaryToInteger(s1);
                int a2=binaryToInteger(s2);
                c1=BigInteger.valueOf(binaryToInteger(s1));
                c2=BigInteger.valueOf(binaryToInteger(s2));
                f2.write("BINARY "+s1+"
"+s2+"\tC1="+c1+"\tC2="+c2+"\n");
                f2.flush();
                BigInteger temp = c1.modPow(secretkey,p);
                temp = temp.modInverse(p);
                BigInteger recover = temp.multiply(c2);
                recover = recover.mod(p);
                char ch=(char) recover.intValue();
                f3.write(ch);
                f3.flush();
                s1="";
                s2="";
            }
        }
        catch (Exception e){e.printStackTrace();}
    }
}

```

```

        catch(Exception e){System.out.println("EXCEPTION AT DECRYPTED
FILE");}
    }

    public int binaryToInteger(String binary)
    {
        char[] numbers = binary.toCharArray();
        int result = 0;
        for(int i=numbers.length - 1; i>=0; i--)
            if(numbers[i]=='1')
                result += Math.pow(2, (numbers.length-i - 1));
        return result;
    }
}

```

Main Method Class –

```

package elGamal;

import java.lang.*;
import java.io.*;
import java.util.*;
import java.math.BigInteger;

public class ElGamal{

    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        KeyGeneration k=new KeyGeneration();
        k.generatekeys();
        System.out.println("\n***Encryption Has Started***");
        Encryption E=new Encryption();
        long encstart=System.currentTimeMillis();
        E.Encrypt(k.getp(),k.getg(),k.getb(),k.getSecretkey());
        long encend=System.currentTimeMillis();
        System.out.println("\n\n***ENCRYPTION DONE***");
        Decryption d=new Decryption();
        long start=System.currentTimeMillis();
        d.decrypt(k.getp(),k.getSecretkey(), null);
        long end=System.currentTimeMillis();
        long diff=end-start;
        long diff2=encend-encstart;
        System.out.println("\n\n"+diff2+"\tMilli Seconds for
Encryption"+" \n"+diff+"\tMilli Seconds for Decryption");
    }
}

```

5. Results

Statement file that is to be encrypted-

```
input - Notepad
File Edit View

Account Number 2679 5654 9878 0348
Name Pratham Rao
Last 3 months Transaction

No. of Transactions 3

transaction_id    message          amount    date
1000001           electricity      5000      10 oct 2022
1000023           zomato          300       11 nov 2022
1000254           bbqnation       100000    12 nov 2022
```

Encryption process stats-

```
Problems Javadoc Declaration Console × Coverage
<terminated> ElGamal [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (Nov 17, 2022, 11:20:19 PM – 11:20:20 PM) [pid: 9112]

