ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра вычислительных систем

ПРАКТИЧЕСКАЯ РАБОТА
«Инфраструктура открытых ключей (PKI)»

Автор: С.Н. Мамойленко

Новосибирск - 2019

Оглавление

1. Введение

С развитием инфокоммуникационных технологий перед человечеством остро встали проблемы защиты передаваемой информации от несанкционированного доступа, обеспечения её достоверности (целостности) и аутентификации участников взаимодействия. Для решения этих проблем разработаны и продолжают разрабатываться методы и алгоритмы криптозащиты и криптоанализа.

Широкое распространение на практике методы криптозащиты получили в виде формирования защищенного соединения через сетевые сокеты (SSL/TLS) и инфраструктуры открытых ключей (PKI), о чем и пойдет речь в данной практической работе.

2. Технология защищенных сокетов (SSL/TLS)

Одной из задач, которая требовала разработки методов защиты информации, стала потребность организации защищённого канала между клиентом и сервером посредствам телекоммуникационной сети Интернет.

Родоначальником технологии организации защищенных соединений через сетевые сокеты считается компания Netscape, которая предложила технологию Secure Socket Layer (SSL) при разработке собственного интернет браузера Netscape Navigator.

Компанией Netscape предложено организовать дополнительный промежуточный слой системного программного обеспечения, располагающегося между сетевым сокетом TCP и прикладным программным приложением, и отвечающего за организацию защищённого канала связи (см. рисунок 1).

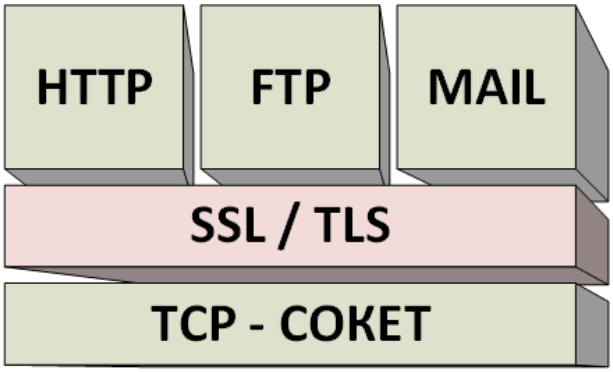

Рисунок 1 – Схема организации защищенных каналов передачи с использованием технологии SSL/TLS

Рассматривая эту технологию с позиции модели организации взаимодействия открытых систем, можно сказать, что слой SSL/TLS реализует функции представительского и сеансового уровней.

Первая версия протокола SSL, опубликованная для широкого применения, была выпущена в 1995 году, но уже получила номер 2.0. Версия 1.0 никогда не публиковалась и, судя по всему, была рабочей версией, используемой внутри компании Netscape для отладки технологии. Версия 3.0 вышла в 1996 году и стала последней версией протокола SSL.

После того, как протокол SSL был стандартизирован IETF (Internet Engineering Task Force), он был переименован в TLS (Transport Layer Security), первая версия которого вышла в 1999 году. Часто TLS-1.0 называют SSL-3.1, но это не совсем верно. Стандарт TLS-1.0 был основан на SSL, но реализовывал другие криптографические алгоритмы и, поэтому, был не совсем совместим с SSL-3.0. Дальнейшее развитие стандарт TLS получил с выходом в 2006 году версии 1.1, в 2008 году версии 1.2, в 2018 году – версии 1.3.

Функциональное назначение слоя SSL/TLS состоит в следующем:

- шифрование – сокрытие информации, передаваемой между участниками взаимодействия;
- аутентификация – проверка подлинности источника полученного сообщения;

- целостность – контроль сохранности (целостности, неизменности) переданной информации.

В каждом конкретном случае приложениями могут использовать как все перечисленные функции, так и разные их комбинации (например, шифрование + аутентификация).

3. Шифрование данных (понятие ключа, сертификата)

С целью решения задачи несанкционированного доступа к информации разработаны и продолжают разрабатываться различные методы шифрования – преобразования информации в другую форму таким образом, чтобы доступ к ней мог получить только тот пользователь, который обладает определёнными сведениями о правилах обратного преобразования (имеет секретный ключ дешифрации).

По сути, алгоритмы шифрования можно разделить на два больших класса: ассиметричные и симметричные. Первые алгоритмы предполагают, что для шифрования информации и для её дешифрования используются одни и те же сведения (сам алгоритм шифрования и его параметры). Алгоритмы, относящие ко второй группе, для шифрования используют одни сведения, а для дешифрования другие. При этом, некоторые сведения, используемые симметричными алгоритмами, все ещё остаются одинаковыми (сам алгоритм и некоторые из его параметров). Параметры алгоритмов, используемые для шифрования и дешифрования, называются **ключом**.

