Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Практическое задание 5

По дисциплине «Основы информационной безопасности»

На тему «Криптографическая защита информации»

Выполнила:

Студентка 2 курса 2 группы

Глухова Д.В.

Вариант 6

Преподаватель: ст. пр. Ржеутская Н. В.

2023, Минск

Практическое задание № 5

Тема «Криптографическая защита информации»

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

**Задание для выполнения:**

1. Объясните последовательность выполнения процедур генерации и проверки ЭЦП.
2. Опишите последовательность действий участников протокола при отправке и проверке ЭЦП.
3. Опишите схему протокола ЭЦП на основе алгоритма RSA.
4. \*На базе алгоритма RSA получить ЭЦП (в проекте можно использовать существующие криптографические алгоритмы). Удостовериться, что ЭЦП принадлежит именно этому сообщению.

**Ответить на следующие вопросы:**

1. Дайте определение понятию "электронная цифровая подпись".
2. Объясните какой порядок использования ключей (открытый; закрытый) при отправке и проверке ЭЦП.
3. Перечислите специальные схемы ЭЦП.
4. Перечислите недостатки алгоритма цифровой подписи RSA.

**Задания**

**1) Последовательность выполнения процедур генерации и проверки ЭЦП на основе алгоритма RSA выглядит следующим образом:**

**Генерация ключей:**

* Генерируется пара ключей: открытый ключ и закрытый ключ.
* Открытый ключ предназначен для проверки подписи и может быть распространен широко.
* Закрытый ключ предназначен для создания подписи и должен храниться в секрете.

**Создание подписи:**

* Исходное сообщение, которое требуется подписать, преобразуется в хеш-значение с использованием надежного алгоритма хеширования (например, SHA256).
* Закрытый ключ используется для создания подписи путем шифрования хеш-значения с использованием приватного RSA-ключа и определенного алгоритма заполнения.
* Полученная подпись является уникальной для данного сообщения и ключа. Она может быть прикреплена к сообщению и отправлена получателю.

**Проверка подписи:**

* Получатель сообщения получает само сообщение и прикрепленную к нему подпись.
* Получатель использует открытый ключ отправителя (который должен быть известен получателю) для расшифровки полученной подписи и получения хеш-значения.
* Исходное сообщение, полученное от отправителя, также хешируется с использованием того же алгоритма хеширования.
* Полученное хеш-значение сравнивается с расшифрованным хеш-значением из подписи.
* Если значения совпадают, подпись считается действительной и можно утверждать, что сообщение не было изменено после его подписи.

**2) Последовательность действий участников протокола при отправке и проверке ЭЦП на основе алгоритма RSA:**

**Отправка:**

* Отправитель генерирует пару ключей: открытый ключ и закрытый ключ.
* Отправитель вычисляет хеш-значение сообщения, которое требуется подписать.
* Отправитель создает подпись, используя закрытый ключ и алгоритм RSA.
* Отправитель отправляет сообщение и подпись получателю.

**Проверка:**

* Получатель получает сообщение и прикрепленную подпись.
* Получатель имеет доступ к открытому ключу отправителя (через доверенный канал или другими средствами).
* Получатель вычисляет хеш-значение полученного сообщения.
* Получатель использует открытый ключ отправителя для проверки подписи, расшифровывая подпись и получая хеш-значение.
* Получатель сравнивает полученное хеш-значение с вычисленным хеш-значением сообщения.
* Если значения совпадают, подпись считается действительной и можно утверждать, что сообщение не было изменено после его подписи.

**3) Схема протокола электронной цифровой подписи (ЭЦП) на основе алгоритма RSA:**

**Генерация ключей:**

* Отправитель генерирует пару ключей: открытый ключ (Kpub) и закрытый ключ (Kpriv) на основе алгоритма RSA.
* Открытый ключ предназначен для проверки подписи, он может быть распространен широко.
* Закрытый ключ предназначен для создания подписи, он должен храниться в секрете.

**Создание подписи:**

* Отправитель имеет сообщение, которое требуется подписать.
* Хэширование сообщения: Исходное сообщение подвергается хэшированию с использованием надежного алгоритма хэширования (например, SHA256), получая хеш-значение сообщения.
* Создание подписи: Закрытый ключ отправителя используется для создания подписи путем шифрования хеш-значения с использованием алгоритма RSA и конкретного алгоритма заполнения.
* Полученная подпись представляет собой уникальную цифровую строку, связанную с конкретным сообщением.

**Отправка сообщения с подписью:**

* Отправитель прикрепляет созданную подпись к исходному сообщению.
* Отправитель отправляет сообщение и подпись получателю.

