Оглавление

1. *Введение.* 2
2. *Обзор протокола SSL* 2
   - 2.1. *Использование SSL* 2
   - 2.2. *HTTP и HTTPS* 3
   - 2.3. *Принцип действия SSL* 3
   - 2.4. *Недостатки SSL* 4
   - 2.5. *Реализации SSL* 5
3. *Криптографмческие основы работы протокола SSL* 6
   - 3.1. *Алгоритм SSL* 6
4. *Обмен данными по протоколу SSL* 7
   - 4.1. *Спецификация протокола записей SSL* 7
   - *Формат заголовка записи SSL* 7
   - *Формат информационных записей SSL* 8
   - 4.2. *Спецификация протокола диалога SSL* 9
   - *Протокол диалога SSL* 9
   - 4.3. *Типовой протокол обмена сообщениями* 10
5. *Обзор протокола TLS* 11
   - 5.1. *Основные положения* 11
   - 5.2. *Цели протокола TLS* 12
   - 5.3. *Криптографические атрибуты* 12
   - 5.4. *Протокол записей TLS* 13
   - 5.5. *Состояния соединений* 13
   - 5.6. *Протокол диалога TLS* 14
   - 5.7. *Протокол изменения спецификации шифра* 15
   - 5.8. *Протокол оповещения* 15
   - 5.9. *Обзор протокола диалога* 15
6. *Заключение* 17
7. *Список литературы:* 18

1. Введение.

С каждым годом число различных сервисов, предоставляемых в сети Internet, растет. Среди них активно развивается и электронная коммерция и другие сервисы, которые используют в свой работе конфиденциальную информацию пользователя, например номер кредитной карты и т.д., в результате возникает необходимость защиты этой информации от несанкционированного доступа со стороны посторонних людей.
Для этого необходимо использовать различные методы защиты данных. Среди них можно выделить механизм обеспечения безопасности на транспортном уровне – протоколы SSL и TLS, о которых и будет рассказано далее.

2. Обзор протокола SSL

Протокол SSL (secure socket layer) был разработан фирмой Netscape, как протокол обеспечивающий защиту данных между сервисными протоколами (такими как HTTP, NNTP, FTP и т.д.) и транспортными протоколами (TCP/IP). Часто для него используется аббревиатура HTTPS. Именно эта латинская буква "s" превращает обычный, не защищенный канал передачи данных в Интернете по протоколу HTTP, в засекреченный или защищенный.
Протокол SSL предоставляет "безопасный канал", который имеет три основные свойства:

- *Канал является частным. Шифрование используется для всех сообщений после простого диалога, который служит для определения секретного ключа.*
- *Канал аутентифицирован. Серверная сторона диалога всегда аутентифицируется, в то время как клиентская - аутентифицируется опционно.*
- *Канал надежен. Транспортировка сообщений включает в себя проверку целостности (с привлечением MAC).*

Не секрет, что можно без особых технических ухищрений просматривать данные, которыми обмениваются между собой клиенты и серверы. Был даже придуман специальный термин для этого - sniffer. А в связи с увеличением объема использования Интернета в коммерческих целях, неизбежно вставал вопрос о защите передаваемых данных. И пользователи не очень были бы рады, если номер их кредитной карточки, был бы перехвачен, каким-нибудь предприимчивым хакером по дороге к виртуальному магазину. И, в общем, появление такого протокола как SSL было вполне закономерным явлением. С одной стороны остаются все возможности сервисных протоколов (для программ-серверов), плюс к этому все данные передаются в зашифрованном виде. И раскодировать их довольно трудно. Следует отметить, что SSL не только обеспечивает защиту данных в Интернете, но так же производит опознание сервера и клиента (server/client authentication). В данный момент протокол SSL принят W3 консорциумом (W3 Consortium) на рассмотрение, как основной защитный протокол для клиентов и серверов (WWW browsers and servers) в сети Интернет.

2.1. Использование SSL

Чаще всего, этот протокол используется в составе любого Интернет-ресурса, осуществляющего манипуляции с личными или финансовыми данными посещающих его пользователей Интернета. Чаще всего, это банки, Интернет-магазины или любые другие виртуальные места, в которых приходящие по своим делам пользователи, вынуждены передавать свои личные, и зачастую, секретные данные. Этого может потребовать и простая регистрация, и процедура оплаты какого-либо товара, или любая другая процедура, при которой пользователи вынуждены честно выдавать свои паспортные данные, пин-коды и пароли. Если бы все жители земного шара являлись бы порядочными и честными людьми, необходимость бы в использовании SSL, отпала бы сама собой, за не надобностью, ведь защищать информацию было бы просто не от кого. Но, поскольку в реалии мы имеем несколько другое положение вещей, то приходится думать о том, как защитить

передаваемую пользователем информацию от посягательств со стороны третьих лиц. Используя обычный HTTP протокол, мы передаем и получаем информацию в чистом, не зашифрованном виде. Таким образом, передаваемая нами информация, может быть легко перехвачена, и использована посторонним человеком. Помимо этого, существует и другая, на первый взгляд просто смешная угроза. Представьте, ваш банк расположен по адресу http://MyHomeBank.com. Теперь представьте, что некий злоумышленник, регистрирует другой адрес, скажем http://MyH0meBank.com, и создает на нем сайт, внешне похожий на сайт вашего банка. Эти адреса так похожи, что рано или поздно, вы наверняка ошибетесь, и случайно попадете не в банк, а на сайт злоумышленника. Ну а далее, схема примерно ясна - ваша персональная информация становится известна третьему лицу. Таким образом, появляются два довольно веских довода:

- *первый, передаваемую информацию надо шифровать*
- *второй, мы должны быть уверены, что передаем информацию именно туда, куда нужно.*

Именно для решения этих двух вопросов и используется SSL.

