МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

**«Исследование алгоритмов генерации и верификации электронной цифровой подписи»**

Студент:

Септилко Анастасия Антоновна

Преподаватель:

Блинова Евгения Александровна

Минск 2020

**ЗАДАНИЯ:**

1. Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться результатами выполнения предыдущих лабораторных работ, а также доступными библиотеками либо программными кодами. Приложение должно реализовывать следующие операции:

• генерацию и верификацию ЭЦП на основе алгоритмов RSA, ЭльГамаля и Шнорра;

• оценку времени выполнения указанных процедур при реальных (требуемых) ключевых параметрах. Для вычисления хешей можно также воспользоваться доступными online-средствами, например, katvin (<https://katvin.com/tools/hash-generator.html>).

2. Для выполнения необходимых операций передачи (по сети)/верификации информации обменяйтесь открытой ключевой информацией с получателем подписанного сообщения для каждого исследуемого алгоритма (по согласованию с преподавателем).

3. Результаты оформить в виде отчета по установленным правилам

Электронная цифровая подпись – контрольная характеристика сообщения, которая вырабатывается с использованием личного ключа, проверяется с использованием открытого ключа, служит для контроля целостности и подлинности сообщения и обеспечивает невозможность отказа от авторства.

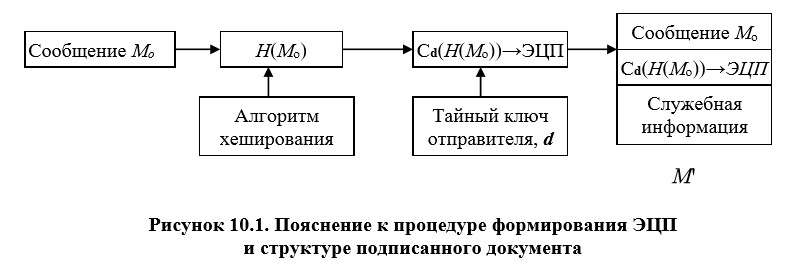
Таким образом, ЭЦП выполняет те же функции, что и собственноручная (поставленная «от руки») подпись:

• аутентифицировать лицо, подписавшее сообщение;

• контролировать целостность подписанного сообщения;

• защищать сообщение от подделок;

• доказать авторство лица, подписавшего сообщение, если это лицо отрицает свое авторство.



**ЭЦП на основе RSA:**

Здесь можно рассматривать две ситуации: • сообщение Мo подписывается и передается в открытом (незашифрованном) виде; • сообщение Мo подписывается и передается в зашифрованном. Первый случай соответствует схеме и операциям, представленным на рис. 10.1 и рис. 10.2. При этом подпись S вычисляется на основе известного из лабораторной работы №8 соотношения (8.5):

S = (H(Mo))dо mod no (10.1)

при указанном выше реверсе в отношении ключевой информации; в (10.1) dо и no – элементы тайного ключа отправителя. Передаваемое сообщение М' = Мo||S. Соответственно, операция расшифрования на приемной стороне (получатель анализирует Мп||S) будет производиться в соответствии с формулой (8.6) с известной модификацией ключей:

H(Mo) = (S)ео mod no. (10.2)

Далее вычисляется Н(Mп). Если H(Mo) = H(Mп), подпись верифицирована. Если подписываемое сообщение М (М') также должно передаваться в зашифрованном виде, то обычно М' шифруется на стороне отправителя стандартным образом: с помощью открытого ключа получателя (еп и nп), который перед основным процессом верификации подписи расшифровывает послание своим тайным ключом: dп и nп . Далее осуществляются вычисления и анализ, как и в первом случае.

На рисунке 1.1 показана функция Create() в которую мы передаем числа р и q, пути к текстовым файлам. Производим необходимые операции, нахождения n, m, d, e. Шифрование текста из файла Source.txt в RSA.txt (в виде чисел как мы это делали ранее в лаб. 8). Весь процесс шифрования и расшифрования приведен в листинге 1.1.

