Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Дисциплина «Основы информационной безопасности»

Отчёт по практическому занятию №5

**Криптографическая защита информации**

Студент: Жук С.С.

ФИТ 2 курс 2 группа

Преподаватель: Ржеутская Н.В.

**Практическое занятие №5**

**«Криптографическая защита информации»**

Цель: изучить и закрепить умение реализации ЭЦП на примере RSA.

**Теоретическое введение**

# Реализация элементов ЭЦП RSA

Протоколы ЭЦП с одной стороны относят к протоколам аутентификации, т.к. гарантируют, что сообщение поступило от достоверного отправителя, а с другой стороны к протоколам контроля целостности, т.к. гарантируют, что сообщение пришло в неискаженном виде. Более того, получатель в дальнейшем может использовать ЭЦП как доказательство достоверности сообщения третьим лицам (арбитру) в том случае, если отправитель впоследствии попытается отказаться от него.

Говоря о схеме цифровой подписи, обычно имеют в виду следующую классическую ситуацию:

* отправитель знает содержание сообщения, которое он подписывает;
* получатель, зная открытый ключ проверки подписи, может проверить правильность подписи полученного сообщения в любое время без какого-либо разрешения и участия отправителя;
* безопасность схемы подписи гарантируется.

**Этап 1.** Выработка ключей (выполняет отправитель **А**) - см. практическую работу 6 " Шифрование методом RSA".

**Этап 2.** Отправка сообщения и электронной подписи (выполняет отправитель **А**).

Отправка сообщения и ЭЦП на базе алгоритма RSA



**Этап 3.** Получение сообщения и проверка электронной подписи (выполняет получатель **B**).

Получение сообщения и проверка ЭЦП на базе алгоритма RSA



**Исполнительская часть**

1. Дайте определение понятию "электронная цифровая подпись".

**Электронная цифровая подпись** – реквизит электронного документа для защиты документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа ЭЦП и позволяющий определить владельца подписи, а также установить отсутствие искажения информации.

1. Объясните какой порядок использования ключей (открытый; закрытый) при отправке и проверке ЭЦП.

При создании цифровой подписи по классической схеме отправитель:

* применяет к исходному сообщению T хеш-функцию h(T) и получает хеш-образ r сообщения;
* вычисляет цифровую подпись s по хеш-образу r с использованием своего закрытого ключа;
* посылает сообщение T вместе с цифровой подписью s получателю.

Получатель, отделив цифровую подпись от сообщения, выполняет следующие действия:

* применяет к полученному сообщению T хеш-функцию h(T)и получает хеш-образ r сообщения;
* расшифровывает хеш-образ r’ из цифровой подписи s с использованием открытого ключа отправителя;
* проверяет соответствие хеш-образов r и r’ и если они совпадают, то отправитель действительно является тем, за кого себя выдает, и сообщение при передаче не подверглось искажению.

Как видно из этой схемы, порядок использования ключей обратный тому, который используется при передаче секретных сообщений. Вначале отправитель использует свой закрытый ключ, а затем получатель применяет открытый ключ отправителя.

1. Перечислите специальные схемы ЭЦП.

* схема "конфиденциальной" (неотвергаемой) подписи – подпись не может быть проверена без участия сгенерировавшего ее лица;
* схема подписи "вслепую" ("затемненной" подписи) - отправитель не знает подписанного им сообщения;
* схема "мультиподписи" - вместо одного отправителя сообщение подписывает группа из нескольких участников;
* схема "групповой" подписи - получатель может проверить, что подписанное сообщение пришло от члена некоторой группы отправителей, но не знает, кем именно из членов группы оно подписано. В тоже время, в случае необходимости, отправитель может быть определен;
* и др.

1. Перечислите недостатки алгоритма цифровой подписи RSA.

**1.** При вычислении ключей для системы цифровой подписи RSA необходимо проверять ряд дополнительных условий. Невыполнение любого из этих условий делает возможным фальсификацию цифровой подписи со стороны того, кто обнаружит такое невыполнение.  
**2.** Для обеспечения криптостойкости цифровой подписи RSA по отношению к попыткам фальсификации, например, на уровне алгоритма шифрования DES, необходимо использовать при вычислениях ключей очень большие целые числа, (около 10 в степени 154), что требует относительно больших вычислительных затрат, превышающих на 20-30% вычислительные затраты других алгоритмов цифровой подписи при сохранении того же уровня криптостойкости.  
**3.** Цифровая подпись RSА уязвима к так называемой мультипликативной атаке. Иначе говоря, *алгоритм цифровой подписи RSA* позволяет злоумышленнику без знания секретного ключа сформировать подписи под теми документами, у которых результат хэширования можно вычислить как произведение результатов хэширования уже подписанных документов. Хотя следует заметить, что вероятность реализации такой атаки весьма незначительна.  
Для работы смарт-карт с цифровыми подписями RSA рекомендуется использование ключей с длиной модуля 1024 бит.

Вычислительная сложность, длина ключа, уязвимость к атакам.

Код:

using System;

using System.Numerics;

using System.Text;

class RSA

{

static void Main()

{

// Генерация ключей

BigInteger p = 3557;

BigInteger q = 2579;

BigInteger n = p \* q;

BigInteger phi = (p - 1) \* (q - 1);

BigInteger e = GeneratePublicKey(phi);

BigInteger d = GeneratePrivateKey(e, phi);

Console.WriteLine("Публичный ключ (e, n): (" + e + ", " + n + ")");

Console.WriteLine("Приватный ключ (d, n): (" + d + ", " + n + ")");

// Шифрование и расшифрование сообщения (для тестирования)

BigInteger originalMessage = 111111;

Console.WriteLine("Исходное сообщение: " + originalMessage);

// Создание ЭЦП

BigInteger signature = SignData(originalMessage, d, n);

Console.WriteLine("ЭЦП: " + signature);

// Проверка ЭЦП

bool isSignatureValid = VerifySignature(originalMessage, signature, e, n);

Console.WriteLine("Проверка ЭЦП: " + isSignatureValid);

}

static BigInteger GeneratePublicKey(BigInteger phi)

{

// Выбираем открытую экспоненту e

BigInteger e = 3;

return e;

}

static BigInteger GeneratePrivateKey(BigInteger e, BigInteger phi)

{

// Вычисляем закрытый ключ d с использованием расширенного алгоритма Евклида

BigInteger d = ModularInverse(e, phi);

return d;

}

static BigInteger ModularInverse(BigInteger a, BigInteger m)

{

BigInteger m0 = m;

BigInteger x0 = 0;

BigInteger x1 = 1;

while (a > 1)

{

BigInteger q = a / m;

BigInteger t = m;

m = a % m;

a = t;

t = x0;

x0 = x1 - q \* x0;

x1 = t;

}

if (x1 < 0)

{

x1 += m0;

}

return x1;

}

static BigInteger SignData(BigInteger data, BigInteger d, BigInteger n)

{

BigInteger s = BigInteger.ModPow(data, d, n);

return s;

}

static bool VerifySignature(BigInteger data, BigInteger signature, BigInteger e, BigInteger n)

{

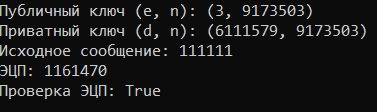
BigInteger decryptedSignature = BigInteger.ModPow(signature, e, n);

return data == decryptedSignature;

}

}

Вывод программы:



**Вывод:** в ходе работы был изучен способ **реализации ЭЦП на примере RSA.**