Министерство образования Республики Беларусь

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет к лабораторной работе № 9

«ИССЛЕДОВАНИЕ АЛГОРИТМОВ ГЕНЕРАЦИИ И ВЕРИФИКАЦИИ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ»

Выполнила:

студентка 3 курса 2 группы

Шастовская М. С.

Вариант: 12

Преподаватель:

Сазонова Д.В.

Минск 2023

**Цель:** изучение алгоритмов генерации и верификации электронной цифровой подписи и приобретение практических навыков их реализации.

**Задачи:**

* Закрепить теоретические знания по алгебраическому описанию и алгоритмам реализации операций генерации и верификации электронной цифровой подписи (ЭЦП).
* Получить навыки практической реализации методов генерации и верификации ЭЦП на основе хеширования подписываемых сообщений и алгоритмов RSA, Эдь-Гамаля и Шнорра, а также DSA.
* Разработать приложение для реализации заданных алгоритмов генерации и верификации ЭЦП
* Оценить скорость генерации и верификации ЭЦП.
* Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента. **1 Теоретические сведения**

*Определение 1.* 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:

* аутентифицирование лица, подписавшего сообщение;
* контроль целостности подписанного сообщения;
* защита сообщения от подделок;
* доказательство авторства лица, подписавшего сообщение, если это лицо отрицает свое авторство.

*Определение 2.* Электронная цифровая подпись – бинарная (или в ином виде) последовательность символов, являющаяся реквизитом электронного документа, зависящая от содержания этого документа и предназначенная для подтверждения целостности и подлинности электронного документа.

Классическая технология использования ЭЦП предусматривает подписание не самого сообщения (обозначим его здесь Мо), а его хеша, H(Mо). Это сокращает время генерации/верификации подписи и снижает вероятность появления случайных ошибок в итоговом документе.

Основу рассматриваемых протоколов составляют методы асимметричной криптографии и эллиптических кривых.

Общая структура подписанного электронного документа – Мо – М' – представляет собой, как правило, конкатенацию этого документа и ЭЦП S. Кроме этих двух элементов, интегральный документ может содержать некоторую служебную информацию (дата, время отправки или различные данные об отправителе), как это схематично показано на рис. 1.1.

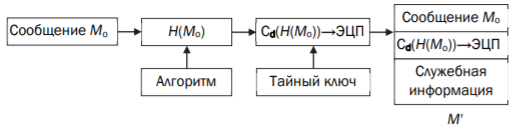


Рис. 1.1 – Пояснение к процедуре формирования ЭЦП и структуре подписанного документа

В алгоритме на основе RSA можно рассматривать две ситуации:

* сообщение Мo подписывается и передается в открытом (незашифрованном) виде;
* сообщение Мo подписывается и передается в зашифрованном виде.

S ≡ (H(Mo))dо mod n0

Операция расшифрования:

H(Mo) ≡ (S)ео mod no

Далее вычисляется Н(Mп). Если H(Mo) = H(Mп), подпись верифицирована.

В алгоритме используются следующие параметры: p – простое число длиной от 64 до 1024 битов (число должно быть кратно 64); q – 160-битный простой множитель (р – 1). Далее вычисляется число g:

g = v(p – 1)/g mod p,

где v – любое число, меньшее (р – 1), для которого выполняется условие:

v(p – 1)/g mod p > 1.

Числа p, q, v могут использоваться группой лиц. Еще один элемент открытого ключа у вычисляется в соответствии с выражением

y ≡ gx mod p,

где x < q; х – закрытый ключ.

Для алгоритма Эль-Гамаля ключевая информация отправителя для ЭЦП создается точно так же, как это описано в материалах к лабораторной работе № 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).

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

gq ≡ 1 mod p.

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.

Секретный ключ имеет длину не менее 160 битов.

Для подписи сообщения Мо выбирается случайное число k (1 < k < q) и вычисляет параметр а:

а ≡ gk mod p.

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.

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

Для проверки подписи получатель вычисляет

Х ≡ gb yh (mod p).

