Лабораторная работа №10

Шифрование с помощью OpenSSL. Использование алгоритмов шифрования для сокрытия содержимого файла с применением OpenSSL.

Цель работы: получить начальные навыки работы в программе OpenSSL

OpenSSL - это система защиты и сертификации данных, название SSL переводится, как система безопасных сокетов (secure socket layer). OpenSSL используется практически всеми сетевыми серверами для защиты передаваемой информацией. Существует программное API SSL (SSLEAY), позволяющее создавать безопасные сокеты с шифрацией передаваемых данных в собственных разработках. Кроме того, OpenSSL—это криптографический пакет с открытым исходным кодом для работы с SSL/TLS. Позволяет создавать ключи RSA, DH, DSA и сертификаты X.509, подписывать их, формировать CSR и CRT. Также имеется возможность шифрования данных и тестирования SSL/TLS соединений.

TLS – это так называемый «транспорт» или от английского — Transport Layer Security — криптографический протокол, обеспечивающий защищённую передачу данных между узлами в сети Интернет.

SSL от английского Secure Sockets Layer — уровень защищённых сокетов — криптографический протокол, который обеспечивает установление безопасного соединения между клиентом и сервером.

Сокеты (англ. socket - разъём) - название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет - абстрактный объект, представляющий конечную точку соединения. Сокеты позволяют реализовать различные модели межпроцессного взаимодействия: один с одним, один с многими и многие с многими. Их называют еще двухточечной моделью, широковещанием и групповым вещанием.

OpenSSL основана на SSLeay, написанной Эриком Янгом (Eric A. Young) и Тимом Хадсоном (Tim Hudson), которые неофициально закончили работать над ней в декабре 1998 года, когда они начали работать в проекте RSA Security.

Шифрование с помощью OpenSSL

Т.к. OpenSSL поддерживает очень много различных стандартов сертификации, шифрования, хеширования, то использование данной команды достаточно сложно. Внутри OpenSSL существуют отдельные

компоненты, отвечающие за то или иное действие, для получения списка доступных компонентов можно вызвать openssl с параметрами list-standart-commands. Можно также получить список доступных алгоритмов хеширования (list-message-digest-commands) и алгоритмов шифрования (list-cipher-commands).

Применение OpenSSL:

- Создание RSA, DH и DSA ключей

- Создание сертификатов X.509, CSR и CRL

- Шифрование и дешифрование с помощью алгоритмов

- SSL/TLS клиент и сервер тесты

-Обработка, дешифровка и шифровка электронной почты

OpenSSL поддерживает разные алгоритмы шифрования и хеширования:

Симметричные: Blowfish, Camellia, DES, RC2, RC4, RC5, IDEA, AES, ГОСТ 28147-89

Хеш-функции: MD5, MD2, SHA, MDC-2, ГОСТ Р 34.11-94

Асимметричные: RSA, DSA, Diffie-Hellman key exchange, ГОСТ Р 34.10-2001 (34.10-94)

Поддержка алгоритмов ГОСТ появилась в версии 1.0.0, выпущенной 29 марта 2010 года, и была реализована сотрудниками фирмы «Криптоком»

OpenSSL может использоваться во множестве случаев и умеет выполнять следующие задачи:

- создавать и управлять ключами RSA и DSA - команды rsa, dsa, dsaparam;

- создавать сертификаты формата x509, запросы на сертификацию, восстановление - команды x509, req, verify, ca, crl, pks12, pks7;

- зашифровывать данные с помощью симметрического или асимметрического шифрования - команды enc, rsautl;

- высчитывать хеши различных типов - команда dgst;

- работать с S/MIME - команда s/mime;

- проверять работу серверов и клиентов ssl - команды s_client, s_server.

Синтаксис OpenSSL следующий:

$ openssl <команда> [<опции команды>] [<параметры команды>]

При вызове openssl без команды и параметров будет открыта интерактивная оболочка, из которой можно выполнять все те же команды, что и в неинтерактивном режиме. При вызове openssl с несуществующей командой будет выведен перечень команд и поддерживаемых криптоалгоритмов.

