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.

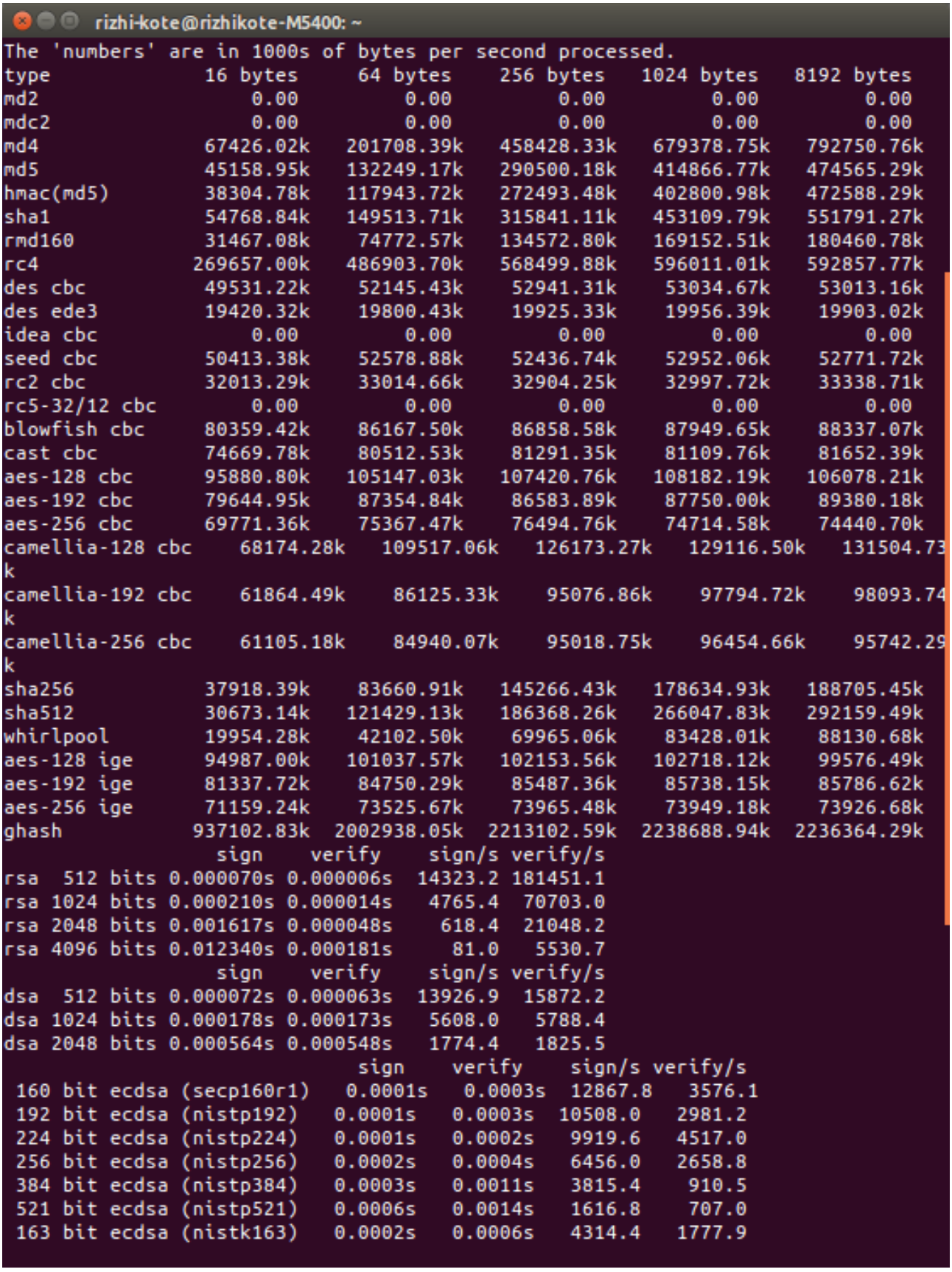

Рисунок 1 – Тестирование скорости выполнения алгоритмов шифрования

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.