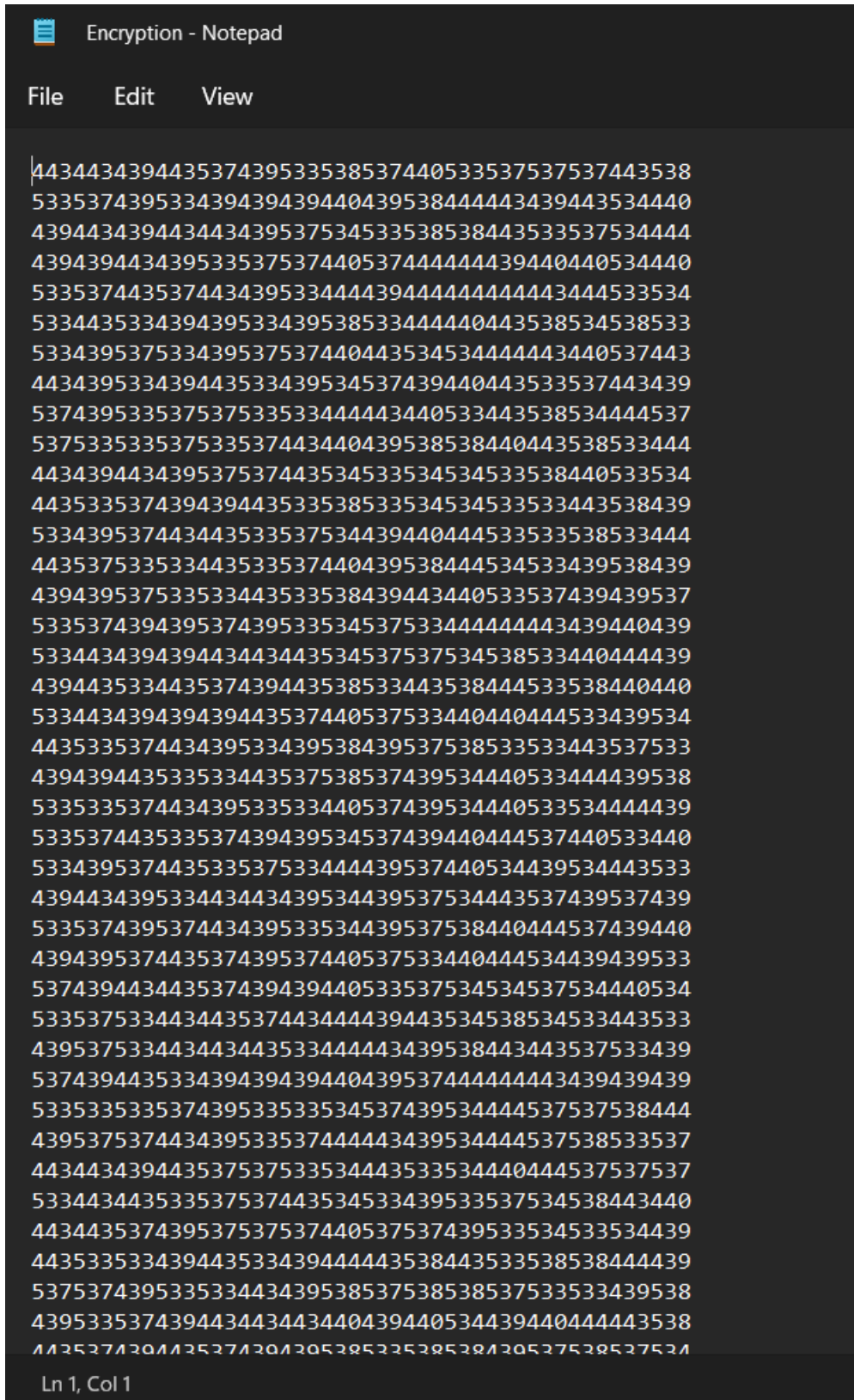
Public keys:: P = 157 G = 15 B = 5
Secret Key :: 47

***Encryption Has Started***

***ENCRYPTION DONE***

59      Milli Seconds for Encryption
52      Milli Seconds for Decryption
```


Encrypted file



```
Encryption - Notepad
File Edit View
443443439443537439533538537440533537537537443538
533537439533439439439440439538444443439443534440
439443439443443439537534533538538443533537534444
439439443439533537537440537444444439440440534440
53353744353744343953344443944444444443444533534
533443533439439533439538533444440443538534538533
533439537533439537537440443534534444443440537443
443439533439443533439534537439440443533537443439
537439533537537533533444443440533443538534444537
537533533537533537443440439538538440443538533444
443439443439537537443534533534534533538440533534
443533537439439443533538533534534533533443538439
533439537443443533537534439440444533533538533444
443537533533443533537440439538444534533439538439
439439537533533443533538439443440533537439439537
53353743943953743953353453753344444443439440439
533443439439443443443534537537534538533440444439
439443533443537439443538533443538444533538440440
533443439439439443537440537533440440444533439534
443533537443439533439538439537538533533443537533
439439443533533443537538537439534440533444439538
533533537443439533533440537439534440533534444439
533537443533537439439534537439440444537440533440
533439537443533537533444439537440534439534443533
439443439533443443439534439537534443537439537439
533537439537443439533534439537538440444537439440
439439537443537439537440537533440444534439439533
537439443443537439439440533537534534537534440534
533537533443443537443444439443534538534533443533
439537533443443443533444443439538443443537533439
537439443533439439439440439537444444443439439439
533533533537439533533534537439534444537537538444
439537537443439533537444443439534444537538533537
443443439443537537533534443533534440444537537537
533443443533537537443534533439533537534538443440
443443537439537537537440537537439533534533534439
44353353343944353343944443538443533538538444439
537537439533533443439538537538538537533533439538
439533537439443443443440439440534439440444443538
4435374395335374394439533853385338537537537537537
```

Ln 1, Col 1

First encrypted file-

