**Для Push-уведомлений**

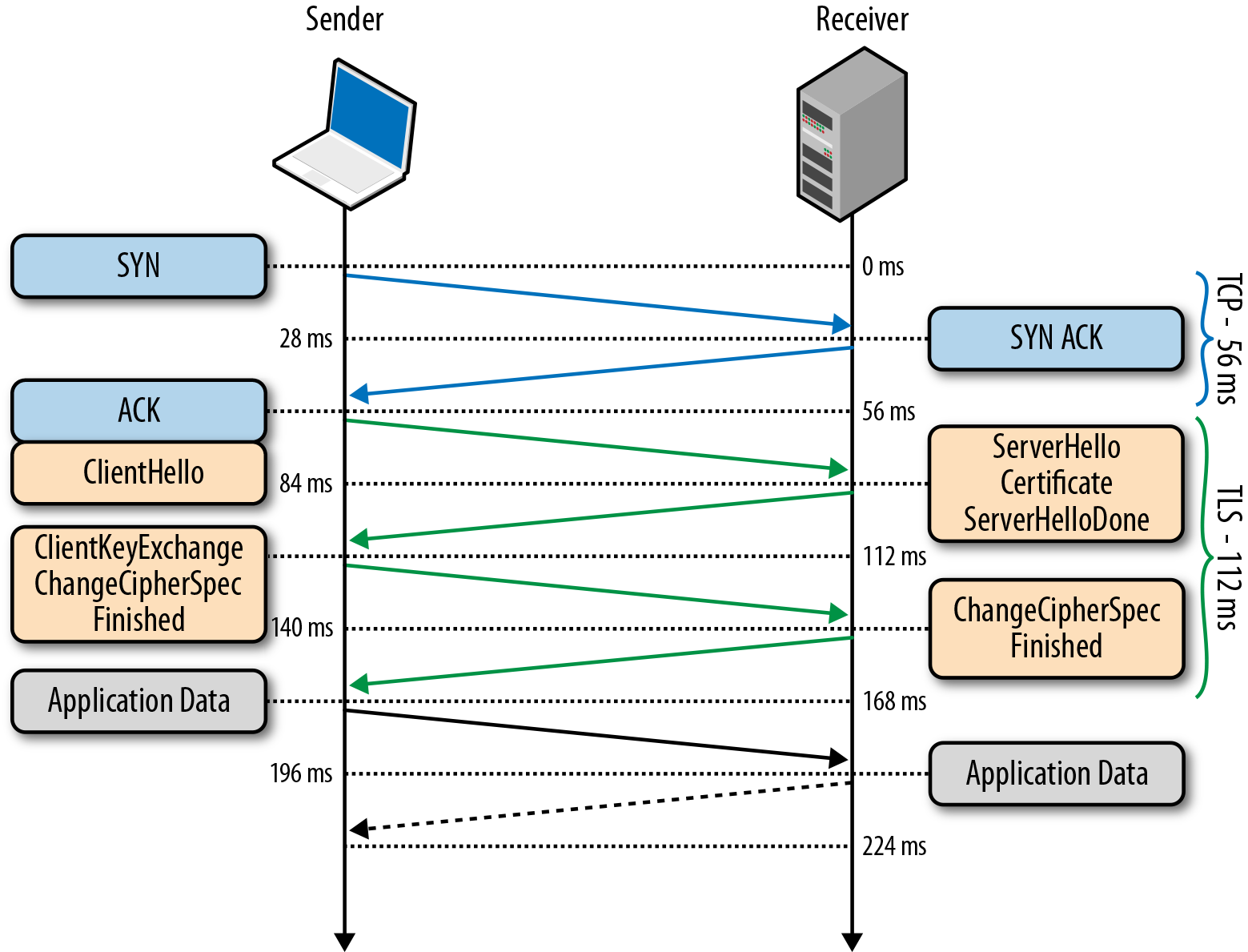
**http(s) с использованием**

**TSL**

Общие сведения о TLS

Протокол TLS (transport layer security) основан на протоколе SSL (Secure Sockets Layer), изначально разработанном в Netscape для повышения безопасности электронной коммерции в Интернете. Протокол SSL был реализован на application-уровне, непосредственно над TCP (Transmission Control Protocol), что позволяет более высокоуровневым протоколам (таким как HTTP или протоколу электронной почты) работать без изменений. Если SSL сконфигурирован корректно, то сторонний наблюдатель может узнать лишь параметры соединения (например, тип используемого шифрования), а также частоту пересылки и примерное количество данных, но не может читать и изменять их.