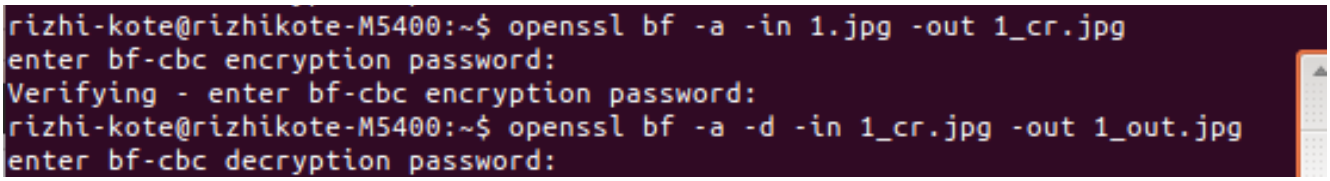

Рисунок 3 – Шифрование алгоритмом blowfish

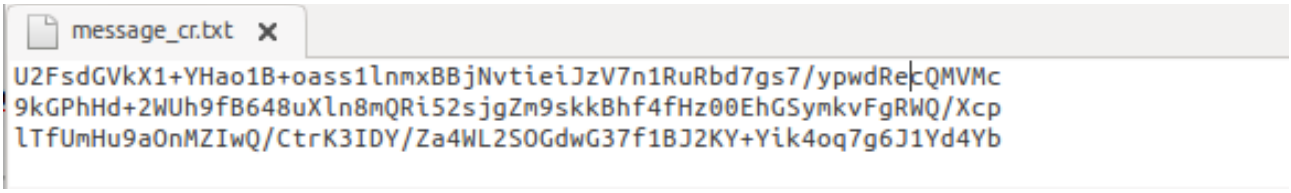

Рисунок 4 – Зашифрованный файл

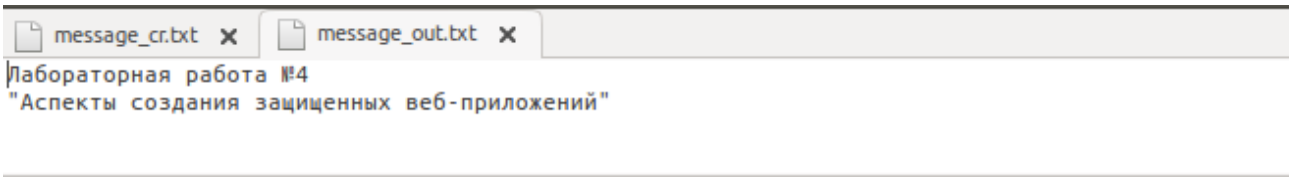

Рисунок 5 – Расшифрованный файл

3. Асимметричное шифрование(алгоритм rsa)

```
rizhi-kote@rizhikote-M5400:~$ openssl genrsa -out privateKey.txt 1024
Generating RSA private key, 1024 bit long modulus
..........++++++
.................................++++++
e is 65537 (0x10001)
rizhi-kote@rizhikote-M5400:~$ openssl rsa -in privateKey.txt -pubout -out public
Key.txt
writing RSA key
rizhi-kote@rizhikote-M5400:~$
```

Рисунок 6 – Генерация приватного и публичного ключей rsa

privateKey.txt

```
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQC1b/kkB3ZmpKlbymmjYkrBjpy6aA+lUSDCvkmUtWbCQ3BIJfAy
dNQGlleMOcu8WVVSvW/1wt+y0iPYn9ViYpuukDtBVtry54BTcevwl/mQz//8cRdp
OleL5CWRnc18OgBk47+GPnxtz05OSdI3qB6IGQorFfJ81B2/XG+6fkKwJQIDAQAB
AoGBALSVr0aR+uGssDgBHeWVdkxyMkJCQphLSC0skjXl21ItJztYL1s49rNbveV+
vO5Jh+fjTuUwQdBM4YrBgvP6q3GFGMT1etM6PGjCyA8UvzVxC3I5WpxKg1WUyxsT
i2yZyqzXKrckxH/tjZPQU60SpLqdO1eW1y5+yQG7nPBiB+cdAkEA4DQSnJlfYlEU
UGUDId33Zekq80mW/FWzHDfsqjZ7MzHqh8mI0ZBQ2CKLxcIjQi8fNMH4QafJ8DyD
xR5hPK8NDwJBAM8rPOkD3c82YrH8WGiODnw6rzvr+sN6SB1QWnSaN7wyuvOUQemR
8yT3LBXCT7zGv9+hqkxg8Dl3Xh4aUELR14sCQQC3lCvBKhxeBoGZqnw8GOHgAmH2
v0Tv7MQx0ODqJp4ON0sQUEbFQWPQ9ch47xOrt6XXrxO1vsus9//1QYDhDXylAkEA
yafr3WF/GoF05o2cvQFyRiiVhD161aaE5cvekNvA3vd+lTU+R45i4BudFP0Uwsuv
344tB7Xgf/+aZLD2aWR7awJAdfppFkohAvALmk8NUoImMlYyZeHrfEqPdmPbyhCb
nlhtb/vviB2kEK+L+l2PoS0lxopNOVjL/3wxcx5rIJN3Lg==
-----END RSA PRIVATE KEY-----
```