```
first_encryptpion - Notepad
File Edit View

Character= A Its ASCII IS= 65 C1 Value is = 1 C2 Value is = 65
Character= c Its ASCII IS= 99 C1 Value is = 1 C2 Value is = 99
Character= c Its ASCII IS= 99 C1 Value is = 1 C2 Value is = 99
Character= o Its ASCII IS= 111 C1 Value is = 1 C2 Value is = 111
Character= u Its ASCII IS= 117 C1 Value is = 1 C2 Value is = 117
Character= n Its ASCII IS= 110 C1 Value is = 1 C2 Value is = 110
Character= t Its ASCII IS= 116 C1 Value is = 1 C2 Value is = 116
Character= Its ASCII IS= 32 C1 Value is = 1 C2 Value is = 32
Character= N Its ASCII IS= 78 C1 Value is = 1 C2 Value is = 78
Character= u Its ASCII IS= 117 C1 Value is = 1 C2 Value is = 117
Character= m Its ASCII IS= 109 C1 Value is = 1 C2 Value is = 109
Character= b Its ASCII IS= 98 C1 Value is = 1 C2 Value is = 98
Character= e Its ASCII IS= 101 C1 Value is = 1 C2 Value is = 101
Character= r Its ASCII IS= 114 C1 Value is = 1 C2 Value is = 114
Character= Its ASCII IS= 32 C1 Value is = 1 C2 Value is = 32
Character= 2 Its ASCII IS= 50 C1 Value is = 1 C2 Value is = 50
Character= 6 Its ASCII IS= 54 C1 Value is = 1 C2 Value is = 54
Character= 7 Its ASCII IS= 55 C1 Value is = 1 C2 Value is = 55
Character= 9 Its ASCII IS= 57 C1 Value is = 1 C2 Value is = 57
Character= Its ASCII IS= 32 C1 Value is = 1 C2 Value is = 32
Character= 5 Its ASCII IS= 53 C1 Value is = 1 C2 Value is = 53
Character= 6 Its ASCII IS= 54 C1 Value is = 1 C2 Value is = 54
Character= 5 Its ASCII IS= 53 C1 Value is = 1 C2 Value is = 53
Character= 4 Its ASCII IS= 52 C1 Value is = 1 C2 Value is = 52
Character= Its ASCII IS= 32 C1 Value is = 1 C2 Value is = 32
Character= 9 Its ASCII IS= 57 C1 Value is = 1 C2 Value is = 57
Character= 8 Its ASCII IS= 56 C1 Value is = 1 C2 Value is = 56
Character= 7 Its ASCII IS= 55 C1 Value is = 1 C2 Value is = 55
Character= 8 Its ASCII IS= 56 C1 Value is = 1 C2 Value is = 56
Character= Its ASCII IS= 32 C1 Value is = 1 C2 Value is = 32
Character= 0 Its ASCII IS= 48 C1 Value is = 1 C2 Value is = 48
Character= 3 Its ASCII IS= 51 C1 Value is = 1 C2 Value is = 51
Character= 4 Its ASCII IS= 52 C1 Value is = 1 C2 Value is = 52
Character= 8 Its ASCII IS= 56 C1 Value is = 1 C2 Value is = 56
Character=
    Its ASCII IS= 13 C1 Value is = 1 C2 Value is = 13
Character=
    Its ASCII IS= 10 C1 Value is = 1 C2 Value is = 10
Character= N Its ASCII IS= 78 C1 Value is = 1 C2 Value is = 78
Character= a Its ASCII IS= 97 C1 Value is = 1 C2 Value is = 97

