Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Практическое задание №5**

**Тема «Криптографическая защита информации»**

Руководитель: Ржеутская Н. В.

Выполнил:

Студент 2 курса 1 группы ФИТ

Немкович Анастасия Вадимовна

Минск 2023

**Цель работы**

Изучить и закрепить умение реализации ЭЦП на примере RSA.

**Теоретический минимум. Ответы на контрольные вопросы**

Электронная подпись – информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию. Это реквизит электронного документа, предназначенный для защиты данного документа от подделки. Он получается в результате криптографического преобразования информации с использованием закрытого ключа электронно-цифровой подписи и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе.

Допустим, некоторый пользователь захочет подписать документ. Во-первых, потребуется сгенерировать два ключа: первый ключ – закрытый, личный ключ, второй – открытый. Они задаются совместно и как правило генерируются одновременно, отличие лишь в том, что открытый ключ находится в открытом доступе, а закрытый ключ доступен только тому, кто подписывает документ. Весь алгоритм строится так, чтобы, имея открытый ключ, было невозможно сгенерировать закрытый.

Есть сообщение М, для которого нужно сгенерировать цифровую подпись. Необходимо применить математическое преобразование, пользователь преобразует сообщение М и свой закрытый ключ – в результате появляется специфическая, сравнительно короткая последовательность чисел Sm, которая называется подписью сообщения М. Алгоритмы электронной подписи разрабатываются таким образом, чтобы только сам владелец закрытого ключа имел возможность генерировать такой код, или электронную подпись.

Проведение верификации похоже на создание подписи, но здесь используется открытый ключ. В процессе верификации задействованы три категории входных данных – сообщение M, подлинность подписи которого мы проверяем, электронно-цифровая подпись Sm (ясно, что электронная подпись будет зависеть от сообщения, изменим сообщение – получим другую подпись) и открытый ключ. Это те входные данные, которые требуются для математического преобразования: в результате мы можем проверить, что подпись, которая прилагается к сообщению M, действительно построена при помощи закрытого ключа, принадлежащего пользователю, отправившему сообщение. Личный закрытый ключ взаимосвязан соткрытым ключом**.**

Примечательно, что для проверки подписи нужен только открытый ключ. Закрытый ключ для верификации не требуется, о нем даже не обязательно знать. С помощью обычного открытого ключа можно установить подлинность любой подписи отправителя (в результате проверки получается нечто вроде сигнала «да» или «нет»).

Процесс создания подписи позволяет связать открытый ключ и личность его владельца. Открытый ключ является своего рода удостоверением личности. Электронная подпись – процесс, связывающий личность и сообщение – таким образом получается математический аналог традиционной подписи. Перед тем, как пользователь подписывает сообщение, применяется криптографическая хэш-функция, получает хэш сообщения, который гораздо короче оригинального сообщения. Затем применяется алгоритм электронной подписи, то есть имеет место двухступенчатый процесс – сначала сообщение хэшируется, а затем подписывается. Это позволяет работать не с длинным сообщением, а с небольшим кодом фиксированной длины.

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.

Асимметричные криптографические системы основаны на так называемых односторонних функциях с секретом. Под односторонней понимается такая функция я y=f(x), которая легко вычисляется при имеющемся x, но аргумент x при заданном значении функции вычислить сложно. Аналогично, односторонней функцией с секретом называется функция y=f(x, k), которая легко вычисляется при заданном x, причём при заданном секрете k аргумент x по заданному y восстановить просто, а при неизвестном k – сложно. Подобным свойством обладает операция возведения числа в степень по модулю.

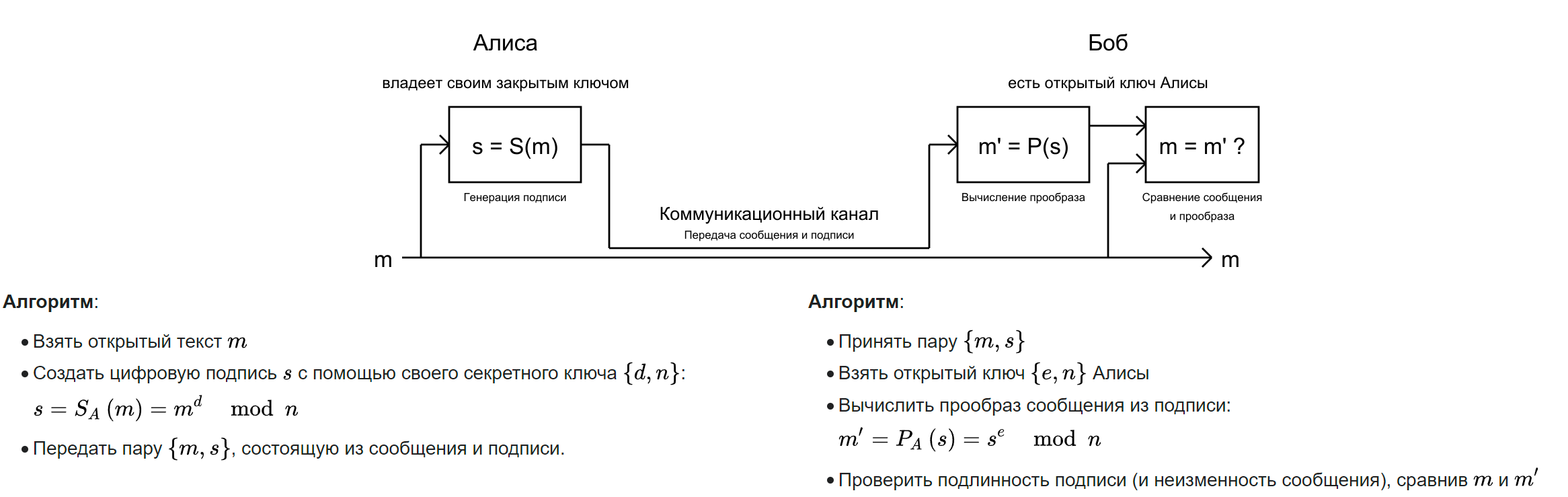




Современные процедуры создания и проверки электронной цифровой подписи основаны на шифровании с открытым ключом.

