Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ  ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

                        Факультет      Информационных технологий и управления

                        Кафедра         Интеллектуальных информационных технологий

**ОТЧЁТ**

по лабораторной работе №7

по дисциплине “Средства и методы защиты

информации в интеллектуальных системах”

Вариант 10

Выполнила:

Демидовец Д.В. , гр. 221703

Проверила:

Крищенович В.А.

Минск, 2024

**УСТАНОВКА, ИСПОЛЬЗОВАНИЕ И АНАЛИЗ**

**СПЕЦИАЛИЗИРОВАННЫХ СРЕДСТВ КРИПТОГРАФИЧЕСКОГО**

**ПАКЕТА OPENSSL**

Теоретические сведения:

В симметричном шифровании используется один и тот же ключ для шифрования и расшифровки данных.

**DES** — это симметричный алгоритм блочного шифрования, который использует фиксированный ключ длиной 56 бит.

* des-cbc (Cipher Block Chaining)

Это вариант DES, использующий режим цепочки блоков. В этом режиме каждый блок шифруемых данных перед шифрованием комбинируется с результатом предыдущего блока (XOR), что делает шифрование более стойким к атакам, зависящим от структуры данных.

* des-ede3 (3DES, Encrypt-Decrypt-Encrypt)

Этот метод выполняет шифрование трижды с использованием двух или трёх ключей. Тройной DES обеспечивает лучшую защиту по сравнению с обычным DES, хотя также считается устаревшим из-за своей медлительности и возможности атаки методом полного перебора (brute-force).

**AES** — это симметричный блочный шифр, пришедший на смену DES и 3DES. Он работает с фиксированными размерами блока данных (128 бит) и поддерживает ключи разной длины: 128, 192 и 256 бит.

* aes-128-cbc:

Использует ключ длиной 128 бит и режим CBC. Этот вариант подходит для большинства задач, если требуется баланс между скоростью и безопасностью.

* aes-192-cbc:

Использует ключ длиной 192 бита и режим CBC. Более безопасен, но медленнее, чем 128-битный.

* aes-256-cbc:

Использует ключ длиной 256 бит и режим CBC. Это самый надежный вариант AES, особенно против современных атак. Рекомендуется для случаев, где требуется высокая степень безопасности.

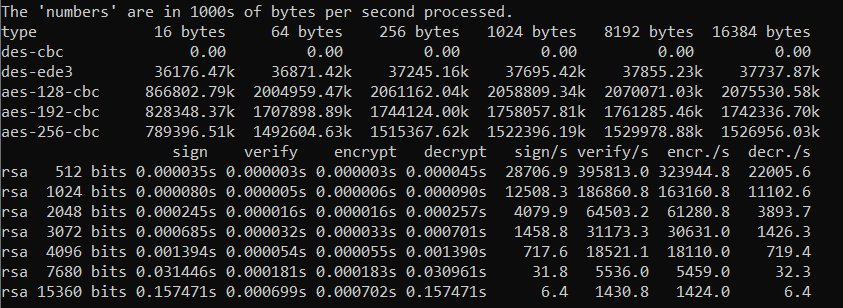
В асимметричном шифровании используются две пары ключей: открытый ключ для шифрования данных(может быть доступен любому), закрытый ключ для расшифровки данных(должен храниться в секрете).

**RSA** — это асимметричный алгоритм, основанный на сложности факторизации больших чисел. Использует пару ключей: открытый (для шифрования) и закрытый (для расшифровки). Применяется для защиты данных, электронной подписи и обмена ключами.

Чем длиннее ключ, тем выше уровень безопасности, но также тем больше вычислительные затраты:

* 512, 1024 бита устарели и считаются небезопасными.
* 2048 бит используется для большинства задач.
* 3072, 4096 бит обеспечивают более высокий уровень безопасности.
* 7680, 15360 бит предназначены для обеспечения безопасности на длительный срок в будущем.

**Результаты тестирования производительности (вывод команды openssl speed aes des rsa):**

****

**Метрики для каждого шифра**

**DES и AES**:

Производительность измеряется в килобайтах в секунду (kB/s) для различных размеров данных: 16, 64, 256, 1024, 8192, и 16384 байт.

