МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №10 НА ТЕМУ:**

**Исследование асимметричных шифров RSA и Эль-Гамаля**

                                                                 Выполнил студент 3 курса 5 группы

Дмитрук Илья Игоревич

Минск 2024

# Задание 1.

# Для измерения времени вычисления уравнения вида:

*y* ≡ *ax* mod *n*

была разработана функция, в которой происходит вычисление *y* и измерения времени данного вычисления. Код данной функции представлен в листинге 1.1.

void SolveEquation(BigInteger a, BigInteger xStart, BigInteger xEnd, BigInteger n)

{

var step = (xEnd - xStart) / 6;

for (BigInteger x = xStart; x < xEnd; x += step)

{

var pow = a;

for (BigInteger i = 1; i < x; i++)

{

pow \*= a;

}

start = DateTime.Now;

Console.WriteLine($"{a}^{x} mod {n} = {pow % n}");

end = DateTime.Now;

Console.WriteLine($"Время вычесления y: {(end - start).TotalMilliseconds}");

}

}

Листинг 1.1 – Функция вычисления *y* и измерения времени вычисления

Результат выполнения данной функции представлен на рисунке 1.1.

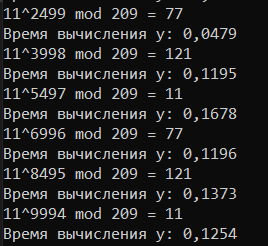


Рисунок 1.1 – Время и результат вычисления *y*

График зависимости времени вычисления *y* от *x* представлен на рисунке 1.1.

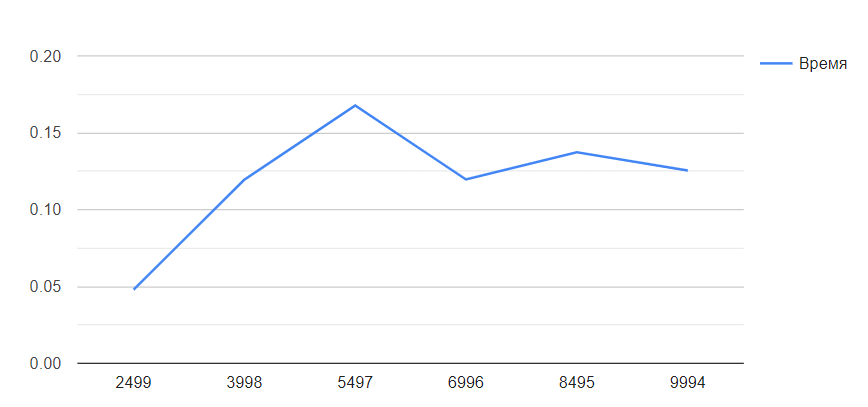


Рисунок 1.1 – График зависимости времени вычисления *y* от *x*

# Задание 2.

Для генерации двух ключей: тайного и открытого (а по сути – двух взаимосвязанных частей одного ключа, т. е. ключа, принадлежащего одному физическому лицу (или группе лиц), либо одному юридическому лицу), используются два больших случайных простых числа *p* и *q*. Для максимальной большей криптостойкости нужно выбирать *p* и *q* равной длины. Рассчитывается произведение: *n* = *pq*. Это есть один из трех компонент ключа, состоящего из чисел *n*, *e*, *d*.

Затем случайным образом выбирается второй компонент ключа (открытый ключ или ключ зашифрования, e, такой что e и (*p* – 1)(*q* – 1) являются взаимно простыми числами; вспомним, что (*p* – 1)(*q* – 1) = φ(*n*) – функция Эйлера).

