

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Российский технологический университет»**

**МИРЭА**

Институт кибернетики

**Кафедра информационной безопасности**

ДОКЛАД

по дисциплине

«Криптографические протоколы»

На тему:

**«Протоколы защиты текстовой/голосовой/видео связи»**

**Подготовили**

студенты групп ККСО−01/03−14

С.А. Крюков

К.И. Нарежный

А.С. Першин

**Руководитель работы**

А.П. Никитин

Москва, 2019

Оглавление

[Протоколы защиты текстовой связи 3](#_Toc5348408)

[Signal 3](#_Toc5348409)

[MTProto 4](#_Toc5348410)

[EL337 5](#_Toc5348411)

[Протоколы защиты звуковой связи 6](#_Toc5348412)

[Протокол XMPP 6](#_Toc5348413)

[Протокол WebRTC 9](#_Toc5348414)

[Протокол RTP 12](#_Toc5348415)

[Протокол SIP 15](#_Toc5348416)

[Протоколы защиты видеосвязи 17](#_Toc5348417)

[Протокол Skype 17](#_Toc5348418)

[Протокол Tox 18](#_Toc5348419)

[Протокол SRTP 21](#_Toc5348420)

[Протокол ZRTP 23](#_Toc5348421)

# **Протоколы защиты текстовой связи**

## Signal

Протокол Signal — криптографический протокол, созданный компанией Open Whisper Systems для обеспечения сквозного шифрования голосовых вызовов, видеозвонков и мгновенных сообщений. Протокол был разработан Open Whisper Systems в 2013 году и впервые был представлен в приложении TextSecure с открытым исходным кодом, которое позже было объединено в приложение Signal. Этот протокол используют во многих сторонних приложениях: WhatsApp, Facebook Messenger, Viber, Google Allo, G Data Secure Chat — все они используют оригинальную или слегка модифицированную версию Signal Protocol, иногда давая им собственные названия. Например, у Viber это протокол Proteus — по сути, тот же Signal с другими кpиптографическими примитивами.

Протокол сочетает в себе алгоритм управления ключами Double Ratchet Algorithm и расширенный протокол тройного обмена ключами Диффи-Хеллмана (3-DH) и использует Curve25519 , AES-256 и HMAC-SHA256 в качестве примитивов.

## MTProto

Для достижения этой безопасности Telegram использует собственную разработку — криптографический протокол MTProto.

Прежде чем сообщение (или составное сообщение / сообщение из нескольких частей) будет передано по сети с использованием транспортного протокола, оно зашифровывается определённым образом, и вверху сообщения добавляется внешний заголовок, который представляет собой: 64-битный идентификатор ключа (который уникально идентифицирует ключ авторизации для сервера, а также для юзера) и 128-битный ключ сообщения.

Ключ юзера вместе с ключом сообщения определяет актуальный (текущий) 256-битный ключ и 256-битный вектор инициализации, который шифрует сообщение, используя AES-256 шифрование с расширением неопределённого искажения. Обратите внимание, что часть сообщения, которая должна быть зашифрована, содержит переменные данные (сессию, ID сообщения, порядковый номер, соль сервера), которые явно оказывают влияние на ключ сообщения (и таким образом на ключ AES и iv). Ключ сообщения определяется 128 битами нижнего порядка от SHA1 тела сообщения (включая сессию, ID сообщения, и т. д.) Составные сообщения шифруются как одно сообщение.

Более подробно описание протокола вы можете найти на <https://tlgrm.ru/docs/mtproto/description>

Если говорить о конфиденциальности передаваемой информации через этот сервис, то стоит обратить внимание на документацию. В документации написано “Ключ авторизации это 2048-битный ключ, которым обмениваются девайс клиента и сервер, созданный непосредственно во время регистрации юзера на устройстве клиента, чтобы обмениваться ключами Диффи-Хеллмана, и никогда не передаваемый через сеть. Каждый ключ авторизации существует только для конкретного пользователя. Ничто не мешает юзеру иметь несколько ключей (которые согласовываются с «перманентными сессиями» на разных девайсах), некоторые из них могут быть заблокированы навсегда, если девайс утерян.” Данный алгоритм выработки используется в этом протоколе и должен обеспечивать конфиденциальность сообщений. Остается вопрос каким образом ключи согласовываются с «перманентными сессиями» на разных девайсах и зачем это сделано… Ведь что получается в итоге, Telegram накрутили множество алгоритмов, исключили возможность перехвата и подмены трафика, а про банальный пароль забыли. Злоумышленнику не нужно слушать трафик мессенджера, а всего лишь нужно перехватить смс и доступ получен без проблем.

## EL337

Eleet Private Messenger — мессенджер популярный среди пользователей мобильных устройств, ценящих свою приватность. Регистрация в приложении в отличие от Viber, WhatsApp, Skype, Telegram не требует подтверждения e-mail, привязки к номеру мобильного телефона или социальным аккаунтам. Это гарантирует защищенность персональных данных пользователей и тайну личной переписки. Для поиска собеседника вам нужно знать его ID или отсканировать QR код.

Все сообщения в приложении Eleet Private Messenger передаются в зашифрованном виде и не хранятся на сервере. В приложении применяются технологии шифрования от устройства к устройству, которые не оставляют следов на серверах и умеют самоликвидироваться в назначенное пользователем время. Теперь вы можете свободно общаться с людьми в любой точке земного шара, не опасаясь утечки информации. Все сообщения в приложении передаются в зашифрованном виде и не хранятся на сервере. Для защиты данных пользователей применяется протокол EL337, это усовершенствованный вариант протокола TextSecure. Во время каждого сеанса использования приложения у пользователя генерируется несколько пар ключей, один из которых открытый, другой закрытый. Открытые ключи можно пересылать другим пользователям, закрытый всегда остается на устройстве пользователя. При общении с собеседником благодаря специальному алгоритму создается общий ключ шифрования. Чтобы исключить подмену ключей, пользователь имеет возможность сравнить цифровой отпечаток ключа с тем, что ему передал собеседник.