Например, aes-128-cbc показывает скорость 866802.79 kB/s для данных размером 16 байт и 2075530.58 kB/s для 16384 байт. Эти значения показывают, сколько килобайтов данных может быть зашифровано за секунду.

**RSA**:

Метрики измеряют время выполнения следующих операций для разных длин ключей:

* sign (подпись): время, необходимое для подписания сообщения.
* verify (верификация): время, необходимое для проверки подписи.
* encrypt (шифрование): время, необходимое для шифрования.
* decrypt (расшифрование): время, необходимое для расшифровки.

Также указаны соответствующие скорости выполнения этих операций (sign/s, verify/s, encrypt/s, decrypt/s) для каждой длины ключа, что показывает, сколько операций в секунду может быть выполнено. Например, для ключа 512 битов RSA может подписывать 28706.9 сообщений в секунду и проверять 395813.0 сообщений в секунду.

**Анализ метрик**

*DES:*

* У алгоритма des-cbc все показатели равны 0, что может свидетельствовать о том, что он не использовался или тест не выполнялся для этого алгоритма.
* У des-ede3 производительность относительно невысока по сравнению с AES, но стабильно увеличивается с размером данных, достигая наибольшей скорости 37737.87 kB/s для 16384 байт.

*AES:*

Все вариации AES демонстрируют высокую производительность, особенно с увеличением размера данных.

AES-128 показывает наивысшую производительность, особенно для больших данных (16384 байт), достигая скорости более 2 млн kB/s, что делает его наиболее эффективным среди вариаций AES.

*RSA:*

С увеличением длины ключа скорость RSA значительно снижается для всех операций.

RSA с меньшими ключами (512, 1024 и 2048 бит) показывает более высокую производительность, особенно для операций верификации и шифрования. Например, 512-битный ключ позволяет выполнять проверку с производительностью 395813 операций в секунду, тогда как для ключа 15360 битов этот показатель падает до 1430.8 операций в секунду.

**Алгоритмы хеширования**

Алгоритмы хеширования — это математические функции, которые преобразуют входные данные произвольного размера в фиксированную строку бит, называемую хешем. Хеш-функции широко применяются в информационной безопасности, особенно для проверки целостности данных и создания цифровых подписей.

**MD5** — один из наиболее известных алгоритмов хеширования, разработанный Рональдом Ривестом в 1991 году. Он преобразует данные в хеш длиной 128 бит.

**SHA-1** был разработан Национальным институтом стандартов и технологий США (NIST) в 1993 году. Алгоритм генерирует хеш длиной 160 бит, что делает его более устойчивым к атакам, чем MD5.

**Результаты тестирования производительности (вывод команды** **openssl speed md5 sha1):**

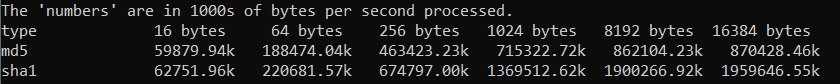
****

Таблица показывает скорость обработки данных (в тысячах байтов в секунду), которую каждый алгоритм достигает при разных размерах входных данных.

**Размер блоков данных**: значения варьируются от 16 байт до 16384 байт, и для каждого размера блоков указано, сколько данных алгоритм может обработать за одну секунду.

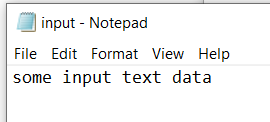
**MD5:**

* Скорость обработки постепенно возрастает с увеличением размера блока, достигая максимума для блоков 16384 байт, где обрабатывается 870428.46 кБ/с.
* В сравнении с SHA-1 MD5 быстрее на небольших блоках данных (16 и 64 байта), но уступает SHA-1 на больших блоках.

