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 №3

Тема: «Несимметричные алгоритмы шифрования»

Выполнил: студент группы РИС-22-1б

Поважный В. Е. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент 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 №21**. Выполнить шифрование текстового файла методом Диффи-Хеллмана, используя в качестве x и y простые числа с разрядностью не меньшей двадцати, выполнив условие случайности x и y для каждого нового шифрования и используя в алгоритме шифрования функцию тангенса.

**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**

На рисунке 1 представлена главная форма программы.

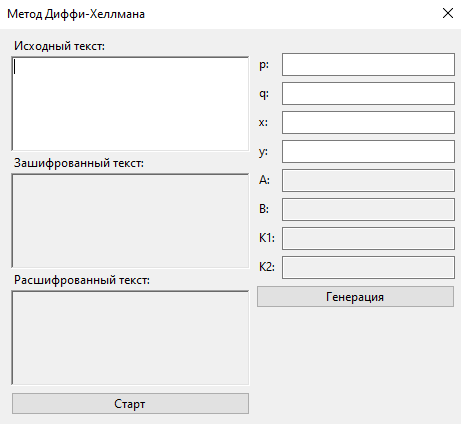


Рисунок 1 – Главная форма программы.

Пример работы программы представлен на рисунке 2.

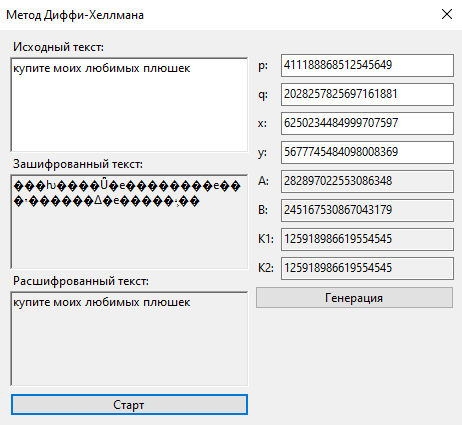


Рисунок 2 – Пример работы программы.

**ПРИЛОЖЕНИЕ**

**Листинг класса программы**

using System;

using System;

using System.IO;

using System.Numerics;

using System.Security.Cryptography;

using System.Security.Cryptography.Pkcs;

using System.Text;

using System.Windows.Forms;

namespace InformationSecureLabs

{

public partial class lab3 : Form

{

private int cap = 64;

private BigInteger p;

private BigInteger g;

private BigInteger x;

private BigInteger y;

private BigInteger A;

private BigInteger B;

private BigInteger Kx;

private BigInteger Ky;

public lab3()

{

InitializeComponent();

}

private BigInteger GenerateRandomPrime(int bits)

{

// Используем криптографически стойкий генератор случайных чисел

using (var rng = new RNGCryptoServiceProvider())

while (true)

{

byte[] bytes = new byte[bits / 8];

rng.GetBytes(bytes);

BigInteger candidate = new BigInteger(bytes);

// Преобразование числа в положительное

candidate = BigInteger.Abs(candidate);

if (IsProbablePrime(candidate))

return candidate;

}

}

// Проверка числа на простоту

private bool IsProbablePrime(BigInteger number, int certainty = 10)

{

if (number < 2)

return false;

if (number == 2 || number == 3)

return true;

if (number % 2 == 0)

return false;

BigInteger d = number - 1;

int s = 0;

while (d % 2 == 0)

{

d /= 2;

s += 1;

}

Random rng = new Random();

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

{

BigInteger a = RandomBigInteger(2, number - 2, rng);

BigInteger x = BigInteger.ModPow(a, d, number);

if (x == 1 || x == number - 1)

continue;

for (int r = 1; r < s; r++)

{

x = BigInteger.ModPow(x, 2, number);

if (x == 1)

return false;

if (x == number - 1)

break;

}

if (x != number - 1)

return false;

}

return true;

}

private BigInteger RandomBigInteger(BigInteger minValue, BigInteger maxValue, Random rng)

{

byte[] bytes = new byte[maxValue.ToByteArray().Length];

rng.NextBytes(bytes);

return new BigInteger(bytes) % (maxValue - minValue) + minValue;

}

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

{

p = GenerateRandomPrime(cap);

g = GenerateRandomPrime(cap);

x = GenerateRandomPrime(cap);

y = GenerateRandomPrime(cap);

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

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

Kx = BigInteger.ModPow(B, x, p);

Ky = BigInteger.ModPow(A, y, p);

textBox1.Text = p.ToString();

textBox2.Text = g.ToString();

textBox3.Text = x.ToString();

textBox4.Text = y.ToString();

textBox5.Text = A.ToString();

textBox6.Text = B.ToString();

textBox7.Text = Kx.ToString();

textBox8.Text = Ky.ToString();

}

private void GenerateKeys()

{

p = Convert.ToInt32(textBox1.Text);

g = Convert.ToInt32(textBox2.Text);

x = Convert.ToInt32(textBox3.Text);

y = Convert.ToInt32(textBox4.Text);

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

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

Kx = BigInteger.ModPow(B, x, p);

Ky = BigInteger.ModPow(A, y, p);

textBox5.Text = A.ToString();

textBox6.Text = B.ToString();

textBox7.Text = Kx.ToString();

textBox8.Text = Ky.ToString();

}

private void Encrypt(double key)

{

byte[] fileBytes = Encoding.UTF8.GetBytes(richTextBox1.Text);

byte[] encryptedBytes = new byte[fileBytes.Length];

for (int i = 0; i < fileBytes.Length; i++)

encryptedBytes[i] = (byte)(fileBytes[i] ^ (byte)(key \* 1000) % 256);

richTextBox2.Text = Encoding.UTF8.GetString(encryptedBytes);

File.WriteAllBytes("crypted.txt", encryptedBytes);

}

private void Decrypt(double key)

{

byte[] fileBytes = File.ReadAllBytes("crypted.txt");

byte[] decryptedBytes = new byte[fileBytes.Length];

for (int i = 0; i < fileBytes.Length; i++)

decryptedBytes[i] = (byte)(fileBytes[i] ^ (byte)(key \* 1000) % 256);

richTextBox3.Text = Encoding.UTF8.GetString(decryptedBytes);

File.WriteAllBytes("decrypted.txt", decryptedBytes);

}

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

{

if (richTextBox1.Text == "")

{

MessageBox.Show("Сообщение не может быть пустым");

return;

}

if (textBox5.Text == "")

GenerateKeys();

double encryptionKey = Math.Tan((double)(Ky % int.MaxValue));

double dencryptionKey = Math.Tan((double)(Kx % int.MaxValue));

Encrypt(encryptionKey);

Decrypt(dencryptionKey);

}

}

}