Наконец, расширенный алгоритм Евклида используется для вычисления третьего компонента ключа: ключа расшифрования d такого, что выполняется условие:

*d*-1 ≡ *e* (mod *φ*(*n*))

Если шифруется сообщение *М*, состоящее из *r* блоков: *m1*, *m2*, …, *mi*, …, *mr*, то шифртекст *С* будет состоять из такого же числа (*r*) блоков, представляемых числами:

*ci*≡ (*mi*)*e*mod *n*

Для расшифрования каждого зашифрованного блока производится вычисление вида:

*mi*≡ (*ci*)*d* mod *n*

Алгоритм шифрования реализован в функции RSAEncrypt. Код данной функции представлен в листинге 2.1.

public static byte[] RSAEncrypt(string plainText, RSAParameters publicKey)

{

using (var rsa = new RSACryptoServiceProvider())

{

rsa.ImportParameters(publicKey);

var bytes = Encoding.UTF8.GetBytes(plainText);

return rsa.Encrypt(bytes, false);

}

}

Листинг 2.1 – Функция шифрования алгоритмом RSA

Для расшифрования сообщения использовалась функция RSADecrypt. Её код представлен в листинге 1.2.

public static string RSADecrypt(byte[] cipherText, RSAParameters privateKey)

{

using (var rsa = new RSACryptoServiceProvider())

{

rsa.ImportParameters(privateKey);

var decryptedBytes = rsa.Decrypt(cipherText, false);

return Encoding.UTF8.GetString(decryptedBytes);

}

}

Листинг 2.2 – Функция расшифрования алгоритмом RSA

Результат шифрования и расшифрования алгоритмом RSA представлен на рисунке 2.1.

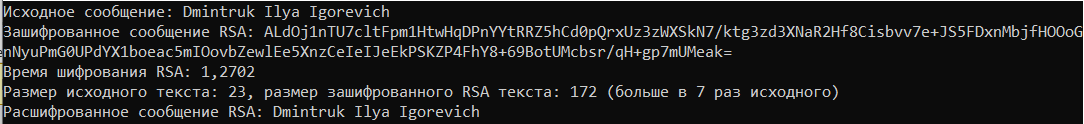


Рисунок 2.1 – Результат шифрования и расшифрования алгоритмом RSA

Как мы видим, размер зашифрованного текста составляет 172 символа, что в 7 раз больше исходного.

# Задание 3.

Для генерации ключевой информации в алгоритме Эль-Гамаля выбирается простое число *р*. Выбирается число (*g*, *g* < *p*), являющееся первообразным корнем числа *р* – очень важный элемент с точки зрения безопасности алгоритма. Далее выбирается число *х* (*х* < *p*) и вычисляется последний компонент ключевой информации:

*y* ≡ *gx* mod *p*

Владельцу сформированной ключевой информации, состоящей из 4 чисел, может посылаться некоторый шифртекст, созданный с использованием открытого ключа получателя: *p*, *g*, *y*. Расшифрование шифртекста получатель производит своим тайным ключом: *p*, *g*, *х*.

Как видим, на самом деле тайным является лишь одно число (как и в RSA): *х*

Как ранее, предположим, что сообщение *М* = {*mi*}, где *mi* – *i*-й блок сообщения. Зашифрование отправителем (каждого отдельного блока *mi* исходного сообщения) предусматривает использование, как это особо подчеркивалось выше, некоторого случайного числа *k* (1 < *k* < *p* – 1).

Блок шифртекста (*ci*) состоит из двух чисел – *аi* и *b*:

*ai* ≡ *gk* mod *p*

*bi* ≡ (*ykmi*) mod *p*

Расшифрование *ci* выполняется по следующей формуле:

*mi* ≡ (*bi*(*ai*) *р* – *x* – 1) mod *p*

где (*ax*) –1 – обратное значение числа ax по модулю *p*. Алгоритм шифрования Эль-Гамаля реализован в функции ElGamalEncrypt. Её код представлен в листинге 3.1.

public static byte[] ElGamalEncrypt(string plainText, ElGamalPublicKeyParameters publicKey)

{

var engine = new ElGamalEngine();

engine.Init(true, publicKey);

var bytes = Encoding.UTF8.GetBytes(plainText);

return engine.ProcessBlock(bytes, 0, bytes.Length);