Для такой безопасности используют асимметричную криптографию на эллиптических кривых Curve25519 и Ed25519, поточный шифр XSalsa20 аутентифицированный алгоритмом Poly1305 и хеш Blake2b, который быстрее SHA3 и при этом такой же безопасный. К тому же реализация не восприимчива к timing атакам, а значит, ключи нельзя узнать даже с помощью специальных средств.

# **Протоколы защиты звуковой связи**

## Протокол XMPP

**XMPP** — [открытый](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82), основанный на [XML](https://ru.wikipedia.org/wiki/XML), свободный для использования [протокол](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) для [мгновенного обмена сообщениями](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B_%D0%BC%D0%B3%D0%BD%D0%BE%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%BC%D0%B5%D0%BD%D0%B0_%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8) и информацией о присутствии в режиме, близком к режиму реального времени. Изначально спроектированный легко расширяемым, протокол, помимо передачи текстовых сообщений, поддерживает передачу [голоса](https://ru.wikipedia.org/wiki/VoIP), видео и [файлов](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB) по сети.

В отличие от коммерческих систем мгновенного обмена сообщениями, таких как [AIM](https://ru.wikipedia.org/wiki/AOL_Instant_Messenger), [ICQ](https://ru.wikipedia.org/wiki/ICQ), [WLM](https://ru.wikipedia.org/wiki/Windows_Live_Messenger) и [Yahoo](https://ru.wikipedia.org/wiki/Yahoo!_Messenger), XMPP является федеративной, расширяемой и открытой системой. Любой желающий может открыть свой сервер мгновенного обмена сообщениями, регистрировать на нём пользователей и взаимодействовать с другими серверами XMPP. На основе протокола XMPP уже открыто множество частных и корпоративных серверов XMPP. Среди них есть достаточно крупные проекты, такие как [Google Talk](https://ru.wikipedia.org/wiki/Google_Talk), [Одноклассники.ru](https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BD%D0%B8%D0%BA%D0%B8.ru), [LiveJournal](https://ru.wikipedia.org/wiki/LiveJournal), [Juick](https://ru.wikipedia.org/wiki/Juick) и др. Ранее протокол поддерживался также социальными сетями [Facebook](https://ru.wikipedia.org/wiki/Facebook), [ВКонтакте](https://ru.wikipedia.org/wiki/%D0%92%D0%9A%D0%BE%D0%BD%D1%82%D0%B0%D0%BA%D1%82%D0%B5) и сервисом [Яндекс.Почта](https://ru.wikipedia.org/wiki/%D0%AF%D0%BD%D0%B4%D0%B5%D0%BA%D1%81.%D0%9F%D0%BE%D1%87%D1%82%D0%B0)

Сообщения шифруются с помощью [криптосистемы с открытым ключом](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC), то есть существует пара ключей и некая процедура, которая может с помощью одного ключа преобразовать сообщение так, что обратное преобразование можно будет выполнить только с помощью другого ключа. Один из этих ключей называется закрытым и держится в тайне, другой же, напротив, называется открытым, и распространяется свободно. Таким образом, зашифровав сообщение открытым ключом можно быть уверенным, что его прочитает только владелец закрытого ключа, напротив, подписав что-то своим ключом автор даёт уверенность получателю сообщения в своём авторстве.

