МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)

ПРОТОКОЛ SSL (version 2)

Эссе по курсу “Теория защиты информации” студента 014 группы
Рафикова Сергея.

Москва 2004

Введение

Еще несколько лет назад работа в сети Интернет ограничивалась использованием достаточно простых протоколов (электронная почта, передача файлов). Однако быстрый рост сети и желание многих пользователей использовать новые возможности (главным образом электронная торговля), привели к быстрому развитию механизмов защиты. Один из таких механизмов, разработанный Netscape Communications Corporation, является протокол Secure Sockets Layer (SSL).

Итак, SSL был создан, во-первых, для обеспечения засекреченности между двумя взаимодействующими приложениями (между клиентом и сервером) и, во-вторых, для аутентификации сервера и, возможно, клиента. Для SSL нужен надежный транспортный протокол (например, TCP) для приема и передачи данных.

Преимуществом протокола SSL является то, что он независим от протоколов прикладного уровня (HTTP, FTP, TELNET), которые могут открыто работать поверх протокола SSL. SSL протокол согласовывает алгоритм шифрования, ключ сессии, а также аутентифицирует сервер перед тем, как протокол прикладного уровня передаст или получит первый байт данных. Все данные протокола прикладного уровня передаются в зашифрованном виде, гарантируя конфиденциальность.

SSL предоставляет “защищенный канал”, имеющий следующие свойства:

1. Канал является засекреченным. Шифрование используется для всех сообщений после простого диалога, который служит для определения секретного ключа.
2. Канал аутентифицирован. Сервер аутентифицируется всегда, клиент аутентифицируется по выбору.
3. Канал надежен. Передача сообщения включает проверку целостности с использованием MAC.

Формат заголовка SSL записи

В SSL все пересылаемые данные представлены в виде записи (record) – объект, состоящий из заголовка и данных. Заголовок каждой записи содержит два или три байта кода длины (length code). Если самый старший бит первого байта установлен, то запись не имеет байта, называемого padding и длина заголовка равно двум байтам, в противном случае – трем. Заголовок записи всегда передается перед частью, содержащей данные.

Заметим, что в случае присутствия padding байта если второй по старшинству бит первого байта равен 0, то запись является информационной, иначе запись является security escape (в настоящее время нет примеров security escape, зарезервировано для будущих версий). В любом случае, код длины определяет сколько данных в записи.

Длина записи не включает длину заголовка. Когда используется 2-х байтовый заголовок, то длина записи вычисляется следующим образом:

RECORD-LENGTH = ((byte[0] & 0x7f) << 8)) | byte[1];

Где byte[0] представляет первый полученный байт, а byte[1] – второй.

В случае 3-х байтового заголовка:

```
RECORD-LENGTH = ((byte[0] & 0x3f) << 8)) | byte[1];
IS-ESCAPE = (byte[0] & 0x40) != 0;
PADDING = byte[2];
```

Заголовок записи определяет значение PADDING. Это значение определяет сколько байт данных было добавлено отправителем к исходной записи и используются для того, чтобы сделать длину записи кратной размеру блока шифра, если применен блочный шифр.

Отправитель "padding" записи добавляет padding данные в конец нормальных данных, а затем шифрует все это, благо длина этого массива кратна размеру блока используемого шифра. Содержимое padding не играет роли. Так как объем передаваемых данных известен, заголовок сообщения может быть корректно сформирован с учетом установленного значения PADDING.

Получатель "padding" записи дешифрует всю запись,получает исходную информацию, вычитает PADDING значение из RECORD-LENGTH и получает истинное значение RECORD-LENGTH. Padding данные удаляются.

Формат поля данных SSL записи.

Часть данных SSL записи состоит из трех компонент (передаваемых и получаемых в приведенном ниже порядке):

```
MAC-DATA[MAC-SIZE]
ACTUAL-DATA[N]
PADDING-DATA[PADDING]
```

ACTUAL-DATA есть именно те данный, которые нужно передать (поле данных сообщения). MAC-DATA является кодом аутентификации сообщения (***Message Authentication Code)***. Когда SSL записи посылаются открытым текстом, никаких шифров не нужно. Следовательно, длина PADDING-DATA будет равна нулю и объем MAC-DATA также будет нулевым. Когда используется шифрование, PADDING-DATA является функцией размера блока шифра. MAC-DATA зависит от CIPHER-CHOICE. MAC-DATA вычисляется следующим образом:

```
MAC-DATA = HASH[ SECRET, ACTUAL-DATA, PADDING-DATA, SEQUENCE-NUMBER ]
```

SEQUENCE-NUMBER представляет собой 32-битовое значение, которое передается хэш-функции в виде 4 байт (первый байт – старший, последний - младший).

MAC-SIZE является функцией используемого алгоритма вычисления дайджеста. Для MD2 и MD5 MAC-SIZE равен 16 байтам (128 битам).

