МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1—98 01 03 «Программное обеспечение информационной

безопасности мобильных систем»

**Отчет**

по дисциплине «Основы информационной безопасности»

**Исполнитель**

студент(ка) 2 курса 7 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Банкузов М.О.

подпись, дата

**Руководитель**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ржеутская Н. В.

должность, ученая степень, ученое звание подпись, дата

**Практическое задание № 5**

**Тема «Криптографическая защита информации»**

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

**Теоретические сведения**

# 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.

**Электронная цифровая подпись** – реквизит электронного документа, предназначенный для защиты данного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа ЭЦП и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе (Федеральный закон "Об электронной цифровой подписи").

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.

**Разновидности ЭЦП**

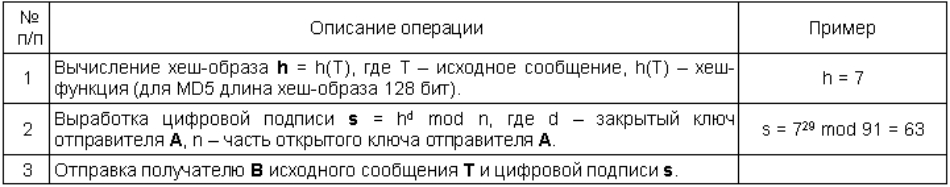
Кроме классической схемы ЭЦП различают еще несколько специальных:

* 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.

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

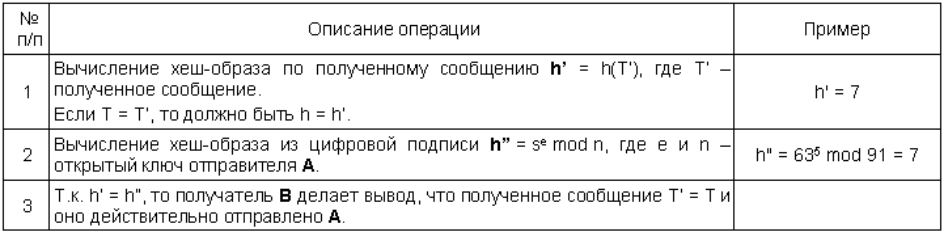
**Этап 2.** Отправка сообщения и электронной подписи (выполняет отправитель **А**).

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



1. Дайте определение понятию "электронная цифровая подпись".

**Электронная цифровая подпись** – реквизит электронного документа, предназначенный для защиты данного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа ЭЦП и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе (Федеральный закон "Об электронной цифровой подписи").

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

Вначале отправитель использует свой закрытый ключ, а затем получатель применяет открытый ключ отправителя.

1. Перечислите специальные схемы ЭЦП. Перечислите недостатки алгоритма цифровой подписи RSA.

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;

Недостатки

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.

**Вывод:** 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. Одним из самых популярных алгоритмов шифрования является RSA (от Rivest, Shamir, Adleman), однако и он имеет свои недостатки, которые могут по неосторожности или невыполнении требуемых условий выборки ключей привести к фальсификации или потере данных.

using System;

using System.Security.Cryptography;

namespace RSAExample

{

class Program

{

static void Main(string[] args)

{

using (RSA rsa = RSA.Create())

{

rsa.KeySize = 2048;

RSAParameters privateKeyParams = rsa.ExportParameters(true);

byte[] privateKeyBytes = ExportPrivateKey(privateKeyParams);

RSAParameters publicKeyParams = rsa.ExportParameters(false);

byte[] publicKeyBytes = ExportPublicKey(publicKeyParams);

byte[] message = System.Text.Encoding.UTF8.GetBytes("Банкузов Михаил Олегович");

Console.WriteLine("Исходное сообщение: " + System.Text.Encoding.UTF8.GetString(message));

Console.WriteLine();

byte[] signature = SignData(message, privateKeyParams);

bool isValid = VerifyData(message, signature, publicKeyParams);

if (isValid)

{

Console.WriteLine("Подпись действительна. Сообщение не было изменено.");

}

else

{

Console.WriteLine("Подпись недействительна. Сообщение было изменено или подпись неверна.");

}

}

}

static byte[] ExportPrivateKey(RSAParameters privateKeyParams)

{

using (RSACryptoServiceProvider rsaCsp = new RSACryptoServiceProvider())

{

rsaCsp.ImportParameters(privateKeyParams);

return rsaCsp.ExportCspBlob(true);

}

}

static byte[] ExportPublicKey(RSAParameters publicKeyParams)

{

using (RSACryptoServiceProvider rsaCsp = new RSACryptoServiceProvider())

{

rsaCsp.ImportParameters(publicKeyParams);

return rsaCsp.ExportCspBlob(false);

}

}

static byte[] SignData(byte[] data, RSAParameters privateKeyParams)

{

using (RSA rsa = RSA.Create())

{

rsa.ImportParameters(privateKeyParams);

byte[] signature = rsa.SignData(data, HashAlgorithmName.SHA256, RSASignaturePadding.Pss);

Console.WriteLine("Хеш исходного сообщения: " + BitConverter.ToString(data).Replace("-", ""));

Console.WriteLine();

Console.WriteLine("Подписанное сообщение (Signature): " + BitConverter.ToString(signature).Replace("-", ""));

Console.WriteLine();

return signature;

}

}

static bool VerifyData(byte[] data, byte[] signature, RSAParameters publicKeyParams)

{

using (RSA rsa = RSA.Create())

{

rsa.ImportParameters(publicKeyParams);

bool isValid = rsa.VerifyData(data, signature, HashAlgorithmName.SHA256, RSASignaturePadding.Pss);

Console.WriteLine("Верифицированное сообщение (подпись): " + BitConverter.ToString(data).Replace("-", ""));

Console.WriteLine();

return isValid;

}

}

}

}