**Проверка подписи:**

* Получатель получает сообщение и прикрепленную к нему подпись.
* Получатель имеет доступ к открытому ключу отправителя (через доверенный канал или другими средствами).
* Хэширование полученного сообщения: Полученное сообщение подвергается хэшированию с использованием того же алгоритма хэширования, что и отправитель, получая хеш-значение.
* Расшифровка подписи: Открытый ключ отправителя используется для расшифровки полученной подписи и получения хеш-значения.
* Сравнение хеш-значений: Полученное хеш-значение сравнивается с хеш-значением сообщения.
* Если значения совпадают, подпись считается действительной, и можно утверждать, что сообщение не было изменено после его подписи.

**4) На базе алгоритма RSA получить ЭЦП:**

|  |
| --- |
|  |
| using System;  using System.Security.Cryptography;  using System.Text;  public class RSADigitalSignatureExample  {  public static void Main()  {  try  {  // Создание экземпляра класса RSA для генерации ключей  using (var rsa = new RSACryptoServiceProvider())  {  // Генерация открытого и закрытого ключей  var publicKey = rsa.ExportParameters(false);  var privateKey = rsa.ExportParameters(true);  // Сообщение, которое требуется подписать  var message = "мяу";  // Получение хеш-значения сообщения  var hash = GetHash(message); // С использованием алгоритма SHA256  // Создание подписи  var signature = CreateSignature(hash, privateKey); // Результатом является подпись, представленная в виде массива байтов.  // Вывод подписи в шестнадцатеричном формате  Console.WriteLine("Подпись: " + BitConverter.ToString(signature).Replace("-", ""));  // Проверка подписи  //Внутри функции также создается экземпляр RSACryptoServiceProvider и импортируются параметры открытого ключа.  //Затем вызывается метод VerifyHash(), который проверяет подлинность подписи,  //сравнивая ее с хеш-значением сообщения при использовании открытого ключа и определенного алгоритма хеширования и заполнения.  //Результатом является логическое значение, указывающее, является ли подпись действительной.  var isValid = VerifySignature(hash, signature, publicKey);  // Вывод результата проверки подписи  Console.WriteLine("Проверка подписи: " + (isValid ? "Действительна" : "Недействительна"));  }  }  catch (CryptographicException e)  {  Console.WriteLine("Ошибка шифрования: " + e.Message);  }  }  // Функция для получения хеш-значения сообщения  static byte[] GetHash(string message)  {  using (var sha256 = SHA256.Create())  {  return sha256.ComputeHash(Encoding.UTF8.GetBytes(message));  }  }  // Функция для создания подписи  static byte[] CreateSignature(byte[] hash, RSAParameters privateKey)  {  using (var rsa = new RSACryptoServiceProvider())  {  rsa.ImportParameters(privateKey);  return rsa.SignHash(hash, HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);  }  }  // Функция для проверки подписи  static bool VerifySignature(byte[] hash, byte[] signature, RSAParameters publicKey)  {  using (var rsa = new RSACryptoServiceProvider())  {  rsa.ImportParameters(publicKey);  return rsa.VerifyHash(hash, signature, HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);  }  }  } |

**Ответы на вопросы**

**1) Электронная цифровая подпись (ЭЦП)** - это криптографический механизм, который используется для обеспечения целостности, подлинности и невозможности отказа от подписанных электронных документов или сообщений. Она является электронным аналогом обычной подписи на бумаге и позволяет установить авторство и неподдельность данных.

**2) Порядок использования ключей при использовании ЭЦП на основе алгоритма RSA:**

* При отправке: Отправитель использует свой закрытый ключ (Kpriv) для создания подписи, шифруя хеш-значение сообщения. Затем отправитель прикрепляет подпись к исходному сообщению и отправляет оба компонента получателю.
* - При проверке: Получатель получает сообщение и прикрепленную подпись. Затем получатель использует открытый ключ отправителя (Kpub), который должен быть известен и достоверен, для расшифровки подписи и получения хеш-значения. Полученное хеш-значение сравнивается с хеш-значением сообщения. Если значения совпадают, подпись считается действительной.

**3) Некоторые специальные схемы ЭЦП включают:**

* DSA (Digital Signature Algorithm) - схема цифровой подписи, основанная на вычислительных сложных задачах в теории чисел.
* ECDSA (Elliptic Curve Digital Signature Algorithm) - схема цифровой подписи, использующая эллиптические кривые в криптографии.
* EdDSA (Edwards-curve Digital Signature Algorithm) - схема цифровой подписи, основанная на кривых Эдвардса.

**4) Недостатки алгоритма цифровой подписи RSA включают:**

* Высокая вычислительная сложность: Генерация и проверка подписей RSA требуют больших вычислительных мощностей, особенно при работе с большими числами.
* Длина ключа: Для достаточной безопасности RSA требует использования ключей большой длины, что может замедлять процессы шифрования и расшифрования.
* Ограниченная поддержка: RSA не поддерживает некоторые функции, такие как прямая передача больших объемов данных без разбиения на блоки.
* Уязвимость к квантовым вычислениям: RSA может быть подвержен атакам с использованием квантовых компьютеров, если ключи не достаточно длинные или не используется соответствующая защита от квантовых вычислений.