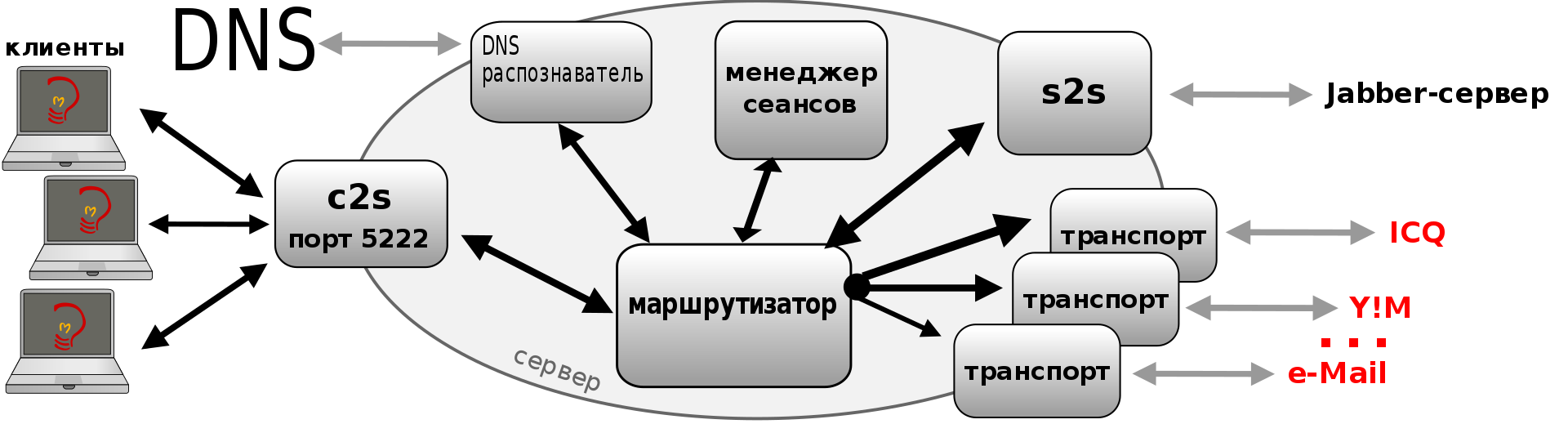


Схема устройства работы XMPP

**Преимущества**

* *Децентрализация*: Архитектура сети XMPP схожа с электронной почтой; кто угодно может запустить свой собственный XMPP-сервер и нет какого-либо центрального сервера.
* [*Открытый стандарт*](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82)*:* [Internet Engineering Task Force](https://ru.wikipedia.org/wiki/Internet_Engineering_Task_Force) формализовал XMPP как стандарт мгновенного обмена сообщениями и технологии присутствия под названием XMPP, и спецификации XMPP были опубликованы как [RFC 3920](https://tools.ietf.org/html/rfc3920) и [RFC 3921](https://tools.ietf.org/html/rfc3921). Никаких привилегий не требуется для добавления поддержки этих спецификаций и их разработка не привязана к какому-либо разработчику. Существует множество реализаций серверов и клиентов, а также библиотек с открытым исходным кодом.
* *Бесконечная расширяемость.* Любой может написать свое расширение и добавить его в протокол. Изначально предполагалось крупные компании будут создавать свои расширения протокола для интеграции собственных мессенджеров. Это бы позволило создать универсальный протокол общения между разными менеджерами. На практике это не произошло. Крупные интернет компании стали предпочитать собственные, закрытые протоколы
* *История*: Технологии XMPP используются с 1998 года. При поддержке таких крупных компаний, как [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) и Google, создано множество дополнений к стандартам XMPP для клиентов, серверов, компонент и библиотек кодов. В настоящее время XMPP существует практически без инвестиций, однако это не мешает дальнейшему развитию протокола сообществом и постоянно появляются новые расширения
* *Безопасность*: XMPP серверы могут быть изолированы от публичных сетей XMPP (например, во внутренней сети компании) и хорошо защищены (благодаря использованию [SASL](https://ru.wikipedia.org/wiki/Simple_Authentication_and_Security_Layer) и [TLS](https://ru.wikipedia.org/wiki/TLS)) встроенными в ядро XMPP спецификациями. Для поддержки использования шифрования канала XMPP Standards Foundation также использовал вспомогательный [certification authority](https://ru.wikipedia.org/w/index.php?title=Certificate_authority&action=edit&redlink=1) в xmpp.net, обеспечивая [цифровые сертификаты](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D1%81%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82) для администраторов XMPP серверов при содействии StartCom Certification Authority (который является основным хранителем сертификатов для всех вспомогательных). Многие реализации серверов используют [SSL](https://ru.wikipedia.org/wiki/SSL) при обмене между клиентом и сервером, и немало клиентов поддерживают шифрование с помощью [PGP](https://ru.wikipedia.org/wiki/PGP)/[GPG](https://ru.wikipedia.org/wiki/GPG) внутри протокола.
* *Гибкость*: Настраиваемая функциональность может быть надстроена поверх XMPP; для поддержки возможности взаимодействия различных сетей стандартные расширения поддерживаются XMPP Software Foundation. Приложения XMPP в дополнение к функциональности клиента сетевого общения включают в себя администрирование сети, распределение ресурсов, утилиты для совместной работы, обмен файлами, игры и мониторинг удалённых систем.

**Слабые стороны**

* *Избыточность передаваемой информации*: Более 70 % межсерверного трафика XMPP составляют сообщения о присутствии, около 60 % которых являются излишними.
* *Масштабируемость*: XMPP сейчас страдает от фактически той же проблемы избыточности, но применительно к [чат-комнатам](https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)) и возможностям [публикации информации](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D1%83%D0%B1%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8&action=edit&redlink=1). Решение этих проблем также ожидается в виде [XEP](https://ru.wikipedia.org/w/index.php?title=XEP&action=edit&redlink=1)-расширений. Пока они не введены, большие чат-комнаты образуют избыточный трафик.
* *Неэффективность передачи бинарных данных*: Так как XMPP является, по сути, одним длинным XML-документом, невозможно передать немодифицированную двоичную информацию.

**Jingle**

**Jingle** — это дополнение к протоколу [XMPP](https://ru.wikipedia.org/wiki/XMPP), позволяющее передавать между двумя клиентами аудио- и видеоданные. Он был разработан компанией [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)) и [XMPP Standards Foundation](https://ru.wikipedia.org/w/index.php?title=XMPP_Standards_Foundation&action=edit&redlink=1). Официальное название стандарта — **XEP-0166**. На начало июня 2010 года последняя версия протокола — 1.1 (от [23 декабря](https://ru.wikipedia.org/wiki/23_%D0%B4%D0%B5%D0%BA%D0%B0%D0%B1%D1%80%D1%8F) [2009 года](https://ru.wikipedia.org/wiki/2009_%D0%B3%D0%BE%D0%B4)).

Jingle - это расширение протокола Jabber, предназначенное для передачи бинарных данных. Подобно [Skype](http://jabberworld.info/Skype), Jingle является [пиринговой](http://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%80%D0%B0%D0%BD%D0%B3%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) (P2P) сетью, это значит, что данные передаются напрямую между клиентами. Прямая передача данных значительно снижает нагрузку на сервера Jabber.

У Jingle существует множество применений, таких как:

* Передача голоса в реальном времени (аудиозвонки или просто разговор голосом, так называемый [VoIP](http://ru.wikipedia.org/wiki/VoIP))
* Передача видео в реальном времени (видеозвонки)
* Передача файлов