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

2 Практическая часть

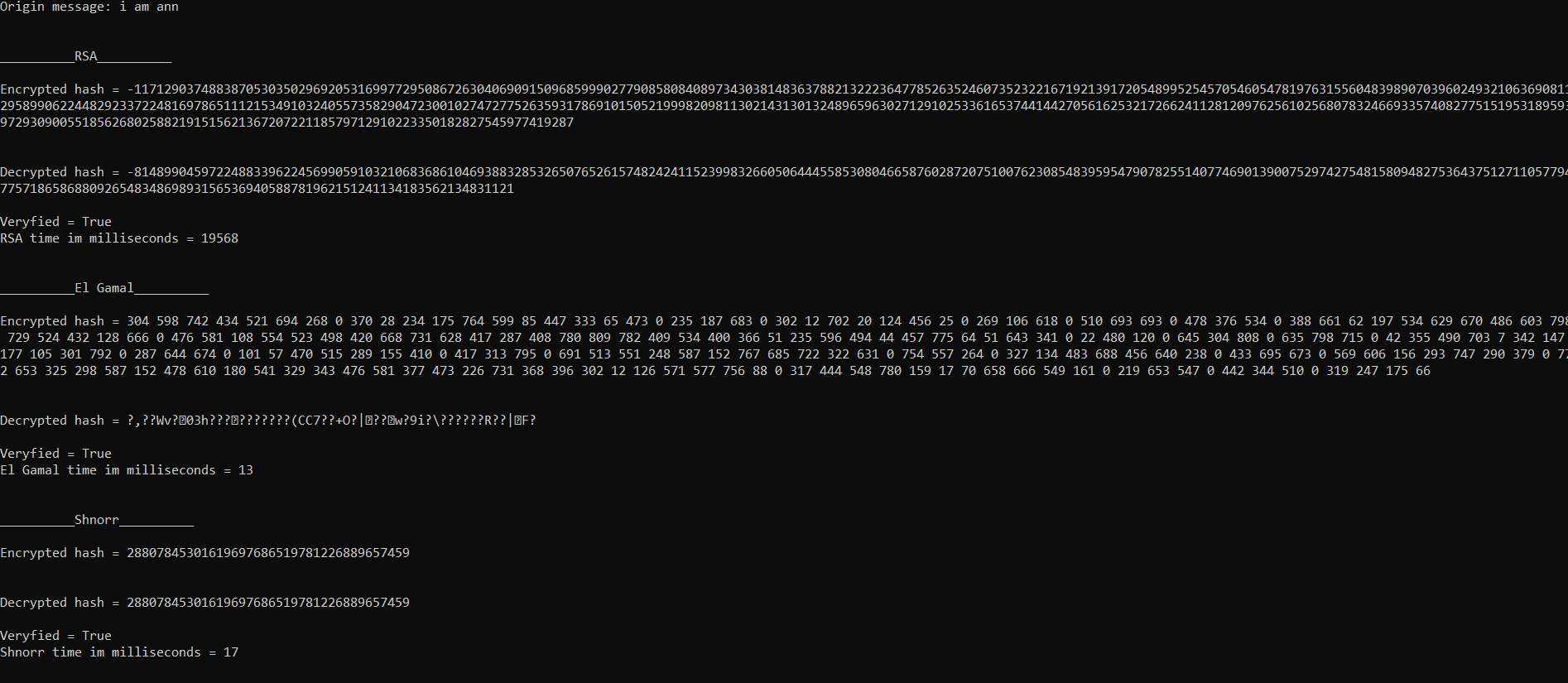


Рисунок 2.1 – Результат генерации и верификации ЭЦП на основе хеширования подписываемых сообщений и алгоритмов RSA, Эль-Гамаля и Шнорра.

На данных скриншотах отображен процесс работы программы хэширования подписываемых сообщений и аллгоритмов RSA, Эль-Гамаля и Шнорра, процесс верификации и время выполнения алгоритма.

Основные функции программы представлены на рисунке 2.2.

public static void Do()

{

BigInteger p = 2267;

BigInteger q = 103;

string text = File.ReadAllText(".\\Test.txt");

BigInteger g = 354;

BigInteger obg = 967;

int x = 30;

BigInteger y = BigInteger.ModPow(obg, x, p);

BigInteger a = BigInteger.Pow(g, 13) % p;

BigInteger hash = CalculateMd5Hash(text + a.ToString());

Console.WriteLine($"\nEncrypted hash = {hash}\n");

File.WriteAllText(".\\shnorr.txt", hash.ToString());

BigInteger b = (13 + x \* hash) % q;

BigInteger dov = BigInteger.ModPow(g, b, p);

BigInteger X = (dov \* BigInteger.ModPow(y, hash, p)) % p;

BigInteger hash2 = CalculateMd5Hash((text + X.ToString()));

Console.WriteLine($"\nDecrypted hash = {hash2}\n");

Console.WriteLine("Veryfied = " + (hash == hash2));

}

public static BigInteger Encrypt(BigInteger message)

{

return BigInteger.ModPow(message, privateKey, modulo);

}

public static BigInteger Decrypt(BigInteger encryptedMessage)

{

return BigInteger.ModPow(encryptedMessage, publicKey, modulo);

}

public static List<BigInteger> EncryptMethod(byte[] m)

{

Random random = new Random();

x = random.Next(2, p - 1);

int g = FindAtiderivativeRoot(p);

int y = (int)(BigInteger.ModPow(g, x, p));

List<BigInteger> cipher = new List<BigInteger>();

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

{

int k = random.Next(2, p - 2);

BigInteger a = BigInteger.ModPow(g, k, p);

BigInteger b = (BigInteger.Pow(y, k) \* m[i]) % p;

cipher.Add(a);

cipher.Add(b);

}

return cipher;

}

public static byte[] DecryptMethod(List<BigInteger> c, string pathToFile)

{

List<byte> m = new List<byte>();

int x = Convert.ToInt32(File.ReadAllText(pathToFile));

for (int i = 0; i < c.Count; i += 2)

{

BigInteger axMinusOne = EvklidExtend(BigInteger.Pow(c[i], x), p);

BigInteger b = c[i + 1];

m.Add((byte)((axMinusOne \* b) % p));

}

return m.ToArray();

}

Вывод

В ходе выполнения лабораторной работы были изучены и приобретены практические навыки разработки и использования приложений для реализации генерации и верификации ЭЦП.