Ln 1, Col 1
```

Second encrypted file-

```
second_encryption - Notepad

File Edit View

C1=1 C2=65 Binary C1 00000001 Binary C2 01000001
C1=1 C2=99 Binary C1 00000001 Binary C2 01100011
C1=1 C2=99 Binary C1 00000001 Binary C2 01100011
C1=1 C2=111 Binary C1 00000001 Binary C2 01101111
C1=1 C2=117 Binary C1 00000001 Binary C2 01110101
C1=1 C2=110 Binary C1 00000001 Binary C2 01101110
C1=1 C2=116 Binary C1 00000001 Binary C2 01110100
C1=1 C2=32 Binary C1 00000001 Binary C2 00100000
C1=1 C2=78 Binary C1 00000001 Binary C2 01001110
C1=1 C2=117 Binary C1 00000001 Binary C2 01110101
C1=1 C2=109 Binary C1 00000001 Binary C2 01101101
C1=1 C2=98 Binary C1 00000001 Binary C2 01100010
C1=1 C2=101 Binary C1 00000001 Binary C2 01100101
C1=1 C2=114 Binary C1 00000001 Binary C2 01110010
C1=1 C2=32 Binary C1 00000001 Binary C2 00100000
C1=1 C2=50 Binary C1 00000001 Binary C2 00110010
C1=1 C2=54 Binary C1 00000001 Binary C2 00110110
C1=1 C2=55 Binary C1 00000001 Binary C2 00110111
C1=1 C2=57 Binary C1 00000001 Binary C2 00111001
C1=1 C2=32 Binary C1 00000001 Binary C2 00100000
C1=1 C2=53 Binary C1 00000001 Binary C2 00110101
C1=1 C2=54 Binary C1 00000001 Binary C2 00110110
C1=1 C2=53 Binary C1 00000001 Binary C2 00110101
C1=1 C2=52 Binary C1 00000001 Binary C2 00110100
C1=1 C2=32 Binary C1 00000001 Binary C2 00100000
C1=1 C2=57 Binary C1 00000001 Binary C2 00111001
C1=1 C2=56 Binary C1 00000001 Binary C2 00111000
C1=1 C2=55 Binary C1 00000001 Binary C2 00110111
C1=1 C2=56 Binary C1 00000001 Binary C2 00111000
C1=1 C2=32 Binary C1 00000001 Binary C2 00100000
C1=1 C2=48 Binary C1 00000001 Binary C2 00110000
C1=1 C2=51 Binary C1 00000001 Binary C2 00110011
C1=1 C2=52 Binary C1 00000001 Binary C2 00110100
C1=1 C2=56 Binary C1 00000001 Binary C2 00111000
C1=1 C2=13 Binary C1 00000001 Binary C2 00001101
C1=1 C2=10 Binary C1 00000001 Binary C2 00001010
C1=1 C2=78 Binary C1 00000001 Binary C2 01001110
C1=1 C2=97 Binary C1 00000001 Binary C2 01100001
C1=1 C2=109 Binary C1 00000001 Binary C2 01101101
C1=1 C2=101 Binary C1 00000001 Binary C2 01100101