Значение SECRET зависит оттого, кто из партнеров посылает сообщение. Если сообщение посылается клиентом, тогда SECRET равен CLIENT-WRITE-KEY (сервер будет использовать SERVER-READ-KEY для проверки MAC). Если клиент получает сообщение, SECRET равен CLIENT-READ-KEY (сервер будет использовать SERVER-WRITE-KEY для генерации MAC).

SEQUENCE-NUMBER является счетчиком, который увеличивается как отправителем, так и получателем. Для каждого направления передачи, используется пара счетчиков (один для отправителя, другой для получателя). Каждый раз при передаче сообщения отправителем, счетчик увеличивается. SEQUENCE-NUMBER однуляется при

переполнении (0xFFFFFFFF).

Получатель сообщения использует ожидаемое значение SEQUENCE-NUMBER как входной параметр для хэш-функции MAC (тип хэш-функции определяется параметром CIPHER-CHOICE). Вычисленная MAC-DATA должна совпадать с переданной MAC-DATA. Если нет совпадения, запись считается поврежденной, такая ситуация рассматривается как "I/O Error" (т.е. как непоправимая ошибка, которая вызывает закрытие соединения).

Окончательная проверка соответствия выполняется, когда используется блочный шифр и соответствующий протокол шифрования. Объем данных в записи (RECORD-LENGTH) должен быть кратным размеру блока шифра. Если полученная запись не кратна размеру блока шифра, то она считается поврежденной, при этом считается, что имела место "I/O Error".

Уровень записей SSL используется для всех SSL коммуникаций, включая сообщения диалога (handshake massages), security escape и информационный обмен. Уровень записей SSL используется как клиентом, так и сервером.

Для двухбайтового заголовка, максимальная длина записи равна 32767 байт. Для трехбайтового заголовка, максимальная длина записи равна 16383 байтов. Сообщения SSL Handshake Protocol должны соответствовать одной записи SSL Record Protocol. Сообщения протокола прикладного уровня могут занимать несколько SSL записей.

Прежде чем послать первую SSL запись, все SEQUENCE-NUMBER обнуляются. При передаче сообщения SEQUENCE-NUMBER увеличивается, начиная с сообщений CLIENT-HELLO и SERVER-HELLO.

Спецификация SSL Handshake Protocol (SSLHP)

SSL Handshake Protocol состоит из 2-х фаз. Первая фаза используется для установления конфиденциального канала, вторая – для аутентификации клиента.

Фаза 1

Первая фаза является фазой инициализации соединения, когда обе стороны посылают сообщения "hello". Клиент инициирует диалог посылкой сообщения CLIENT-HELLO. Сервер получает сообщение CLIENT-HELLO, обрабатывает его и откликается сообщением SERVER-HELLO.

К этому моменту, как клиент, так и сервер имеют достаточно информации, чтобы знать, нужен ли новый контрольный ключ. Если новый контрольный ключ не нужен, клиент и сервер немедленно переходят в фазу 2.

В случае если нужен новый контрольный ключ, сообщение SERVER-HELLO содержит достаточно данных, чтобы клиент мог сформировать такой ключ. Сюда входит сертификат, подписанный сервером, список спецификаций шифра, и идентификатор соединения (последний представляет собой случайное число, сформированное сервером и используемое клиентом и сервером на протяжении сессии). Клиент генерирует контрольный ключ и посылает сообщение CLIENT-MASTER-KEY (или сообщение ERROR, если информация сервера указывает, что клиент и сервер не могут согласовать спецификацию шифра).

Здесь следует заметить, что каждая конечная точка SSL использует пару шифров для каждого соединения (т.е. всего 4 шифра). На каждой конечной точке, один шифр используется для исходящих соединений и один - для входящих. Когда клиент или сервер генерирует ключ сессии, они фактически генерируют два ключа, SERVER-READ-KEY (известный также как CLIENT-WRITE-KEY) и SERVER-WRITE-KEY (известный также как CLIENT-READ-KEY). Контрольный ключ используется клиентом и сервером для генерации различных ключей сессий.

Наконец, после того как контрольный ключ определен, сервер посылает клиенту сообщение SERVER-VERIFY. Этот заключительный шаг аутентифицирует сервер, так как только сервер, который имеет соответствующий общедоступный ключ, может знать контрольный ключ.