2.2. HTTP и HTTPS

*Попытки разработать универсальный сетевой протокол, способный обеспечить надлежащий уровень безопасности при работе в Интернет, предпринимались достаточно давно, и достаточно большим количеством различных фирм и организаций. HTTP протокол предлагал достаточно простой, парольный способ идентификации того или иного пользователя. В момент соединения с сервером, пользователь вводил пароль, пароль передавался серверу в открытом, не зашифрованном виде, и далее, проверив соответствие пароля и имени пользователя, сервер открывал или не открывал затребованное соединение. Далее, по мере развития Интернета, было создано несколько различных безопасных протоколов. Официальный протокол, разработку которого спонсировала IETF, назывался Secure HTTP (SHTTP), Помимо него, разрабатывались, и были созданы, еще несколько не официальных проектов, один из которых, под названием SSL (***Secure Sockets Layer***), созданный* **Netscape**, *получил большую популярность и широкое распространение. Правда, не смотря на свою популярность, SSL не является официальным Интернет стандартом.*

2.3. Принцип действия SSL

Главным назначением SSL-протокола, является обеспечение частного и надежного способа обмена информацией между двумя удаленно взаимодействующими приложениями. Протокол реализуется в виде двухслойной (многослойной) среды, специально предназначенной для безопасного переноса секретной информации, через не засекреченные каналы связи. В качестве первого слоя, в такой среде используется некоторый надежный транспортный протокол; TCP к примеру. По слову "транспортный", не трудно догадаться, что TCP берет на себя функции "несущей", и в дальнейшем, становится извозчиком, для всех лежащих выше слоев (протоколов). Вторым по счету слоем, накладываемым на TCP, является SSL Record Protocol. Вместе, эти два слоя, TCP и SSL Record Protocol, формируют своеобразное ядро SSL. В дальнейшем, это ядро становится первичной герметизирующей оболочкой, для всех последующих более сложных протокольных инфраструктур. В качестве одной из таких структур, используется SSL Handshake Protocol - позволяющий серверу и клиенту идентифицировать друг друга и согласовывать криптографические алгоритмы и ключи, перед тем как приложения, работающие на серверной и клиентской стороне, смогут начать передачу или прием информационных байтов в защищенном режиме.

Одним из не мало важных преимуществ SSL, является его полная программно-платформенная независимость. Протокол разработан на принципах переносимости, и идеология его построения, не зависит, от тех приложений, в составе которых он используется. Помимо этого, важно и то, что поверх протокола SSL, могут прозрачно накладываться и другие протоколы; либо для еще большего увеличения степени защиты целевых информационных

потоков, либо, для адаптации криптографических способностей SSL под какую-нибудь другую, вполне определенную задачу.
Вы начинаете использовать SSL в тот момент, когда вводите в адресной строке своего браузера URL начинающийся с аббревиатуры HTTPS. В результате, вы подключаетесь к порту за номером 443, который для SSL обычно используется по умолчанию; для стандартного HTTP соединения, чаще всего используется порт 80. В процессе подключения, браузер пользователя (в дальнейшем клиент), посылает серверу приветственное сообщение (hello message). В свою очередь сервер, также должен посылать клиенту свое приветственное сообщение. Приветственные сообщения, являются первичными, инициализирующими сообщениями и содержат информацию, используемую при дальнейшей настройке открываемого секретного канала.
В общем случае, приветственное сообщение устанавливает четыре основных параметра:

- *версия протокола,*
- *идентификатор сессии,*
- *способ шифрования,*
- *метод компрессии,*
- *а также, два специально сгенерированных случайных числа;*

И сервер, и клиент, генерируют такие числа независимо друг от друга, а затем, просто обмениваются ими друг с другом.
После получения приветственного сообщения от клиента, сервер отсылает свой сертификат, если таковой у него имеется. Также, при необходимости, сервер может послать и некое ключевое сообщение, например в случае отсутствии сертификата. Если сервер авторизирован (т.е. имеет соответствующий сертификат), он может потребовать и клиентский сертификат, если того потребует выбранный способ шифрования данных. После этого, производится еще ряд промежуточных обменных операций, в процессе которых, производится окончательное уточнение выбранного алгоритма шифрования, ключей и секретов, и далее, сервер посылает клиенту некое финальное сообщение, после чего обе стороны приступают к обмену зашифрованной информации.
На практике, процесс обмена ключами и сертификатами, иногда может занимать относительно много времени. С этой целью, часто предусматривается возможность повторного использования одних и тех же идентификационных данных. Бывают ситуации, когда после установления соединения с SSL-сервером, у пользователя появляется желание открыть еще одно окно браузера, и через него, осуществить еще одно подключение к тому же SSL-серверу. В этом случае, чтобы не повторять весь цикл предварительных обменных операций, браузер может отправить серверу идентификатор сессии предыдущего соединения, и если сервер примет этот идентификатор, весь набор шифровочных и компрессионных параметров, будет взят от предыдущего соединения. Браузеры от Netscape, также могут осуществлять и так называемый "keep alive" запрос. При этом по завершению передачи шифрованных данных, установленное SSL-соединение закрывается не сразу, а лишь по истечении некоторого времени.

2.4. Недостатки SSL

SSL как таковой, теоретически, может обеспечить практически полную защиту любого Интернет соединения. Но, любая вещь в этом мире не существует в пустоте. Это означает, что для успешного функционирования SSL, кроме него самого, необходимы также и чисто программные средства, претворяющие технологию SSL в жизнь. Программы, так или иначе использующие SSL протокол, как ни странно, является порой самым уязвимым местом этой технологии. Именно из-за ошибок в этих программах, возможна почти полная потеря, всех, достигнутых после использования SSL щитов и заслонов. К таким программным инструментам, прежде всего, относятся активно используемые нами Интернет-браузеры.
Одним из самых показательных критериев уровня защиты, является размер используемых ключей. Чем больше этот размер, тем соответственно надежнее защита. Браузеры в

основном используют три размера: 40, 56 и 128 бит, соответственно. Причем, 40-а битный вариант ключа недостаточно надежен. Таким образом, предпочтительнее использовать именно 128-ми битные ключи. Применительно к Internet Explorer от Microsoft, это означает загрузку дополнительного пакета (security pack). Так как интернациональные версии этого браузера, всегда снабжаются исключительно 40-а или 56-и битной защитой, а 128-ми битная защита, ставится только на североамериканские версии этого браузера (США, Канада). Для того чтобы установить, какой именно размер ключа используется в вашем браузере, в Netscape Navigator вам достаточно открыть подменю "Options/Security Preferences", а в Internet Explorer, подменю "Help/About".