Ln 1, Col 1
```

Decryption Input-

```
Encryption - Notepad
File Edit View

443443439443537439533538537440533537537537443538
533537439533439439439440439538444443439443534440
439443439443443439537534533538538443533537534444
439439443439533537537440537444444439440440534440
53353744353744343953344443944444444443444533534
533443533439439533439538533444440443538534538533
53343953753343953753744044353453444443440537443
443439533439443533439534537439440443533537443439
53743953353753753353344443440533443538534444537
537533533537533537443440439538538440443538533444
443439443439537537443534533534534533538440533534
443533537439439443533538533534534533533443538439
533439537443443533537534439440444533533538533444
443537533533443533537440439538444534533439538439
439439537533533443533538439443440533537439439537
53353743943953743953353453753344444443439440439
53344343943944344344353453753753453853344044439
439443533443537439443538533443538444533538440440
533443439439439443537440537533440440444533439534
443533537443439533439538439537538533533443537533
439439443533533443537538537439534440533444439538
53353353744343953353344053743953444053353444439
533537443533537439439534537439440444537440533440
533439537443533537533444439537440534439534443533
439443439533443443439534439537534443537439537439
533537439537443439533534439537538440444537439440
439439537443537439537440537533440444534439439533
537439443443537439439440533537534534537534440534
533537533443443537443444439443534538534533443533
439537533443443443533444443439538443443537533439
537439443533439439439440439537444444443439439439
533533533537439533533534537439534444537537538444
439537537443439533537444443439534444537538533537
443443439443537537533534443533534440444537537537
533443443533537537443534533439533537534538443440
443443537439537537537440537537439533534533534439
4435335334394435334394444353844353353853844439
537537439533533443439538537538538537533533439538
439533537439443443443440439440534439440444443538
4435374394435337439439538533538538538439537537534
```

Ln 1, Col 1

First decrypted file-