Имеются следующие команды:

• псевдокоманда list-standard-commands - выводит перечень поддерживаемых команд;

• псевдокоманда list-message-digest-commands - выводит перечень команд для применения алгоритмов хеширования;

• псевдокоманда list-cipher-commands - выводит перечень команд для применения алгоритмов шифрования и кодирования;

• ca - управление центром сертификации;

• crl - управление списками отзыва сертификатов;

• dgst - вычисление хэш-функций;

• dsa - управление ключами алгоритма DSA;

• ec - обработка ключей алгоритмов на основе эллиптических кривых;

• ecparam - генерация и обработка файлов с параметрами эллиптических кривых;

• enc - применение алгоритмов симметричного шифрования;

• genpkey - генерация закрытых ключей;

• ocsp - управление протоколом OCSP;

• pkcs12 - управление контейнерами PKCS12;

• pkcs7 - управление контейнерами PKCS7;

• pkey - обработка открытых и закрытых ключей для асимметричных алгоритмов шифрования;

• pkeyparam - просмотр параметров ключей;

• pkeyutl - выполнение операций с применением асимметричных алгоритмов шифрования;

• rand - генерация псевдослучайных последовательностей;

• req - управление запросами на подпись (CSR);

• rsa - обработка ключей RSA;

• speed - вычисление скорости алгоритмов;

• verify - проверка сертификатов X.509;

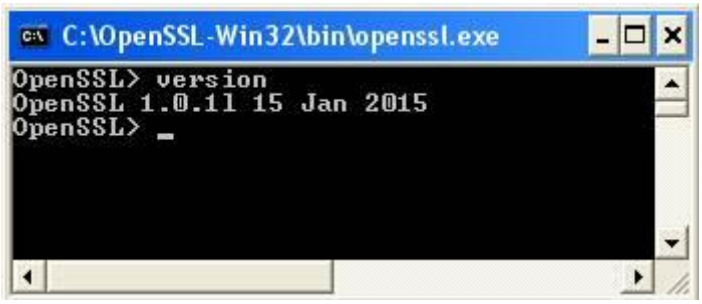
• version - информация о версии OpenSSL;

• x509 - операции над сертификатами X.509.

Практическая часть

1. Установите Visual C++ 2008 Redistributables и OpenSSL на компьютер. Запустите файл openssl.exe из папки bin в директории, куда установился OpenSSL. Должно появиться окно с командной строкой OpenSSL, куда нужно будет вводить последующие команды

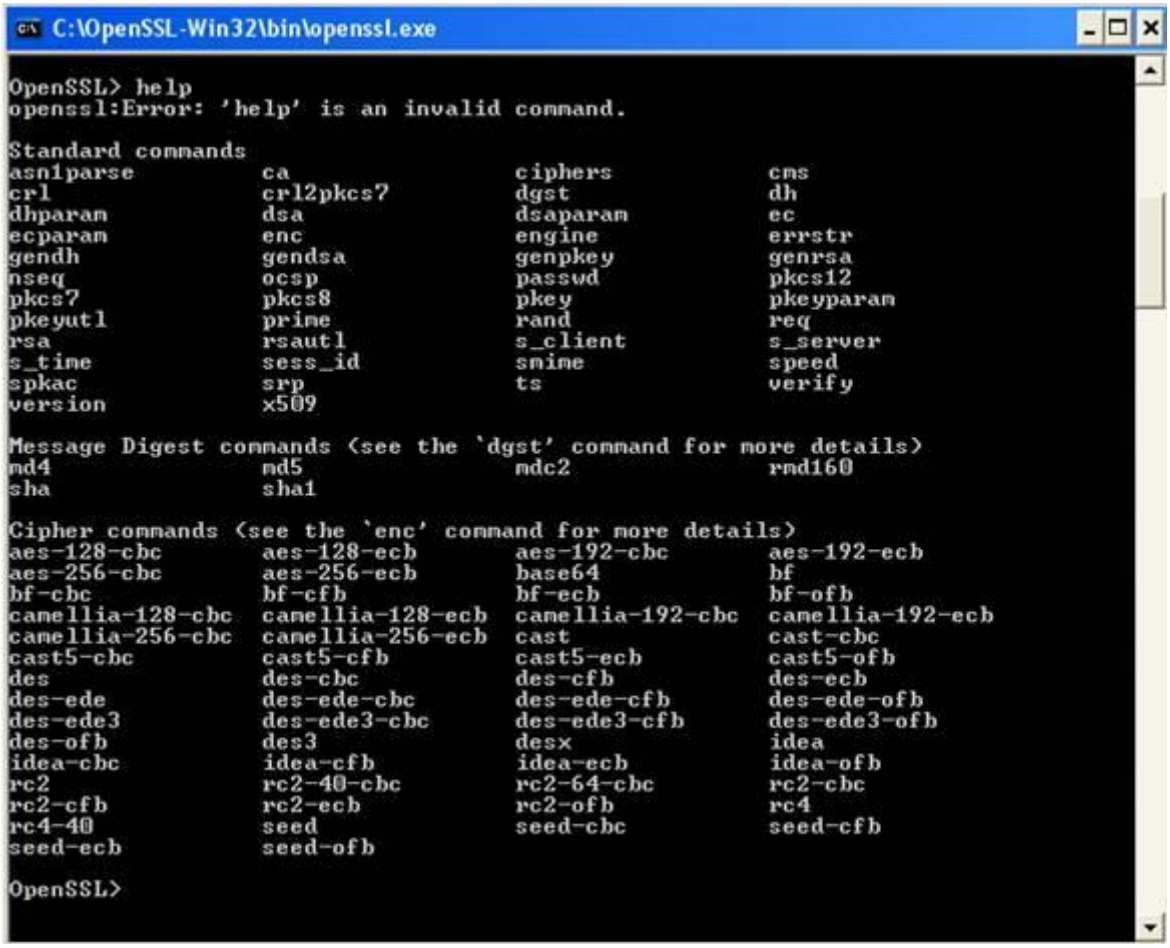
2. Определим используемую версию OpenSSL, с помощью опции version: Openssl> version