Но размер ключа, не будет играть решающий роли, если в защите браузера имеется внутренняя брешь. Сообщения об открытии таких брешей, в тех или иных браузерах, появляются с регулярными интервалами. Такая брешь напоминает открытую форточку в протапливаемой комнате - все тепло мгновенно выветривается. По этому поводу, уместно вспомнить случай произошедший с Netscape Navigator, в мае 2000 года. Тогда, один корейский студент обнаружил такую, довольно не приятную особенность, этого браузера. При попытке соединения с сервером, обладающим не годным сертификатом, с дальнейшим отказом от продолжения такого соединения, происходило следующее. Netscape, по ошибке, помещал этот сертификат в список годных, и соответственно, при последующем подключение уже не выдавал пользователю ни каких предупреждающих сообщений, спокойно подключаясь к этому, не вполне надежному, серверу

Но все эти и им подобные прорехи, не идут ни в какое сравнение с той угрозой, которую могут представлять для пользователя вовремя не отозванные сертификаты. Дело в том, что браузеры обычно поставляются с неким, вполне определенным набором действительных сертификатов, но автоматического механизма проверки этой годности по прошествию некоторого времени - не существует. Таким образом, возможно, что действие, того или иного, используемого вашим браузером сертификата, уже, давно кончилось; мог истечь срок годности, мог быть потерян контроль над личным ключом соответствующим этому сертификату и.т.д. В любом из этих случаев, сертификат автоматически отзывается, и помещается в специальный, так называемый revocation list, или список не годных сертификатов, создаваемый и обновляемый тем или иным сертификационным сообществом (CA). Теперь, если не удалить такой сертификат из вашего браузера, он по прежнему будет числиться как годный, со всеми вытекающими отсюда последствиями.

Следует заметить, что идея, заложенная в протоколе SSL безусловно, хороша. Хотя у нее есть и свои плюсы, и свои минусы, но в целом, этот протокол можно назвать одним из наиболее удачных решений проблемы защиты пользовательских данных при их распространении "открытым" каналом. Этот протокол вполне бы мог стать некой сетевой панацеей. Но, к сожалению, практика, показывает, что идея это еще не решение. Без соответствующей практической составляющей, идея так и остается идеей, а потому, пользователи безусловно, должны помнить, что символ замка, появляющийся в строке состояния их Интернет-браузеров, еще не гарантия того, что все наши секреты и тайны находятся под действительно надежной защитой

2.5. Реализации SSL

Теперь несколько слов о реализации SSL. Наиболее распространенным пакетом программ для поддержки SSL является SSLeay. Последняя версия (SSLeay v. 0.8.0) поддерживает SSLv3. Эта версия доступна в исходных текстах. Этот пакет предназначен для создания и управления различного рода сертификатами. Так же в его состав входит и библиотека для поддержки SSL различными программами. Эта библиотека необходима, например, для модуля SSL в распространенном HTTP сервере Apache. Если Вы устанавливаете версию, вне США, то особых проблем с алгоритмом RSA быть не должно. Но только накладывается ограничение на длину ключа в 40 бит. Действует это ограничение и на другой пакет от фирмы Netscape - SSLRef. А вот если компьютер с SSLeay находится на территории США, то за использование

алгоритма RSA необходимо заплатить. Но об этом нужно заключать договор с самой фирмой RSA Data Security Inc.

3. Криптографмческие основы работы протокола SSL

3.1. Алгоритм SSL

Алгоритм работы SSL построен на принципе публичных ключей. Этот принцип построен на использовании пары асимметричных ключей (публичном и приватном) для кодирования/декодирования информации. Публичный ключ раздается всем желающим. И с его помощью шифруются необходимые данные, которые можно дешифровать только с помощью приватного ключа. Отходя от темы, можно сказать, что так оно выглядит в теории. На практике все несколько менее строго. Из-за юридических ограничений на длину ключей, они поддаются взлому, хотя для этого и необходимы достаточно большие вычислительные мощности.

Теперь рассмотрим, каким образом все-таки работает SSL. Представьте себе, что есть два человека, которые общаются посредством Интернета и соответственно не видят друг друга. И лишены возможности, узнать, о том кто же его абонент. Их имена - Алиса и Боб. Допустим Алисе надо, узнать действительно она разговаривает с Бобом или нет. В этом случае диалог может выглядеть следующим образом: Алиса отправляет Бобу случайное сообщение. Боб шифрует его с помощью своего приватного ключа и отправляет его Алисе. Алиса дешифрует это сообщение (с помощью публичного ключа Боба). И сравнив это сообщение с посланным, может убедиться в том, что его действительно послал Боб. Но на самом деле со стороны Боба не очень удачная идея шифровать сообщение от Алисы с помощью своего приватного ключа. И возвращать его. Это аналогично подписи документа, о которой Боб мало что знает. С такой позиции Боб должен сам придумать сообщение. И послать его Алисе в двух экземплярах. В первом сообщение передается открытым текстом, а второе сообщение зашифровано с помощью приватного ключа Боба. Такое сообщение называется message digest. А способ шифрования сообщения с помощью своего приватного ключа - цифровой подписью (digital signature).

Теперь закономерно встает вопрос о том, каким образом распространять свои публичные ключи. Для этого (и не только) была придумана специальная форма - сертификат (certificate). Сертификат состоит из следующих частей:

- *Имя человека/организации выпускающего сертификат.*
- *Для кого был выпущен данный сертификат (субъект сертификата).*
- *Публичный ключ субъекта.*
- *Некоторые временные параметры (срок действия сертификата и т.п.).*

Сертификат подписывается приватным ключом человека (или организации), который выпускает сертификаты. Организации, которые производят подобные операции, называются Certificate authority **(CA)**. *Если в стандартном Web-клиенте (web-browser), который поддерживает SSL, зайти в раздел security, то там можно увидеть список известных организаций, которые подписывают сертификаты. С технической стороны, создать свою собственную CA достаточно просто. Но против этого могут действовать скорее юридические препятствия.*

Теперь рассмотрим, каким образом происходит обмен данными в Интернете. Воспользуемся все теми же действующими лицами.