Рисунок 7 – Приватный ключ

publicKey.txt

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC1b/kkB3ZmpKlbymmjYkrBjpy6
aA+lUSDCvkmUtWbCQ3BIJfAydNQGlleMOcu8WVVSvW/1wt+y0iPYn9ViYpuukDtB
Vtry54BTcevwl/mQz//8cRdpOleL5CWRnc18OgBk47+GPnxtz05OSdI3qB6IGQor
FfJ81B2/XG+6fkKwJQIDAQAB
-----END PUBLIC KEY-----
```

Рисунок 8 – Публичный ключ

```
rizhi-kote@rizhikote-M5400:~$ openssl rsautl -encrypt -inkey publicKey.txt -pubi
n -in message.txt -out des.txt
rizhi-kote@rizhikote-M5400:~$ openssl rsautl -decrypt -inkey privateKey.txt -in
des.txt -out desd.txt
rizhi-kote@rizhikote-M5400:~$
```

Рисунок 9 – Зашифровка файла и последующая расшифровка

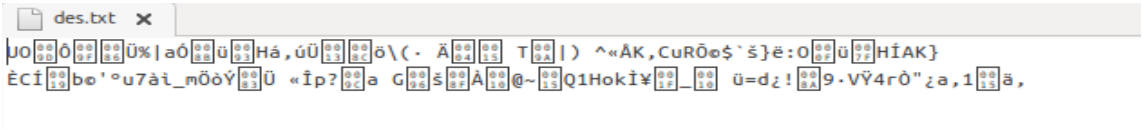

Рисунок 10 – Зашифрованный файл

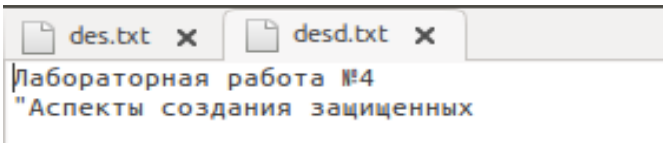

Рисунок 11 – Расшифрованный файл

4. Хеширование файлов (md5, sha1, rmd160)

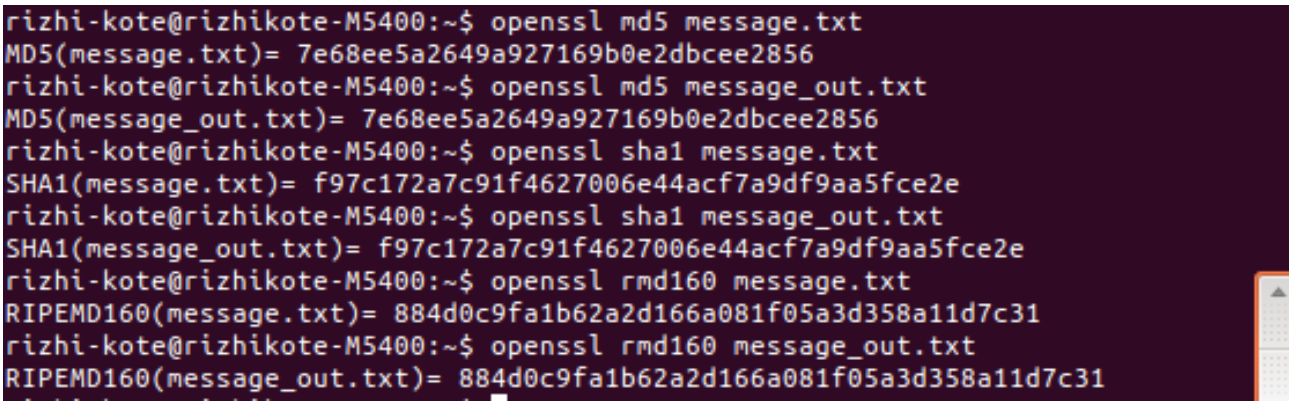

Рисунок 12 – Вычисление хеш-значений различными алгоритмами

5. Создание самоподписанного сертификата X509

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

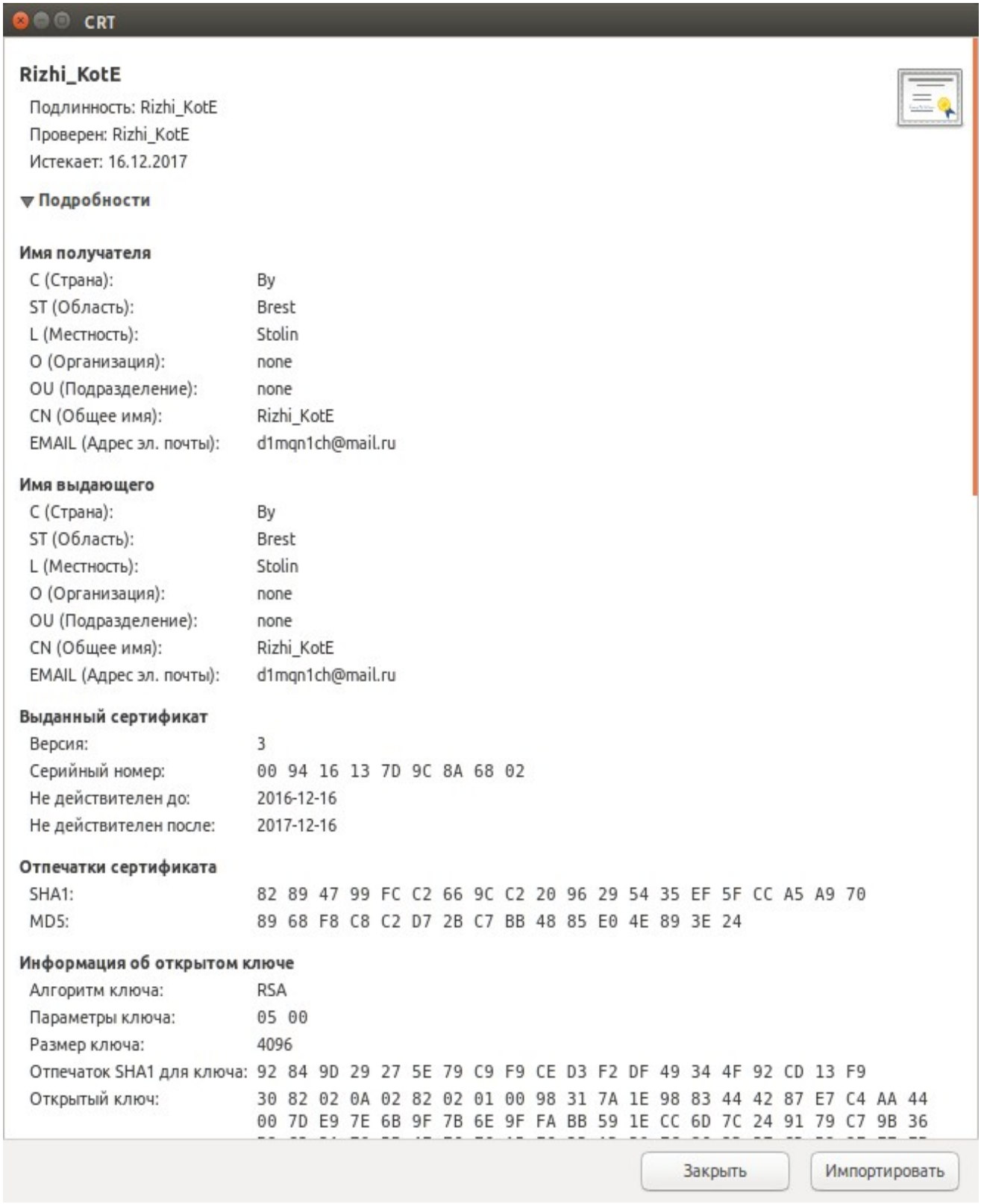

Рисунок 13. Сертификат

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.