# SSL – Secure Socket Layer, TLS – Transport Layer Security

## Что такое SSL и что такое TLS?

SSL – Secure Socket Layer, уровень защищенных сокетов. TLS – Transport Layer Security, безопасность транспортного уровня. SSL является более ранней системой, TLS появился позднее и он основан на спецификации SSL 3.0, разработанной компанией Netscape Communications. Тем не менее, задача у этих протоколов одна – обеспечение защищенной передачи данных между двумя компьютерами в сети Интернет. Такую передачу используют для различных сайтов, для электронной почты, для обмена сообщениями и много еще для чего. В принципе, можно передавать любую информацию таким образом, об этом чуть ниже.

Безопасная передача обеспечивается при помощи аутентификации и шифрования передаваемой информации. По сути эти протоколы, TLS и SSL, работают одинаково, принципиальных различий нет. TLS, можно сказать, является преемником SSL, хотя они и могут использоваться одновременно, причем даже на одном и том же сервере. Такая поддержка необходима для того, чтобы обеспечить работу как с новыми клиентами (устройствами и браузерами), так и с устаревшими, которые TLS не поддерживают.

## Многослойная среда SSL

Безопасный SSL протокол размещается между двумя протоколами: протоколом, который использует программа-клиент (HTTP, FTP, IMAP, LDAP, Telnet и т.д.) и транспортным протоколом TCP/IP. Создавая своего рода заслонки с обеих сторон, он защищает и передает данные на транспортный уровень. Благодаря работе по

многослойному принципу, SSL протокол может поддерживать много разных протоколов программ-клиентов.

Работу протокола SSL можно разделить на два уровня.

*Первый уровень* – слой протокола подтверждения подключения (Handshake Protocol Layer). Он состоит из трех подпротоколов: протокол подтверждения подключения (Handshake Protocol), протокол изменения параметров шифра (Change Cipher Spec Protocol) и предупредительный протокол (Alert protocol).

*Второй уровень* – это слой протокола записи. На рис.1 схематически изображены уровни слоев SSL

# Уровень подтверждения подключения состоит из трех подпротоколов:

1. ***Подтверждение подключения.*** Этот подпротокол используется для согласования данных сессии между клиентом и сервером. В данные сессии входят:
* идентификационный номер сессии;
* сертификаты обеих сторон;
* параметры алгоритма шифрования, который будет использован;
* алгоритм сжатия информации, который будет использоваться;
* «общий секрет», применён для создания ключей; открытый ключ
2. ***Изменения параметров шифрования.*** Этот подпротокол используется для изменения данных ключа (keyingmaterial), который используется для шифрования данных между клиентом и сервером. Данные ключа – это информация, которая используется для создания ключей шифрования. Подпротокол изменения параметров шифрования состоит из одного единственного сообщения. В этом сообщении сервер говорит, что отправитель хочет изменить набор ключей. Дальше, ключ вычисляется из информации, которой обменялись стороны на уровне подпротокола подтверждения подключения.

3. ***Предупреждение.*** Предупредительное сообщение показывает сторонам изменение статуса или о возможной ошибке. Существует множество предупредительных сообщений, которые извещают стороны, как при нормальном функционировании, так и при возникновении ошибок. Как правило, предупреждение отсылаются тогда, когда подключение закрыто и получено неправильное сообщение, сообщение невозможно расшифровать или пользователь отменяет операцию.

Подпротокол подтверждения подключения обеспечивает реализацию многих функций безопасности. Он производит цепочку обмена данными, что в свою очередь начинает проверку подлинности сторон и согласовывает шифрование, алгоритмы хеширования и сжатия.

# Установление подлинности участников

Для определения подлинности участников обмена данных, протокол подтверждения подключения использует сертификат стандарта X.509. Это является доказательством для одной стороны, так как помогает подтвердить подлинность другой стороны, которая владеет сертификатом и секретным ключом. Сертификат – это цифровой способ идентификации, который выпускает центр сертификации. В сертификате содержится идентификационная информация, период действия, публичный ключ, серийный номер, и цифровые подписи эмитента.