Алиса*: привет.*

Боб*: привет, я Боб (выдает свой сертификат).*

Алиса*: а ты точно Боб?*

Боб*: Алиса я Боб. (Сообщение передается два раза, один раз в открытую, второй раз, зашифрованный с помощью приватного ключа Боба).*

Алиса*: все нормально, ты действительно Боб. (И присылает Бобу секретное сообщение, зашифрованное с помощью публичного ключа Боба).*

Боб: *А вот и мое сообщение (посылает сообщение, которое было зашифровано с помощью секретного ключа, например того же шифрованного сообщения Алисы).*
Поскольку Боб знает сообщение Алисы, потому что он владеет приватным ключом и Алиса знает, что было в том сообщении. Теперь они могут использовать симметричный шифровальный алгоритм (где в качестве секретного ключа выступает сообщение Алисы) и безбоязненно обмениваться шифрованными сообщениями. А для контроля над пересылкой сообщений (от случайного/преднамеренного изменения) используется специальный алгоритм - Message Authentication Code (MAC). Довольно распространенным является алгоритм MD5. Обычно, и сам MAC-code так же шифруется. В связи с этим достоверность сообщений повышается в несколько раз. И внести изменения в процесс обмена практически невозможно.

4. Обмен данными по протоколу SSL

Преимуществом SSL является то, что он независим от прикладного протокола. Протоколы приложения, такие как HTTP, FTP, TELNET и т.д. могут работать поверх протокола SSL совершенно прозрачно. Протокол SSL может согласовывать алгоритм шифрования и ключ сессии, а также аутентифицировать сервер до того как приложение примет или передаст первый байт данных. Все протокольные прикладные данные передаются зашифрованными с гарантией конфиденциальности.

4.1. Спецификация протокола записей SSL

Формат заголовка записи SSL

В SSL все данные пересылаются в виде записей, объектов, которые состоят из заголовка и некоторого количества данных. Каждый заголовок записи содержит два или три байта кода длины. Если старший бит в первом байте кода длины записи равен 1, тогда запись не имеет заполнителя и полная длина заголовка равна 2 байтам, в противном случае запись содержит заполнитель и полная длина заголовка равна 3 байтам. Передача всегда начинается с заголовка.
Заметим, что в случае длинного заголовка (3 байта), второй по старшинству бит первого байта имеет специальное значение. Когда он равен нулю, посылаемая запись является информационной. При равенстве 1, посылаемая запись является security escape (в настоящее время не определено ни одного значения security escapes; это зарезервировано для будущих версий протокола).
Код длины записи не включает в себя число байт заголовка (2 или 3). Для 2-байтового заголовка его длина вычисляется следующим образом (используется Си-подобная нотация):
RECORLENGTH = ((byte[0] & 0x7F << 8)) | byte[1];
Где byte[0] представляет собой первый полученный байт, а byte[1] – второй полученный байт.
Когда используется 3-байтовый заголовок, длина записи вычисляется следующим образом:
RECORD-LENGTH = ((byte[0] & 0x3F) << 8)) | byte[1];
IS-ESCAPE = (byte[0] & 0x40) != 0;
PADDING = byte[2];
Заголовок записи определяет значение, называемое PADDING. Значение PADDING специфицирует число байтов добавленных отправителем к исходной записи. Данные заполнителя используются для того, чтобы сделать длину записи кратной размеру блока шифра, если применен блочный шифр.
Отправитель "заполненной" записи добавляет заполнитель после имеющихся данных, а затем шифрует все это, так как длина этого массива кратна размеру блока используемого шифра. Содержимое заполнителя не играет роли. Так как объем передаваемых данных известен, заголовок сообщения может быть корректно сформирован с учетом объема субполя PADDING.

Получатель этой записи дешифрует все поле данных и получает исходную информацию. После этого производится вычисление истинного значения RECORD-LENGTH (с учетом наличия опционного PADDING), при этом заполнитель из поля данные удаляется.

Формат информационных записей SSL

Часть данных записи SSL состоит из трех компонентов (передаваемых и получаемых в приведенном ниже порядке):
MAC-DATA[MAC-SIZE]
ACTUAL-DATA[N]
PADDING-DATA[PADDING]
ACTUAL-DATA представляет собой реальные переданные данные (поле данных сообщения). PADDING-DATA – это данные заполнителя, посылаемые, когда используется блочный код шифрования. MAC-DATA является кодом аутентификации сообщения (Message Authentication Code).
Когда записи SSL посылаются открытым текстом, никаких шифров не используется. 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]
Где SECRET передается хэш-функции первым, далее следует ACTUAL-DATA и PADDING-DATA, за которыми передается SEQUENCE-NUMBER. Порядковый номер (SEQUENCE-NUMBER) представляет собой 32-битовый код, который передается хэш-функции в виде 4 байт. Первым передается старший байт (т.е., используется сетевой порядок передачи - "big endian").
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 верификации MAC). Если клиент получает сообщение, SECRET равен CLIENT-READ-KEY (сервер будет использовать SERVER-WRITE-KEY для генерации MAC).
SEQUENCE-NUMBER является счетчиком, который инкрементируется как сервером, так и получателем. Для каждого направления передачи, используется пара счетчиков (один для отправителя, другой для получателя). При отправлении сообщения счетчик инкрементируется. Порядковыми номерами являются 32-битовые целые числа без знака, которые при переполнении обнуляются.
Получатель сообщения использует ожидаемое значение порядкового номера для передачи хэш-функции MAC (тип хэш-функции определяется параметром CIPHER-CHOICE). Вычисленная MAC-DATA должна совпадать с переданной MAC-DATA. Если сравнение не прошло, запись считается поврежденной, такая ситуация рассматривается как случай "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) должна быть кратной размеру блока шифра. Если полученная запись не кратна размеру блока шифра, запись считается поврежденной, при этом считается, что имела место "I/O Error" (что вызовет разрыв соединения).
Уровень записей SSL используется для всех коммуникаций SSL, включая сообщения диалога и информационный обмен. Уровень записей SSL применяется как клиентом, так и сервером.
Для двухбайтового заголовка, максимальная длина записи равна 32767 байтов. Для трехбайтового заголовка, максимальная длина записи равна 16383 байтов. Сообщения протокола диалога SSL должны соответствовать одиночным записям протокола SSL (Record Protocol). Сообщения прикладного протокола могут занимать несколько записей SSL.