## Протокол WebRTC

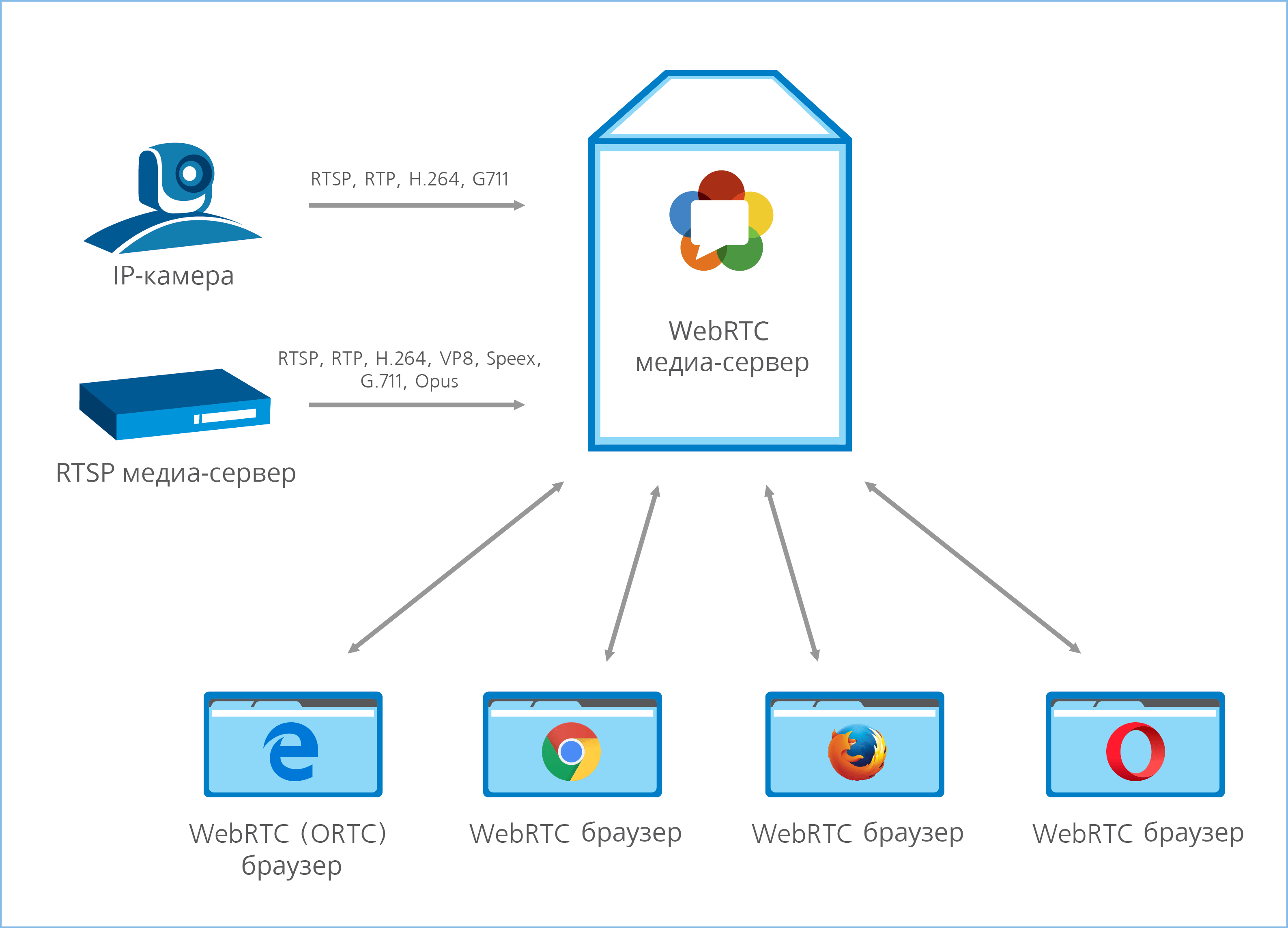
**WebRTC (Web Real Time Communications) — это стандарт, который описывает передачу потоковых аудиоданных, видеоданных и контента от браузера и к браузеру в режиме реального времени без установки плагинов или иных расширений. Стандарт позволяет превратить браузер в оконечный терминал видеоконференцсвязи, достаточно просто открыть веб-страницу, чтобы начать общение.Как работает WebRTC**

#### **На стороне клиента**

* Пользователь открывает страницу, содержащую HTML5 тег <video>.
* Браузер запрашивает доступ к веб-камере и микрофону пользователя.
* JavaScript код на странице пользователя контролирует параметры соединения (IP-адреса и порты сервера WebRTC или других WebRTC клиентов) для обхода NAT и Firewall.
* При получении информации о собеседнике или о потоке со смикшированной на сервере конференцией, браузер начинает согласование используемых аудио и видео кодеков.
* Начинается процесс кодирования и передача потоковых данных между WebRTC клиентами (в нашем случае, между браузером и сервером).

#### **На стороне WebRTC сервера**

Для обмена данными между двумя участниками видеосервер не требуется, но если нужно объединить в одной конференции несколько участников, сервер необходим.

**Схема работы WebRTC сервера**

Видеосервер будет получать медиа-трафик с различных источников, преобразовывать его и отправлять пользователям, которые в качестве терминала используют WebRTC.

Также WebRTC сервер будет получать медиа-трафик от WebRTC пиров и передавать его участникам конференции, которые используют приложения для настольных компьютеров или мобильных устройств, в случае наличия таковых.

*Преимущества стандарта*

* Не требуется установка ПО.
* Очень высокое качество связи, благодаря:
  + Использованию современных видео (VP8, H.264) и аудиокодеков (Opus).
  + Автоматическое подстраивание качества потока под условия соединения.
  + Встроенная система эхо- и шумоподавления.
  + Автоматическая регулировка уровня чувствительности микрофонов участников (АРУ).
* Высокий уровень безопасности: все соединения защищены и зашифрованы согласно протоколам TLS и SRTP.
* Есть встроенный механизм захвата контента, например, рабочего стола.
* Возможность реализации любого интерфейса управления на основе HTML5 и JavaScript.
* Возможность интеграции интерфейса с любыми back-end системами с помощью WebSockets.
* Проект с открытым исходным кодом — можно внедрить в свой продукт или сервис.
* Настоящая кросс-платформенность: одно и то же WebRTC приложение будет одинаково хорошо работать на любой операционной системе, десктопной или мобильной, при условии, что браузер поддерживает WebRTC. Это значительно экономит ресурсы на разработку ПО.

*Недостатки стандарта*

* Для организации групповых аудио и видеоконференций требуется сервер ВКС, который бы микшировал видео и звук от участников, т.к. браузер не умеет синхронизировать несколько входящих потоков между собой.
* Все WebRTC решения несовместимы между собой, т.к. стандарт описывает лишь способы передачи видео и звука, оставляя реализацию способов адресации абонентов, отслеживания их доступности, обмена сообщениями и файлами, планирования и прочего за вендором.
* Другими словами вы не сможете позвонить из WebRTC приложения одного разработчика в WebRTC приложение другого разработчика.
* Микширование групповых конференций требует больших вычислительных ресурсов, поэтому такой тип видеосвязи требует покупки платной подписки либо инвестирования в свою инфраструктуру, где на каждую конференцию требуется 1 физическое ядро современного процессора.