```
first_decrypton - Notepad
File Edit View

INPUT IS = 443443439443537439533538537440533537537537443538 BINARY - 00000001 01000001
INPUT IS = 533537439533439439439440439538444443439443534440 BINARY - 00000001 01100011
INPUT IS = 439443439443443439537534533538538443533537534444 BINARY - 00000001 01100011
INPUT IS = 439439443439533537537440537444444439440440534440 BINARY - 00000001 01101111
INPUT IS = 53353744353744343953344443944444444443444533534 BINARY - 00000001 01110101
INPUT IS = 533443533439439533439538533444440443538534538533 BINARY - 00000001 01101110
INPUT IS = 53343953753343953753744044353453444443440537443 BINARY - 00000001 01110100
INPUT IS = 443439533439443533439534537439440443533537443439 BINARY - 00000001 00100000
INPUT IS = 537439533537537533533444443440533443538534444537 BINARY - 00000001 01001110
INPUT IS = 537533533537533537443440439538538440443538533444 BINARY - 00000001 01110101
INPUT IS = 443439443439537537443534533534533538440533534 BINARY - 00000001 01101101
INPUT IS = 443533537439439443533538533534534533533443538439 BINARY - 00000001 01100010
INPUT IS = 533439537443443533537534439440444533533538533444 BINARY - 00000001 01100101
INPUT IS = 443537533533443533537440439538444534533439538439 BINARY - 00000001 01110010
INPUT IS = 439439537533533443533538439443440533537439439537 BINARY - 00000001 00100000
INPUT IS = 53353743943953743953353453753344444443439440439 BINARY - 00000001 00110010
INPUT IS = 533443439439443443443534537537534538533440444439 BINARY - 00000001 00110110
INPUT IS = 439443533443537439443538533443538444533538440440 BINARY - 00000001 00110111
INPUT IS = 53344343943944353744053753344044044533439534 BINARY - 00000001 00111001
INPUT IS = 443533537443439533439538439537538533533443537533 BINARY - 00000001 00100000
INPUT IS = 439439443533533443537538537439534440533444439538 BINARY - 00000001 00110101
INPUT IS = 533533537443439533533440537439534440533534444439 BINARY - 00000001 00110110
INPUT IS = 533537443533537439439534537439440444537440533440 BINARY - 00000001 00110101
INPUT IS = 533439537443533537533444439537440534439534443533 BINARY - 00000001 00110100
INPUT IS = 439443439533443443439534439537534443537439537439 BINARY - 00000001 00100000
INPUT IS = 533537439537443439533534439537538440444537439440 BINARY - 00000001 00111001
INPUT IS = 439439537443537439537440537533440444534439439533 BINARY - 00000001 00111000
INPUT IS = 537439443443537439439440533537534534537534440534 BINARY - 00000001 00110111
INPUT IS = 533537533443443537443444439443534538534533443533 BINARY - 00000001 00111000
INPUT IS = 439537533443443443533444443439538443443537533439 BINARY - 00000001 00100000
INPUT IS = 537439443533439439439440439537444444443439439439 BINARY - 00000001 00110000
INPUT IS = 533533533537439533533534537439534444537537538444 BINARY - 00000001 00110011
INPUT IS = 439537537443439533537444443439534444537538533537 BINARY - 00000001 00110100
INPUT IS = 443443439443537537533534443533534440444537537537 BINARY - 00000001 00111000
INPUT IS = 533443443533537537443534533439533537534538443440 BINARY - 00000001 00001101
INPUT IS = 443443537439537537537440537537439533534533534439 BINARY - 00000001 00001010
INPUT IS = 443533533439443533439444443538443533538538444439 BINARY - 00000001 01001110
INPUT IS = 537537439533533443439538537538538537533533439538 BINARY - 00000001 01100001
INPUT IS = 4395335374394434434440439440534439440444443538 BINARY - 00000001 01101101
INPUT IS = 443537439443537439439538533538538439537538537534 BINARY - 00000001 01100101
```

Ln 1, Col 1

Second decrypted file-

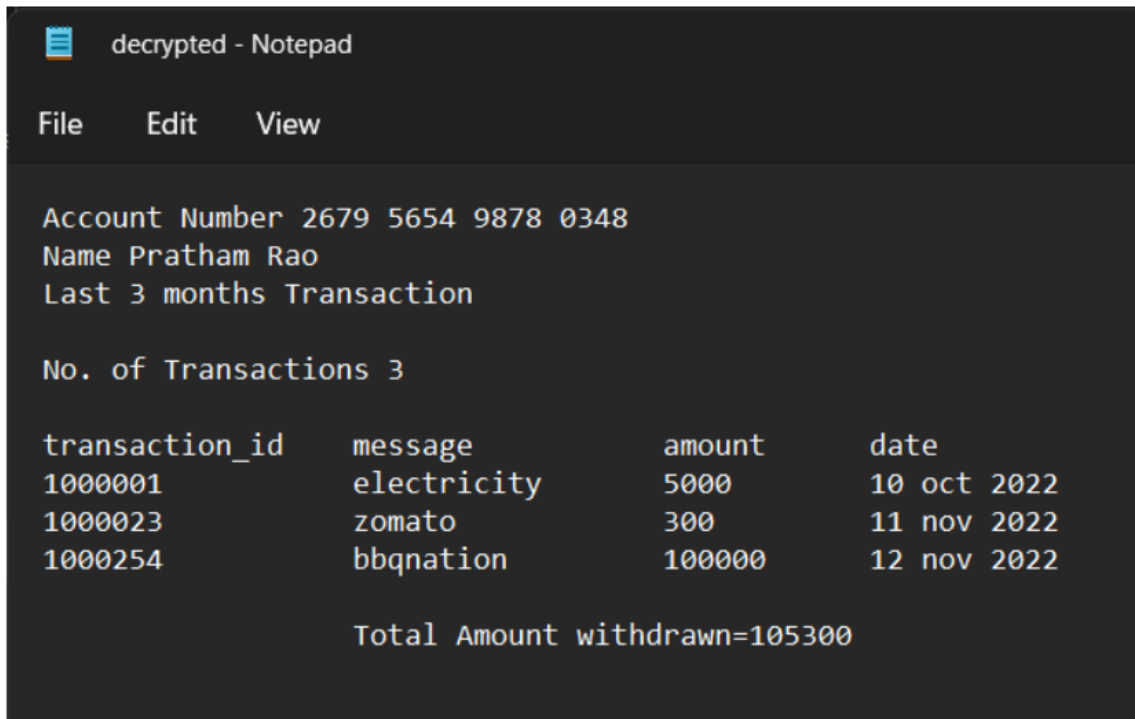
```
second_decryption - Notepad