**TLS Handshake**

Перед тем, как начать обмен данными через TLS, клиент и сервер должны согласовать параметры соединения, а именно: версия используемого протокола, способ шифрования данных, а также проверить сертификаты, если это необходимо. Схема начала соединения называется TLS Handshake и показана на рисунке:  
  
  
Разберём подробнее каждый шаг данной процедуры:

1. Так как TLS работает над TCP, для начала между клиентом и сервером устанавливается TCP-соединение.
2. После установки TCP, клиент посылает на сервер спецификацию в виде обычного текста (а именно версию протокола, которую он хочет использовать, поддерживаемые методы шифрования, etc).
3. Сервер утверждает версию используемого протокола, выбирает способ шифрования из предоставленного списка, прикрепляет свой сертификат и отправляет ответ клиенту (при желании сервер может так же запросить клиентский сертификат).
4. Версия протокола и способ шифрования на данном моменте считаются утверждёнными, клиент проверяет присланный сертификат и инициирует либо RSA, либо обмен ключами по Диффи-Хеллману, в зависимости от установленных параметров.
5. Сервер обрабатывает присланное клиентом сообщение, сверяет MAC, и отправляет клиенту заключительное (‘Finished’) сообщение в зашифрованном виде.
6. Клиент расшифровывает полученное сообщение, сверяет MAC, и если всё хорошо, то соединение считается установленным и начинается обмен данными приложений.

**Обмен ключами в протоколе TLS**

По различным историческим и коммерческим причинам чаще всего в TLS используется обмен ключами по алгоритму RSA: клиент генерирует симметричный ключ, подписывает его с помощью открытого ключа сервера и отправляет его на сервер. В свою очередь, на сервере ключ клиента расшифровывается с помощью закрытого ключа. После этого обмен ключами объявляется завершённым. Данный алгоритм имеет один недостаток: эта же пара отрытого и закрытого ключей используется и для аутентификации сервера. Соответственно, если злоумышленник получает доступ к закрытому ключу сервера, он может расшифровать весь сеанс связи. Более того, злоумышленник может попросту записать весь сеанс связи в зашифрованном варианте и занять расшифровкой потом, когда удастся получить закрытый ключ сервера. В то же время, обмен ключами Диффи-Хеллмана представляется более защищённым, так как установленный симметричный ключ никогда не покидает клиента или сервера и, соответственно, не может быть перехвачен злоумышленником, даже если тот знает закрытый ключ сервера. На этом основана служба снижения риска компрометации прошлых сеансов связи: для каждого нового сеанса связи создаётся новый, так называемый «временный» симметричный ключ. Соответственно, даже в худшем случае (если злоумышленнику известен закрытый ключ сервера), злоумышленник может лишь получить ключи от будущих сессий, но не расшифровать ранее записанные.

На текущий момент, все браузеры при установке соединения TLS отдают предпочтение именно сочетанию алгоритма Диффи-Хеллмана и использованию временных ключей для повышения безопасности соединения.

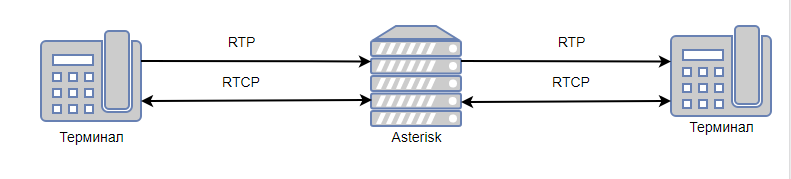
Следует ещё раз отметить, что шифрование с открытым ключом используется только в процедуре TLS Handshake во время первоначальной настройки соединения. После настройки туннеля в дело вступает симметричная криптография, и общение в пределах текущей сессии зашифровано именно установленными симметричными ключами. Это необходимо для увеличения быстродействия, так как криптография с открытым ключом требует значительно больше вычислительной мощности.

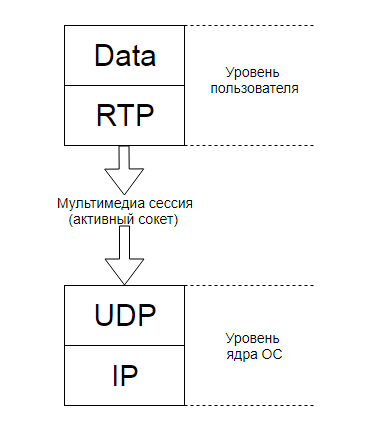
**Для защиты звонков**

Протокол RTP является одним из технических основ в VoIP-телефонии, обеспечивая передачу голоса в таких протоколах, как SIP

**Взаимодействие RTP и RTCP**

Взаимодействие механизмов работы RTP и RTCP происходит в режиме сессии. Для передачи видео либо аудио потока устанавливается отдельная сессия. В заголовке каждого пакета указывается тип полезной нагрузки, которая в него заключена. Работа протоколов RTP и RTCP происходит по двум разным портам (могут использоваться различные варианты портов), как правило для RTP назначается порт с четным номером, а на RTCP следующий порт с нечетным номером. В соответствии со спецификациями, нет четко определенных портов для данного взаимодействия.