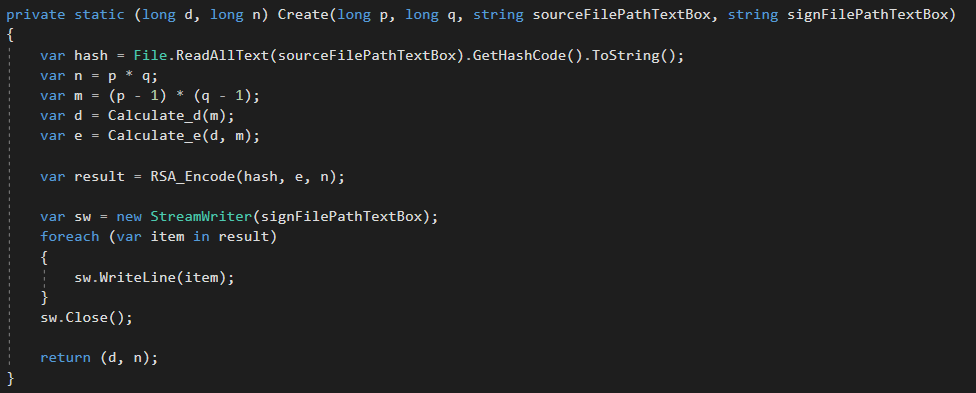


Рисунок 1.1 – Функция Create()

Листинг 1.1 – Функция RSA\_Encode

private static List<string> RSA\_Encode(string s, long e, long n)

{

var result = new List<string>();

foreach (var t in s)

{

var index = Array.IndexOf(Characters, t);

var bi = new BigInteger(index);

bi = BigInteger.Pow(bi, (int)e);

var bn = new BigInteger((int)n);

bi %= bn;

result.Add(bi.ToString());

}

return result;

}

Функция RSA\_Decode

private static string RSA\_Decode(List<string> input, long d, long n)

{

var result = "";

var bn = new BigInteger((int)n);

foreach (var item in input)

{

var bi = new BigInteger(Convert.ToDouble(item));

bi = BigInteger.Pow(bi, (int)d);

bi = bi % bn;

var index = Convert.ToInt32(bi.ToString());

result += Characters[index].ToString();

}

return result;

}

Чтобы проверить ЭЦП действительна или нет была разработана специальная функция Verify. В ней мы передаем файлы и сравниваем содержимое файлов (их хеш) если файлы идентичны, то ЭЦП верифицирована, если нет, то программа напишет False (рисунок 1.2).

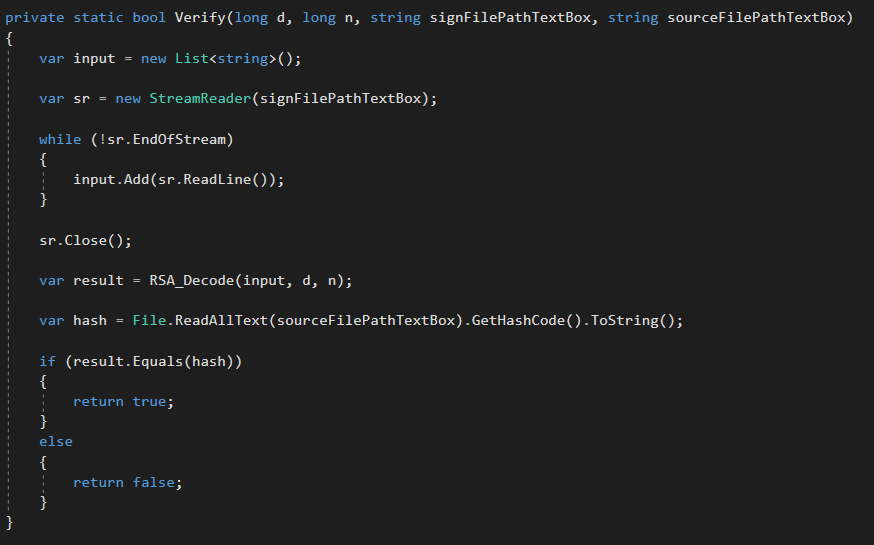


Рисунок 1.2 – Функция Verify

Давайте проверим работоспособность приложения. Запишем некий текст в файлы и сравним их (рисунок 1.3). Мы видим, что файлы различны значит программа скажет нам что ЭЦП не действительна.

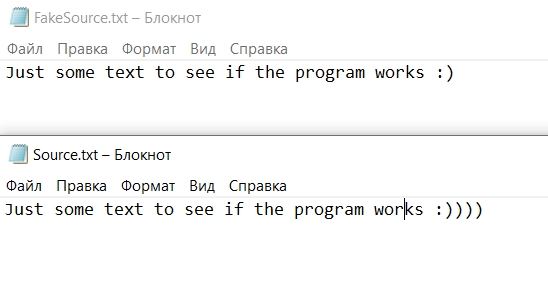


Рисунок 1.3 – Содержимое текстовых файлов

На рисунке 1.4 мы видим вывод работы программы и можем сделать вывод о том, что программа работает правильно.