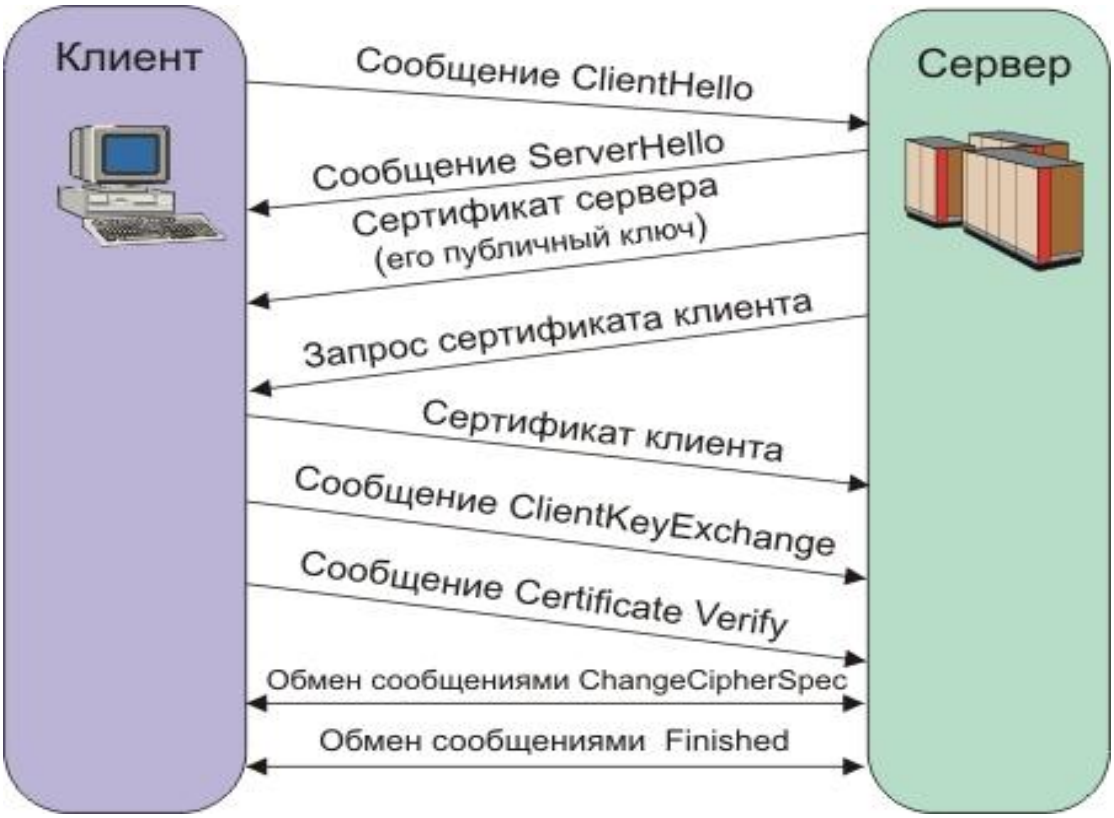
Фаза 2

Вторая фаза является фазой аутентификации. Сервер уже аутентифицирован клиентом на первой фазе, по этой причине здесь осуществляется аутентификация клиента. При типичном сценарии, серверу необходимо получить что-то от клиента, и он посылает запрос. Клиент пришлет положительный ответ, если располагает необходимой информацией, или пришлет сообщение об ошибке, если нет. Эта спецификация протокола не определяет семантику сообщения ERROR, посылаемого в ответ на запрос сервера (например, конкретная реализация может игнорировать ошибку, закрыть соединение, и т.д. и, тем не менее, соответствовать данной спецификации). Когда одна сторона аутентифицировала другую, она посылает сообщение **finished**. В случае клиента сообщение CLIENT-FINISHED содержит зашифрованную форму идентификатора CONNECTION-ID, которую должен проверить сервер. Если проверка терпит неудачу, сервер посылает сообщение ERROR.

Поскольку партнер послал сообщение **finished,** он должен продолжить воспринимать сообщения до тех пор, пока не получит сообщение **finished** от другой стороны. Как только оба партнера послали и получили сообщения **finished**, SSL handshake protocol закончил свою работу. С этого момента начинает работать протокол прикладного уровня (Заметим, что протокол прикладного уровня лежит поверх SSL Record Protocol).

Типичный протокол обмена сообщениями

В несколько упрощенном варианте диалог SSL можно представить:



Приведем несколько вариантов обмена в рамках SSL Handshake Protocol. Если что-то помещено в фигурные скобки, например, "{нечто}key", это означает, что “нечто” зашифровано с помощью ключа "key".

Таблица 1. При отсутствии идентификатора сессии

Client-hello	C → S:	challenge, cipher_specs
server-hello	S → C:	connection-id,server_certificate,cipher_specs
client-master-key	C → S:	{master_key}server_public_key
client-finish	C → S:	{connection-id}client_write_key
server-verify	S → C:	{challenge}server_write_key
server-finish	S → C:	{new_session_id}server_write_key

Таблица 2. Идентификатор сессии найден клиентом и сервером

client-hello	C → S:	challenge, session_id, cipher_specs
server-hello	S → C:	connection-id, session_id_hit
client-finish	C → S:	{connection-id}client_write_key
server-verify	S → C:	{challenge}server_write_key
server-finish	S → C:	{session_id}server_write_key

Таблица 3. Использован идентификатор сессии и аутентификация клиента

client-hello	C → S:	challenge, session_id, cipher_specs
server-hello	S → C:	connection-id, session_id_hit
client-finish	C → S:	{connection-id}client_write_key
server-verify	S → C:	{challenge}server_write_key
request-certificate	S → C:	{auth_type,challenge'}server_write_key
client-certificate	C → S:	{cert_type,client_cert, response_data} client_write_key
server-finish	S → C:	{session_id}server_write_key

