Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по лабораторной работе №10

«Исследование асимметричных шифров RSA и Эль-Гамаля»

Студентка: Пунько А.А,

ФИТ 3 курс 5 группа

Преподаватель: Берников В. О.

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.

. 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.

# Теоретические сведения

Как отмечалось выше, асимметричная криптография основана на сложности решения некоторых математических задач. По существу таких задач две:

• разложение больших чисел на простые сомножители (задача факторизации),

• вычисление дискретного логарифма в конечном поле,

а также:

• вычислительные операции над точками эллиптической кривой.

Эти задачи объединяет то, что они используют операцию получения остатка от целочисленного деления. В силу этого практически все системы асимметричного зашифрования/расшифрования основаны либо на проблеме факторизации (среди них – RSA), либо на проблеме дискретного логарифмирования (среди них – Эль-Гамаля).

Рассматриваемый алгоритм появился (1977 г.) после алгоритма рюкзака Меркла. Он стал первым полноценный алгоритмом с открытым ключом, который впоследствии стал одним из основных для шифрования и для электронных цифровых подписей. Из всех предложенных алгоритмов с открытыми ключами RSA проще всего понять и реализовать. Названный в честь трех его создателей: Рона Ривеста (RonRivest), Ади Шамира (Adi Shamir) и Леонарда Эдлемана (Leonard Adleman). Как было отмечено, безопасность RSA основана на трудности разложения на множители больших чисел. Открытый и закрытый ключи являются функциями двух больших простых чисел. Предполагается, что восстановление открытого текста по шифртексту и открытому ключу эквивалентно разложению на множители двух больших чисел. 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. Этой 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. Б. Шнайер рекомендует число е выбирать из ряда: 3, 17, 216 + 1. Наконец расширенный алгоритм Евклида используется для вычисления третьего компонента ключа: ключа расшифрования, d, такого, что выполняется условие: ed = 1 (mod φ(n)). Другими словами: d-1 = e(mod φ(n)).

Таким образом, сформирован ключ, состоящий из трех чисел, которые, в свою очередь, образуют две вышеупомянутые взаимосвязанные части: открытый (публичный) ключ, (e, n), и тайный ключ, (d, n; на самом деле, как видим, тайным здесь является лишь первое из пары чисел). Для зашифрования/расшифрования используется ключ получателя: отправитель шифрует сообщение открытым ключом, а получатель расшифровывает шифртекст своим тайным ключом. Зашифрование. 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. Разработаны несколько версий стандарта рассматриваемого алгоритма. Среди прочего, в этих документах обсуждаются размеры безопасного ключа. Доступна одна из последних версий стандарта RSA: RFC 3447. Размер ключа в алгоритме RSA связан с размером модуля,n. Два числа p и q, произведение которых равно n, должны иметь приблизительно одинаковую длину, поскольку в этом случае найти сомножители (факторы) сложнее, чем в случае, когда длина чисел значительно различается. Например, если предполагается использовать 768-битный модуль, то каждое число должно иметь длину приблизительно 384 бита. В 1999 году 512-битный ключ был вскрыт за семь месяцев. Это означает, что 512-битные ключи уже не обеспечивают достаточную криптостойкость. Сейчас в критических системах применяются ключи длиной 1024 и 2048 бит.

Алгоритм Эль-Гамаля предложен Эль-Гамалем (T. El-Gamal) в 1985 г. Он может быть использован для решения трех основных криптографических задач: для зашифрования/расшифрования данных, для формирования цифровой подписи и для согласования общего ключа. Кроме того, возможны модификации алгоритма для схем проверки пароля, доказательства идентичности сообщения и другие варианты. Как подчеркивалось выше, безопасность алгоритма Эль-Гамаля, как и безопасность алгоритма Диффи-Хеллмана, основана на трудности вычисления дискретных логарифмов. Алгоритм Эль-Гамаля фактически использует схему Диффи-Хеллмана, чтобы сформировать общий секретный ключ для абонентов, передающих друг другу сообщение, и затем сообщение шифруется путем умножения его на этот ключ. И в случае шифрования, и в случае формирования цифровой подписи каждому пользователю необходимо сгенерировать пару ключей. Рассматриваемый алгоритм отличается от алгоритма RSA несколькими параметрами и особенностями:

1) генерацией ключевой информации и числом компонент, составляющих ключ;

2) каждому блоку (символу) открытого сообщения в шифртексте на основе алгоритма Эль-Гамаля соответствуют 2 блока (в RSA – один-один);

3) в алгоритме Эль-Гамаля при зашифровании используется число (обозначим его k), которое практически никак не связано с ключевой информацией получателя и которое принимает (по определению) различные значения при зашифровании различных блоков сообщения.

