Лабораторная работа № 12

Тема: Использование алгоритмов хеширования для подтверждения неизменности файла с применением OpenSSL. Применение электронной цифровой подписи для проверки авторства и неизменности файла.

Цель работы: получить навык применения программного продукта OpenSSL для применения алгоритмов хеширования. Получение навыков применения программного продукта OpenSSL для электронного подписывания документов и проверки подписи, изучить принципы ЭЦП.

Применения алгоритмов хеширования.

Хеширования преобразует данные (большого или малого размеров), в относительно короткий отрезок данных, таких как строка или целое число.

Это осуществляется с помощью односторонней хэш-функции. “Односторонняя” означает, что очень трудно (практически невозможно) получить первоначальные данные.

Типичный пример хэш-функции — md5(), которая очень популярна в разных языках программирования.

Хэш-функцией называется *односторонняя функция*, предназначенная для получения *дайджеста* или "отпечатков пальцев" файла, сообщения или некоторого блока данных.

Для хеширования используются следующие команды:

$ openssl dgst -<алгоритм> <параметры>

$ openssl <алгоритм> <параметры>

Если используется первый вариант команды, один дефис перед именем алгоритма обязателен. Во втором варианте дефисов перед именем алгоритма быть не должно.

Среди алгоритмов хеширования могут применяться следующие:

- md2 (128 бит);
- md4 (128 бит);

- md5 (128 бит);

- mdc2 (128 бит);

- sha (160 бит);

- sha1 (160 бит);

- ripemd160 (160 бит).

В качестве параметров используются следующие ключи (дефисы перед ключами обязательны):

-c - вывести результат с разделяющими байты двоеточиями;

- hex - вывести результат без разделителей;

- binary - вывести результат как строку символов (не все символы являются печатаемыми);

- out <имя_файла> - вывести результат в указанный файл;

- sign <имя_файла> - подписать результат указанным закрытым ключом (поддерживаются только ключи в формате PEM);

- passin <источник_пароля> - указание на источник парольной фразы для ключа;

- verify <имя_файла> - проверить значение, используя открытый ключ в указанном файле;

- prverify <имя_файла> - проверить значение, используя закрытый ключ в указанном файле;

signature <имя_файла> - файл с подписью, которую необходимо проверить;

- hmac <ключ> - создать хешированный код аутентификации сообщения, используя указанный ключ;

- rand <имя_файла> - использовать указанный файл как источник энтропии для создания зерна ГПСЧ;

- <имя_файла> - выполнить заданную операцию над указанным файлом (указывается последним параметром).

В качестве алгоритма может использоваться одно из следующих значений, выдаваемых командой list-message-digest-commands.

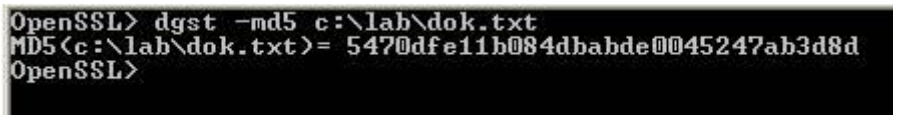
Практическая часть

1. Изучите теоретическое руководство.

2. Подготовьте (создайте или выберите) текстовый файл с семантически понятным содержимым. Дайте ему имя dok.txt.

3. С помощью OpenSSL вычислите значение хэш-функции MD5 от подготовленного текста. Измерьте время хеширования и запомните (запишите) его.

- openssl> dgst -md5 dok.txt



4. Выполните действие 3 для алгоритма SHA1.

- $ openssl> dgst –sha1 dok.txt

5. Сравните время хеширования с применением двух алгоритмов.

6. Измените содержимое исходного файла.

7. Посчитайте хеш-суммы MD5 и SHA1 от измененного файла. Убедитесь, что значения сумм от исходного и измененного файлов не совпадают.

8. Получите аутентификатор выбранного файла с применением алгоритма DES-CBC с помощью OpenSSL и вручную.