Среди симметричных алгоритмов широкое распространение получили:

- AES (англ. Advanced Encryption Standard) — американский стандарт шифрования;
- ГОСТ 28147-89 — советский и российский стандарт шифрования, также является стандартом СНГ;
- DES (англ. Data Encryption Standard) — стандарт шифрования данных в США.

Ассиметричные алгоритмы шифрования:

- RSA (Rivest-Shamir-Adleman);
- DSA (Digital Signature Algorithm);
- ГОСТ Р 34.10-2012.

Основное отличие симметричных алгоритмов – это наличие двух ключевых данных: одни из них используются для шифрования, другие – для дешифрования. При этом данные, которые используются для шифрования свободно передаются по телекоммуникационному каналу, а данные, используемые для дешифрования, хранятся в тайне.

4. Цифровая подпись

При передаче данных помимо шифрования важной задачей является контроль целостности полученных данных, или, другими словами, проверка изменились ли данные в процессе их передачи или нет. Распространённым методом решения этой задачи является дополнение передаваемой информации служебными данными, описывающими передаваемую информацию. В качестве таких служебных данных чаще всего используется так называемое хеширование – вычисление на основе передаваемых данных определённой математической функции, гарантирующей получение одного строго определённого результата для входящей комбинации данных.

Очевидно, что если передавать результат вычислений хеш-функции также открытым образом, то подменить её не составит никакого труда. Чтобы избежать подобной ситуации, результат хеш-функции шифруется ассиметричным образом и передается в зашифрованном виде. Получатель информации, расшифровывает хеш, рассчитывает хеш на основе полученных данных и, если результат его вычислений совпадет с тем, что получено после расшифровки, то полученные данные будут считаться целостными.

Результат вычисления хеш-функции, зашифрованный закрытым ключом, называется цифровой подписью.

Широкое распространение получили следующие функции хеширования:

- MD5 (Message Digest 5);
- SHA1 (Secure Hash Algorithm 1).

5. Инфраструктура открытых ключей.

В ситуации, когда используется асимметричный алгоритм и открытый ключ свободно передается по телекоммуникационным каналам, возникает задача подтверждения факта, что открытый ключ получен от желаемого собеседника. Для решения этой задачи в диалоге необходимо появление третьей стороны, которая будет подтверждать, что ключ принадлежит тому собеседнику, который нужен. Для хранения открытого ключа и информации, подтверждающей владельца открытого ключа, используется специальные форматы данных, называемые **сертификатами**. Внутри сертификата содержится открытый ключ и его цифровая подпись.

Совокупность технических и программных средств, используемых для создания сертификатов, содержащих открытый ключ, информацию о владельце открытого ключа и о том, кто является гарантом подлинности этой информации, называется **инфраструктурой открытых ключей (PKI, public key infrastructure)**. Сторона, которая выступает гарантом принадлежности открытого ключа его владельцу, называется **центром сертификации** (CA, Certificate authority).

6. Библиотека функций OpenSSL.

На практике для решения вышеперечисленных задач широкое распространение получила библиотека функций OpenSSL. Эта библиотека используется как для разработки приложений (реализации необходимого функционала в программном обеспечении), так и для выполнения системных функций по работе с шифрованием, созданием и манипуляцией цифровых сертификатов, организации инфраструктуры открытых ключей.

В рамках практической работы будет рассматриваться только системный режим работы OpenSSL. В этом режиме библиотека предоставляет доступ к интерактивному приложению, способному выполнять следующие функции:

- шифровать и расшифровывать данные с помощью симметрического или асимметрического шифрования — команды `enc`, `rsautl`;
- получать хеш — команда `dgst`;
- создавать и управлять ассиметричными ключами RSA и DSA — команды `rsa`, `genrsa`, `dsa`, `gendsa`, `dsaparam`;
- создавать сертификаты формата x509, запросы на сертификацию, восстановление — команды `x509`, `req`, `verify`, `ca`, `crl`, `pks12`, `pks7`.
- работа с S/MIME — команда `s/mime`;
- Выполнение служебных функций: `s_client`, `s_server`, `speed`, `rand`, `ciphers`.

Для работы с системной версией библиотеки используется интерактивное приложение `openssl`:

```
[user@host ~]openssl rand 15 && echo -e "\n"
.P7@▒m▒▒}[▒(▒

[user@host ~]
```

В примере показано каким образом возможно используя утилиту `openssl` сгенерировать последовательность из случайных чисел определённой длинны.

