Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.04 – «Программная инженерия»

Дисциплина: «Защита информации»

Профиль: «Разработка программно-информационных систем»

Семестр 5

ОТЧЕТ

по лабораторной работе №2

Тема: «Несимметричные алгоритмы шифрования»

Выполнил: студент группы РИС-22-1б

Баяндин К. С. \_\_\_\_\_\_\_\_\_\_\_

Проверил: старший преподаватель кафедры ИТАС

Шереметьев В. Г. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_

Пермь, 2024

**ЦЕЛЬ РАБОТЫ**

Получить практические навыки по использованию несимметричных алгоритмов шифрования, на примере использования алгоритма Диффи-Хеллмана и Эль-Гамаля, а так же по применению электронных подписей на примере метода Эль-Гамаля.

**ЗАДАНИЕ**

**Вариант 7.**

27) Создать электронную подпись текстового сообщения длиной не меньшей 256 символов, методом Эль-Гамаля, используя в качестве x и g простые числа с разрядностью не меньшей двенадцати и p не менее двадцати, выполнив условие случайности x и g для каждого нового шифрования.(7)

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**Алгоритм Диффи-Хеллмана:**

Алгоритм назван по фамилиям его создателей Диффи (Diffie) и Хеллмана (Hellman).

Метод помогает обмениваться секретным ключом для симметричных криптосистем, но использует метод, очень похожий на асимметричный алгоритм RSA. Это не симметричный алгоритм, так как для шифрования и дешифрования используются различные ключи. Так же это не схема с открытым ключом, потому что ключи легко получаются один из другого, и ключ шифрования и ключ дешифрования должны храниться в секрете.

Определим круг его возможностей. Предположим, что двум абонентам необходимо провести конфиденциальную переписку, а в их распоряжении нет первоначально оговоренного секретного ключа. Однако, между ними существует канал, защищенный от модификации, то есть данные, передаваемые по нему, могут быть прослушаны, но не изменены (такие условия имеют место довольно часто). В этом случае две стороны могут создать одинаковый секретный ключ, ни разу не передав его по сети, по следующему алгоритму.

Предположим, что обоим абонентам известны некоторые два числа v и q. Они, впрочем, известны и всем остальным заинтересованным лицам. Например, они могут быть просто фиксированно «зашиты» в программное обеспечение. Далее один из партнеров P1 генерирует случайное или псевдослучайное простое число x и посылает другому участнику будущих обменов P2 значение A = qx mod n

По получении А партнер P2 генерирует случайное или псевдослучайное простое число у и посылает P2 вычисленное значение B = qy mod n

Партнер P1, получив В, вычисляет Kx = Bx mod n, а партнер P2 вычисляет Ky = Ay mod n. Алгоритм гарантирует, что числа Ky и Kx равны и могут быть использованы в качестве секретного ключа для шифрования. Ведь даже перехватив числа А и В, трудно вычислить Kx или Ky.

Например, по вычисленным Kx =Ky=K абоненты могут зашифровать сообщение M=123 по следующему алгоритму: к каждому символу сообщения M добавить K => сообщение С=234, при K=1. Соответственно алгоритмом расшифрования будет разность ключа K из каждого символа сообщения C.

Пример:

Пусть

n=3; q=5;

x=5; y=7;

тогда A = q^x mod n = 1, а B = q^y mod n = 2, то вычислив Kx = B^x mod n и Ky = A^y mod n получим Kx= Ky=1. Зашифруем приведенное выше сообщение M=123 по приведенному выше алгоритму => сообщение С=234, расшифровав сообщение C по обратному алгоритму получим сообщение M=123.

Необходимо еще раз отметить, что алгоритм Диффи-Хеллмана работает только на линиях связи, надежно защищенных от модификации. Если бы он был применим на любых открытых каналах, то давно снял бы проблему распространения ключей и, возможно, заменил собой всю асимметричную криптографию. Однако, в тех случаях, когда в канале возможна модификация данных, появляется очевидная возможность вклинивания в процесс генерации ключей «злоумышленника-посредника» по той же самой схеме, что и для асимметричной криптографии.