В последнем примере, **response_data** является функцией **auth_type.**

Ошибки

Обработка ошибок в протоколе SSL соединения очень проста. Когда ошибка обнаружена, сторона, которая обнаружила ошибку, посылает другой стороне сообщение. Ошибки, которые являются неустранимыми, вызывают разрыв соединения. Сервер и клиент должны "забыть" все идентификаторы сессии, связанные с разорванным соединением. Протокол диалога SSL определяет следующие ошибки:

NO-CIPHER-ERROR

Эта ошибка присылается клиентом серверу, когда он не может найти шифр или размер ключа, который поддерживается клиентом, также поддерживается и сервером. Эта ошибка неустранима.

NO-CERTIFICATE-ERROR

Когда послано сообщение REQUEST-CERTIFICATE, эта ошибка может быть прислана, если клиент не имеет сертификата, чтобы ответить. Эта ошибка устранима.

BAD-CERTIFICATE-ERROR

Такая ошибка возникает, когда сертификат по какой-то причине считается принимающей стороной плохим. Плохой означает, что, либо некорректна подпись сертификата, либо некорректно его значение (например, имя в сертификате не соответствует ожидаемому). Эта ошибка устранима (только для аутентификации клиента).

UNSUPPORTED-CERTIFICATE-TYPE-ERROR

Возникает, когда клиент/сервер получает тип сертификата, который он не поддерживает. Эта ошибка устранима (только для аутентификации клиента).

Сообщения SSLHP

Сообщения SSLHP инкапсулируются в SSL Record Protocol и состоят из двух частей: однобайтового кода типа сообщения, и некоторых данных. Клиент и сервер обмениваются сообщениями, пока обе стороны не пошлют сообщения **finished**, указывающие, что они удовлетворены диалогом SSLHP.

После того как каждый из партеров определил пару ключей сессии, тела сообщений кодируются с помощью этих ключей. Для клиента это происходит, после того как он проверил идентификатор сессии или создаст новый ключ сессии и пошлет его серверу. Для сервера это происходит, после того как идентификатор сессии признан корректным, или сервер получил сообщение клиента с ключом сессии. Для сообщений SSLHP используется следующие определения:

```
char MSG-EXAMPLE
char FIELD1
char FIELD2
char THING-MSB
char THING-LSB
```

```
char THING-DATA[(MSB<<8)|LSB];
...
```

Эта нотация определяет данные в протокольном сообщении, включая код типа сообщения. Порядок передачи соответствует порядку перечисления.

Для записи "THING-DATA", значения MSB и LSB в действительности равны THING-MSB и THING-LSB (соответственно) и определяют число байт данных, имеющихся в сообщении. Например, если THING-MSB был равен нулю, а THING-LSB был равен 8, тогда массив THING-DATA будет иметь 8 байт.

CLIENT-HELLO (Фаза 1; посылается открыто)

```
char MSG-CLIENT-HELLO
char CLIENT-VERSION-MSB
char CLIENT-VERSION-LSB
char CIPHER-SPECS-LENGTH-MSB
char CIPHER-SPECS-LENGTH-LSB
char SESSION-ID-LENGTH-MSB
char SESSION-ID-LENGTH-LSB
char CHALLENGE-LENGTH-MSB
char CHALLENGE-LENGTH-LSB
char CIPHER-SPECS-DATA[(MSB<<8)|LSB]
char SESSION-ID-DATA[(MSB<<8)|LSB]
char CHALLENGE-DATA[(MSB<<8)|LSB]
```

Когда клиент впервые подключается к серверу, он должен послать сообщение CLIENT-HELLO. Сервер ожидает это сообщение от клиента первым. Любое другое сообщение от клиента в данных обстоятельствах рассматривается как ошибка клиента.

Клиент посылает серверу свою версию SSL, спецификации шифра, некоторые данные вызова (*challenge data*), и данные идентификатора сессии. Данные идентификатора сессии посылаются клиентом только в том случае, когда в его кэше имеется идентификатор сессии для клиента (в этом случае SESSION-ID-LENGTH не равно нулю). Когда идентификатора сессии нет, то значение SESSION-ID-LENGTH должно быть равно нулю. Данные вызова используются для аутентификации сервера. После того как клиент и сервер согласовали пару ключей сессии, сервер присылает сообщение SERVER-VERIFY с зашифрованной формой CHALLENGE-DATA.

Заметим также, что сервер не пошлет сообщения SERVER-HELLO пока не получит сообщения CLIENT-HELLO. Это делается так, чтобы сервер мог в первом сообщении клиенту определить состояние идентификатора сессии клиента (т.е. улучшить эффективность протокола и уменьшить объем обменов).

Сервер рассматривает сообщение CLIENT-HELLO и проверяет, поддерживает ли он версию программы клиента и хотя бы одну из спецификации шифра клиента. Сервер может выборочно отредактировать спецификации шифра, удалив записи, которые он решил не поддерживать. Отредактированная версия будет прислана в сообщении SERVER-HELLO, если идентификатор сессии не находится в кэше сервера.