**SHA-1:**

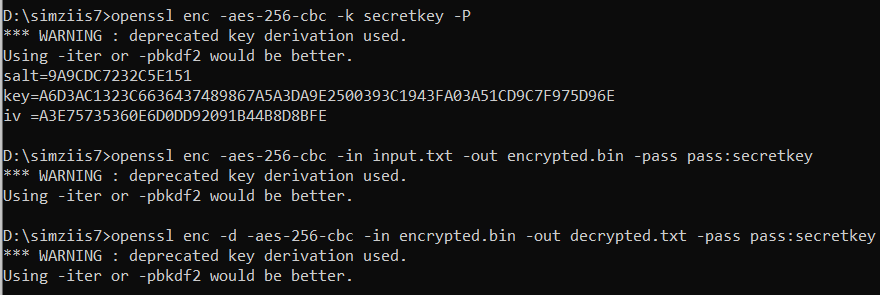
* SHA-1 демонстрирует значительный рост скорости обработки данных на больших блоках, особенно при размерах от 1024 байт и выше.
* При размере блока 16384 байта SHA-1 обрабатывает 1959646.55 кБ/с, что почти в два раза превышает скорость MD5 на этом же размере блока.

**Шифрование и расшифровка текста. Создание ключей.**

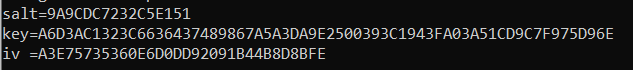
Создадим входной файл “input.txt” в который запишем текст, который мы хотим зашифровать.

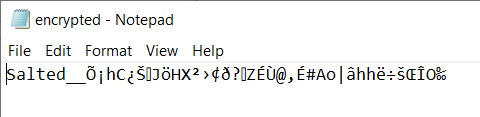
Будем последовательно применять команды openssl:

1. Создать ключ
2. Зашифровать файл
3. Расшифровать файл

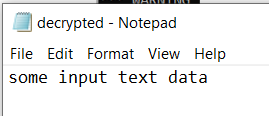
**AES**

В результате отработки команды “enc” c флагами “-k secretkey -P” создался ключ:

  
  
В результате отработки команды “enc” создался бинарный файл “decrypted.bin”, который при открытии в «блокноте» выглядит следующим образом:

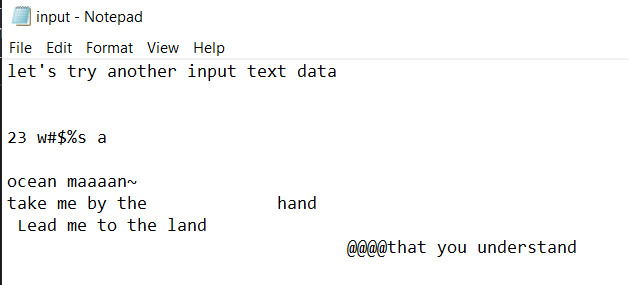


В результате отработки команды “enc” с флагом “-d” создался текстовый файл “decrypted.txt”:

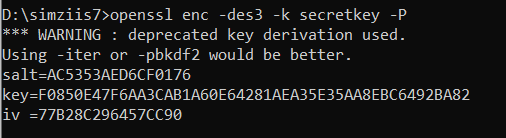


**DES**

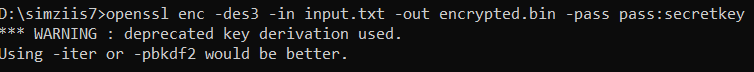
Input.txt:

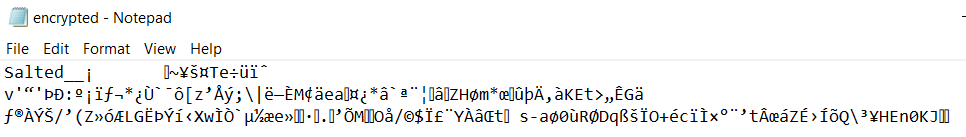
****

Создание ключа:

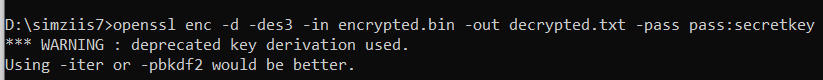


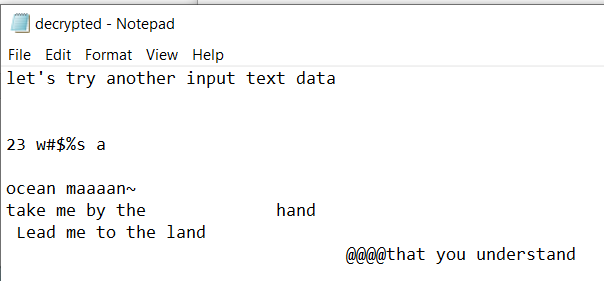
Шифрование:





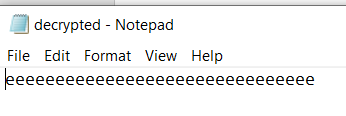
Расшифровка:





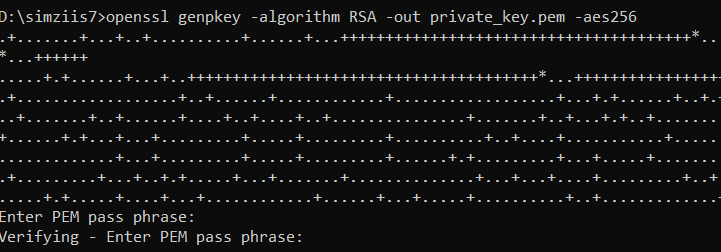
**RSA**

Input.txt:

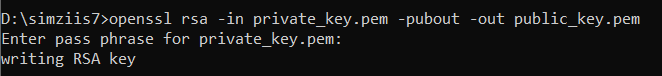
****

Создание ключа:

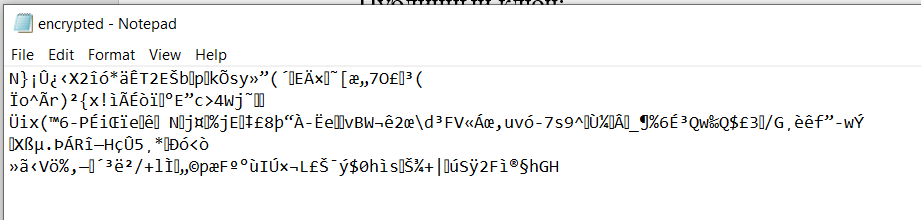
Приватный ключ

****

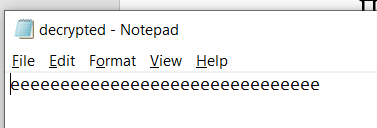
Публичный ключ



Шифрование:

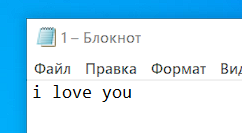


Расшифрование:

**Создание хеш-значений.**

Input.txt:



**Md5**

****

**Sha1**

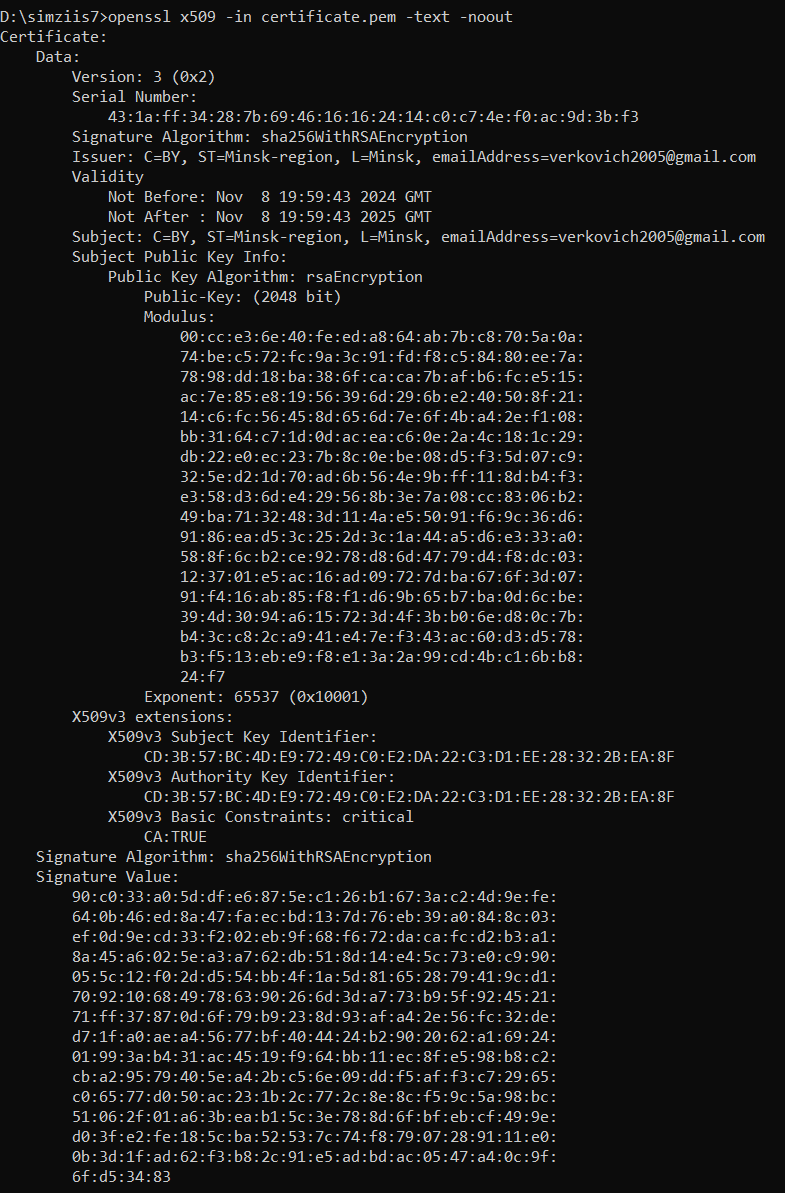
****

**Создание самоподписанного сертификата X509.**

Сертификат будет создан на основе приватного ключа, который был создан при шифровании данных алгоритмом RSA.

Поэтому сразу введём команду для создания цифрового сертификата:



Сертификат:

Из чего состоит сертификат:

* **Версия сертификата (Version),** которая указывает стандарт X.509, применённый для его создания.
* **Серийный номер (Serial Number)** представляет собой уникальный идентификатор, назначаемый сертификату. Этот номер помогает различать сертификаты, даже если остальные их параметры совпадают.
* **Алгоритм подписи (Signature Algorithm)** определяет, какой криптографический алгоритм использовался для создания цифровой подписи сертификата. Например, популярным выбором является SHA256 с RSA.
* **Поле подписант (Issuer)** содержит информацию о центре сертификации (Certificate Authority, CA), который выдал сертификат. Если сертификат самоподписан, то подписант совпадает с владельцем, поскольку сертификат подписан самим собой.
* **Раздел субъект (Subject)** включает информацию о владельце сертификата. Здесь указываются такие параметры, как:
  + общее имя (Common Name, CN), которое может содержать доменное имя или название организации;
  + организация (Organization, O);
  + местоположение (Locality, L);
  + регион (State, ST) и страна (Country, C).

Эти данные помогают идентифицировать владельца.

* **Публичный ключ (Public Key)** — это важный элемент сертификата, содержащий открытый ключ владельца. Этот ключ используется для проверки цифровых подписей и шифрования данных, которые могут быть расшифрованы только соответствующим приватным ключом.
* **Срок действия (Validity)** показывает период, в течение которого сертификат считается действительным. Поле Not Before указывает дату начала действия сертификата, а Not After определяет дату его окончания. После истечения срока сертификат становится недействительным.
* **Расширения (Extensions)** добавляют дополнительные возможности. Они определяют, для чего может использоваться сертификат, например для серверного SSL, подписания кода или электронной почты.
* **Подпись (Signature**) — это результат применения приватного ключа центра сертификации к данным сертификата. Эта подпись служит для проверки его подлинности и подтверждает, что сертификат выдан доверенным центром сертификации и не был изменён после подписания.

**Вывод:** В ходе лабораторной работы были изучены ключевые аспекты работы с криптографическими алгоритмами, включая их использование для шифрования, хеширования и создания цифровых подписей. Были реализованы и применены алгоритмы RSA, DES, AES, MD5 и SHA-1 для выполнения различных криптографических операций. Кроме того, был изучен процесс создания самоподписанных сертификатов X.509, а также их структура и назначение.