File Edit View

|BINARY 00000001 01000001 C1=1 C2=65
BINARY 00000001 01100011 C1=1 C2=99
BINARY 00000001 01100011 C1=1 C2=99
BINARY 00000001 01101111 C1=1 C2=111
BINARY 00000001 01110101 C1=1 C2=117
BINARY 00000001 01101110 C1=1 C2=110
BINARY 00000001 01110100 C1=1 C2=116
BINARY 00000001 00100000 C1=1 C2=32
BINARY 00000001 01001110 C1=1 C2=78
BINARY 00000001 01110101 C1=1 C2=117
BINARY 00000001 01101101 C1=1 C2=109
BINARY 00000001 01100010 C1=1 C2=98
BINARY 00000001 01100101 C1=1 C2=101
BINARY 00000001 01110010 C1=1 C2=114
BINARY 00000001 00100000 C1=1 C2=32
BINARY 00000001 00110010 C1=1 C2=50
BINARY 00000001 00110110 C1=1 C2=54
BINARY 00000001 00110111 C1=1 C2=55
BINARY 00000001 00111001 C1=1 C2=57
BINARY 00000001 00100000 C1=1 C2=32
BINARY 00000001 00110101 C1=1 C2=53
BINARY 00000001 00110110 C1=1 C2=54
BINARY 00000001 00110101 C1=1 C2=53
BINARY 00000001 00110100 C1=1 C2=52
BINARY 00000001 00100000 C1=1 C2=32
BINARY 00000001 00111001 C1=1 C2=57
BINARY 00000001 00111000 C1=1 C2=56
BINARY 00000001 00110111 C1=1 C2=55
BINARY 00000001 00111000 C1=1 C2=56
BINARY 00000001 00100000 C1=1 C2=32
BINARY 00000001 00110000 C1=1 C2=48
BINARY 00000001 00110011 C1=1 C2=51
BINARY 00000001 00110100 C1=1 C2=52
BINARY 00000001 00111000 C1=1 C2=56
BINARY 00000001 00001101 C1=1 C2=13
BINARY 00000001 00001010 C1=1 C2=10
BINARY 00000001 01001110 C1=1 C2=78
BINARY 00000001 01100001 C1=1 C2=97
BINARY 00000001 01101101 C1=1 C2=109
BINARY 00000001 01100101 C1=1 C2=101

Ln 1, Col 1
```

Decryption output-



```
decrypted - Notepad

File Edit View

Account Number 2679 5654 9878 0348
Name Pratham Rao
Last 3 months Transaction

No. of Transactions 3

transaction_id    message          amount    date
1000001           electricity      5000      10 oct 2022
1000023           zomato          300       11 nov 2022
1000254           bbqnation       100000    12 nov 2022