Выбирается простое число, р. Выбирается число (g, g < p), являющееся первообразным корнем числа р – очень важный элемент с точки зрения безопасности алгоритм. Далее выбирается число х (х < p) и вычисляется последний компонент ключевой информации: y =gх mod р.

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.

Первообразный корень (primary (residual ) root ) по модулю р является таким числом, что его степени (gi, 1 ≤i≤p-1 ) дают все возможные по модулю р вычеты (остатки), которые взаимно просты с p. Понятно, что для больших значений р количество всех неповторяющихся остатков (р – 1) будет также большим.

Для схемы вероятностного шифрования само сообщение и ключ не определяют шифртекст однозначно. 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). В силу использования случайной величины k шифр Эль-Гамаля называют также шифром многозначной замены, а также схемой вероятностного шифрования. Вероятностный характер шифрования является преимуществом для схемы Эль-Гамаля по сравнению, например, с алгоритмом RSA. Блок шифртекста (ci) состоит из двух чисел: аi и bi: ai = gk mod p, bi = (yk \*mi) mod p.

Здесь стал очевидный упомянутый недостатком алгоритма шифрования Эль-Гамаля: удвоение (реально – примерно в 1,5 раза) длины зашифрованного текста по сравнению с начальным текстом. Случайное число k должно сразу после вычисления уничтожаться.

Расшифрование ci выполняется по следующей формуле: mi = (bi \*(ai)x)-1) mod p или mi = (bi \*(ai)р-x-1) mod p, где (ax)-1 – обратное значение числа ax по модулю p. Нетрудно проверить, что ((ai)x)-1) = gkх mod p. Еще раз возвратимся к криптостойкости рассмотренного алгоритма. Если для зашифрования двух разных блоков (m1 и m2) некоторого сообщения использовать одинаковые k, то для соответствующих шифртекстов c1 = (a1, b1) и c2 = (a2, b2) выполняется соотношение b1(b2)-1 = m1(m2)-1. Из этого выражения можно легко вычислить m2, если известно m1. При примерно одинаковой размерности ключей рассмотренные алгоритмы обеспечивают примерно одинаковый уровень криптостойкости.

# Практическая часть

При создании приложения я использовала язык C# и встроенный алгоритм для реализации шифрования и расшифрования алгоритмом RSA. Стартовое окно приложения приведено на рисунке 1.

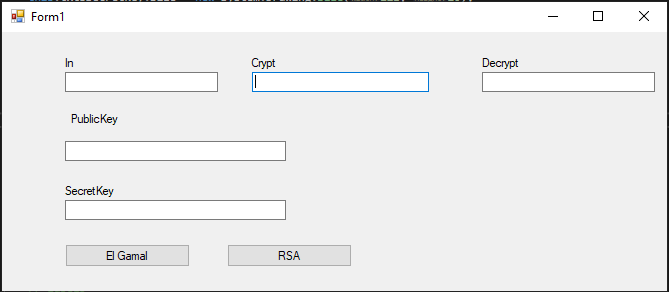


Рисунок 1 – Стартовое окно приложения

Результат зашифрования и расшифрования с использованием алгоритма Эль-Гамаля приведен на рисунке 2.

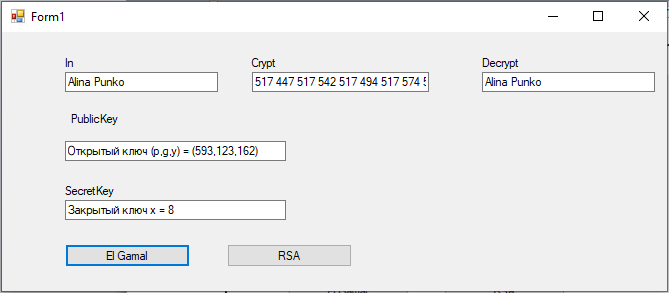


Рисунок 2 – Использование алгоритма Эль-Гамаля

Результат использования алгоритма RSA приведен на рисунке 3.