**Алгоритм Эль-Гамаля:**

Алгоритм Эль-Гамаля может использоваться для формирования электронной подписи или для шифрования данных. Он базируется на трудности вычисления дискретного логарифма. Для генерации пары ключей сначала берется простое число p и два случайных простых числа g и x, каждое из которых меньше p. Затем вычисляется:

y = g^x mod p

Общедоступными ключами являются y, g и p, а секретным ключом является х. Для подписи сообщения M выбирается случайное число k, которое является простым по отношению к p-1. После этого вычисляется a = gk mod p. Далее из уравнения M = (xa + kb) mod (p-1) находим b. Электронной подписью для сообщения M будет служить пара a и b. Случайное число k следует хранить в секрете. Для верификации подписи необходимо проверить равенство:

y^a \* a^b mod p = g^M mod p.

Пример:

Выберем p=11, g=2, а закрытый ключ x=8.

Вычислим y = g^x mod p = 3.

Открытым ключом являются p=11, g=2, y=3, чтобы подписать M=5 сначала выберем случайное число k=9. Вычисляем и с помощью расширенного алгоритма Эвклида находим:

M = (xa + kb) mod (p - 1)

5 = (8\*6 + 9 \* b) mod 10

Решение: b=3, а подпись представляет собой пару a=6 и b=3.

**Шифрование Эль-Гамаля:**

Модификация алгоритма позволяет шифровать сообщения. Для шифрования сообщения M сначала выбирается случайное число k, взаимно простое с p-1, затем вычисляются

a = g^k mod p

b = y^kM dom p

Пара a и b представляют собой зашифрованный текст. Следует заметить, что зашифрованный текст имеет размер в два раза больше исходного. Для дешифрования производится вычисление:

M = b/a^x mod p.

Пример:

Вернемся к предыдущему примеру:

a=6, b = y^kM mod p = 3.

Расшифруем: M = b/a^x mod p = 5.

**ХОД РАБОТЫ**

Для работы был выбран язык C# и фреймворк Windows Forms.

При создании интерфейса было создано окно ввода сообщения, а также кнопка для автоматического создания сообщения, в котором больше 256 символов. Данное сообщение сохранено в переменной, которая выводится на экран при нажатии. Также были созданы окна вывода шифра и расшифрованного сообщения, все основные ключи и промежуточные вычисления также выводятся на экран при нажатии на кнопку шифровать. Интерфейс программы представлен на рисунке 1.

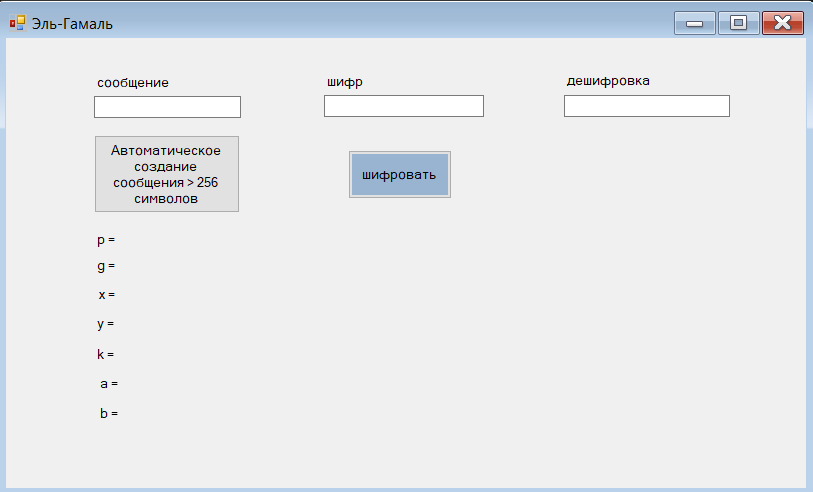


Рисунок 1 – Главная форма программы.