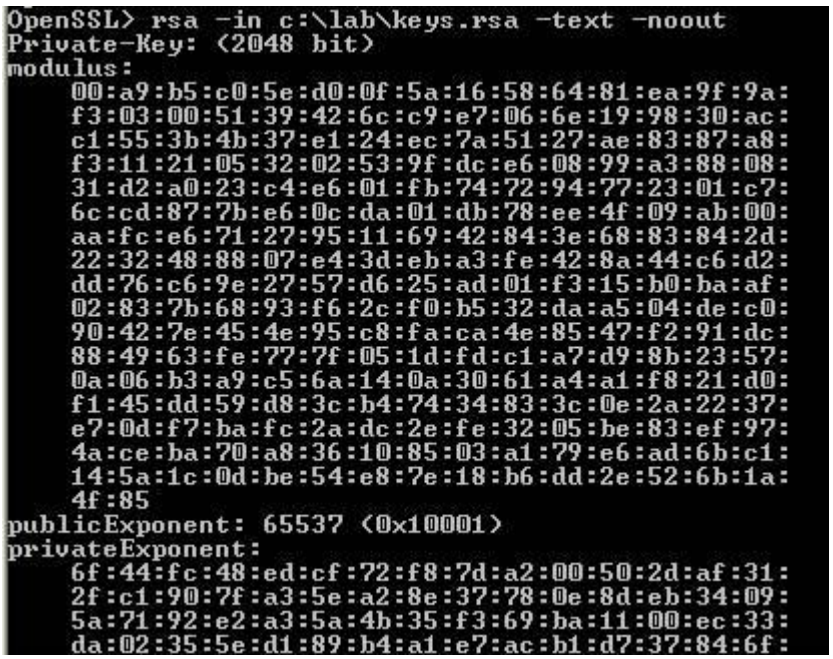
9. Выполните операции с хэш-функцией, используя закрытый и открытый ключ

1) Сгенерировать закрытый ключ длиной 2048 бит без парольной фразы для алгоритма RSA и записать их в файл keys.rsa:

- $ openssl> genpkey -out [путь] keys.rsa -algorithm RSA -pkeyopt rsa_keygen_bits:2048

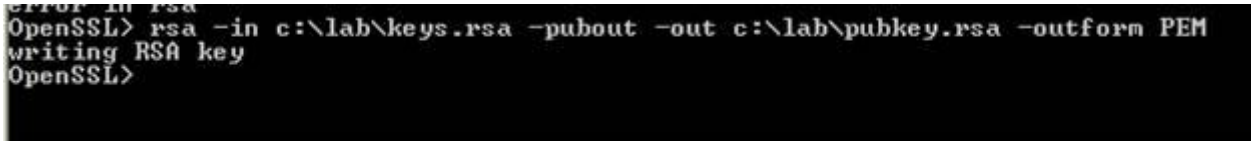
2) Вывести информацию о закрытом ключе:

- $ openssl> rsa -in [путь]keys.rsa -text –noout



3) Извлечь открытый ключ в формате PEM из закрытого ключа:

- $ openssl> rsa -in [путь]keys.rsa -pubout -out [путь]pubkey.rsa -outform PEM



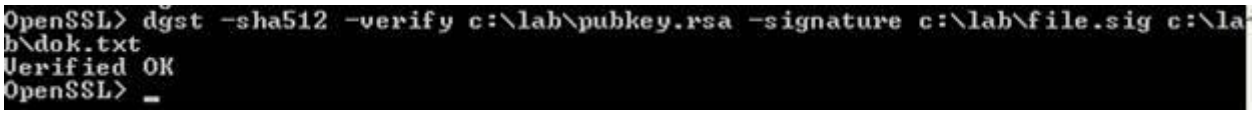
Откройте папку и убедитесь, что ключ pubkey.rsa создан.

4) Подписать хэш-сумму sha512 от файла file закрытым ключом алгоритма RSA, записать подпись в файл file.sig:

- $ openssl> dgst -sha512 -sign [путь]keys.rsa -out [путь]file.sig [путь]file

5) Проверить подпись хэш-суммы sha512 из файла file.sig для файла file по алгоритму RSA:

$ openssl >dgst -sha512 -verify [путь]pubrsa.pem -signature [путь]file.sig [путь]file



Данная команда выводит «Verification OK» при правильной подписи или «Verification Failure» в любом другом случае.

6) Снять защиту ключа парольной фразой:

$ openssl> rsa -in [путь]keys.rsa -passin pass:1234 -out keys_nopass.rsa

Изучение ЭЦП.

