Министерство образования Республики Беларусь

Учреждение образования

“Белорусский государственный университет

информатики и радиоэлектроники”

Кафедра интеллектуальных информационных технологий

**Лабораторная работа №4**

**"Аспекты создания защищенных веб-приложений"**

|  |  |
| --- | --- |
| Выполнил студент  группы 421701 | Бруцкий Д.С. |
| Проверил | Захаров В.В. |

МИНСК

2016

**1 Задание**

1. Установить OpenSSL и ознакомиться с возможностями библиотеки.

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) хеширование различных файлов различными алгоритмами (обязательно

md5 и sha1).

4. Создать самоподписанный сертификат X509. Изучить состав сертификата и назначение его компонентов.

5. Оформить отчёт.

**2 ход работы**

**1. Тестирование скорости алгоритмов**

После установки OpenSSL было проведено тестирование скорости работы различных алгоритмов шифрования посредством выполнения команды speed.

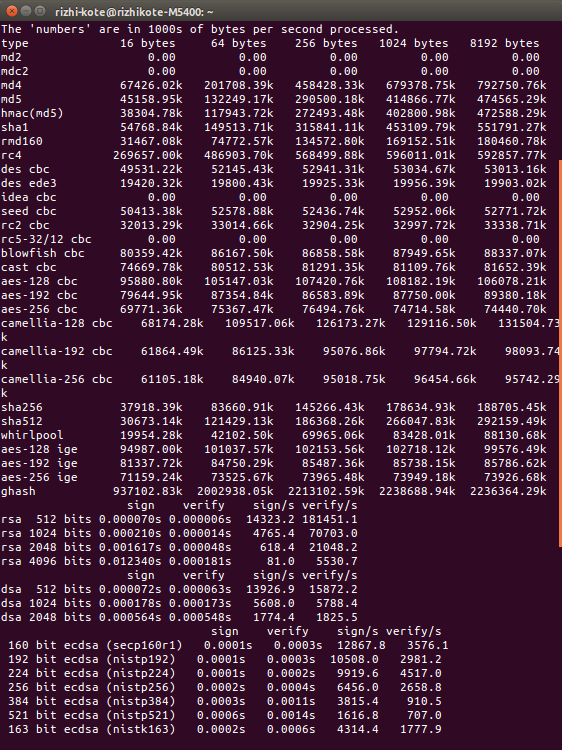


Рисунок 1 – Тестирование скорости выполнения алгоритмов шифрования

Результаты говорят о том, что самая высокая скорость работы была у алгоритмов rc4(при размере блока от 16 байт до 256 байт) и hmac(md5)(при размере блока от 1024 до 8192), эти алгоритмы обработали больше всего информации за секунду; самая низкая скорость была у алгоритма md2. Полученные данные не противоречат сведениям о данных алгоритмах. md2 действительно является относительно медленным алгоритмом хэширования и практически не используется в настоящее время, т.к. считается устаревшим. Об алгоритме rc4, напротив, известно, что он выполняется достаточно быстро и используется широко. Алгоритм HMAC — алгоритм хеширования с ключом для аутентификации сообщения — алгоритм позволяет хешировать входное сообщение L с некоторым ключом K, такое хеширование позволяет аутентифицировать подпись. На расматриваемом интервале заметно увеличение производительности алгоритма с увеличением размера блока.

**3. Симметричное шифрование(алгоритм blowfish)**

При шифровании указывается пароль.