## Протокол RTP

Протокол **RTP** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Real-time Transport Protocol*) работает на [уровне приложений (OSI - 7)](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F) и используется при передаче трафика реального времени. Протокол был разработан Audio-Video Transport Working Group в [IETF](https://ru.wikipedia.org/wiki/IETF) и впервые опубликован в 1996 году как [RFC 1889](https://tools.ietf.org/html/rfc1889) ([RFC 1889](https://tools.ietf.org/html/rfc1889) является устаревшим с момента выхода [RFC 3550](https://tools.ietf.org/html/rfc3550) в 2003 году).

Протокол RTP переносит в своём заголовке данные, необходимые для восстановления аудиоданных или видеоизображения в приёмном узле, а также данные о типе кодирования информации ([JPEG](https://ru.wikipedia.org/wiki/JPEG), [MPEG](https://ru.wikipedia.org/wiki/MPEG) и т. п.). В заголовке данного протокола, в частности, передаются временная метка и номер пакета. Эти параметры позволяют при минимальных задержках определить порядок и момент декодирования каждого пакета, а также интерполировать потерянные пакеты.

Аутентификация и подтверждение целостности получаемых RTP-сообщений реализуется с помощью HMAC-SHA-1. Однако из-за недостатков HMAC-SHA-1 экспертами рекомендуется использование SHA-256, несмотря на нестандартизованность такого решения. Рекомендуется также использовать шифрование полезных данных RTP с последующим расчетом цифрового отпечатка (fingerprint) зашифрованных данных.

RTP не имеет стандартного зарезервированного номера порта. Единственное ограничение состоит в том, что соединение проходит с использованием чётного номера, а следующий нечётный номер используется для связи по протоколу [RTCP](https://ru.wikipedia.org/wiki/RTCP). Тот факт, что RTP использует динамически назначаемые адреса портов, создаёт ему трудности для прохождения [межсетевых экранов](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%8D%D0%BA%D1%80%D0%B0%D0%BD), для обхода этой проблемы, как правило, используется [STUN](https://ru.wikipedia.org/wiki/STUN)-сервер.

Установление и разрыв соединения не входит в список возможностей RTP, такие действия выполняются [сигнальным протоколом](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) (например, [RTSP](https://ru.wikipedia.org/wiki/RTSP) или [SIP](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D1%83%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D1%81%D0%B5%D0%B0%D0%BD%D1%81%D0%B0) протоколом).

Приложения, формирующие потоки [реального времени](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D0%B8%D0%B0), требуют своевременной доставки информации и для достижения этой цели могут допустить некоторую потерю пакетов. Например, потеря пакета в аудио-приложении может привести к доле секунды тишины, которая может быть незаметна при использовании подходящих алгоритмов скрытия ошибок. Протокол [TCP](https://ru.wikipedia.org/wiki/TCP), хотя и стандартизирован для передачи RTP, как правило не используется в RTP-приложениях, так как надежность передачи в TCP формирует временные задержки. Вместо этого, большинство реализаций RTP базируется на [UDP](https://ru.wikipedia.org/wiki/UDP). Кроме этого, существуют другие спецификации для транспортных протоколов [SCTP](https://ru.wikipedia.org/wiki/SCTP) и [DCCP](https://ru.wikipedia.org/wiki/DCCP), но они мало распространены.

**Спецификация RTP описывает два подпротокола:**

* Протокол передачи данных, RTP, который взаимодействует с передачей данных реального времени. Информация, предоставляемая посредством этого протокола, включает в себя отметку времени (для синхронизации), последовательный номер (для детектирования потери и дублирования пакетов) и формат полезной нагрузки, который определяет формат кодирования данных.
* Протокол контроля, [RTCP](https://ru.wikipedia.org/wiki/RTCP), используемый для определения качества обслуживания ([QOS](https://ru.wikipedia.org/wiki/QOS)), обратной связи и синхронизации между медиа-потоками. Занимаемая полоса пропускания RTCP мала в сравнении с RTP, обычно около 5 %.
* Управляющий сигнальный протокол, такой как [SIP](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D1%83%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D1%81%D0%B5%D0%B0%D0%BD%D1%81%D0%B0), [H.323](https://ru.wikipedia.org/wiki/H.323), [MGCP](https://ru.wikipedia.org/wiki/MGCP) или [H.248](https://ru.wikipedia.org/wiki/H.248). Сигнальные протоколы управляют открытием, модификацией и закрытием RTP-сессий между устройствами и приложениями реального времени.
* Управляющий протокол описания медиа, такой как [Session Description Protocol](https://ru.wikipedia.org/wiki/Session_Description_Protocol).

**Установление сессии**

RTP-сессия устанавливается для каждого потока мультимедиа. Сессия состоит из [IP-адреса](https://ru.wikipedia.org/wiki/IP-%D0%B0%D0%B4%D1%80%D0%B5%D1%81) и пары портов для RTP и RTCP. Например, аудио и видео потоки будут иметь различные RTP-сессии, позволяющие приемнику для этого выделить конкретный поток. Порты, которые образуют сессию, связываются друг с другом средствами других протоколов, таких как SIP (содержащий в своих сообщениях протокол SDP) и [RTSP](https://ru.wikipedia.org/wiki/RTSP) (используя SDP в методе Setup). В соответствии со спецификацией, RTP не имеет стандартного зарезервированного номера порта. Единственное ограничение состоит в том, что соединение проходит с использованием чётного номера, а следующий нечётный номер используется для связи по протоколу [RTCP](https://ru.wikipedia.org/wiki/RTCP). RTP и RTCP обычно используют непривилегированные UDP-порты (16k-32k), но могут использовать и другие порты, поскольку сам протокол RTP независим от транспортного уровня.