Прежде чем послать первую запись SSL все порядковые номера делаются равными нулю. При передаче сообщения порядковый номер инкрементируется, начиная с сообщений CLIENT-HELLO и SERVER-HELLO.

4.2. Спецификация протокола диалога SSL

Протокол диалога SSL

Протокол диалога SSL имеет две основные фазы. Первая фаза используется для установления конфиденциального канала коммуникаций. Вторая - служит для аутентификации клиента.

Фаза 1

Первая фаза является фазой инициализации соединения, когда оба партнера 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 мастерный ключ. Когда новый мастерный ключ не нужен, клиент и сервер немедленно переходят в фазу 2.

Когда нужен новый мастерный ключ, сообщение SERVER-HELLO будет содержать достаточно данных, чтобы клиент мог сформировать такой ключ. Сюда входит подписанный сертификат сервера, список базовых шифров (см. ниже), и идентификатор соединения (последний представляет собой случайное число, сформированное сервером и используемое на протяжении сессии). Клиент генерирует мастерный ключ и посылает сообщение CLIENT-MASTER-KEY (или сообщение ERROR, если информация сервера указывает, что клиент и сервер не могут согласовать базовый шифр).

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

Наконец, после того как мастерный ключ определен, сервер посылает клиенту сообщение SERVER-VERIFY. Этот заключительный шаг аутентифицирует сервер, так как только сервер, который имеет соответствующий общедоступный ключ, может знать мастерный 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 позитивный отклик, 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 один партнер выполнил аутентификацию другого партнера, он посылает сообщение finished. В случае клиента сообщение CLIENT-FINISHED содержит зашифрованную форму идентификатора CONNECTION-ID, которую должен верифицировать сервер. Если верификация терпит неудачу, сервер посылает сообщение ERROR.

Раз партнер послал сообщение finished он должен продолжить воспринимать сообщения до тех пор, пока не получит сообщение finished от партнера. Как только оба партнера послали и получили сообщения finished, протокол диалога SSL закончил свою работу. С этого момента начинает работать прикладной протокол.

4.3. Типовой протокол обмена сообщениями

В несколько упрощенном варианте диалог SSL представлен на рис.

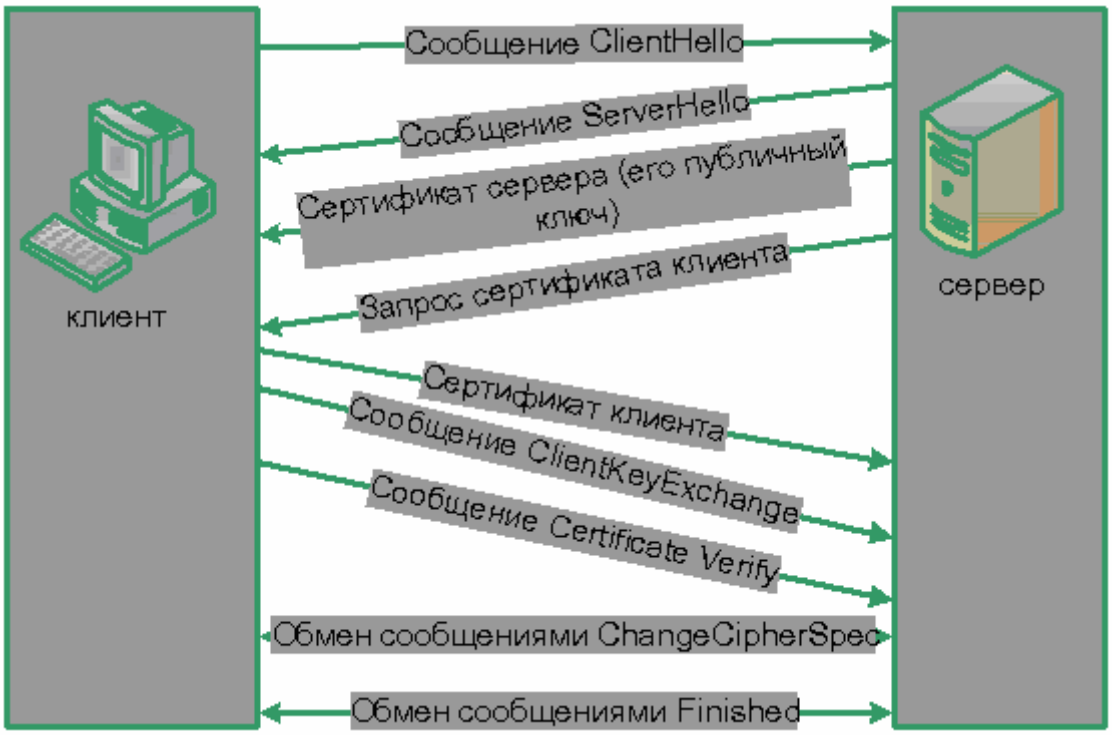

Рис 1. Алгоритм работы SSL

Ниже представлено несколько вариантов обмена сообщениями в рамках протокола диалога SSL. В этих примерах представлены два участника диалога: клиент (C) и сервер (S). 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*

Ошибки

Обработка ошибок в протоколе соединений SSL весьма проста. Когда ошибка детектирована, обнаруживший его посылает своему партнеру сообщение. Ошибки, которые являются неустранимыми, требуют от клиента и сервера разрыва соединения. Серверы и клиент должны "забыть" все идентификаторы сессии, сопряженные 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 также и сервером. Эта ошибка неустранима.
- *NO-CERTIFICATE-ERROR*
  Когда послано сообщение REQUEST-CERTIFICATE, эта ошибка может быть прислана, если клиент не имеет сертификата. Эта ошибка устранима.
- *BAD-CERTIFICATE-ERROR*
  Такой отклик присылается, 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*
  Этот отклик присылается, когда клиент/сервер получает тип сертификата, который он не поддерживает. Эта ошибка устранима (только для аутентификации клиента).