Представление сессии взаимодействия RTP/RTCP при звонке через АТС Asterisk



Представление сессии взаимодействия при передаче мультимедиа данных

**Безопасность при передаче мультимедиа данных**

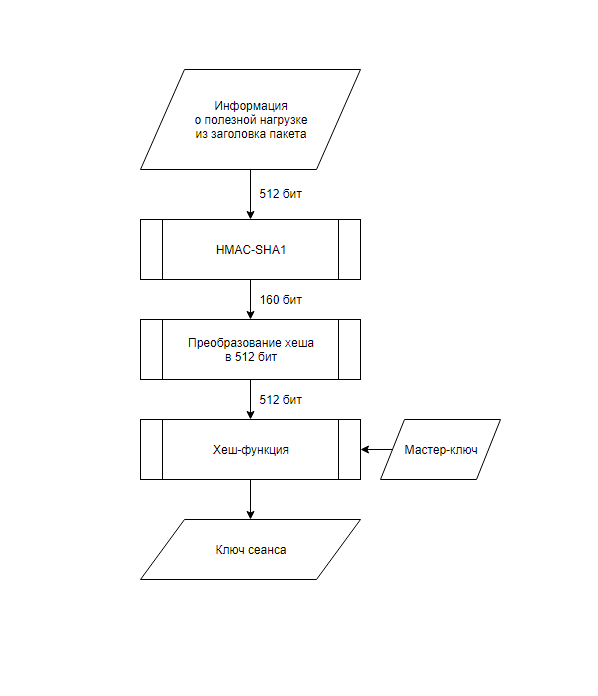
Существуют две реализации шифрования медиаданных при их передаче. Обе эти реализации основаны на протоколе RTP, отличает их способ передачи ключа. SRTP передает ключ на этапе установления сессии, а ZRTP — уже после, перед началом передачи данных. Для работы SRTP в качестве протокола установления сессии может быть использован только SIP\TLS для защиты от того, что передаваемый ключ попадет в руки злоумышленников.

**Протокол SRTP**

SRTP является наиболее часто используемым решением, в том случае, если вам необходимо шифровать передаваемую мультимедиа. SRTP предоставляет возможности по шифрованию передаваемых сообщений, их аутентификации, проверке целостности, и предохраняет от возможности проведения плейбек-атак (тип атак, проводимых с перехватом и подменой ключа). Важным моментом в работе SRTP является необходимость поддержки шифрования с обеих сторон, между которыми происходит передача данных.

Шифрование информации выполняется методом AES, что положительно отражается на криптозащите информации и скорости работы протокола.Возможности по конфигурированию шифрования в SRTP весьма широки вплоть до использования любого, описанного в RFC стандарта шифрования (с четко описанным алгоритмом) или отключения шифрования вовсе (используя NULL-шифр). Отключение шифрования может быть полезно для экономии аппаратных ресурсов в том случае, если для вас важно обеспечение целостности и правильного порядка доставки данных, но не требуется шифрование. При этом сохраняется защита от playback-атак, так как блоки передаваемой информации нумеруются и исключена возможность повторной передачи уже ранее принятого сообщения. Методы шифрования SRTP обеспечивают не только безопасную передачу, но так же и безопасное воспроизведение мультимедиа, т.е. злоумышленник не только не сможет расшифровать данные, но и также не сможет их подменить либо воспроизвести. Для аутентификации сообщений и защите его целостности используется алгоритм HMAC-SHA1, в качестве аргумента функция хеширования получает данные из заголовка пакета о полезной нагрузке им переносимой, т.е. обеспечивается высокий уровень криптозащиты. Слабым звеном в данной схеме является то, что заголовки пакетов SRTP не шифруются и по данным о передаваемой информации, злоумышленник может сделать косвенное предположение что происходит обмен мультимедиа и каким-то образом помешать передаче.

Еще одним механизмом обеспечения безопасности данных в протоколе SRTP является обновление ключей идентификации. На основе главного ключа генерируются дополнительные идентификационные ключи для каждой отдельной сессии. Таким образом, даже если злоумышленник получит ключ, передаваемый в сессии, то он не поможет ему для расшифровки других сессий, и объем полученных им данных не будет сколь-либо значительным.

Механизм генерации ключей безопасности