Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Отчет**

«

»

Выполнил:

студент 3 курса 4 группы

специальности ПОИТ

Чернявский А. Л.

Минск 2020

**Цель**: изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров RSA и Эль-Гамаля (рассчитана на 4 часа аудиторных занятий).

**Задачи**:

1.Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости асимметричных шифров RSA и Эль-Гамаля.

2. Разработать приложение для реализации асимметричного зашифрования/расшифрования на основе алгоритмов RSA и Эль-Гамаля.

3. Выполнить анализ криптостойкости асимметричных шифров RSA и Эль-Гамаля.

4.Оценить скорость зашифрования/расшифрования реализованных шифров.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Практическое задание**:

1. С помощью простого консольного приложения составить табличную

или графическую форму зависимости времени вычисления параметра у,

функционально заданного выражением вида:

**у = ax mod n**,

от параметров: а (десятичные числа от 5 до 35; можно взять 1 или 2 числа), х (числа, желательно – простые, из диапазона от 103 до 10100; для примера взять 5-10 чисел, равномерно распределенных в указанном диапазоне), n (для примера взять числа, в двоичном виде состоящие из 1024 и 2048 бит).

2. Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться доступными библиотеками либо программными кодами. В основе вычислений – кодировочные таблицы Base64 и ASCII.

Приложение должно реализовывать следующие операции:

• зашифрование и расшифрование текстовых документов на основе алгоритмов RSA и Эль-Гамаля;

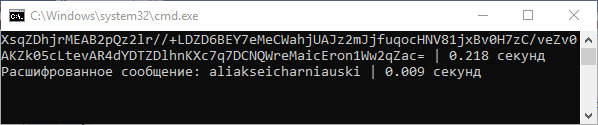
• определение времени выполнения операций.

Исходный текст для зашифрования – собственные фамилия, имя, отчество. Для численного представления блоков текста можно, в том числе, пользоваться указанными выше кодировочными таблицами. Ключевую информацию для обоих алгоритмов можно сгенерировать самостоятельно либо воспользоваться, например, одной из утилит криптографической библиотеки OpenSSL, с помощью которой, в частности, можно сгенерировать ключевую информацию для алгоритма RSA.

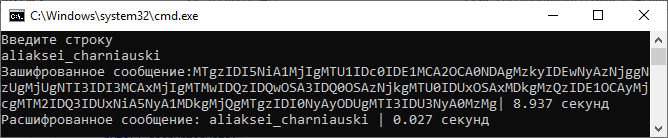
3. Используя примерно одинаковый порядок ключевой информации, оценить производительность обоих алгоритмов и относительное изменение объемов криптотекстов (по отношению к объемам открытых текстов).

**Результат выполнения работы приложения:**

RSA:



El-Gamal:



**Листинги**

**RSA**:

using System;

using System.Security.Cryptography;

using System.Diagnostics;

using System.Text;

namespace Lab8

{

internal class Program

{

private static void Main()

{

Stopwatch time = new Stopwatch();

UnicodeEncoding ByteConverter = new UnicodeEncoding();

RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

time.Start();

byte[] text = Encoding.UTF8.GetBytes("aliakseicharniauski");

byte[] crypted = RSAcl.Encryption(text, RSA.ExportParameters(false), false);

string cryptedText = Convert.ToBase64String(crypted);

time.Stop();

Console.WriteLine($"Зашифрованное сообщение: {cryptedText} | {(float)time.ElapsedMilliseconds / 1000} секунд");

time.Reset();

time.Start();

string decryptedText = RSAcl.Decryption(crypted, RSA.ExportParameters(true), false);

time.Stop();

Console.WriteLine($"Расшифрованное сообщение: {decryptedText} | {(float)time.ElapsedMilliseconds / 1000} секунд");

Console.ReadKey();

}

}

public class RSAcl

{

static public byte[] Encryption(byte[] Data, RSAParameters RSAKey, bool DoOAEPPadding)

{

try

{

byte[] encryptedData;

using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider(512))

{

RSA.ImportParameters(RSAKey);

encryptedData = RSA.Encrypt(Data, DoOAEPPadding);

}

return encryptedData;

}

catch (CryptographicException e)

{

Console.WriteLine(e.Message);

return null;

}

}

static public string Decryption(byte[] Data, RSAParameters RSAKey, bool DoOAEPPadding)

{

try

{

byte[] decryptedData;

using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider(512))

{

RSA.ImportParameters(RSAKey);

decryptedData = RSA.Decrypt(Data, DoOAEPPadding);

}

return Encoding.UTF8.GetString(decryptedData);

}

catch (CryptographicException e)

{

Console.WriteLine(e.ToString());

return null;

}

}

}

}

**El**-**Gamal**:

using System;

using System.Security.Cryptography;

using System.Diagnostics;

using System.Text;

using System.Linq;

namespace Lab10ElGam

{

internal class Program

{

private static void Main()

{

Stopwatch time = new Stopwatch();

Console.WriteLine("Введите строку");

time.Start();

string text = Console.ReadLine();

string cryptedText = ElGam.EnCrypt(text);

time.Stop();

var plainTextBytes = System.Text.Encoding.UTF8.GetBytes(cryptedText);

var entxt = System.Convert.ToBase64String(plainTextBytes);

Console.WriteLine($"Зашифрованное сообщение:{entxt}| {(float)time.ElapsedMilliseconds / 1000} секунд");

time.Reset();

time.Start();

string decryptedText = ElGam.DeCrypt(cryptedText);

time.Stop();

Console.WriteLine($"Расшифрованное сообщение: {decryptedText} | {(float)time.ElapsedMilliseconds / 1000} секунд");

Console.ReadKey();

}

}

public class ElGam

{

private static string Crypt(int p, int g, int x, string inString)

{

var result = "";

var y = Power(g, x, p);

var rand = new Random();

//Console.WriteLine($"Открытый ключ (p,g,y)=({p},{g},{y})");

//Console.WriteLine($"Закрытый ключ x={x}");

foreach (int code in inString)

if (code > 0)

{

var k = rand.Next() % (p - 2) + 1; // 1 < k < (p-1)

var a = Power(g, k, p);

var b = Mul(Power(y, k, p), code, p);

result += a + " " + b + " ";

}

return result;

}

private static string Decrypt(int p, int x, string inText)

{

var result = "";

var arr = inText.Split(' ').Where(xx => xx != "").ToArray();

for (var i = 0; i < arr.Length; i += 2)

{

var a = int.Parse(arr[i]);

var b = int.Parse(arr[i + 1]);

if (a != 0 && b != 0)

{

var deM = Mul(b, Power(a, p - 1 - x, p), p);

// m=b\*(a^x)^(-1)mod p =b\*a^(p-1-x)mod p - т

var m = (char)deM;

result += m;

}

}

return result;

}

private static int Power(int a, int b, int n)

{

// a^b mod n

var tmp = a;

var sum = tmp;

for (var i = 1; i < b; i++)

{

for (var j = 1; j < a; j++)

{

sum += tmp;

if (sum >= n)

{

sum -= n;

}

}

tmp = sum;

}

return tmp;

}

private static int Mul(int a, int b, int n)

{

// a\*b mod n

var sum = 0;

for (var i = 0; i < b; i++)

{

sum += a;

if (sum >= n)

{

sum -= n;

}

}

return sum;

}

public static string EnCrypt(string str)

{

return Crypt(593, 123, 8, str);

}

public static string DeCrypt(string str)

{

return Decrypt(593, 8, str);

}

}

}