Сертификаты X.509 — это цифровые документы, которые представляют пользователя, компьютер, службу или устройство. Они могут выдаваться корневым или подчиненным центром сертификации (ЦС) либо центром регистрации и содержат открытый ключ субъекта сертификата. Они не содержат закрытый ключ субъекта, который должен храниться безопасно. Они имеют цифровую подпись и обычно содержат следующую информацию:

- сведения о субъекте сертификата;
- открытый ключ, который соответствует открытому ключу субъекта сертификата;
- сведения о ЦС, выдавшем сертификат;
- поддерживаемые алгоритмы шифрования и (или) цифровой подписи;
- сведения для определения состояния отзыва или допустимости сертификата.

Для работ, связанных с центром сертификации X.509, используется следующая команда:

$ openssl ca <параметры>

В качестве параметров используются следующие ключи (дефисы перед ключами обязательны):

• - config <имя_файла> — указывает на файл с настройками центра сертификации (описание файла дано ниже; также на этот файл указывает содержимое переменной окружения OPENSSL_CONF);

• - name <имя_секции> — указывает на секцию файла настроек, которую нужно использовать, иначе используется секция [ca] или секция, указанная в параметре default_ca в секции [ca];

• - in <имя_файла> — указывает на входной файл, содержащий запрос на подпись;

• - ss_cert <имя_файла> — указывает на входной файл, содержащий самоподписанный сертификат, требующий подписи центра сертификации;

• - out <имя_файла> — указывает выходной файл, в котором будет храниться сертификат;

• - outdir <имя_каталога> — указывает выходной каталог, в котором сертификат будет размещѐн под именем <серийный_номер>.pem;

• - cert <имя_файла> — указывает на сертификат центра сертификации;

• - keyfile <имя_файла> — указывает на зарытый ключ центра сертификации;

• - key <пароль> — парольная фраза, которой защищѐн ключ;

• - selfsign — указывает на то, что сертификат должен быть подписан тем же ключом, что и запрос на сертификацию

• - passin <источник_пароля> — указывает на источник парольной фразы (более безопасный метод, чем параметр -key);

• - verbose — повышенный уровень детализации выполняемых операций;

• - notext — не выводить сертификат в текстовой форме в выходной файл;

• - startdate — установить дату начала действия сертификата (в формате ГГММДДЧЧММСС);

• - enddate — установить дату окончания действия сертификата (в формате ГГММДДЧЧММСС);

Для работ, связанных с центром сертификации X.509, используется следующая команда:

$ openssl ca <параметры>

В качестве параметров используются следующие ключи (дефисы перед ключами обязательны):

• - config <имя_файла> — указывает на файл с настройками центра сертификации (описание файла дано ниже; также на этот файл указывает содержимое переменной окружения OPENSSL_CONF);

• - name <имя_секции> — указывает на секцию файла настроек, которую нужно использовать, иначе используется секция [ca] или секция, указанная в параметре default_ca в секции [ca];

• - in <имя_файла> — указывает на входной файл, содержащий запрос на подпись;

• - ss_cert <имя_файла> — указывает на входной файл, содержащий самоподписанный сертификат, требующий подписи центра сертификации;

• - out <имя_файла> — указывает выходной файл, в котором будет храниться сертификат;

• - outdir <имя_каталога> — указывает выходной каталог, в котором сертификат будет размещѐн под именем <серийный_номер>.pem;

• - cert <имя_файла> — указывает на сертификат центра сертификации;

• - keyfile <имя_файла> — указывает на зарытый ключ центра сертификации;

• - key <пароль> — парольная фраза, которой защищѐн ключ;

• - selfsign — указывает на то, что сертификат должен быть подписан тем же ключом, что и запрос на сертификацию

• - passin <источник_пароля> — указывает на источник парольной фразы (более безопасный метод, чем параметр -key);

• - verbose — повышенный уровень детализации выполняемых операций;

• - notext — не выводить сертификат в текстовой форме в выходной файл;

• - startdate — установить дату начала действия сертификата (в формате ГГММДДЧЧММСС);

• - enddate — установить дату окончания действия сертификата (в формате ГГММДДЧЧММСС);

$ openssl verify -CAfile cacert.pem -CRLfile crl.pem cert.pem

Практическая часть

Для реализации электронной цифровой подписи используется утилита Open SSl.