5. Обзор протокола TLS

5.1. Основные положения

Первоначальной целью протокола TLS (Transport Layer Security) является обеспечение конфиденциальности и целостности данных при коммуникации двух приложений. Протокол имеет два уровня: протокол записей TLS и протокол диалога TLS. На нижнем уровне, работающем поверх надежного транспортного протокола (например, TCP), размещается протокол записей TLS. Этот протокол обеспечивает безопасность соединений, которые имеют два основных свойства:

- *Соединение является конфиденциальным. Для шифрования данных используется симметричная криптография. Ключи для шифрования генерируются независимо для каждого соединения и базируются на секретном коде, получаемом с помощью другого*

протокола (такого как протокол диалога TLS). Протокол записей может использоваться и без шифрования.

- *Соединение является надежным. Процедура передачи сообщения включает в себя проверку целостности с помощью вычисления MAC. Для расчета MAC используются хэш-функции. Протокол записей может работать и без MAC, но в этом режиме он применяется только в случае, когда другой протокол использует протокол записей в качестве транспортного при выяснении параметров безопасности.*

Протокол записей TLS используется для инкапсуляции различных протоколов высокого уровня. Один из таких инкапсулируемых объектов, протокол диалога TLS, позволяет серверу и клиенту аутентифицировать друг друга и согласовать алгоритм шифрования и крипто-ключи до того как приложение передаст или примет первый байт информации. Протокол диалога TLS обеспечивает безопасное соединение, которое имеет три базовых свойства:

- *Идентичность партнеров может быть выяснена с использованием асимметричной криптографии. Эта аутентификация может быть сделана опционной, но она необходима, по крайней мере, для одного из партнеров.*
- *Выявление общего секретного кода является безопасным: этот секретный код недоступен злоумышленнику, даже если он ухитрится подключиться к соединению.*
- *Диалог надежен: атакующий не может модифицировать обсуждаемое соединение, без того чтобы быть обнаруженным партнерами обмена.*

Одним из преимуществ TLS является то, что он не зависит от протокола приложения. Протоколы верхнего уровня могут размещаться поверх протокола TLS прозрачным образом. Стандарт TLS, однако, не специфицирует то, как протоколы увеличивают безопасность с помощью TLS; решение о том, как инициализировать TLS-диалог и как интерпретировать сертификаты аутентификации, оставляется на усмотрение разработчиков протоколов и программ, которые работают поверх TLS.

5.2. Цели протокола TLS

Целями протокола TLS в порядке приоритетности являются:

- *Криптографическая безопасность. TLS должен использоваться для установления безопасного соединения между двумя партнерами.*
- *Совместимость. Независимые программисты должны быть способны разрабатывать приложения, использующие TLS, которые будут способны успешно обмениваться криптографическими параметрами без знания особенностей программ друг друга.*
- *Расширяемость. TLS ищет способ, как при необходимости встроить в систему новые ключи и методы шифрования. Здесь имеются две побочные цели: исключить необходимость создания нового протокола (что может быть сопряжено с введением новых слабых мест) и сделать ненужным внедрение новой библиотеки, обеспечивающей безопасность.*
- *Относительная эффективность. Криптографические операции требуют больших мощностей CPU, особенно этим славятся операции с открытыми ключами. По этой причине, протокол TLS имеет опционную схему кэширования сессии, что позволяет уменьшить число соединений, устанавливаемых с использованием новых временных буферов. Были приняты меры, чтобы уменьшить сетевую активность.*

5.3. Криптографические атрибуты

В TSL используются четыре криптографические операции:

- *цифровая подпись,*
- *блочное шифрование*
- *поточное шифрование*
- *шифрование с помощью общедоступного ключа.*

Криптографические ключи соответствуют состоянию текущей сессии.
Алгоритм цифровой подписи включает в себя однопроходные хэш-функции, служащие для преобразования подписываемого текста. Элемент с цифровой подписью кодируется как непрозрачный вектор <0..2^{16-1}>, где длина специфицируется алгоритмом подписи и ключом.
В подписи RSA, 36-байтовая структура двух хэшей (один SHA и один MD5) кодируется с помощью секретного ключа.
В DSS, 20 байтов хэша SHA передаются непосредственно алгоритму цифровой подписи DSA (Digital Signing Algorithm) без дополнительного хэширования. В результате получаются числа **r** *и* **s**. *Подпись DSS представляет собой непрозрачный вектор, содержимое которого представляет собой результат DER-кодирования.*
При поточном шифровании, исходный текст сначала объединяется с псевдослучайным кодом идентичной длины (формируется специальным генератором) с помощью операции исключающее ИЛИ.
При использовании блочного шифра, каждый блок исходного текста преобразуется в зашифрованный кодовый блок той же длины. Все блочные шифрования выполняются в режиме CBC (Cipher Block Chaining), и все зашифрованные блочные элементы будут иметь размер, кратный длине шифрового блока.
При шифровании с использованием общедоступного ключа, алгоритм открытого ключа используется для шифрования данных так, чтобы их можно было дешифровать только с помощью секретного ключа, который образует пару с открытым ключом. Элемент, зашифрованный с помощью открытого ключа, выглядит как непрозрачный вектор <0..2^{16-1}>, где длина определяется алгоритмом подписи и ключом.

5.4. Протокол записей TLS

Протокол записей TLS является послойным. На каждом уровне, сообщения могут включать поля длины, описания и содержимого. Протокол записей берет сообщения, подлежащие пересылке, разбивает их на блоки, опционно сжимает данные, применяет MAC, шифрует и передает результат. Полученные данные дешифруются, верифицируются, декомпрессируются, восстанавливается их первоначальный вид, результат передается клиентам верхнего уровня.
Четыре протокола описаны в данном документе: протокол диалога, протокол уведомления, протокол спецификации изменения шифра, и прикладной информационный протокол. Для того чтобы позволить расширение протокола TLS, разрешена поддержка протоколом дополнительных типов записей. Если реализация TLS получает рекорд нераспознаваемого типа, она должна его игнорировать. Любой протокол, предназначенный для использования поверх TLS, должен быть тщательно сконфигурирован, для того чтобы противостоять любым атакам. Заметим, что из-за того, что тип и длина записи не защищены шифрованием, следует принимать меры, чтобы минимизировать трафик анализа этих величин.