**Сертификационный центр** – это третья сторона, которой по умолчанию доверяют обе стороны. При попытке установить подключение в режиме SSL сессии, сертификационный центр проверяет инициатора (обычно в этой роли выступает пользователь, компьютер клиента), а затем выдает ему сертификат. Если необходимо, сертификационный центр обновляет или конфискует сертификаты. Проверка подлинности проходит по схеме:

* клиенту предоставлен сертификат сервера;
* компьютер клиента пытается сопоставить эмитента сертификата

сервера со списком доверительных сертификационных центров;
* если эмитент сертификата – доверительный сертификационный центр, то клиент связывается и этим центром и проверяет, является ли сертификат настоящим, а не подделанным;
* после проверки сертификата у сертификационного центра, клиент принимает сертификат как свидетельство подлинности сервера.

## Шифрование данных

Существует два основных способа шифрования данных: симметричный ключ (еще называется «общий секретный ключ») и ассиметричный ключ (второе название «открытый ключ» или «схема открытый-секретный ключ»). Протокол SSL использует как симметричные, так и ассиметричные ключи для шифрования данных.

**SSL-ключ** – это зашифрованные данные, которые используются для определения схемы шифрования данных во время сессии. Чем длиннее ключ, тем труднее его взломать. Как правило, алгоритмы ассиметричных ключей более стойкие их практически невозможно взломать.

**Симметричный ключ.** При шифровании симметричным ключом, используется один и тот же ключ для шифрования данных. Если две стороны хотят обмениваться зашифрованными сообщениями в безопасном режиме, то обе стороны должны иметь одинаковые симметричные ключи. Шифрование симметричным ключом обычно используется для шифрования большого объема данных, так как это процесс проходит быстрее, чем при ассиметричном шифровании. Обычно используются алгоритмы DES (Data Encryption Standard – стандарт шифрования данных), 3-DES (тройной DES), RC2, RC4, и AES (Advanced Encryption Standard – современный стандарт шифрования).

**Ассиметричный ключ.** Шифрование с применением ассиметричного (открытого) ключа использует пару ключей, которые оба были получены, пройдя целый комплекс математических вычислений. Один из ключей используется в качестве открытого, как правило,

сертификационный центр публикует открытый ключ в самом сертификате владельца (обычно это является заголовком (subject)). Секретный ключ держится в тайне и никогда никому не окрывается. Эти ключи работают в паре: один ключ используется для запуска противоположных функций второго ключа. Так, если открытый ключ используется чтоб шифровать данные, то расшифровать их можно только секретным ключом. Если данные шифруются секретным ключом, то открытый ключ должен это расшифровывать. Такая взаимосвязь позволяет, используя схему шифрования открытым ключом, делать две важные вещи. Во-первых, любой пользователь может получить открытый ключ по назначению и использовать его для шифрования данных, расшифровать которые может только пользователь, у которого есть секретный ключ. Во-вторых, если заголовок шифрует данные, используя свой секретный ключ, каждый может расшифровать данные, используя соответствующий открытый ключ. Именно это является основой для цифровых подписей. Самый распространенный алгоритм, который используется при шифровании с ассиметричными ключами – RSA (назван в честь разработчиков Rivest, Shamir, Adleman).

Протокол SSL использует шифрование с открытым ключом для того, чтоб подтвердить клиенту подлинность сервера, и наоборот. Шифрование открытым ключом также используется для определения ключа сессии. Ключ сессии используется симметричными алгоритмами для шифровки большого объема данных. Это объединяет ассиметричное шифрование (для проверки подлинности) и быстрое симметричное шифрование объемных данных (что не требует больших вычислительных ресурсов и больших затрат времени).

## Хэширование

Во время подтверждения подключения согласовывается также и хэш-алгоритм. Хэш–функция – это односторонняя математическая функция, которая принимает на входе сообщение произвольной длинны и вычисляет из него строку фиксированной длины. Хеш-значение играет роль идентификационной отметки, «отпечаток

сообщения». Как и отпечатки пальцев уникальны для каждого человека, хеш-значения тоже уникальны. Кроме того, как отпечатки пальцев значительно меньше, чем сам человек, так и хеш-значение намного меньше оригинального сообщения. Хэширование используется для обеспечения целостности передачи данных. Самыми популярными хэш-алгоритмами являются MD5 (Message Digest 5 – дайджест сообщения, 5 версия) и SHA-1 (Standard Hash Algorithm – стандартный алгоритм хэширования). MD5 создает 128 битное хэш-значение, а SHA-1 создает 160 битное хэш-значение. Есть также новые, более устойчивые алгоритмы хэширования: WHIRLPOOL, SHA-512, SHA-384, HAVAL, Tiger(2).