Для того, чтобы зашифровать данные или расшифровать их с использованием симметричного шифрования используется режим `enc`:

```
[user@host ~]$ cat file.txt
This is file
[user@host ~]$ openssl enc -aes128 -in file.txt -out file.txt.enc
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[user@host ~]$ cat file.txt.enc
Salted__-FvU▒"h?G▒▒▒zp▒x▒U▒A▒`
[user@host ~]$ openssl enc -d -aes128 -in file.txt.enc -out file.txt.dec
enter aes-128-cbc decryption password:
[user@host ~]$ cat file.txt.dec
```

```
This is file
[msn@jet openssl]$
```

Во всех режимах работы утилиты openssl опции `-in` и `-out` имеют одинаковое значение – входной файл и выходной файл соответственно.

Первым параметром в режиме enc указывается алгоритм шифрования (в примере это алгоритм AES с длинной ключа 128 бит). Узнать перечень поддерживаемых алгоритмов шифрования можно с использованием опции `-chipers`, указываемой вместо алгоритма шифрования:

```
[user@host ~]$ openssl enc -ciphers
Supported ciphers:
-aes-128-cbc               -aes-128-cfb               -aes-128-cfb1
-aes-128-cfb8              -aes-128-ctr               -aes-128-ecb
-aes-128-ofb               -aes-192-cbc               -aes-192-cfb
…
[user@host ~]$
```

Указать пароль можно прямо в командной строке. Для этого используется опция `-k`. Чтобы результирующий файл был записан не в бинарном виде, а в текстовом виде, закодированном с помощью методы Base64, можно использовать опцию `-a`.

Для того, чтобы получить хеш от файла или сформировать цифровую подпись используется режим `dgst`:

```
[user@host ~]$ openssl dgst -md5 file.txt
MD5(file.txt)= a5bc3c6ce3eafea85c78e20c485571e5
```

Алгоритм, по которому генерируется хеш, указывается вторым параметром. Список поддерживаемых алгоритмов можно узнать, выполнив команду `list --digest-commands`.

Для генерирования закрытых ассиметричных ключей используются команды `genrsa` и `gendsa`:

```
[user@host ~]$ openssl genrsa -out key.pem
Generating RSA private key, 2048 bit long modulus
.....................................................+++++
..............+++++
e is 65537 (0x010001)
[user@host ~]$
```

Чтобы получить открытый ключ на основе закрытого, используются команды rsa и dsa:

```
[user@host ~]$ openssl rsa -in key.pem  -pubout
writing RSA key
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEApb/BH6F4stCg/0OPffH4
W64eO1XqLSJ6d/GjQLd17ZA09SpB1wkLLFMRBzHB+qUYsYoBnj4VQ7TWmj5sz8Gj
C5SCfLEbyfca0k8olyzAg89rqNKW8I6Tmcw2oSLxDAReQvivdcJ5GotPVfsBujGS
vP5NO1i2ZtovHigjPFLd2b57BBPmpg72cIq5vSDd7f4mxkyjndtkBB3FzweiSglU
6OYP5Hr3M3HKxV2egmscI8xiOLcFqHLTgzXuRedYjqZYpsP3R710su2z2D84VhIv
sUx4BoQH1lCP9KwbIRrUi8lVWD4Mg2ta+fdLbqWzcQsCiruBxcD+FAxJO0NyLtPt
nwIDAQAB
-----END PUBLIC KEY-----
[user@host ~]$
```

Для шифрования и дешифрования данных ассиметричным методом используется команда `rsautil`:

```
[user@host ~]$ openssl rsautl -encrypt -inkey rsa.pem -in file.txt -out file.enc
```

7. Практическое задание

1. Создайте текстовый файл с произвольным содержанием.

2. Используя командный интерфейс OpenSSL зашифруйте файл, созданный в п.1, с использованием симметричного алгоритма.

3. Попробуйте расшифровать файл используя неверный ключ. Расшифруйте файл с правильным ключом.

4. Используя командный интерфейс библиотеки OpenSSL сгенерируйте пару открытый/закрытый ключ.

5. Используя командный интерфейс OpenSSL зашифруйте файл, созданный в п.1, с использованием асимметричного алгоритма.

6. Попробуйте расшифровать файл используя открытый ключ. Расшифруйте файл с правильным ключом.

7. Получите контрольную сумму файла, созданного в п. 1. Измените несколько символов в файле и заново получите контрольную сумму. Результаты совпадают?