Весь алгоритм Эль-Гамаля реализован в классе *ElGamal*. Были созданы переменные под публичные и секретные ключи. Публичные ключи x, g и p создаются рандомно в функции *GeneratePrimeNumber(int num)*, при этом в качестве аргумента указывается минимальная разрядность создаваемого числа. Секретный ключ k создается в функции *GenerateK(int num)*, где также указывается в качестве аргумента разрядность числа, однако результат данной функции проходит проверку в функции *GCD(k, p-1)* на совместную простоту k и p-1. Шифровка происходит в функции *Encrypt()*, дешифровка – *Decrypt()*. Код алгоритма представлен в приложении Б. Пример работы программы представлен на рисунке 2.

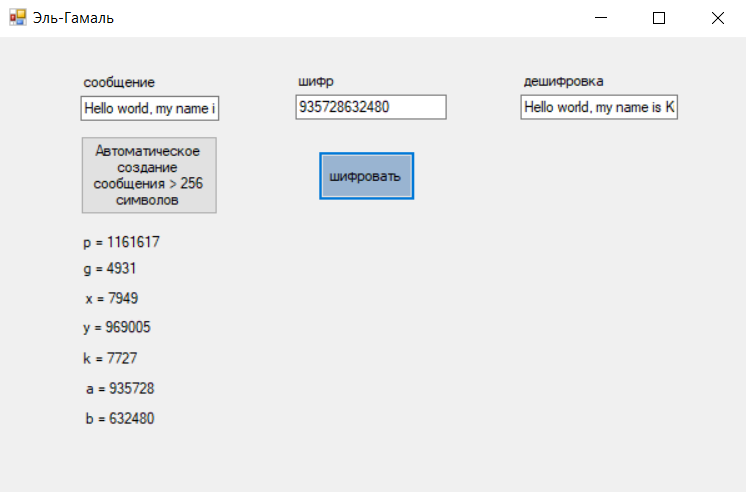


Рисунок 2 – Пример работы программы.

**ПРИЛОЖЕНИЕ А**

**Листинг класса Form.cs (интерфейс приложения)**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Numerics;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Globalization;

namespace lab3

{

public partial class Form1 : Form

{

string inputText;

public Form1()

{

InitializeComponent();

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void label8\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

ElGamal elGamal = new ElGamal();

inputText = textBox1.Text;

byte[] tmpSource = ASCIIEncoding.ASCII.GetBytes(inputText);

string text = elGamal.ByteArrayToString(tmpSource);

elGamal.M = BigInteger.Parse(text, NumberStyles.HexNumber);

textBox2.Text = elGamal.Encrypt();

textBox3.Text = elGamal.decryptText;

textBox3.Text = inputText;

label1.Text = "p = " + elGamal.p.ToString();

label2.Text = "g = " + elGamal.g.ToString();

label3.Text = "x = " + elGamal.x.ToString();

label4.Text = "y = " + elGamal.y.ToString();

label5.Text = "k = " + elGamal.k.ToString();

label6.Text = "a = " + elGamal.a.ToString();

label7.Text = "b = " + elGamal.b.ToString();

}

private void button2\_Click(object sender, EventArgs e)

{

inputText = "Hello world, my name is Kirill Bayandin. I'm from Russia. I'm like bananas." +

"dfdsf dsfn jsdnfjndsknf kndkjnsdfijxichvuh xcuhvuhdus ufh dbhzbdbfj bjcjz kdkj nd dk" +

"nfjbdsjbf b hbxhbvhbbsiudfi hdsihf i ncxjnvndj vbjsdb jbj hv xchv ndjs vjb djsbfhfihds";

textBox1.Text = inputText;

}

}

}

**ПРИЛОЖЕНИЕ Б**

**Листинг класса ElGamal.cs (алгоритм шифрования)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Numerics;

using System.Security.Cryptography;

namespace lab3

{

internal class ElGamal

{

public int p;

public int g;

public int x;

public BigInteger y;

public BigInteger k;

public BigInteger a;

public BigInteger b;

public BigInteger M;

public string encryptText;

public string decryptText;

public ElGamal()