**Структура пакета**



0-1 — **Ver**. (2 бита) указывает версию протокола. Текущая версия — 2.   
2 — **P** (один бит) используется в случаях, когда RTP-пакет дополняется пустыми байтами на конце.   
3 — **X** (один бит) используется для указания расширений протокола, задействованных в пакете.   
4-7 — **CC** (4 бита) содержит количество CSRC-идентификаторов, следующих за постоянным заголовком.   
8 — **M** (один бит) используется на уровне приложения и определяется профилем. Если это поле установлено, то данные пакета имеют какое-то особое значение для приложения.   
9-15 — **PT** (7 бит) указывает формат полезной нагрузки и определяет её интерпретацию приложением.   
64-95 — **SSRC** указывает источник синхронизации.   
**EHL** (Extension Header Length) — количество 32-битных слов в блоке данных расширения заголовка.   
**L** — последний байт в пакете, определяющий длину области заполнения в байтах (используется для выравнивания в последнем пакете).

## Протокол SIP

**SIP** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Session Initiation Protocol* — протокол установления сеанса) — [протокол передачи данных](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), описывающий способ установки и завершения пользовательского интернет-сеанса, включающего обмен [мультимедийным](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D0%B8%D0%B0) содержимым ([IP-телефония](https://ru.wikipedia.org/wiki/IP-%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD%D0%B8%D1%8F), [видео](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE%D0%BA%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D1%8F) и [аудиоконференции](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D0%B4%D0%B8%D0%BE%D0%BA%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D1%8F), [мгновенные сообщения](https://ru.wikipedia.org/wiki/%D0%9C%D0%B3%D0%BD%D0%BE%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F), [онлайн-игры](https://ru.wikipedia.org/wiki/%D0%9E%D0%BD%D0%BB%D0%B0%D0%B9%D0%BD-%D0%B8%D0%B3%D1%80%D1%8B)).

Протокол описывает, каким образом клиентское приложение может запросить начало соединения у другого, возможно, физически удалённого клиента, находящегося в той же сети, используя его уникальное имя. Протокол определяет способ согласования между клиентами об открытии каналов обмена на основе других протоколов, которые могут использоваться для непосредственной передачи информации (например, [RTP](https://ru.wikipedia.org/wiki/Real-time_Transport_Protocol)). Допускается добавление или удаление таких каналов в течение установленного сеанса, а также подключение и отключение дополнительных клиентов (то есть допускается участие в обмене более двух сторон — конференц-связь). Протокол также определяет порядок завершения сеанса.

Шифрование в SIP базируется на передаче «общего секрета», который состоит из контрольной суммы поверх nonce (уникальной комбинации для каждой сессии) и параметров (пользовательского имени, пароля, nonce, SIP-метода, Request URI). SIP не осуществляет обмен паролями. Общий секрет хэшируется с использованием MD5 или SHA-1 (IETF рекомендовано использование SHA-1).

Разработкой занималась организация [IETF](https://ru.wikipedia.org/wiki/IETF) MMUSIC Working Group. Протокол начал разрабатываться в 1996 году Хенингом Шулзри (Henning Schulzrinne, [Колумбийский университет](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D1%83%D0%BC%D0%B1%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82)) и Марком Хэндли ([Университетский колледж Лондона](https://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BB%D0%BB%D0%B5%D0%B4%D0%B6_%D0%9B%D0%BE%D0%BD%D0%B4%D0%BE%D0%BD%D0%B0)). В ноябре 2000 года SIP был утверждён как [сигнальный протокол](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) проекта [3GPP](https://ru.wikipedia.org/wiki/3GPP) и основной протокол архитектуры [IMS](https://ru.wikipedia.org/wiki/IMS_(%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D1%81%D0%B2%D1%8F%D0%B7%D1%8C)) (модификация 3GPP TS.24.22). SIP — один из протоколов, лежащих в основе [Voice over IP](https://ru.wikipedia.org/wiki/Voice_over_IP).

В основу протокола рабочая группа MMUSIC заложила следующие принципы:

* *Простота*: включает в себя только шесть методов (функций)
* *Независимость* от транспортного уровня, может использовать [UDP](https://ru.wikipedia.org/wiki/UDP), [TCP](https://ru.wikipedia.org/wiki/TCP), [ATM](https://ru.wikipedia.org/wiki/ATM) и т. д.
* *Персональная мобильность пользователей*. Пользователи могут перемещаться в пределах сети без ограничений. Это достигается путём присвоения пользователю уникального идентификатора. При этом набор предоставляемых услуг остается неизменным. О своих перемещениях пользователь сообщает с помощью сообщения **REGISTER** своему серверу.
* *Масштабируемость сети*. Структура сети на базе протокола SIP позволяет легко её расширять и увеличивать число элементов.
* *Расширяемость протокола*. Протокол характеризуется возможностью дополнять его новыми функциями при появлении новых услуг.
* *Интеграция в стек существующих протоколов*[*Интернет*](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Протокол SIP является частью глобальной архитектуры мультимедиа, разработанной комитетом [IETF](https://ru.wikipedia.org/wiki/IETF). Кроме SIP, эта архитектура включает в себя протоколы [RSVP](https://ru.wikipedia.org/wiki/RSVP_(%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB)), [RTP](https://ru.wikipedia.org/wiki/Real-time_Transport_Protocol), [RTSP](https://ru.wikipedia.org/wiki/RTSP), [SDP](https://ru.wikipedia.org/wiki/SDP).
* *Взаимодействие с другими протоколами сигнализации*. Протокол SIP может быть использован совместно с другими протоколами [IP-телефонии](https://ru.wikipedia.org/wiki/VoIP), протоколами [ТфОП](https://ru.wikipedia.org/wiki/%D0%A2%D1%84%D0%9E%D0%9F), и для связи с интеллектуальными сетям.

# **Протоколы защиты видеосвязи**

## Протокол Skype

Протокол Skype – это закрытый протокол интернет–телефонии, используемый продуктом Skype. Спецификации данного протокола никогда не были опубликованы Skype, а официальные приложения, использующие протокол, являются закрытыми.