Рисунок 1.4 – Итог выполнения программы

**ЭЦП Эль-Гамаля**:

Ключевая информация отправителя для ЭЦП создается точно так же, как это описано в материалах к ЛР №8. Она состоит из тех же элементов, что и ключи в DSA. Основное отличие в применении расчетов состоит в том, что результатом зашифрования является только одна пара чисел, а не пара – для каждого блока исходного сообщения. Причем в рассматриваемом случае таким сообщением является хеш подписываемого документа: Н(Mо).

Итак, ключевая информация отправителя: открытый ключ: y, g и р; тайный ключ: х. Чтобы подписать сообщение Мо, обладатель используемых для ЭЦП ключей должен выбрать, как и в предыдущей схеме, случайное число k, взаимно простое с р-1. Затем вычисляется числа а и b, являющиеся цифровой подписью (S = {a, b}): a = gk mod p; (10.5) для вычисления b с помощью расширенного алгоритма Евклида решается уравнение Н(Mо) = (xa + kb) mod (p-1). (10.6)

Получателю отправляется сообщение М' = Мо||S. Для верификации подписи вычисляется хеш полученного сообщения, Н(Мп) = h. Далее нужно убедиться, что выполняется равенство ya \* ab = gh (mod p). (10.7) Если равенство выполняется, подпись верифицируется.

В случае этого алгоритма мы используем заранее указанные строки "Hello world". Для получения хеша строки я использовала стандартную функцию MD5. Эта функция продемонстрирована на рисунке 1.5.

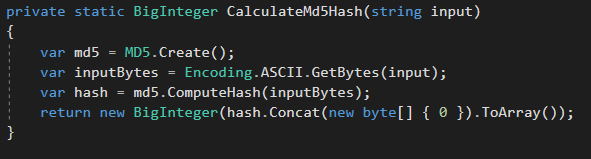


Рисунок 1.5 – Функция CalculateMd5Hash

Шифрование и расшифрование описывалось и производилось ранее в лаб. 8, поэтому я просто привела листинг 1.2 кода этих функций.

Листинг 1.2– Функция Crypt

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

{

var result = "";

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

var rand = new Random();

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;

}

Функция Decrypt

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)

{

//wcout<<a<<" "<<b<<endl;

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 - трудно было найти нормальную формулу, в ней вся загвоздка

var m = (char)deM;

result += m;

}

}

return result;

}

Пример как сравниваются строки (их хеш) приведен на рисунке 1.6.

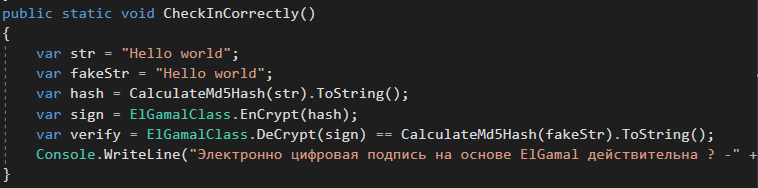


Рисунок 1.6 – Функция CheckInCorrectly

Мы можем заметить, что строки (из рисунка 1.6) абсолютно равны, а это значит, что программа вернет нам True. Итог выполнения программы продемонстрирован на рисунке 1.7.

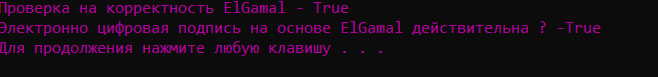


Рисунок 1.7 – Выполнение программы

**ЭЦП Шнорра:**

Рассматриваемая схема является основой стандарта ЭЦП в Беларуси. Алгоритм ЭЦП К. Шнорра (К. Schnorr ) является вариантом алгоритма ЭЦП Эль-Гамаля.

Одной из особенностей ЭЦП Эль-Гамаля является то, что число p должно быть очень большим, чтобы сделать действительно трудной проблему дискретного логарифма. Рекомендуется длина p, по крайней мере, должна составлять 1024 бита. Чтобы уменьшить размер подписи, Шнорр предложил новую схему, но с уменьшенным размером подписи.

Ключевая информация: p – простое число в диапазоне от 512 до 1024 бит; q –160-битное простое число, делитель p-1; любое число g (g≠1) такое, что gq=1 mod p. (10.8)

Числа p, g, q являются открытыми и могут применяться группой пользователей.

Выбирается число х < q (х является тайным ключом) и вычисляется последний элемент открытого ключа: y = g-х mod p. (10.9)

Секретный ключ имеет длину не менее 160 бит.