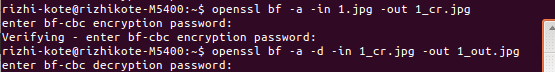


Рисунок 3 – Шифрование алгоритмом blowfish

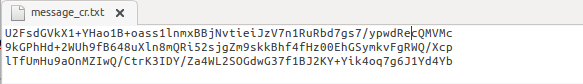


Рисунок 4 – Зашифрованный файл

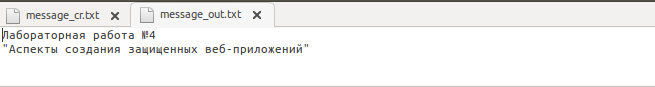


Рисунок 5 – Расшифрованный файл

**3. Асимметричное шифрование(алгоритм rsa)**

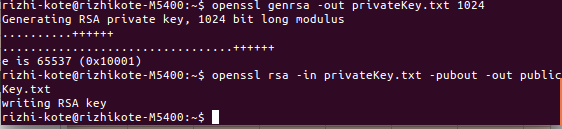


Рисунок 6 – Генерация приватного и публичного ключей rsa



Рисунок 7 – Приватный ключ

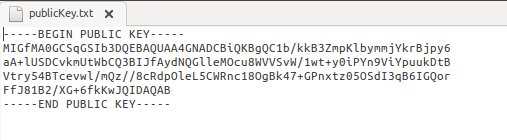


Рисунок 8 – Публичный ключ

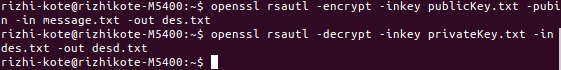


Рисунок 9 – Зашифровка файла и последующая расшифровка

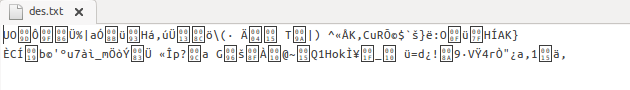


Рисунок 10 – Зашифрованный файл

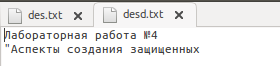


Рисунок 11 – Расшифрованный файл

**4. Хеширование файлов (md5, sha1, rmd160)**

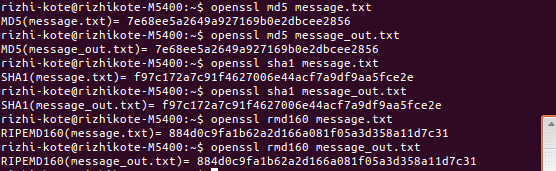
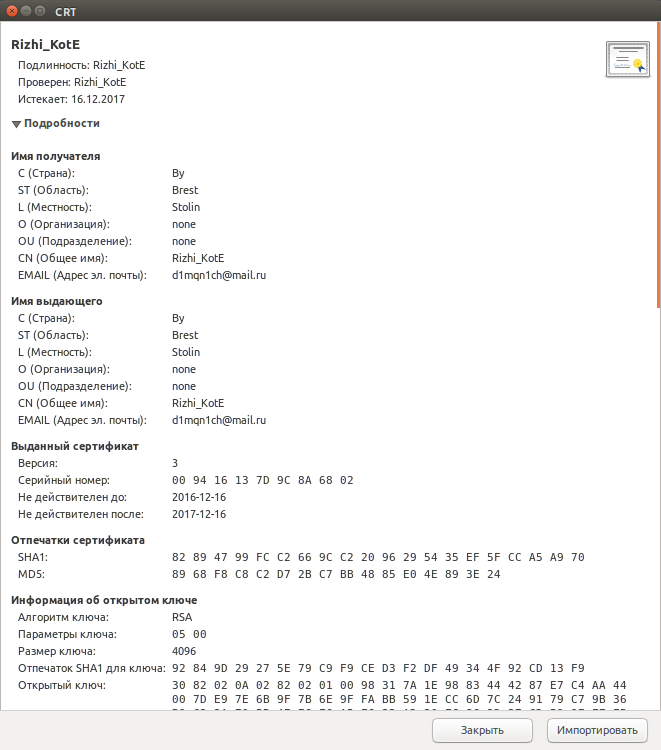


Рисунок 12 – Вычисление хеш-значений различными алгоритмами

**5. Создание самоподписанного сертификата Х509**

Генерируем запрос на получение публичного ключа, публичный ключ и сертификат. Далее необходимо добавить сгенерированный ключ в хранилище ключей.

Рисунок 13. Сертификат

**6 ВЫВОД**

В результате выполнения работы был проанализирован http и https траффик. В результате анализа было выявлено, что защищенные данные сложнее интерпретировать, как следствие обеспечивает безопасность https протокол более высокую.

Также были проанализированы различные алгоритмы шифрования, при этом было выявлено, что алгоритмы rc4 и hmac(md5) обработали наибольшее количество байт за секунду.

Был создан и проанализирован сертификат безопасности. Сертификат включает в себя: версию, срок действия, алгоритм шифрования, имя компании, выдавшей сертификат, а также другие атрибуты.