Сеть Skype не совместима с большинством других сетей VoIP без надлежащего лицензирования от Skype (например, комерческое решение от Skype: SkyStone – выделенный шлюз на несколько каналов). Были предприняты многочисленные попытки изучить и/или перестроить протокол, чтобы выявить принцип его работы, а также исследовать на безопасность.

Видео зашифровано с помощью поточного шифра RC4. Голосовые данные шифруются с помощью AES.

Skype был первой одноранговой IP – телефонной сетью. Раньше, любой клиент имел хорошую пропускную способность видеосвязи за счет технологии p2p. В 2012 году Microsoft изменила дизайн сети и передала все суперноды под свой контроль в качестве размещенных серверов в центрах обработки данных. Microsoft, в то время, защищал свой принцип, считая, что такой подход имеет не менее быструю пропускную способность, масштабируемость и преимущества доступности для сотен миллионов пользователей, которые составляют сообщество продукта Skype". В то время среди узкого круга исследователей была некоторая озабоченность относительно последствий изменения конфиденциальности из–за централизации супернодов. Позже это было доказано публикацией в открытых источниках деятельности программы наблюдения PRISM в июне 2013.

## Протокол Tox

Tox – это одноранговый протокол прикладного уровня, созданный для обмена мгновенными сообщениями и видеосвязи, который обеспечивает сквозное шифрование. Заявленная цель проекта – обеспечить безопасное, но легкодоступное общение для всех. Эталонная реализация протокола публикуется как свободное программное обеспечение с открытым исходным кодом.

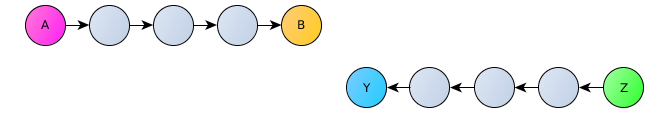
Взаимодействие участников Tox реализовано с использованием пирингового обмена информацией для улучшения пропускной способности, но в отличие от Skype, не требует регистрации для использования, а идентификатор пользователя создаётся локально. После установки Tox автоматически создаётся пара ключей. Публичный ключ можно передавать кому угодно – он служит как уникальный идентификатор для поиска собеседника. Секретный ключ хранится только у владельца и подтверждает его подлинность, не раскрывая персональные данные. Центральный сервер отсутствует, поиск собеседников происходит через DHT сеть.

DHT – это класс децентрализованных распределённых систем поисковой службы, работающей подобно хеш–таблице. Как структура данных, хеш–таблица может представлять собой ассоциативный массив, содержащий пары (ключ–значение). Также, с термином DHT связан ряд принципов и алгоритмов, позволяющих записывать данные, распределяя информацию среди некоторого набора узлов–хранителей, и восстанавливать их путём распределённого поиска по ключу. Особенностью распределённой таблицы является возможность распределить информацию среди некоторого набора узлов–хранителей таким образом, что каждый участвующий узел смог бы найти значение, ассоциированное с данным ключом. Ответственность за поддержание связи между именем и значением распределяется между узлами, в силу чего изменение набора участников является причиной минимального количества разрывов. Это позволяет легко масштабировать DHT, а также постоянно отслеживать добавление и удаление узлов и ошибки в их работе.

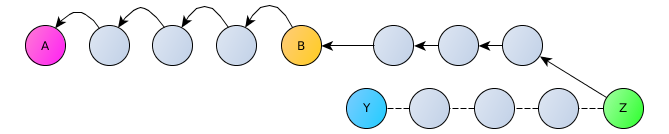
В отличии от DHT–нод, вся информация о которых известна или может быть получена любым клиентом DHT–сети Tox, клиентские приложения скрыты от стороннего наблюдателя – простого знания ToxID контакта (содержащего его публичный ключ) недостаточно для того, чтобы установить местонахождение этого контакта. Для соединения двух приложений Tox используется механизм луковой маршрутизации («onion routing»).

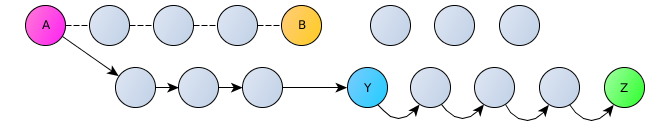
Установка связи

Механизм установления связи между двумя клиентами можно представить следующим образом. Два клиента (A и Z) анонсируют свой публичный ключ на ближайших (для своего публичного ключа) нодах через три случайные промежуточные DHT–ноды, каждая из которых знает только своих соседей по маршрутизации пакета, но не может прочитать содержимое пакета.

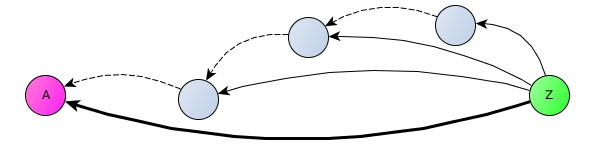


Второй клиент (Z), желающий соединиться с первым (A), так же через три случайные DHT–ноды делает запрос на установку соединения на ближайшую к искомому ключу (A) ноду, которая знает об анонсе первого клиента и маршруте, по которому требуется передать запрос от второго клиента.

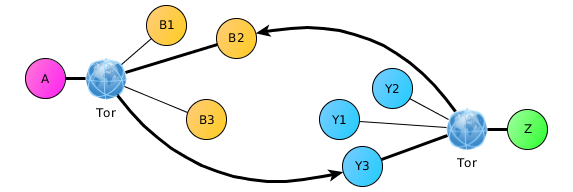


Первый клиент (A) в случае принятия запроса на установку соединения, проделывает обратную операцию к ближайшей DHT–ноде второго клиента (Z).  


После обмена информацией о местоположении друг–друга и временных ключах клиенты могут соединиться напрямую.



Если клиенты не желают делиться информацией о своем местоположении даже со своими контактами, они могут использовать ноды поддерживающие TCP–relay через прокси типа Tor.



Tox использует криптографические примитивы, присутствующие в криптобиблиотеке NaCl. В частности, Tox использует Curve25519 для обмена ключами (ECDH), алгоритм xsalsa20 для поточного шифрования и алгоритм Poly1305 для подсчета MAC (massage access code – режима имитовставки).