3. Расширенная информация о версии: Openssl> version –a

4. Вывод на экран списка доступных команд с неправильным ключом

OpenSSL> help или OpenSSL > help –h



5. Вывод на экран списка доступных команд с подкомандами:

openssl>dgst –h

unknown option '-h'options are-c to output the digest with separating colons-d to output debug info-hex output as hex dump-binary output in binary form-sign file sign digest using private key in file-verify file verify a signature using public key in file-prverify file verify a signature using private key in file-keyform arg key file format (PEM or ENGINE)-signature file signature to verify-binary output in binary form-engine e use engine e, possibly a hardware device.-md5 to use the md5 message digest algorithm (default)-md4 to use the md4 message digest algorithm-md2 to use the md2 message digest algorithm-sha1 to use the sha1 message digest algorithm-sha to use the sha message digest algorithm-sha256 to use the sha256 message digest algorithm-sha512 to use the sha512 message digest algorithm-mdc2 to use the mdc2 message digest algorithm-ripemd160 to use the ripemd160 message digest algorithm

6. Вывод списка доступных шифров: команда ciphers

- все доступные шифры:
Openssl> ciphers -v- только шифры TLSv1: Openssl> ciphers -v -tls1- только шифры длиною больше 128 битов (high ciphers): Openssl> ciphers -v 'HIGH'-

только шифры длиною больше 128 битов, использующие AES: Openssl> ciphers -v 'AES+HIGH'

7. Команда для измерения производительности системы: openssl> speed

Как видно, результаты собраны в таблицу, где по вертикали идут названия алгоритмов, а по горизонтали (в первой части) – размер блока данных для шифрования/подсчета контрольной суммы. На пересечении стоит цифра скорости обработки данных в «тысячи байт в секунду». Во второй части (тест алгоритмов RSA/DSA) результаты выдаются в единицах «подписей в секунду» (при работе с подписями используется уже готовый дайджест сообщения).

8. Команда для измерения производительности сетевого соединения:

$ openssl s_time -connect remote.host:443

Задание. Шифрование и дешифрование файлов с помощью различных алгоритмов

1.Создайте в директории C папку lab (для ОС Linux – домашний каталог)

2. В папке lab создайте текстовый файл dok.txt

3. Зашифруйте файл dok.txt используя алгоритм шифрования AES и сохраните его в text.txt:

$ Openssl> enc -e -aes-256-cbc -in c/lab/dok.txt -out c/lab/text.txt

После ввода команды, она запросит у вас пароль.

Ключи:

enc – говорит утилите, что шифруем с помощью алгоритма (AES256)

-e – указывает шифровать

-in – входящий файл

-out – выходящий, зашифрованный файл

- 256bit AES - это криптографический алгоритм, который использует правительство Соединенных Штатов для шифрования информации на самом секретном уровне.