Система RSA может использоваться не только для шифрования, но и для цифровой подписи.

Предположим, что Алисе (стороне A �) нужно отправить Бобу (стороне B �) сообщение �m, подтверждённое электронной цифровой подписью.

****

Поскольку цифровая подпись обеспечивает как аутентификацию автора сообщения, так и подтверждение целостности содержимого подписанного сообщения, она служит аналогом подписи, сделанной от руки в конце рукописного документа.

Важное свойство цифровой подписи заключается в том, что её может проверить каждый, кто имеет доступ к открытому ключу её автора. Один из участников обмена сообщениями после проверки подлинности цифровой подписи может передать подписанное сообщение ещё кому-то, кто тоже в состоянии проверить эту подпись. Например, сторона �A может переслать стороне �B электронный чек. После того как сторона �A проверит подпись стороны �B на чеке, она может передать его в свой банк, служащие которого также имеют возможность проверить подпись и осуществить соответствующую денежную операцию.

Заметим, что подписанное сообщение �m не зашифровано. Оно пересылается в исходном виде и его содержимое не защищено от нарушения конфиденциальности. Путём совместного применения представленных выше схем шифрования и цифровой подписи в системе RSA можно создавать сообщения, которые будут и зашифрованы, и содержать цифровую подпись. Для этого автор сначала должен добавить к сообщению свою цифровую подпись, а затем — зашифровать получившуюся в результате пару (состоящую из самого сообщения и подписи к нему) с помощью открытого ключа, принадлежащего получателю. Получатель расшифровывает полученное сообщение с помощью своего секретного ключа. Если проводить аналогию с пересылкой обычных бумажных документов, то этот процесс похож на то, как если бы автор документа поставил под ним свою печать, а затем положил его в бумажный конверт и запечатал, с тем чтобы конверт был распечатан только тем человеком, кому адресовано сообщение.

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.

Использование электронной подписи позволяет осуществить:

* Контроль целостности передаваемого документа.
* Защиту от изменений (подделки) документа.
* Невозможность отказа от авторства.
* Доказательное подтверждение авторства документа.

Алгоритм RSA, используемый для цифровых подписей, обладает рядом недостатков, которые важно учитывать:

* Вычислительная сложность: RSA требует значительных вычислительных ресурсов, особенно для генерации ключей и шифрования. Это может быть проблемой для устройств с ограниченными ресурсами, таких как смарт-карты или IoT-устройства.
* Длина ключа: Для обеспечения безопасности RSA требуются длинные ключи. Типичные длины ключей начинаются от 2048 бит и выше, что приводит к большему размеру подписи по сравнению с некоторыми другими алгоритмами, такими как ECC (Эллиптические кривые).
* Медленное шифрование и дешифрование: Процесс шифрования и дешифрования в RSA медленнее по сравнению с симметричными алгоритмами из-за сложных математических операций, которые нужно выполнить.
* Уязвимости: Если алгоритм не реализован правильно, RSA может быть уязвим для различных атак, таких как атака по времени или выборочное шифрование. Также существуют теоретические угрозы от квантовых компьютеров, которые могут нарушить безопасность RSA.
* Масштабируемость: В системах с большим количеством пользователей управление ключами RSA может быть сложным и ресурсоемким из-за необходимости генерации, распределения, хранения и отзыва больших ключей.
* Эффективность передачи: Поскольку RSA увеличивает размер данных из-за шифрования, это может быть неэффективным для систем, где важен размер данных, например, при передаче по каналам с ограниченной пропускной способностью.
* Сложность реализации: Правильная реализация RSA требует тщательной работы и понимания криптографических принципов, что увеличивает риск ошибок и уязвимостей.

Все эти недостатки не означают, что RSA не следует использовать, но они подчеркивают важность внимательного планирования и реализации при использовании RSA в системах цифровой подписи.

**Ход работы**

4. На базе алгоритма RSA получить ЭЦП (в проекте можно использовать существующие криптографические алгоритмы). Удостовериться, что ЭЦП принадлежит именно этому сообщению.

|  |
| --- |
| using System.Security.Cryptography;//используем библиотеку  // Функция создания подписи  public static byte[] SignData(byte[] dataToSign, string privateKey)  {  using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())  {  // Импортируем приватный ключ  rsa.FromXmlString(privateKey);  // Используем SHA256 как алгоритм хеширования  using (SHA256 sha256 = new SHA256Managed())  {  // Получаем хеш от данных  byte[] hash = sha256.ComputeHash(dataToSign);  // Подписываем хеш и возвращаем подпись  return rsa.SignHash(hash, CryptoConfig.MapNameToOID("SHA256"));  }  }  }  // Функция проверки подписи  public static bool VerifyData(byte[] dataToVerify, byte[] signature, string publicKey)  {  using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())  {  // Импортируем публичный ключ  rsa.FromXmlString(publicKey);  // Используем SHA256 как алгоритм хеширования  using (SHA256 sha256 = new SHA256Managed())  {  // Получаем хеш от данных  byte[] hash = sha256.ComputeHash(dataToVerify);  // Проверяем соответствие подписи и хеша  return rsa.VerifyHash(hash, CryptoConfig.MapNameToOID("SHA256"), signature);  }  }  } |