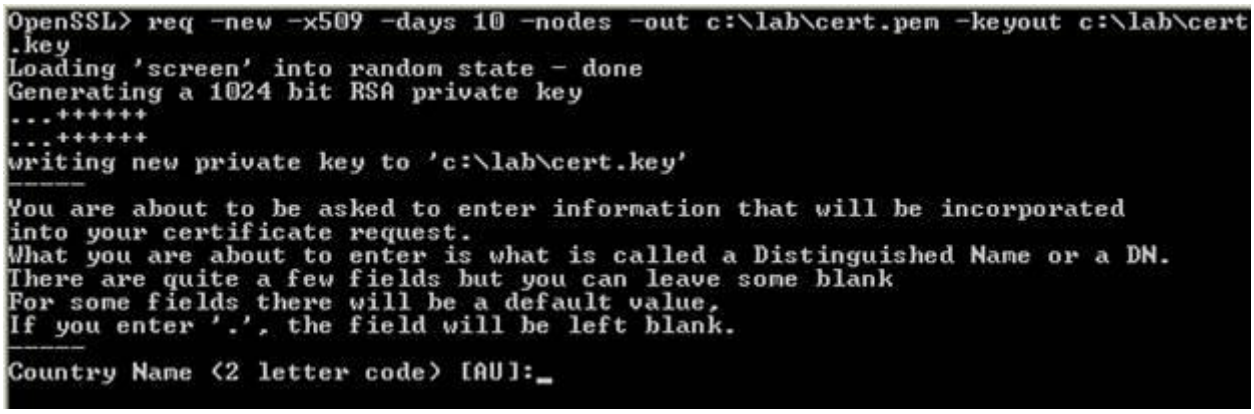
1. Формирование самоподписного сертификата

Openssl> req -new -x509 -days 365 -nodes -out cert.pem -keyout cert.key

Здесь:

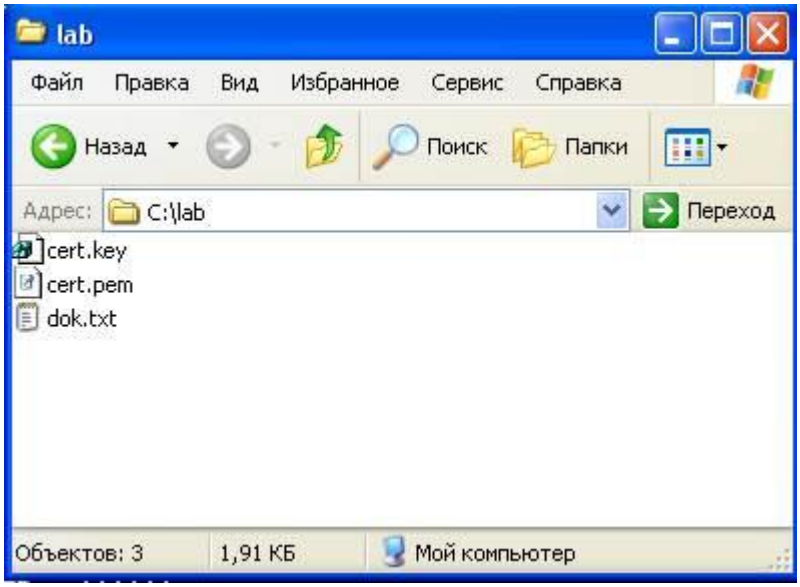
- cert.pem — сертификат, содержащий открытый ключ и информацию о том, кому выдан;

- cert.key — закрытый ключ, предназначенный для создания подписей или расшифровки.



2. Далее необходимо ответить на вопросы:

3. Проверим наличие в папке lab созданного сертификата **cert.pem,**и закрытого ключа **cert.key**



4. Выполним проверку сертификата:

openssl > verify c:\lab\cert.pem

```
OpenSSL> verify c:\lab\cert.pem
c:\lab\cert.pem: C = KZ, ST = TARAZ, L = TARAZ, O = TARGU, OU = INFORMATIKA, CN
= TIMUR, emailAddress = tina@gmail.com
error 18 at 0 depth lookup:self signed certificate
OK
OpenSSL>
```