4. Для дешифрования полученного файла выполните команду:

$ Openssl> enc -d -aes-256-cbc -in c/lab/text.txt -out c/lab/newtext.txt

Ключи:

-d – ключ указывает на дешифровку

5. Проверьте скорость алгоритма aes-256-cbc:

$ Openssl>speed aes-256-cbc

Задание. Шифрование и дешифрование файлов с помощью пароля

1. Зашифровать файл file с использованием алгоритма DES в режиме CBC, используя предоставленный в командной строке пароль для создания ключа, без «соли», результат записать в файл file.enc:

$ openssl> enc -des-cbc -in file -out file.enc -pass pass:1234 -nosalt

2. Расшифровать файл:

$ openssl> enc -des-cbc -d -in file.enc -out file –nosalt

Ключи:

-nosalt - не использовать «соль» при шифровании;

-salt(соль) - увеличивает стойкость шифрования, должна использоваться всегда, если секретный ключ формируется на основе пароля, указывает использовать случайно сгенерированную соль при шифровании (по умолчанию);

- **pass <источник_пароля>** - указание на источник парольной фразы. Синтаксис одинаков и для последующих команд: **pass:<пароль>** - пароль задается прямо в командной строке.

3. Зашифруйте файл dok.txt используя парольную фразу из файла key.txt, без «соли», результат записать в файл file.enc:

$ openssl> enc -des-cbc -in dok.txt -out file.enc –pass pass:key.txt –nosalt

4. Расшифровать файл file.enc, используя парольную фразу из указанного файла, без соли, результат записать в файл newfile.txt:

$ openssl> enc -des-cbc –d -in file.enc –out newfile.txt –pass pass:key.txt –nosalt

Задание. Изучение метода генерации ключей.

(В созданной паре один ключ является закрытым (private) ключом, который бережно хранится владельцем. Второй ключ является открытым (public), т.е. его совершенно спокойно можно передать по доступным врагам каналам своему товарищу.

Для создания ключей алгоритма RSA используется команда genrsa:

openssl genrsa [-out file] [-des | -des3 | -idea] [-rand file] [bits]

Команда genrsa создает секретный ключ длиной bits в формате PEM, шифрует его одним из алгоритмов (des, des3, idea).

PEM - текстовый формат, в нём возможно хранение как сертификатов и секретных ключей, так и прочей информации. Опция out позволяет указать, в какой файл выполнять вывод данных, т. е. созданного секретного ключа.)

1. Создание RSA-ключей, используя команду genrsa. Генерируем секретный ключ RSA: openssl> genrsa -out c:\lab\private.pem 1024

В результате выполнения этой команды на диске будет создан файл private.pem со следующим содержимым:

-----BEGIN RSA PRIVATE KEY-----

MIICXQIBAAKBgQCjOxZWuzGJzYlFYQTYZtGdFSHHAzpg606xncVHrSPJO
OdyOEZV

mM8f7fwzfdJvOnS1hlvfvZRa4b+xZcs6ztED5uA2QClTmyhqDrm/qu3Bh4t5dYO
W

exX6altnXUJ42xHriSdHh4Rzen7d05YF/eYYgikGPb9iyxlA71lWgGzH2wIDAQ
AB

AoGAS9GBQc33Z6PBtCkpX/76NDWBl/gG4MfLqLK41N82NRwqXSKIgO/Wn
WUkdSJn

YWTMmxiKijV7uG8NZC/9Ixa3fJOSoG55R1fG5l1kVCBT7KK2VdhrcazLzddk
GNu/

sqfwo1hDWvhCtFza/g3hDSUbXuhrB+f05CnaofsQ8tw9oTECQQDOa/JvDf9KVjSd

IAEOABYHs+AcXwLrO0EkSeUAkbsPXjdNW2MAB2ENyR1cP06+hB800R15SvZzciKz

cHjKGwQZAkEAym+BRi5HKZZ5MJ8OHw0qIsfTfJIPx516lglvDf79hw0QL6nshY6c

ALfHgZS0Hisd/tbZ4d+niDZR7ljziKaKEwJBAKtCKhhzT3v4O5yk4dsgvSZU2ToQ

pdZOCgS2fhVT0xO0AkPe7ysl9CLA2egf6g/shAuI92Abjo9HgXzIcp6TfXkCQEwT

bXynZZzubKrmC1OGCgC6IfNdnMqClxANiYuz+Skosp2G+VkTx/LJHhPHg40W3RXp

PkaRW49oApBVI2iGVbsCQQCBZohvjWkUsQ6TC0sofLhxqXsiBeEhTmmERDDW8+Jn

l13JoFqYUCH0eYzQSXGdY2qEjFFMPcRUka/mkyJKsH24

-----END RSA PRIVATE KEY-----

2. Генерируем по готовому секретному ключу открытый ключ.

(Для создания открытого ключа на основе созданного секретного ключа используется команда rsa, которая имеет следующий синтаксис:

openssl rsa –in filename [-out file] [-des | -3des | -idea] [-check] [-pubout]

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

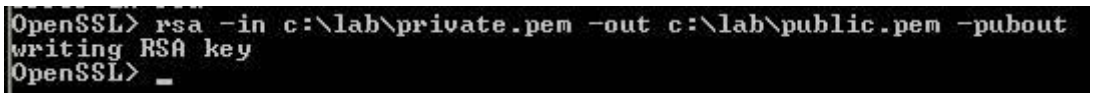
- **pubout** указывает на необходимость создания открытого ключа (public key) в файле, указанном в параметре –out;

- **in** указывает на файл с секретным ключом

- **pubin** - указание на то, что передается открытый ключ.

1) Введите команду для создания открытого ключа:

openssl> rsa -in c:\lab\private.pem -out c:lab\public.pem –pubout



2) Откройте файл ключа.

Содержимое файла public.pem

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDWa3o0k75DJ23LneuIKChgZyEy

11czWRUDuEnerfgFiN/qHVDDaW7v9mevNyw35xYg0ntu65cS33ugx/0/RBYEMW8o

qikRjY7b9rFaiDtYpVStZM+oRhbzYOyqHLTnJdusmg7V3CwQWUE+48Q12WmPmKtw

UKo7yPVv4QyVHkud5wIDAQAB

-----END PUBLIC KEY-----

3. Шифрование данных с использованием открытого ключа RSA:

Используемые параметры:

- **inkey** <имя_файла> - имя файла с ключом (по умолчанию из файла считывается закрытый ключ);

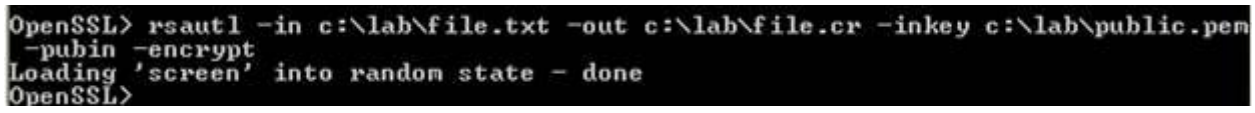
- **encrypt** - зашифровать открытым ключом;

- **decrypt** - расшифровать закрытым ключом;

1) Создайте в редакторе Блокнот текстовый файл и сохраните его в папке под именем file.txt

2) зашифруйте созданный файл, для этого в командной строке введите команду:

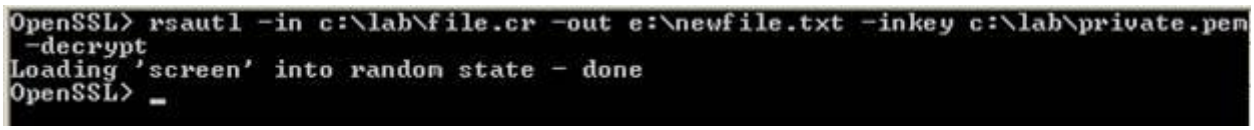
Openssl> rsautl –in c:\lab\file.txt –out c:\lab\file.cr –inkey c:\lab\public.pem –pubin –encrypt



Попробуйте открыть зашифрованный файл и просмотрите его содержимое.

4. Расшифруйте файл file.cr закрытым ключом RSA и сохраните его в директории E:

Openssl> rsautl –in c:\lab\file.cr –out e:\newfile.txt –inkey c:\lab\private.pem –decrypt



5. Откройте файл newfile.txt и убедитесь в расшифровке файла file.cr.