Министерство образования Республики Беларусь

Учреждение образования “Белорусский государственный университет информатики и радиоэлектроники”

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Дисциплина: Средства и методы защиты информации в интеллектуальных системах

Отчёт к лабораторной работе №7

Выполнил: Потоцкий Д.А.

Группа: 221702

Проверила: Крищенович В.А.

Минск 2024

Лабораторная работа №7

УСТАНОВКА, ИСПОЛЬЗОВАНИЕ И АНАЛИЗ СПЕЦИАЛИЗИРОВАННЫХ СРЕДСТВ КРИПТОГРАФИЧЕСКОГО ПАКЕТА OPENSSL

**Задание:** 1) Установить OpenSSL на виртуальную машину (или рабочую версию ОС Windows 7/8/10 пользователя) и ознакомиться с возможностями библиотеки (команда «?»).

2) Выполнить тестирование скорости выполнения различных алгоритмов шифрования.

3) Создать криптографические ключи. Выбрать несколько произвольных файлов и выполнить:

а) шифрование (зашифрование и расшифрование) посредством различных симметричных алгоритмов;

б) шифрование (зашифрование и расшифрование) посредством различных асимметричных алгоритмов;

в) хэширование различных файлов различными алгоритмами (обязательно md5 и sha1).

4) Создать самоподписанный сертификат X509. Изучить состав

сертификата и назначение его компонентов.

5) Оформить отчет. В отчет поместить:

а) результаты тестирования производительности;

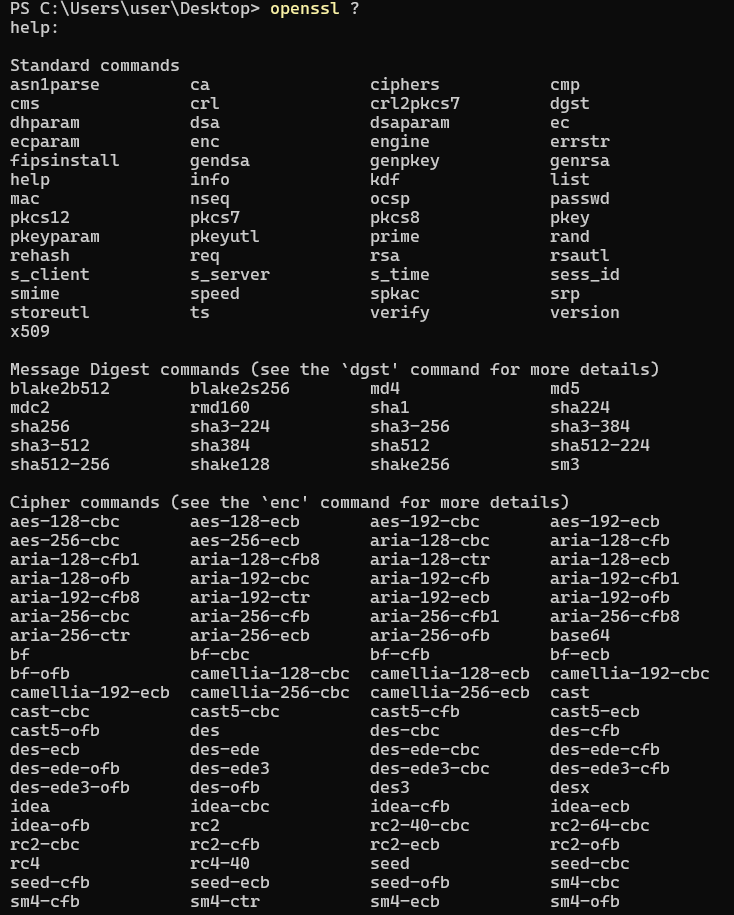
б) времена шифрования (выполнить сравнительную оценку скорости

шифрования DES и AES, AES и RSA, объяснить полученные результаты);

в) полученные хэш значения;

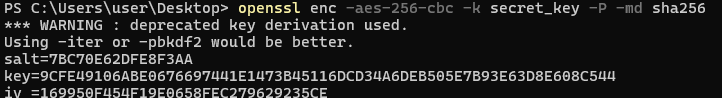
г) сертификат с описанием его компонентов.

**Ход работы:** для выполнения использовался ПК с установленной версией Windows 11 и пакет OpenSSL версии 3.3.2 3

1. Ознакомление с возможностями библиотеки  
   
2. Протестируем скорость алгоритмов шифрования AES, DES, RSA и выведем времена их выполнения:  
   

**Пример сравнительной оценки скорости (DES vs AES vs RSA)**

1. **AES vs DES**: AES быстрее DES из-за более современной структуры и лучшей оптимизации. При прочих равных условиях, AES с ключом в 128 бит работает быстрее, чем DES с его 56-битным ключом. AES также более безопасен, так как DES уже считается устаревшим.
2. **AES vs RSA**: RSA гораздо медленнее, чем AES, особенно при шифровании больших данных. Это объясняется тем, что в RSA нужно выполнять сложные математические операции (модулярная арифметика), в то время как AES использует более простые побитовые операции.
3. Создадим криптографические ключи для алгоритма AES:



**openssl enc**: указывает, что мы будем использовать симметричное шифрование.

**-aes-256-cbc**: алгоритм шифрования AES с 256-битным ключом и режимом CBC (Cipher Block Chaining).

**-k secret\_key**: задает ключ (в данном случае "secret\_key") для шифрования.

**-P**: выводит ключ и вектор инициализации (IV), но не выполняет шифрование.

**-md sha256**: использует алгоритм SHA-256 для генерации ключа на основе указанного пароля.

1. Для шифрования симметричным алгоритмом AES необходимо ввести команду   
   

**-in file.txt**: входной файл для шифрования.

**-out file.enc**: выходной файл с зашифрованным содержимым.

**-k secret\_key**: ключ для шифрования.

Для расшифровки используется следующая команда  


**-d**: указывает на операцию расшифровки. Остальные параметры аналогичны команде для шифрования, только файл будет расшифрован в file\_decrypted.txt.

б) Для ассиметричного шифрования создадим RSA ключ:

   
 **genrsa**: генерация закрытого RSA ключа.

**-out private\_key.pem**: сохранение закрытого ключа в файл private\_key.pem.

**2048**: длина ключа (2048 бит).

Для шифрования необходимо ввести команду



**pkeyutl**: утилита для шифрования/расшифровки данных с использованием RSA.

**-encrypt**: указывает операцию шифрования.

**-inkey public\_key.pem**: указывает файл с публичным ключом для шифрования.

**-pubin**: указывает, что используется публичный ключ (без этого параметра OpenSSL ожидает закрытый ключ).

**-in file.txt**: входной файл для шифрования.

**-out file.enc**: выходной зашифрованный файл.

Для расшифрования



**-decrypt**: указывает операцию расшифровки.

**-inkey private\_key.pem**: закрытый ключ для расшифровки.

**-in file.enc**: зашифрованный файл.

**-out file\_decrypted.txt**: расшифрованный файл.

в) Для хэширования файлов алгоритмами md5 и sha1 выполним следующие команды:   
 md5:  


**dgst**: утилита для создания хэш-сумм.

**-md5**: указывает алгоритм MD5.

**file.txt**: файл, для которого нужно создать хэш.

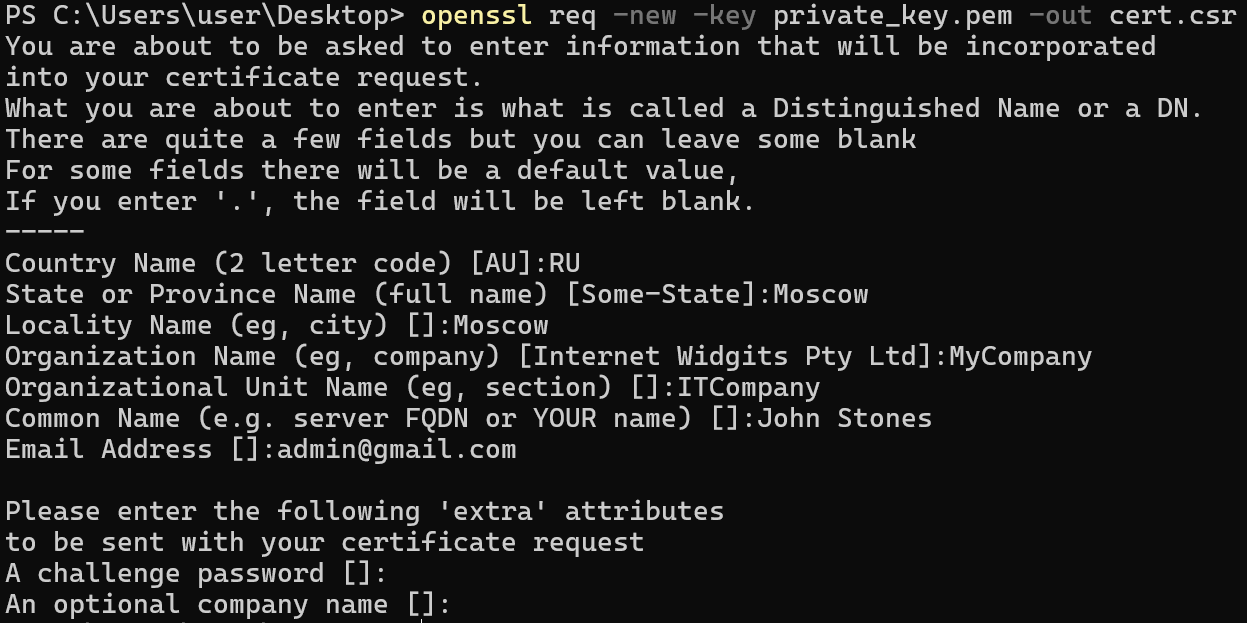
sha1:



-**sha1**: указывает алгоритм SHA-1.