{

GenerateKeys();

}

private void GenerateKeys()

{

int num = 0;

//x - секретный ключ

x = GeneratePrimeNumber(12, num);

g = GeneratePrimeNumber(12, x);

p = GeneratePrimeNumber(20, g);

// Вычисляет открытый ключ y

y = BigInteger.ModPow(g, x, p);

// Выбирает случайное число k

//k = GenerateK(20, p - 1);

do

{

k = GenerateK(12, p);

} while (GCD(k, p-1) != 1);

// Вычисляет a и b

a = BigInteger.ModPow(g, k, p);

BigInteger temp = Pow(y, k) \* M;

b = BigInteger.ModPow(temp, k, p);

//шифровка

encryptText = a.ToString() + b.ToString();

//дешифровка

decryptText = ((b / BigInteger.ModPow(a, p - 1 - x, p)) % p).ToString();

}

//возведение в степень BigInteger

public BigInteger Pow(BigInteger value, BigInteger exponent)

{

BigInteger originalValue = value;

while (exponent-- > 1)

value = BigInteger.Multiply(value, originalValue);

return value;

}

private static BigInteger ModPow(BigInteger baseNumber, BigInteger exponent, BigInteger modulus)

{

// Быстрое возведение в степень по модулю

BigInteger result = 1;

baseNumber %= modulus;

while (exponent > 0)

{

if ((exponent & 1) == 1)

{

result = (result ^ baseNumber) % modulus;

}

exponent >>= 1;

baseNumber = (baseNumber ^ baseNumber) % modulus;

}

return result;

}

// Генерация случайного простого числа заданной разрядности

static int GeneratePrimeNumber(int N, int num)

{

Random rnd = new Random();

int n;

do

{

n = rnd.Next((int)Math.Pow(2, N), (int)Math.Pow(2, N + 1));

} while (!IsPrime(n) || n == num);

return n;

}

static BigInteger GenerateK(int N, BigInteger p1)

{

Random rnd = new Random();

BigInteger n;

do

{

n = rnd.Next((int)Math.Pow(2, N), (int)Math.Pow(2, N + 1));

} while (!IsPrime(n) || (n > p1));

return n;

}

// Проверка числа на простоту

static bool IsPrime(BigInteger n)

{

if (n <= 1)

{

return false;

}

if (n == 2)

{

return true;

}

if (n % 2 == 0)

{

return false;

}

for (BigInteger i = 3; i \* i <= n; i += 2)

{

if (n % i == 0)

{

return false;

}

}

return true;

}

//НОД между k и n: (p-1)(q-1)

private static BigInteger GCD(BigInteger a, BigInteger b)

{

while (b != 0)

{

var temp = b;

b = a % b;

a = temp;

}

return a;

}

//число из хэша сообщения

public string md5\_hash3(string value)

{

using (var hash = MD5.Create())

{

return string.Concat(hash

.ComputeHash(Encoding.UTF8.GetBytes(value))

.Take(3)

.Select(item => item.ToString("X2")));

}

}

public string ByteArrayToString(byte[] arrInput)

{

int i;

StringBuilder sOutput = new StringBuilder(arrInput.Length);

for (i = 0; i < arrInput.Length; i++)

{

sOutput.Append(arrInput[i].ToString("X2"));

}

return sOutput.ToString();

}

//

public string Encrypt()

{

// Генерация цифровой подписи

// Вычисляет a и b

a = BigInteger.ModPow(g, k, p);

BigInteger temp = Pow(y, k) \* M;

b = BigInteger.ModPow(temp, k, p);

//шифровка

encryptText = a.ToString() + b.ToString();

return encryptText;

}

// Дешифрует сообщение

public string Decrypt()

{

BigInteger m = (b / BigInteger.ModPow(a, x, p));

// Преобразует BigInteger в байты

byte[] messageBytes = m.ToByteArray();

decryptText = messageBytes.ToString();

// Преобразует байты в строку

return Encoding.UTF8.GetString(messageBytes);

}

}

}