5. Для вывода подробной информации о сертификате поступим так

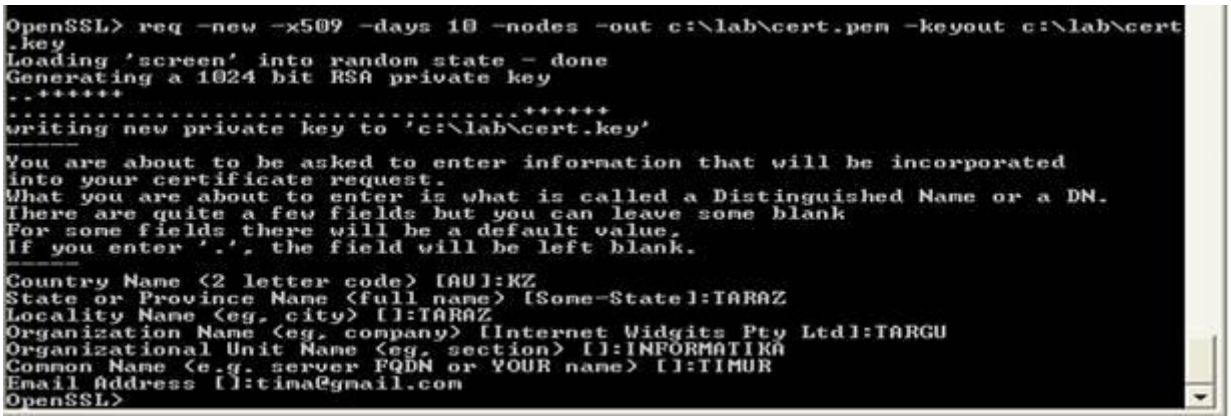
openssl >x509 -in c:\lab\cert.pem -noout –text

```
Ярлык для openssl.exe
OpenSSL> x509 -in c:\lab\cert.pem -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            de:66:5a:ae:93:0a:97:b6
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=KZ, ST=TARAZ, L=TARAZ, O=TARGU, OU=INFORMATIKA, CN=TIMUR/email
Address=tina@gmail.con
        Validity
            Not Before: Apr 27 14:57:03 2016 GMT
            Not After : May  7 14:57:03 2016 GMT
        Subject: C=KZ, ST=TARAZ, L=TARAZ, O=TARGU, OU=INFORMATIKA, CN=TIMUR/emai
lAddress=tina@gmail.con
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:d0:55:25:89:57:9f:9a:ca:16:86:4f:ee:61:cd:
                    a0:72:84:81:b4:f8:b9:87:8f:82:8e:4d:35:3a:c0:
                    e7:6c:dc:72:7a:ab:e9:11:c2:a9:21:32:35:60:b0:
                    8d:f2:86:81:34:ff:8d:5f:98:83:3a:37:6f:39:f7:
                    0a:cc:68:b8:08:db:c0:85:17:5f:1e:a7:9c:7a:d8:
                    eb:00:58:40:f4:f8:d2:74:3d:9b:1c:b9:d2:91:4c:
                    67:c2:04:c9:21:cc:7b:d6:69:21:6d:c6:b2:77:8c:
                    0a:08:1a:cf:8f:20:20:78:c6:ee:43:0d:5a:72:1e:
                    72:e4:af:ef:89:17:08:8b:c3
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                80:4F:F0:1E:22:C0:BB:8B:32:FC:72:59:A4:24:5D:7B:4E:7F:11:FE
            X509v3 Authority Key Identifier:
                keyid:80:4F:F0:1E:22:C0:BB:8B:32:FC:72:59:A4:24:5D:7B:4E:7F:11:F
E
```

В выводе и открытый ключ, и подпись сертификата и информация о владельце и сроки использования.

6. Генерация сертификата:

Openssl> req -new -x509 -days 10 -nodes -out c:\lab\cert.pem -keyout c:\lab\cert.key



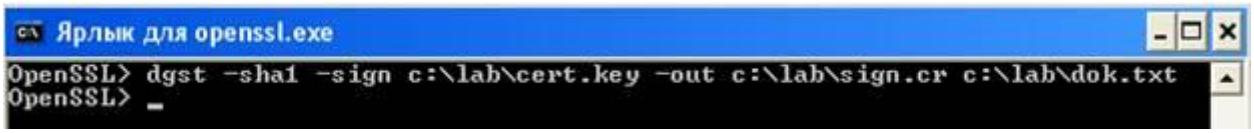
7. Создание электронной подписи файла. Сначала создаем секретный ключ