1. Создание самоподписного сертификата X509  
   Создаём закрытый ключ  
   

Создаём запрос на сертификат (CSR)

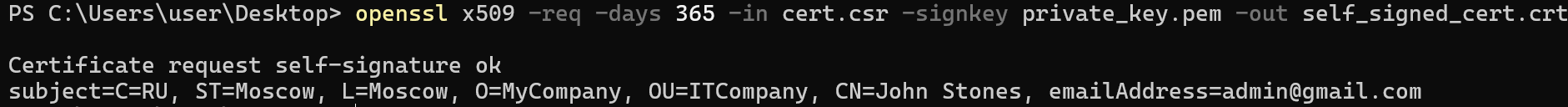


**req**: утилита для создания сертификатных запросов.

**-new**: указывает, что создается новый запрос.

**-key private\_key.pem**: использует закрытый ключ для подписания запроса.

-**out cert.csr**: выходной файл с CSR.

Создаём самоподписный сертификат  


**x509**: утилита для работы с сертификатами X509.

**-req**: использует запрос на сертификат (CSR).

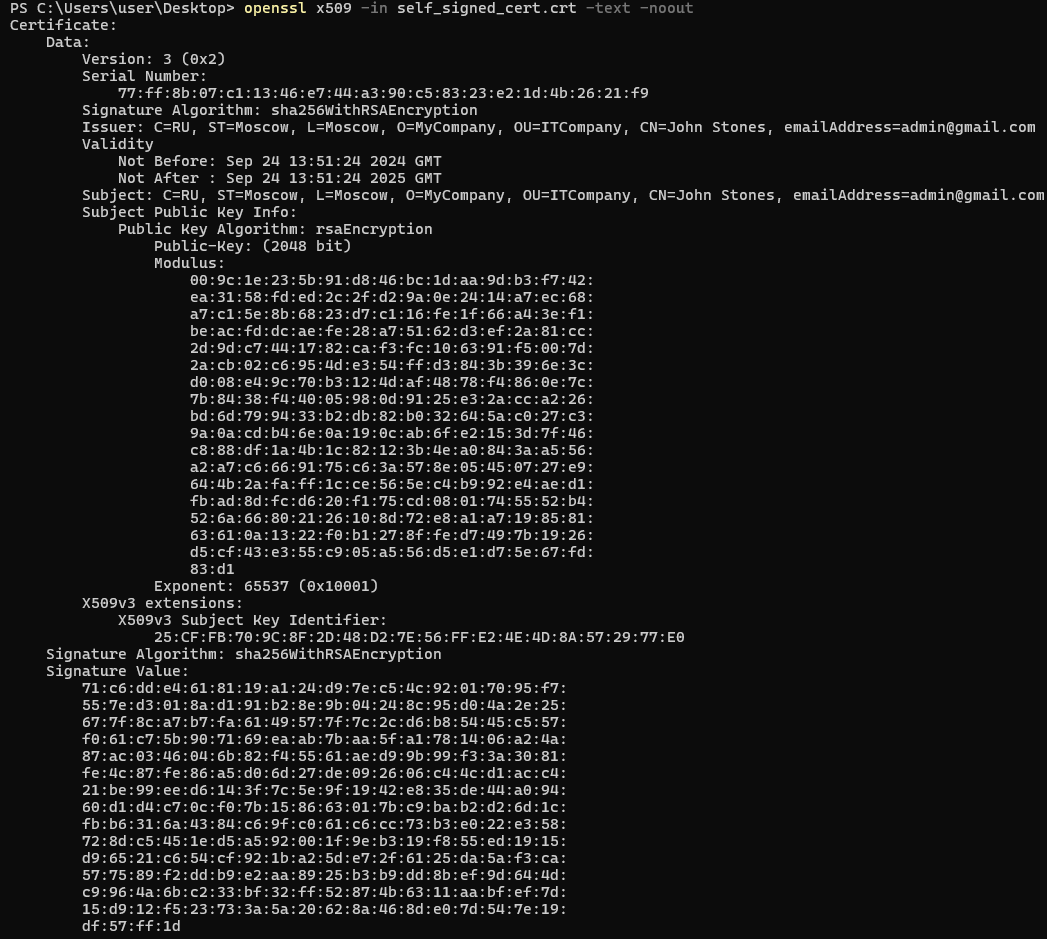
**-days 365**: задает срок действия сертификата (365 дней).

**-in cert.csr**: входной файл с запросом на сертификат.

**-signkey private\_key.pem**: использует закрытый ключ для подписи сертификата.

**-out self\_signed\_cert.crt**: выходной файл с самоподписанным сертификатом.

Изучим содержимое сертификата



**Вывод:** врезультате проделанной работы были получены сведения о специализированных средствах криптографического пакета OpenSSL, а также были приобретены навыки по работе с некоторыми функциями криптографического пакета. Были применены алгоритмы синхронного и асинхронного шифрования, а также был создан самоподписанный сертификат