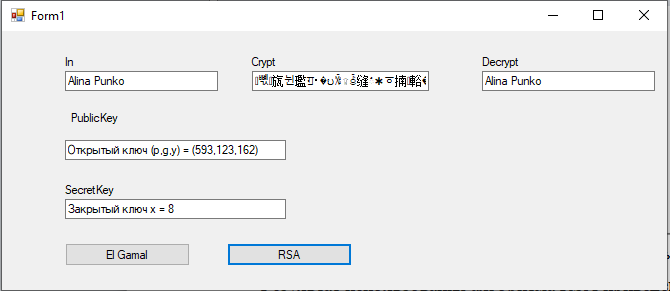


Рисунок 3 – Использование алгоритма RSA

Функция шифрования с использованием алгоритма Эль Гамаля приведена в листинге 1.

void crypt(int p, int g, int x, string strIn)

{

txtBCrypt.Text = "";

int y = power(g, x, p);

txtBPublicKey.Text = "Открытый ключ (p,g,y) = (" + p + "," + g + "," + y + ")";

txtBSecretKey.Text = "Закрытый ключ x = " + x;

if (strIn.Length > 0)

{

char[] temp = new char[strIn.Length - 1];

temp = strIn.ToCharArray();

for (int i = 0; i <= strIn.Length - 1; i++)

{

int m = (int)temp[i];

if (m > 0)

{

int k = Rand() % (p - 2) + 1; // 1 < k < (p-1)

int a = power(g, k, p);

int b = mul(power(y, k, p), m, p);

txtBCrypt.Text = txtBCrypt.Text + a + " " + b + " ";

}

}

}

}

Листинг 1 – Зашифрование с использованием алгоритма Эль-Гамаля

Функция расшифрования с использованием алгоритма Эль Гамаля приведена в листинге 2.

void decrypt(int p, int x, string strIn)

{

if (strIn.Length > 0)

{

txtBDecrypt.Text = "";

string[] strA = strIn.Split(' ');

if (strA.Length > 0)

{

for (int i = 0; i < strA.Length - 1; i += 2)

{

char[] a = new char[strA[i].Length];

char[] b = new char[strA[i + 1].Length];

int ai = 0;

int bi = 0;

a = strA[i].ToCharArray();

b = strA[i + 1].ToCharArray();

for (int j = 0; (j < a.Length); j++)

{ ai = ai \* 10 + (int)(a[j] - 48); }

for (int j = 0; (j < b.Length); j++)

{ bi = bi \* 10 + (int)(b[j] - 48); }

if ((ai != 0) && (bi != 0))

{

int deM = mul(bi, power(ai, p - 1 - x, p), p)

char m = (char)deM;

txtBDecrypt.Text = txtBDecrypt.Text + m;

}

}

}

}

Листинг 2 – Расшифрование с использованием алгоритма Эль-Гамаля

Функция шифрования с использованием алгоритма RSA приведена в листинге 3.

public static byte[] RSAEncrypt(byte[] DataToEncrypt, RSAParameters RSAKeyInfo, bool DoOAEPPadding)

{

try

{

byte[] encryptedData;

using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())

{

RSA.ImportParameters(RSAKeyInfo);

encryptedData = RSA.Encrypt(DataToEncrypt, DoOAEPPadding);

}

return encryptedData;

}

catch (CryptographicException e)

{

Console.WriteLine(e.Message);

return null;

}

}

Листинг 3 – Зашифрование с использованием алгоритма RSA

Функция расшифрования с использованием алгоритма RSA приведена в листинге 4.

public static byte[] RSADecrypt(byte[] DataToDecrypt, RSAParameters RSAKeyInfo, bool DoOAEPPadding)

{

try

{

byte[] decryptedData;

using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())

{

RSA.ImportParameters(RSAKeyInfo);

decryptedData = RSA.Decrypt(DataToDecrypt, DoOAEPPadding);

}

return decryptedData;

}

catch (CryptographicException e)

{

Console.WriteLine(e.ToString());

return null;

}

Листинг 4 – Расшифрование с использованием алгоритма RSA

# Вывод

В результате выполнения данной лабораторной работы было создано приложение, реализующее работу с алгоритмами Эль-Гамаля и RSA и были подробно изучены эти алгоритмы.