Значение CIPHER-SPECS-LENGTH должно быть больше нуля и кратно 3. Код SESSION-ID-LENGTH должен быть равен нулю или 16. Значение CHALLENGE-LENGTH должно быть больше чем ≥ 16 и ≤ 32.

Это сообщение должно быть первым, посланным клиентом серверу. После его посылки клиент ждет сообщения SERVER-HELLO. Любое другое сообщение, присланное сервером (кроме ERROR) не допустимо.

CLIENT-MASTER-KEY (Фаза 1; посылается вначале открыто)

char MSG-CLIENT-MASTER-KEY
char CIPHER-KIND[3]
char CLEAR-KEY-LENGTH-MSB
char CLEAR-KEY-LENGTH-LSB
char ENCRYPTED-KEY-LENGTH-MSB
char ENCRYPTED-KEY-LENGTH-LSB
char KEY-ARG-LENGTH-MSB
char KEY-ARG-LENGTH-LSB
char CLEAR-KEY-DATA[MSB<<8|LSB]
char ENCRYPTED-KEY-DATA[MSB<<8|LSB]
char KEY-ARG-DATA[MSB<<8|LSB]

Клиент посылает это сообщение, когда он определил контрольный ключ для работы с сервером. Заметим, что когда идентификатор сессии согласован, это сообщение не посылается.

Поле CIPHER-KIND указывает, какой шифр выбран из спецификации CIPHER-SPECS сервера.

Данные CLEAR-KEY-DATA содержат открытую часть MASTER-KEY. CLEAR-KEY-DATA вместе с SECRET-KEY-DATA образовуют MASTER-KEY, при этом SECRET-KEY-DATA составляет младшие байты MASTER-KEY. ENCRYPTED-KEY-DATA содержит секретные части MASTER-KEY, зашифрованные с использованием общедоступного ключа сервера. Информационная часть блока имеет следующий формат:

char SECRET-KEY-DATA[SECRET-LENGTH]

SECRET-LENGTH равно числу байт каждого из ключей сессии. SECRET-LENGTH плюс CLEAR-KEY-LENGTH равно числу байт в ключе шифра (как это определено CIPHER-KIND). Если после дешифрования SECRET-LENGTH окажется неравным ожидаемому значению, регистрируется ошибка. Ошибкой считается и ситуация, когда CLEAR-KEY-LENGTH не равно нулю и CIPHER-KIND является не экспортным шифром.

Вычисление ключей сессии клиента и сервера является функцией CIPHER-CHOICE:

SSL_CK_RC4_128_WITH_MD5
SSL_CK_RC4_128_EXPORT40_WITH_MD5
SSL_CK_RC2_128_CBC_WITH_MD5
SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5
SSL_CK_IDEA_128_CBC_WITH_MD5
KEY-MATERIAL-0 = MD5[MASTER-KEY, "0", CHALLENGE, CONNECTION-ID]
KEY-MATERIAL-1 = MD5[MASTER-KEY, "1", CHALLENGE, CONNECTION-ID]
CLIENT-READ-KEY = KEY-MATERIAL-0[0-15]
CLIENT-WRITE-KEY = KEY-MATERIAL-1[0-15]

Где KEY-MATERIAL-0[0-15] означает первые 16 байт данных KEY-MATERIAL-0, с KEY-MATERIAL-0[0], образующим старший байт CLIENT-READ-KEY.

Данные передаются хэш-функции MD5, начиная с MASTER-KEY, далее следует "0" или "1", затем вызов (CHALLENGE) и, наконец, CONNECTION-ID.

Заметим, что "0" означает ASCII символ нуль (0x30), а не значение нуль. "1" означает ASCII символ 1 (0x31). MD5 выдает 128 бит выходных данных, которые используются в качестве ключа алгоритма шифрования.

SSL_CK_DES_64_CBC_WITH_MD5
KEY-MATERIAL-0 = MD5[MASTER-KEY, CHALLENGE, CONNECTION-ID]
CLIENT-READ-KEY = KEY-MATERIAL-0[0-7]
CLIENT-WRITE-KEY = KEY-MATERIAL-0[8-15]

SSL_CK_DES_192_EDE3_CBC_WITH_MD5
KEY-MATERIAL-0 = MD5[MASTER-KEY, "0", CHALLENGE, CONNECTION-ID]
KEY-MATERIAL-1 = MD5[MASTER-KEY, "1", CHALLENGE, CONNECTION-ID]
KEY-MATERIAL-2 = MD5[MASTER-KEY, "2", CHALLENGE, CONNECTION-ID]
CLIENT-READ-KEY-0 = KEY-MATERIAL-0[0-7]
CLIENT-READ-KEY-1 = KEY-MATERIAL-0[8-15]
CLIENT-READ-KEY-2 = KEY-MATERIAL-1[0-7]
CLIENT-WRITE-KEY-0 = KEY-MATERIAL-1[8-15]
CLIENT-WRITE-KEY-1 = KEY-MATERIAL-2[0-7]
CLIENT-WRITE-KEY-2 = KEY-MATERIAL-2[8-15]

Всего генерируется 6 ключей, 3 ключа читающей стороны для шифра DES-EDE3 и 3 - для пишущей стороны для функции DES-EDE3. Вектор инициализации формируется в KEY-ARG-DATA.