Для подписи сообщения Мо выбирается случайное число k (1<k<q) и вычисляет параметр, а: а = gk mod p. (10.10)

Далее вычисляется хеш от канкатенации сообщения Мо и числа а: h = H(Mo||a). Обратим внимание, что хэш-функция непосредственно не применяется к сообщению. Создается хеш-образ подписываемого сообщения, спереди присоединенного к числу а. Далее вычисляется значение b: b = (k + xh ) mod q. (10.11)

Получателю отправляются М' = Мо||S; S = {h, b}.

Для проверки подписи получатель вычисляет Х = gb yh (mod p). (10.12)

Затем он проверяет выполнение равенства: h = Н(Mп||Х). Подпись достоверна, если равенство выполняется.

*Для реализации алгоритма* ***Шнорра*** *были написаны:*

Функция, которая подбирает простое число P в диапазоне от 1 до 999999, простое число Q, которое является делителем P – 1. Также от 1 до P – 1 выбирается случайное число H (для полинома).

Число G равно H^(P – 1 / G) mod P. Это один из наших открытых ключей.

T – случайное число в диапазоне от Q – 1 до 100000 – наше сообщение.

Следующая функция позволяет задать секретный ключ S (можно сгенерировать случайный) от 1 до Q – 1.

Число A является публичным ключом и находится по формуле A = Ginv^S mod P, где Ginv – число, обратное числу G по модулю P.

Следующая функция позволяет задать случайное число R, необходимое для разового шифрования (с ним будет происходить конкатенация), а также позволяет рассчитать тайный ключ X, который находится по формуле G^R mod P.

Большое значение E – это хеш, полученный при помощи конкатенации нашего исходного сообщения (число T) и тайного ключа X.

Далее вычисляется значение Y, которое можно найти по формуле: Y = (R(случайное число, участвующее в генерации X) + E(хеш) \* X(тайный ключ)) mod P.

Получателю отправляется сообщение, являющееся конкатенацией входного сообщения T и пары (E(хеш), Y).

Для того, чтобы узнать тайный ключ, нужно подставить соответствующие элементы в формулу:

Z = (G^Y \* A^E) mod P.

Если Z==X, то ЭЦП подлинна и сообщение не изменялось.

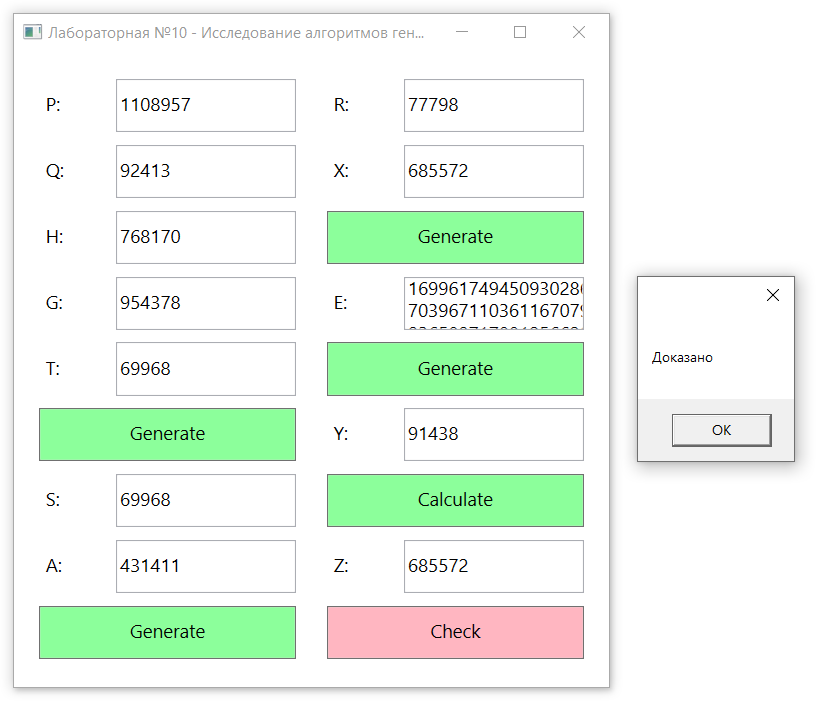


Рисунок 2.5 – Проверка ЭЦП (Шнорр)

**Вывод**

В данной лабораторной работе я закрепила теоретические знания по алгебраическому описанию, алгоритмам реализации операций генерации и верификации электронной цифровой подписи (ЭЦП). Получила навыки практической реализации методов генерации и верификации ЭЦП на основе хеширования подписываемых сообщений и алгоритмов RSA, Эль-Гамаля и Шнорра. Разработала приложение для реализации заданных алгоритмов генерации и верификации ЭЦП. А также оценить скорость генерации и верификации ЭЦП.