5.5. Состояния соединений

Состояние соединения TLS является операционной средой протокола записей TLS. Оно специфицирует алгоритмы сжатия, шифрования и MAC. Кроме того, известны параметры этих алгоритмов: секретный код MAC, а также ключи шифрования и IV соединения для направлений чтения и записи. Логически существует четыре состояния соединения: текущие состояния чтения и записи, и отложенные состояния чтения и записи. Все записи обрабатываются в текущих состояниях чтения или записи. Параметры безопасности для отложенных состояний могут быть установлены протоколом диалога TLS. Протокол диалога может селективно переводить любое отложенное состояние в текущее, при этом соответствующее текущее состояние становится отложенным. Не допускается формировать состояние, которое не инициализировано с учетом параметров безопасности текущего состояния. Исходное текущее состояние всегда специфицировано без компрессии,

шифрования или MAC. Параметры безопасности для состояния чтения и записи соединения TLS задаются путем определения следующих величин:

Конец соединения	*Клиент или сервер участник соединения.*
Алгоритм массового шифрования	*Алгоритм, используемый для массового шифрования. Эта спецификация включает размер ключа алгоритма, степень секретности ключа, является ли этот шифр блочным или поточным, размер блока и является ли шифр экспортным.*
Алгоритм MAC	*Алгоритм аутентификации сообщений. Эта спецификация включает размер хэша, который возвращается алгоритмом MAC.*
Алгоритм сжатия	*Алгоритм сжатия данных. Эта спецификация должна включать всю информацию, необходимую для выполнения компрессии.*
Секретный код сервера (master secret)	*48 байтовый секретный код, общий для обоих партеров в соединении.*
Случайный код клиента	*32 байтный код, предоставляемый клиентом.*
Случайный код сервера	*32 байтный код, предоставляемый сервером.*

5.6. Протокол диалога TLS

Протокол диалога TLS содержит набор из трех суб-протоколов, которые используются, чтобы партнеры могли согласовать используемые параметры безопасности для уровня записи, аутентифицировать себя, и уведомлять друг друга об ошибках.
Протокол диалога ответственен за согласования характеристик сессии, куда входят следующие объекты:

идентификатор сессии	*Произвольная последовательность байтов, выбранная сервером для идентификации состояния сессии (активная/ возобновляемая).*
сертификат партнера	*X509v3 [X509] сертификат партнера. Этот элемент состояния может быть равен нулю.*
метод сжатия	*Алгоритм, используемый для сжатия информации перед шифрованием.*
спецификация шифра	*Специфицирует алгоритм массового шифрования (такой как нуль, DES, и т.д.) и алгоритм MAC (такой как MD5 или SHA). Она определяет также криптографические атрибуты, такие как hash_size.*
мастерный секретный код	*48-байтовый секретный код, общий для сервера и клиента.*
'is resumable'	*Флаг, указывающий, может ли сессия использоваться для инициализации нового соединения.*

Эти объекты используются затем для определения параметров безопасности для уровня записей при защите прикладных данных. Многие соединения могут реализоваться в рамках той же сессии с помощью процедуры возобновления (resumption) протокола диалога.

5.7. Протокол изменения спецификации шифра

Протокол изменения спецификации шифра предназначен для оповещения об изменении стратегии шифрования. Протокол использует одно сообщение, которое зашифровано и архивировано в рамках текущего состояния соединения. Сообщение состоит из одного байта со значением 1.

Сообщение изменения спецификации шифра посылается как клиентом, так и сервером, для того чтобы уведомить партнера о том, что последующие записи будут защищены с помощью только что согласованных ключей и спецификации CipherSpec. Получение этого сообщения заставляет получателя на уровне записей немедленно скопировать состояние ожидания чтения в текущее состояние чтения. Сразу после посылки сообщения отправитель должен дать команду уровню записей преобразовать состояние ожидания записи в активное состояние записи. Сообщение изменения спецификации шифра посылается во время диалога после согласования набора параметров безопасности, но до посылки проверочного завершающего сообщения.

5.8. Протокол оповещения

Одним из типов содержимого, поддерживаемого слоем записей TLS, является оповещение. Сообщения оповещения передают описание возникшей ситуации. Оповещения с аварийным уровнем вызывают немедленное прерывание соединения. В этом случае, другие соединения сессии могут оставаться в рабочем состоянии, но идентификатор сессии должен быть объявлен не действительным, блокируя установление новых соединений. Подобно другим сообщениям, оповещения шифруются и сжимаются, как это специфицировано состоянием текущего соединения.

Оповещения закрытия

Клиент и сервер должны оба знать, что соединение завершается, для того чтобы избежать атаки усечения (truncation). Оба партнера могут запустить обмен сообщениями закрытия.

Оба партнера могут инициализировать закрытие, послав уведомление close_notify. Любые данные, полученные после оповещения о закрытии, игнорируются.

Каждый из партнеров обязан послать уведомление close_notify, прежде чем разрывать соединение со стороны записи. Требуется, чтобы другой партнер реагировал своим уведомлением close_notify и закрывал соединение немедленно, аннулируя все не завершенные записи. Для инициатора закрытия не требуется ждать получения отклика close_notify, прежде чем закрыть соединение со стороны чтения. Если прикладной протокол, использующий TLS, гарантирует, что любые данные могут быть переданы через используемое TLS-соединение после его закрытия, реализация TLS должна получить уведомление-отклик close_notify до оповещения прикладного уровня о том, что соединение TLS завершает свою работу. Если прикладной протокол не передает никаких дополнительных данных, но лишь закрывает ниже лежащее транспортное соединение, тогда реализация может выбрать вариант закрытия транспорта, не дожидаясь отклика close_notify.

Предполагается, что закрытие соединения надежно доставляет все данные, ждущие передачи, прежде чем транспортная система будет блокирована.

Оповещения об ошибках

Обработка ошибок в протоколе диалога TLS очень проста. Когда обнаруживается ошибка, обнаруживший партнер посылает сообщение другому партнеру. При передаче или получении сообщения о фатальной ошибке, оба партнера немедленно закрывают соединение. Серверы и клиенты должны забыть любые идентификаторы сессии, ключи и секретные коды, связанные с неудачным соединением.