Вспомним, что MASTER-KEY передается серверу в сообщении CLIENT-MASTER-KEY. CHALLENGE выдается серверу клиентом в сообщении CLIENT-HELLO. CONNECTION-ID передается клиенту от сервера в сообщении SERVER-HELLO. Это делает получаемые в результате ключи, зависящими от исходной и текущей сессии. Заметим, что контрольный ключ никогда не используется для шифрования данных, и следовательно не может быть легко раскрыт.

Сообщение CLIENT-MASTER-KEY должно быть послано после сообщения CLIENT-HELLO и до сообщения CLIENT-FINISHED. Сообщение CLIENT-MASTER-KEY должно быть послано, если сообщение SERVER-HELLO содержит значение SESSION-ID-HIT равное 0.

CLIENT-CERTIFICATE (Фаза 2; посылается шифрованным)

char MSG-CLIENT-CERTIFICATE
char CERTIFICATE-TYPE
char CERTIFICATE-LENGTH-MSB
char CERTIFICATE-LENGTH-LSB
char RESPONSE-LENGTH-MSB
char RESPONSE-LENGTH-LSB
char CERTIFICATE-DATA[MSB<<8|LSB]
char RESPONSE-DATA[MSB<<8|LSB]

Это сообщение посылается клиентом SSL в ответ на сообщение сервера REQUEST-CERTIFICATE. CERTIFICATE-DATA содержит данные, определенные значением CERTIFICATE-TYPE. Сообщение об ошибке ERROR посылается с кодом NO-CERTIFICATE-ERROR, если данный запрос не может быть обработан корректно (например, получатель сообщения не имеет зарегистрированного сертификата).

Это сообщение должно быть послано клиентом только в ответ на сообщение REQUEST-CERTIFICATE сервера.

CLIENT-FINISHED (Фаза 2; посылается шифрованным)

char MSG-CLIENT-FINISHED
char CONNECTION-ID[N-1]

Клиент посылает это сообщение, после успешной обработки соответствующего сообщения сервера. Заметим, что клиент должен быть готов к приему сообщений от сервера, пока не получит сообщение SERVER-FINISHED. Данные CONNECTION-ID представляют собой исходный идентификатор соединения сервера, посланный в его сообщении SERVER-HELLO и зашифрованный посредством согласованного ключа сессии.

"N" равно числу байт в посланном сообщении, таким образом "N-1" равно числу байт в сообщении за вычетом одного байта заголовка.

Для версии протокола 2, клиент должен посылать это сообщение после получения сообщения SERVER-HELLO. Если в сообщении SERVER-HELLO флаг SESSION-ID-HIT не равен нулю, тогда сообщение CLIENT-FINISHED посылается немедленно, в противном случае сообщение CLIENT-FINISHED посылается после сообщения CLIENT-MASTER-KEY.

Существует несколько сообщений, которые генерируются только серверами.

SERVER-HELLO (Фаза 1; посылается открыто)

char MSG-SERVER-HELLO
char SESSION-ID-HIT
char CERTIFICATE-TYPE
char SERVER-VERSION-MSB
char SERVER-VERSION-LSB
char CERTIFICATE-LENGTH-MSB
char CERTIFICATE-LENGTH-LSB
char CIPHER-SPECS-LENGTH-MSB
char CIPHER-SPECS-LENGTH-LSB
char CONNECTION-ID-LENGTH-MSB
char CONNECTION-ID-LENGTH-LSB
char CERTIFICATE-DATA[MSB<<8|LSB]
char CIPHER-SPECS-DATA[MSB<<8|LSB]
char CONNECTION-ID-DATA[MSB<<8|LSB]

Сервер посылает это сообщение после получения CLIENT-HELLO. Сервер возвращает флаг SESSION-ID-HIT, указывающий, известен ли серверу полученный идентификатор сессии (т.е. хранится ли он в кэше сервера). Флаг SESSION-ID-HIT будет не равен нулю, если клиент посылает серверу идентификатор сессии (в сообщении CLIENT-HELLO с SESSION-ID-LENGTH != 0), а сервер обнаружит этот идентификатор в своем кэше. Если флаг SESSION-ID-HIT не равен нулю, то поля CERTIFICATE-TYPE, CERTIFICATE-LENGTH и CIPHER-SPECS-LENGTH будут содержать код нуль.

Когда флаг SESSION-ID-HIT равен нулю, сервер посылает свой сертификат, спецификацию шифров и идентификатор соединения клиенту. Используя эту информацию, клиент может сформировать ключ сессии и послать его серверу в сообщении CLIENT-MASTER-KEY.