}

Листинг 3.1 – Функция шифрования алгоритмом Эль-Гамаля

Алгоритм расшифрования реализован в функции ElGamalDecrypt. Её код представлен в листинге 3.2.

public static string ElGamalDecrypt(byte[] cipherText, ElGamalPrivateKeyParameters privateKey)

{

var engine = new ElGamalEngine();

engine.Init(false, privateKey);

var decryptedBytes = engine.ProcessBlock(cipherText, 0, cipherText.Length);

return Encoding.UTF8.GetString(decryptedBytes);

}

Листинг 3.2 – Функция расшифрования алгоритмом Эль-Гамаля

Результат шифрования и расшифрования алгоритмом Эль-Гамаля представлен на рисунке 3.1.

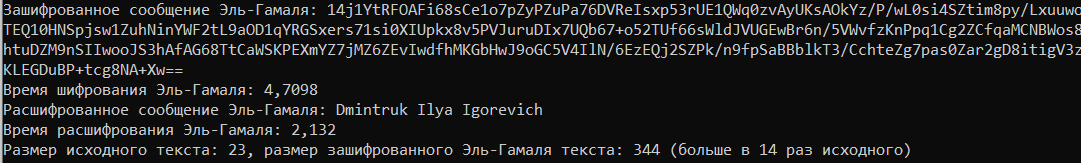


Рисунок 3.1 – Результат шифрования и расшифрования алгоритмом Эль-Гамаля

Как мы видим, размер зашифрованного текста составляет 344 символа, что в 14 раз больше исходного.

# Задание 4.

На рисунке 4.1 представлен график времени работы алгоритмов шифрования и расшифрования RSA, а на рисунке 4.2, алгоритмов шифрования и расшифрования Эль-Гамаля.

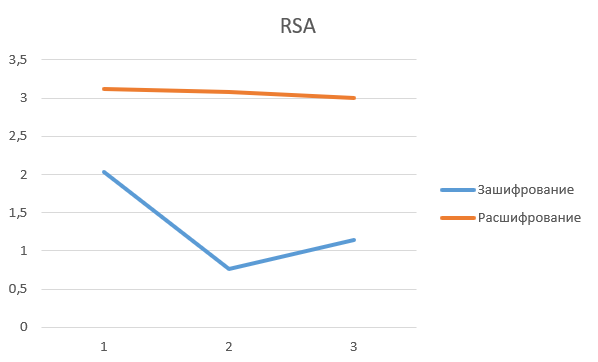


Рисунок 4.1 – Время шифрования и расшифрования алгоритмом RSA

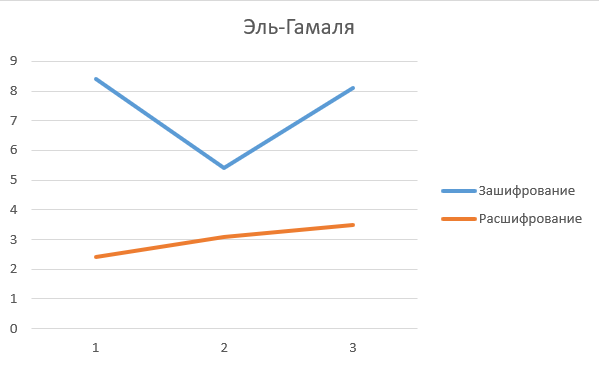


Рисунок 4.2 – Время шифрования и расшифрования алгоритмом Эль-Гамаля

**Вывод**

В ходе лабораторной работы были приобретены навыки в генерации публичных и закрытых ключей, а также в шифровании и расшифровании алгоритмами RSA и Эль-Гамаля. Было разработано приложение, реализующее генерацию ключей и шифрование и расшифрование алгоритмами RSA и Эль-Гамаля. Так же была измерена скорость выполнения данных алгоритмов.