5.9. Обзор протокола диалога

Криптографические параметры состояния сессии формируются протоколом диалога TLS, который работает поверх уровня записей TLS. Когда клиент и сервер TLS впервые начинают взаимодействие, они согласуют версию протокола, выбирают криптографические алгоритмы,

опционно аутентифицируют друг друга и используют методику с общедоступным ключом для формирования общего секретного кода. Протокол диалога TLS включает в себя следующие шаги:

- *Обмен сообщениями hello, чтобы согласовать алгоритмы, обмен случайными кодами, и проверка перезапуска сессии.*
- *Обмен необходимыми криптографическими параметрами, чтобы позволить клиенту и серверу согласовать предмастерные секретные коды.*
- *Обмен сертификатами и криптографической информацией, чтобы позволить клиенту и серверу аутентифицировать друг друга.*
- *Генерация мастерного секретного кода из предмастерного и обмен случайными кодами.*
- *Предоставление параметров безопасности уровню записей.*
- *Разрешение клиенту и серверу проверить, что их партнер вычислил те же самые параметры безопасности и что диалог прошел без вмешательства хакера.*

Заметим, что верхние слои не должны слишком полагаться на TLS, всегда согласуя самые безопасные из возможных соединений между партнерами: существует много способов, с помощью которых злоумышленник, включившийся в разрыв соединения, может попытаться заставить партнеров принять наименее безопасный метод связи из числа поддерживаемых ими. Протокол был устроен так, чтобы минимизировать этот риск, но, тем не менее, существуют некоторые возможности атак. Фундаментальным правилом является то, что верхние уровни должны знать, каковы требования безопасности и никогда не передавать данные по каналам, которые менее безопасны, чем это предписано этими требованиями. Протокол TLS является безопасным, здесь любой шифровой набор предлагает свой уровень безопасности. Если вы согласуете использование 3DES с 1024-битовым RSA-ключом при связи с ЭВМ, чей сертификат вы проверили, вы можете быть уверены в безопасности. Однако вы никогда не должны посылать данные по каналу, где используется 40-битовая шифрование, если только вы не уверены, что данные не стоят того, чтобы кто-то тратил силы на их дешифрование.

Эти цели достигаются протоколом диалога, который может быть суммирован следующим образом. Клиент посылает сообщение hello, на которое сервер должен также откликнуться сообщением hello, в противном случае возникает ситуация фатальной ошибки и соединение разрывается. Сообщения client hello и server hello используются для установления более безопасного взаимодействия клиента и сервера. Сообщения client hello server hello устанавливают следующие атрибуты: версия протокола, ID-сессии, шифровой набор и метод сжатия. Кроме того, партнеры генерируют и пересылают друг другу два случайных числа: ClientHello.random и ServerHello.random.

Реальный обмен ключами использует до четырех сообщений: сертификат сервера, ключевой обмен сервера, сертификат клиента и ключевой обмен клиента. Новые методы ключевого обмена могут быть созданы с помощью спецификации формата для этих сообщений, чтобы позволить клиенту и серверу согласовать использование общего секретного кода. Этот секретный код должен быть достаточно длинным. Современные методы ключевого обмена пересылают коды длиной от 48 до 128 байт.

Вслед за сообщениями hello, сервер, если он должен быть аутентифицирован, посылает свой сертификат. Кроме того, если необходимо, может быть послано сообщение ключевого обмена (например, если сервер не имеет сертификата, или если его сертификат служит только для подписи). Если сервер аутентифицирован, он может затребовать сертификат от клиента, если выбран соответствующий шифровой набор. После этого сервер пошлет сообщение hello done, указывающее, что фаза диалога hello завершена. Сервер ждет отклика клиента. Если сервер послал сообщение сертификатного запроса, клиент должен послать сообщение сертификата. Сообщение ключевого обмена клиента послано, и его содержимое зависит от алгоритма с общедоступным ключом, который выбрали клиент и сервер при обмене сообщениями hello.

В этой точке клиентом посылается сообщение об изменении спецификации шифра, и клиент копирует записанную шифровую спецификацию в текущую спецификацию. После этого клиент немедленно посылает сообщение finished для новых алгоритмов, ключей и секретных кодов. В качестве отклика сервер пошлет свое сообщение об изменении шифровой спецификации, перенесет записанную шифровую спецификацию в текущую, и пошлет свое сообщение finished с использованием новой шифровой спецификации. В этой точке диалог завершается, а клиент и сервер могут начать обмен прикладными данными.

Когда клиент и сервер решают возобновить предыдущую сессию или задублировать существующую сессию (вместо согласования новых параметров безопасности), следует обмен следующими сообщениями:

Клиент посылает ClientHello, используя ID-сессии, которая должна быть возобновлена. Сервер проверяет свой кэш сессий на соответствие. Если соответствие имеется, а сервер желает возобновить соединение со специфицированным состоянием сессии, он посылает ServerHello с тем же значением ID-сессии. В этой точке, как клиент, так и сервер должны послать сообщения об изменении шифровой спецификации, после чего перейти к завершающим сообщениям finished. Раз восстановление сессии завершилось, клиент и сервер могут начать обмен прикладными данными. Смотри диаграмму на рис. .2. Если соответствия с ID-сессии не найдено, сервер генерирует новый ID сессии, а клиент TLS и сервер осуществляют полный диалог.

6. Заключение

При рассмотрении данного реферата были затронуты содержание протоколов, их общий принцип работы, недостатки и положительные моменты. Видно, что они имеют непростую организацию, вызванную необходимостью шифрования и обеспечения надежной и конфиденциальной передачи данных между клиентом и сервером. Однако безусловно необходимость использование безопасных протоколов не вызывает сомнение. И по мере развития электронной коммерции и различных финансовых сервисов в сети Интернет применение протоколов безопасности будет только расти, и будет происходить их усовершенствование и повышение надежности и безопасности.

7. Список литературы:

1. *Д. Ф. Куроуз. Компьютерные сети. 2-е изд. – СПб. Питер, 2004 г.*
2. *В. Столингс. Передача данных. 4-е изд. – СПб. Питер, 2004 г.*
3. *www.excode.ru*
4. *www.ssl.ru*