Когда флаг SESSION-ID-HIT не равен нулю, как сервер так и клиент вычисляют новую пару ключей сессии, базируясь на контрольном ключе MASTER-KEY, который они получили при создании идентификатора SESSION-ID. SERVER-READ-KEY и SERVER-WRITE-KEY получаются из исходных ключей MASTER-KEY тем же способом, что CLIENT-READ-KEY и CLIENT-WRITE-KEY:

SERVER-READ-KEY = CLIENT-WRITE-KEY
SERVER-WRITE-KEY = CLIENT-READ-KEY

Заметим, что когда ключи получены и установлен флаг SESSION-ID-HIT, а сервер обнаружил идентификатор сессии клиента в своем кэше, тогда данные KEY-ARG-DATA используются с момента, когда определен идентификатор SESSION-ID. Это делается, потому что клиент не посылает новых данных KEY-ARG-DATA (напомним, что данные KEY-ARG-DATA посланы в сообщении CLIENT-MASTER-KEY).

CONNECTION-ID-DATA представляет собой строку случайных байт. Сообщение CLIENT-FINISHED содержит зашифрованную версию CONNECTION-ID-DATA. Длина CONNECTION-ID должна лежать между 16 и 32 байтами, включительно.

CIPHER-SPECS-DATA определяет тип шифра и длину ключа (в битах), которые поддерживает принимающая сторона. Каждая спецификация SESSION-CIPHER-SPEC имеет длину 3 байта и выглядит как:

char CIPHER-KIND-0
char CIPHER-KIND-1
char CIPHER-KIND-2
Где CIPHER-KIND равен одному из:

- SSL_CK_RC4_128_WITH_MD5
- SSL_CK_RC4_128_EXPORT40_WITH_MD5
- SSL_CK_RC2_128_CBC_WITH_MD5
- SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5
- SSL_CK_IDEA_128_CBC_WITH_MD5
- SSL_CK_DES_64_CBC_WITH_MD5
- SSL_CK_DES_192_EDE3_CBC_WITH_MD5

Этот список не является исчерпывающим и может быть расширен в будущем. Конфигурации этих средств безопасности стандартизованы (см. табл.1).

Таблица 4. Используемые аллритмы

Набор	**Уровень безопасности**	**Описание**
DES-CBC3-MD5	Очень высокий	Тройной DES в режиме CBC, хэш MD5, 168-битный ключ сессии
DES-CBC3-SHA	Очень высокий	Тройной DES в режиме CBC, хэш SHA, 168-битный ключ сессии
RC4-MD5	Высокий	RC4, хэш MD5, 128-битный ключ
RC4-SHA	Высокий	RC4, хэш SHA, 128-битный ключ
RC2-CBC-MD5	Высокий	RC2 в режиме CBC, хэш MD5, 128-битный ключ

DES-CBC-MD5	Средний	DES в режиме CBC, хэш MD5, 56-битный ключ
DES-CBC-SHA	Средний	DES в режиме CBC, хэш SHA, 56-битный ключ
EXP-DES-CBC-SHA	Низкий	DES в режиме CBC, хэш SHA, 40-битный ключ
EXP-RC4-MD5	Низкий	Экспортное качество RC4, хэш MD5, 40-битный ключ
EXP-RC2-CBC-MD5	Низкий	Экспортное качество RC2, хэш MD5, 40-битный ключ
NULL-MD5	-	Без шифрования, хэш MD5, только аутентификация
NULL-SHA	-	Без шифрования, хэш SHA, только аутентификация

Шифр SSL_CK_RC4_128_EXPORT40_WITH_MD5 имеет тип RC4, где некоторые ключи сессии посылаются открыто, а остальные в зашифрованном виде. MD5 используется в качестве хэш-функции для получения MAC и ключей сессии. Этот тип шифра предлагается для поддержки "экспортных" версий (т.е. версий протокола, которые могут быть применены за пределами Соединенных Штатов) клиента или сервера.

Экспортные реализации протокола диалога SSL будут иметь длины секретных ключей не более 40 бит. Для не экспортных реализаций длины ключей могут быть больше (рекомендуется, по крайней мере, 128 бит). Клиенты и серверы могут иметь не перекрывающийся набор поточных шифров. Но это означает, что они не могут общаться.

Версия 2 протокола SSL определяет, что SSL_CK_RC4_128_WITH_MD5 должен иметь длину ключа 128 бит. SSL_CK_RC4_128_EXPORT40_WITH_MD5 также имеет длину ключа 128 бит. Однако только 40 бит являются секретными (другие 88 пересылаются от клиента к серверу открыто).

Сообщение SERVER-HELLO посылается, после того как сервер получит сообщение CLIENT-HELLO, и до того как сервер пошлет SERVER-VERIFY.