Данный протокол не проходил аудит на безопасность специалистами по защите информации. Ведутся его активные разработки, например, на безопасном языке Rust

## Протокол SRTP

Secure Real–time Transport Protocol – Безопасный протокол передачи данных в реальном времени (или SRTP) определяет профиль RTP (транспортный протокол в реальном времени) и предназначен для шифрования, установления подлинности сообщения, целостности, защиту от замены данных RTP в однонаправленных и multicast передачах медиа и приложениях. Он был разработан небольшой командой криптоэкспертов IP протоколов компаний Cisco и Ericsson в составе David Oran, David McGrew, Mark Baugher, Mats Naslund, Elisabetta Carrara, Karl Norman, и Rolf Blom. Был впервые опубликован в IETF в марте 2004 как RFC 3711.

Для шифрования медиа потока (в целях конфиденциальности соединения), SRTP стандартизирует использование только единственного шифра, AES, который может использоваться в двух режимах, превращающих изначально блочный шифр AES в потоковый шифр:

1. Сегментированный целочисленный счётчик – типичный режим, который осуществляет произвольный доступ к любым блокам, что является существенным для трафика RTP, передающегося в публичных сетях с непредсказуемым уровнем надежности и возможной потерей пакетов. В общем случае почти любая функция может использоваться в роли «счётчика», предполагая, что эта функция не повторяется для большого числа итераций. Но стандарт для шифрования данных RTP – только обычное целочисленное значение счётчика. AES, работающий в этом режиме, является алгоритмом шифрования по умолчанию, с длиной шифровального ключа по умолчанию в 128 бит.
2. f8–режим – вариант режима способа обратной связи, расширенного, чтобы быть доступным с изменённой функцией инициализации. Значения по умолчанию для шифровального ключа и ключа сессии – то же, что и в AES в режиме, описаном выше. (AES, работающий в этом режиме, был выбран для использования в мобильных сетях передачи 3G UMTS.)

Хотя технически в SRTP можно легко встраивать новые алгоритмы шифрования, стандарт SRTP заявляет, что новые алгоритмы шифрования, помимо описанных, не могут просто быть добавлены в реализацию протокола SRTP. Единственный юридически легальный способ добавить новый алгоритм шифрования для совместимости со стандартом SRTP, состоит в том, чтобы опубликовать новый стандарт RFC, где должно быть ясно определено использование нового алгоритма.

Чтобы подтвердить подлинность сообщения и защитить его целостность, алгоритм хеширования HMAC–SHA1, определенный в RFC 2104, используется для получения 160–битового хэша, который затем урезается до 80 или до 32 битов, чтобы стать опознавательным признаком, включаемым в пакет. HMAC вычисляется по типу payload пакета и данных в заголовке пакета, включая номер последовательности пакета. Чтобы защитить против встраивания сообщений Человека\_посередине, приёмник поддерживает индексы ранее полученных пакетов, сравнивает их с индексом каждого нового полученного пакета и пропускает новый пакет, только если он не проигрался (то есть был послан), прежде. Такой подход в большой степени полагается на полную защиту целостности (чтобы лишить возможности изменить индексы последовательности пакетов для обмана).

SRTP полагается на внешний протокол обмена ключами, чтобы установить главный начальный ключ. Разработан специальный протокол для защиты видеосвязи с использованием SRTP – ZRTP.

## Протокол ZRTP

ZRTP – криптографический протокол согласования ключей шифрования, используемый в системах передачи голоса/видео по IP–сетям (VoIP). ZRTP описывает метод получения ключей по алгоритму Диффи – Хелмана для организации Secure Real–time Transport Protocol (SRTP). ZRTP осуществляет согласование ключей в том же потоке RTP, по которому установлена аудио/видео связь, то есть не требует отдельного канала связи. Разработан Филипом Циммерманом (Phil Zimmermann, автор Pretty Good Privacy), Джоном Калласом (Jon Callas) и Аланом Джонстоном (Alan Johnston) в 2006 году.

Протокол RTP (англ. Real–time Transport Protocol) работает на прикладном уровне (OSI – 7) и используется при передаче трафика реального времени. Протокол был разработан Audio – Video Transport Working Group в IETF и впервые опубликован в 1996 году как RFC 1889 (RFC 1889 является устаревшим с момента выхода RFC 3550 в 2003 году).

Протокол RTP переносит в своём заголовке данные, необходимые для восстановления аудиоданных или видеоизображения в приёмном узле, а также данные о типе кодирования информации (JPEG, MPEG и т. п.). В заголовке данного протокола, в частности, передаются временная метка и номер пакета. Эти параметры позволяют при минимальных задержках определить порядок и момент декодирования каждого пакета, а также интерполировать потерянные пакеты.

SAS

Сам по себе алгоритм обмена ключами Диффи – Хелмана не может обеспечить защиту от присутствия человека посередине (man in the middle). Для аутентификации ZRTP использует Short Authentication String (SAS) "короткую строку аутентификации" (4 старших байта хеша), являющуюся сокращённым представлением криптографического хеша полученных ключей Диффи – Хелмана. Значения SAS вычисляются на каждой стороне соединения, абоненты передают их друг другу голосом для сверки. Если значения не совпали, то с большой уверенностью можно предположить присутствие Человека\_посередине. Использование алгоритма Диффи – Хелмана даёт потенциальному Человеку\_посередине всего одну попытку сгенерировать правильную SAS при попытке атаки.

Непрерывность ключевого материала

Вторым уровнем защиты ZRTP от атак "человек посередине" является непрерывность ключевого материала. Хеш ключевой информации предыдущего вызова подмешивается к параметрам алгоритма Диффи – Хеллмана при следующем вызове (между этими же абонентами), что придаёт протоколу ZRTP сходство с SSH. Если "человека посередине" не было при первом вызове, он исключается из всех последующих.

Данный протокол является стандартизованным, а значит он проходил аудит безопасности работы.