Openssl> dgst -sha1 -sign c:\lab\cert.key -out c:\lab\sign.cr c:\lab\dok.txt

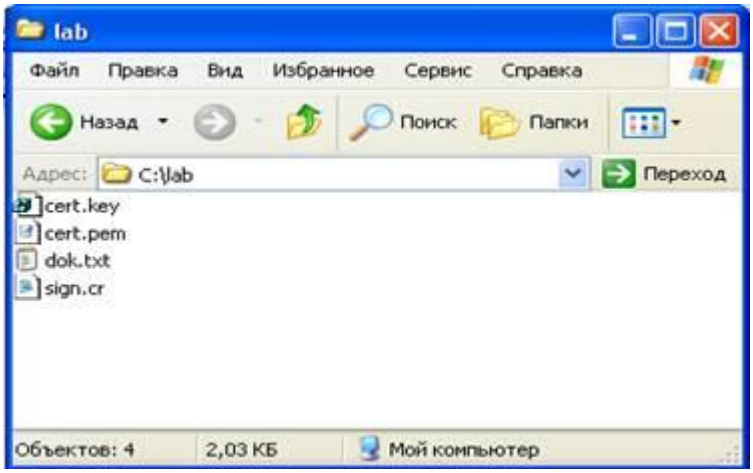
Здесь:

- sign cert.key - указываем закрытый ключ,
- filename - подписываемый файл,
- out sign.cr - указываем файл, в который сохранится подпись,
- sha1 - метод хэширования содержимого файла.

Подписать можно лишь очень ограниченный объём информации, поэтому, как правило хэш-сумму файла. То есть сначала создаётся SHA1-хэш файла.



В результате создаётся файл подписи длинной **sign.cr** 120 байт.



8. Проверка электронной подписи файла. Для начала нам надо преобразовать сертификат, с помощью которого мы будем проверять подпись в открытый ключ

Openssl> x509 -pubkey -noout -in c:\lab\cert.pem > c:\lab\pubkey.pem

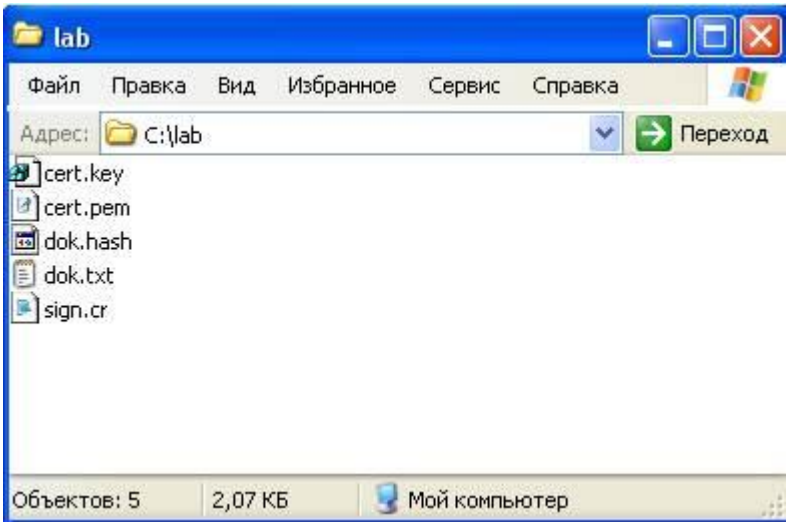
```
OpenSSL> x509 -pubkey -noout -in c:\lab\cert.pem>c:\lab\pubkey.pem
Error opening Certificate c:\lab\cert.pem>c:\lab\pubkey.pem
4292:error:0200107B:system library:fopen:Unknown error:.\crypto\bio\bss_file.c
98:fopen('c:\lab\cert.pem>c:\lab\pubkey.pem','rb')
4292:error:20074002:BIO routines:FILE_CTRL:system lib:.\crypto\bio\bss_file.c:
0:
unable to load certificate
error in x509
OpenSSL>
```

При использовании вместо публичного ключа сертификата мы получим сообщение **unable to load key file**, так что преобразование - операция обязательная.

9. Затем, с помощью этого ключа, произведём проверку подписи:

Openssl> rsautl -verify -certin -inkey cert.pem -out filename.hash -in sign.cr

```
OpenSSL> rsautl -verify -certin -inkey c:\lab\cert.pem -out c:\lab\dok.hash -i
c:\lab\sign.cr
Loading 'screen' into random state - done
OpenSSL>
```



Если подпись верна, то получим:

Verified O