Результатом работы хэш-алгоритма выступает значение, которое используется для проверки целостности переданных данных. Это значение создается с использованием либо MAC либо HMAC. MAC – Message Authentication Code – код проверки сообщения. Он использует отображающую функцию и предоставляет данные в виде значений фиксированного размера, а затем – хэширует само сообщение. MAC гарантирует, что данные не были изменены во время передачи. Разница между MAC и цифровой подписью состоит в том, что цифровая подпись это также способ подтверждения подлинности. SSL использует MAC.

HMAC – Hashed Message Authentication Code – хэшированный код проверки сообщения. HMAC похож на MAC, но при этом используется хэш-алгоритм вместе с общим секретным ключом. Общий секретный ключ прикрепляется к данным, которые хэшируются. Это позволяет сделать хэширование более безопасным, так как обе стороны должны иметь одинаковые секретные ключи для подтверждения подлинности данных. HMAC используется только протоколом TLC.

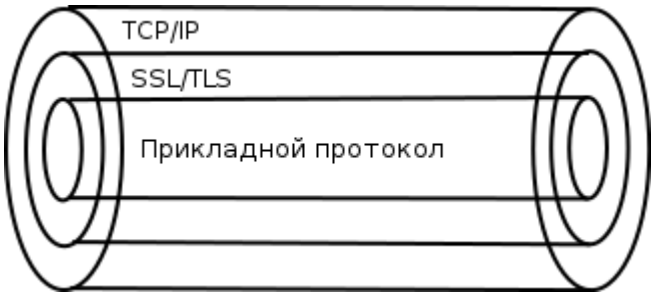
# Уровень записи

Протокол на уровне слоя записи получает зашифрованные данные от программы-клиента и передает его на транспортный слой.

Протокол записи берет данные, разбивает на блоки размером, который подходит криптографическому алгоритму, использует MAC (или HMAC) и потом шифрует (расшифровывает) данные. При этом используется информация, которая была согласованна во время протокола подтверждения данных. В некоторых случая на этом уровне проходит сжатие (распаковка) данных.

## Принцип работы SSL и TLS

Принцип работы SSL и TLS, как я уже сказал, один и тот же. Поверх протокола TCP/IP устанавливается зашифрованный канал, внутри которого передаются данные по прикладному протоколу — HTTP, FTP, и так далее. Вот как это можно представить графически:




Прикладной протокол «заворачивается» в TLS/SSL, а тот в свою очередь в TCP/IP. По сути данные по прикладному протоколу передаются по TCP/IP, но они зашифрованы. И расшифровать передаваемые данные может только та машина, которая установила соединения. Для всех остальных, кто получит передаваемые пакеты, эта информация будет бессмысленной, если они не смогут ее расшифровать.

Установка соединения обеспечивается в несколько этапов:

1) Клиент устанавливает соединение с сервером и запрашивает защищенное подключение. Это может обеспечиваться либо установлением соединения на порт, который изначально предназначен для работы с SSL/TLS, например, 443, либо дополнительным запросом клиентом установки защищенного

соединения после установки обычного.

2) При установке соединения клиент предоставляет список алгоритмов шифрования, которые он «знает». Сервер сверяет полученный список со списком алгоритмов, которые «знает» сам сервер, и выбирает наиболее надежный алгоритм, после чего сообщает клиенту, какой алгоритм использовать

3) Сервер отправляет клиенту свой цифровой сертификат, подписанный удостоверяющим центром, и открытый ключ сервера.

4) Клиент может связаться с сервером доверенного центра сертификации, который подписал сертификат сервера, и проверить, валиден ли сертификат сервера. Но может и не связываться. В операционной системе обычно уже установлены корневые сертификаты центров сертификации, с которыми сверяют подписи серверных сертификатов, например, браузеры.

5) Генерируется сеансовый ключ для защищенного соединения. Это делается следующим образом:
– Клиент генерирует случайную цифровую последовательность
– Клиент шифрует ее открытым ключом сервера и посылает результат на сервер
– Сервер расшифровывает полученную последовательность при помощи закрытого ключа
Учитывая, что алгоритм шифрования является асимметричным, расшифровать последовательность может только сервер. При использовании асимметричного шифрования используется два ключа – приватный и публичный. Публичным отправляемое сообщение шифруется, а приватным расшифровывается. Расшифровать сообщение, имея публичный, ключ нельзя.

6) Таким образом устанавливается зашифрованное соединение. Данные, передаваемые по нему, шифруются и расшифровываются до тех пор, пока соединение не будет разорвано.