SERVER-VERIFY (Фаза 1; посылается шифрованным)

char MSG-SERVER-VERIFY
char CHALLENGE-DATA[N-1]

Сервер посылает это сообщение после пары ключей сессии (SERVER-READ-KEY и SERVER-WRITE-KEY), согласованных посредством идентификатора сессии или явно через спецификацию сообщения CLIENT-MASTER-KEY. Сообщение содержит зашифрованную копию данных CHALLENGE-DATA, посланных клиентом в сообщении CLIENT-HELLO.

"N" равен числу байт в сообщении, которое было послано, таким образом, "N-1" соответствует числу байт в CHALLENGE-DATA за вычетом байта заголовка.

Это сообщение используется для проверки сервера следующим образом. Настоящий сервер имеет секретный ключ, который соответствует общедоступному ключу, содержащемуся в его сертификате, переданном в сообщении SERVER-HELLO. Таким образом, настоящий сервер сможет извлечь и реконструировать пару ключей сессии (SERVER-READ-KEY и SERVER-WRITE-KEY). Наконец, только сервер, который выполнил корректно извлечение и дешифрование может правильно зашифровать CHALLENGE-DATA. Это, по существу, "доказывает", что сервер имеет секретный ключ, который образует пару с открытым ключом из сертификата сервера.

Данные CHALLENGE-DATA должны иметь в точности ту же длину, что и в сообщении клиента CLIENT-HELLO. Их значение должно в точности совпадать с посланным клиентом открыто в сообщении CLIENT-HELLO. Клиент должен дешифровать это сообщение и сравнить полученный результат с тем, что было послано, и только в случае успешного сравнения сервер считается достойным доверия. Если длины не совпадают или не согласуются значения, клиент соединение разрывает.

Это сообщение должно быть послано сервером клиенту либо после обнаружения идентификатора сессии (при этом посылается отклик SERVER-HELLO с флагом SESSION-ID-HIT не равным нулю) или когда сервер получает сообщение CLIENT-MASTER-KEY. Это сообщение должно быть послано до любого сообщения фазы 2 или до сообщения SEVER-FINISHED.

SERVER-FINISHED (Фаза 2; посылается зашифрованным)

char MSG-SERVER-FINISHED
char SESSION-ID-DATA[N-1]

Сервер посылает это сообщение, когда он удовлетворен результатом диалога с клиентом по поводу безопасности и готов продолжить передачу/прием протокольных данных верхнего уровня. Кэши идентификаторов сессии должны содержать копию MASTER-KEY, посланного в сообщении CLIENT-MASTER-KEY, в качестве контрольного ключа, предназначенного для генерации всех последующих ключей сессии.

Здесь "N" имеет то же значение, что и в определениях, представленных выше. Это сообщение должно посылаться после сообщения SERVER-VERIFY.

REQUEST-CERTIFICATE (Фаза 2; посылается шифрованным)

char MSG-REQUEST-CERTIFICATE
char AUTHENTICATION-TYPE
char CERTIFICATE-CHALLENGE-DATA[N-2]

Сервер может выдать этот запрос в любое время диалога второй фазы, требуя присылки сертификата клиента. Клиент немедленно откликается посылкой сообщения CLIENT-CERTIFICATE, если он имеет сертификат, в противном случае присылается уведомление об ошибке с кодом NO-CERTIFICATE-ERROR. CERTIFICATE-CHALLENGE-DATA представляет собой короткую строку байтов (с длиной ≥ 16 байт и ≤ 32 байтам), которую клиент использует для отклика на этот запрос.

Значение AUTHENTICATION-TYPE используется, чтобы выбрать конкретные средства для аутентификации клиента. Определены следующие типы:

- SSL_AT_MD5_WITH_RSA_ENCRYPTION

Тип SSL_AT_MD5_WITH_RSA_ENCRYPTION требует, чтобы клиент сформировал MD5-дайджест сообщения, используя информацию, как это описано выше в разделе о сообщении CLIENT-CERTIFICATE. Раз дайджест сформирован, клиент шифрует его, используя свой секретный ключ. Сервер аутентифицирует клиента, когда получает сообщение CLIENT-CERTIFICATE.

Это сообщение может быть послано после сообщения SERVER-VERIFY и до сообщения SERVER-FINISHED.

Эти сообщения генерируются как клиентом, так и сервером.

ERROR (посылается открыто или зашифровано)

char MSG-ERROR
char ERROR-CODE-MSB
char ERROR-CODE-LSB

Это сообщение посылается, когда обнаружена ошибка. После посылки сообщения, отправитель закрывает соединение. Получатель регистрирует ошибку и затем также разрывает соединение.

Это сообщение посылается открыто, если произошла ошибка при согласовании ключа сессии. После того как ключ сессии согласован, сообщения об ошибках шифруются также как и обычные сообщения.

Литература

1. The SSL Protocol, Kipp E.B. Hickman, Netscape Communications Corp. http://wp.netscape.com/eng/security
2. http://byerley.cs.waikato.ac.nz/~tonym/articles/ssl/ssl.html - краткое описание протокола SSL
3. http://www.kunegin.narod.ru – сайт Кунегина Сергея Владимировича