Total Amount withdrawn=105300
```

6. Conclusion

In conclusion, with the help of Homomorphic and ElGamal Algorithm the banking application will enable the user to download the transaction statement in accordance with the account type and other user's requirements, while providing user authentication and security features during the e-bank transactions. ElGamal cryptosystem has a number of useful properties such as being semantically secure. This makes it a popular choice when developing implementations of procedures in which individuals wish to hide their identities in some way, such as secure bank e-transactions and recipient identity. Unlike, ancient cryptography which uses simple transposition and substitution methods, ElGamal cryptosystem uses different mathematical concepts to generate public and private key. And the security depends on the difficulty in computing the cyclic group.

Thus, for a more secured transaction we prefer to use ElGamal algorithm to secure the banking details.

7. Team Members' Contribution

Team Leader - Siddharth Kumar 20MIS0087

Register Number	Name	Contribution / Role in this Project
20MIS0085	Akash PS	Research paper [1] [2] [3] Decryption algorithm Decryption coding
20MIS0087	Siddharth Kumar	Research paper [4] [5] [6] Documentation Encryption algorithm Encryption coding
20MIS0124	Anjali Jha	Research paper [7] [8] [9] PPT, Homomorphic coding Elgamal coding
20MIS0131	Niharika Sharma	Research paper [10] [11] [12] PPT, Modules description Elgamal sample example Implementation
20MIS0163	Rose Mariya	Research paper [13] [14] [15] Conclusion, Implementation Modules identification, Elgamal sample example

References

- [1] Setyoko, Y. A., & Abdurrohman, M. (2017). SMS Banking Encryption Scheme. *2017 IEEE International Conference on Communication, Networks and Satellite (Comnetsat)*. <https://doi.org/10.1109/comnetsat.2017.8263569>
- [2] Xu, D., & Chen, W. (2010). 3G communication encryption algorithm based on ECC-Elgamal. *2010 2nd International Conference on Signal Processing Systems*. <https://doi.org/10.1109/icsps.2010.5555894>
- [3] Xiang, G., Yu, B., & Zhu, P. (2012). A algorithm of fully homomorphic encryption. *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*. <https://doi.org/10.1109/fskd.2012.6234023>

- [4] Huang, K., & Tso, R. (2012). A commutative encryption scheme based on elgamal encryption. *2012 International Conference on Information Security and Intelligent Control*. <https://doi.org/10.1109/isic.2012.6449730>
- [5] Wang, D., Guo, B., Shen, Y., Cheng, S.-J., & Lin, Y.-H. (2017). A faster fully homomorphic encryption scheme in Big Data. *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*(. <https://doi.org/10.1109/icbda.2017.8078836>
- [6] Chen, Z., Zhang, R., Yang, Y., & Li, Z. (2013). A homomorphic elgamal variant based on BGN's method. *2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. <https://doi.org/10.1109/cyberc.2013.10>
- [7] Sun, R., Peng, Q., & Tian, Y. (2017). A hybrid encryption scheme based on the satisfiability problem. *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. <https://doi.org/10.1109/globalsip.2017.8309184>
- [8] Bhanumathi, S., & Sakthivel, P. (2013). A new model for Privacy Preserving Multiparty Collaborative Data Mining. *2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT)*. <https://doi.org/10.1109/iccpct.2013.6529007>
- [9] Shishkov, B., & Janssen, M. (2016). Towards a service-oriented architecture for eVoting. *Proceedings of the Sixth International Symposium on Business Modeling and Software Design*. <https://doi.org/10.5220/0006223601870195>
- [10] Zhang, P., Sun, X., Wang, T., Gu, S., Yu, J., & Xie, W. (2016). An accelerated fully homomorphic encryption scheme over the integers. *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)*. <https://doi.org/10.1109/ccis.2016.7790295>
- [11] Nan, C., Anle, L., Gu, C., & Zhu, Y. (2010). Automated security proof of the elgamal encryption scheme. *2010 2nd International Conference on Future Computer and Communication*. <https://doi.org/10.1109/icfcc.2010.5497804>
- [12] Zhang, J., Yang, Y., Chen, Y., Chen, J., & Zhang, Q. (2017). A general framework to design secure cloud storage protocol using homomorphic encryption scheme. *Computer Networks*, 129, 37–50. <https://doi.org/10.1016/j.comnet.2017.08.019>
- [13] B.D., P., & Soyjaudah, K. M. (2012). Analysis and comparison of fully layered image encryption techniques and partial image encryption techniques. *Wireless Networks and Computational Intelligence*, 599–604. https://doi.org/10.1007/978-3-642-31686-9_70

- [14] Jain, A., & Soni, S. (2017). Visual cryptography and image processing based approach for secure transactions in banking sector. *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*.
<https://doi.org/10.1109/tel-net.2017.8343545>
- [15] Muthi Reddy, P., Manjula, S. H., & Venugopal, K. R. (2018). Secured privacy data using multi key encryption in cloud storage. *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*.
<https://doi.org/10.1109/eait.2018.8470399>