Информационная безопасность цифрового пространства: коллективная монография /под ред. Е.В. Стельмашонок, И.Н. Васильевой. – СПб.: Изд-во СПбГЭУ, 2019. (155 с.) – С.20-36.

1.4. Безопасность мессенджеров

Широкое распространение и рост функциональности мобильных устройств (планшетных ПК, смартфонов), развитие мобильных операционных систем и разнообразие предоставляемых ими средств коммуникации в условиях относительной ограниченности аппаратных ресурсов остро ставит вопрос безопасности. Ниже будут более подробно рассмотрены проблемы и методы обеспечения безопасности мессенджеров, тесно связанные с общими вопросами обеспечения информационной безопасности мобильных платформ.

Под мессенджером (Instant Messenger, IM) будем понимать систему для обмена сообщениями в режиме реального времени через Интернет, голосовой и видеосвязи, обмена файлами, организации групповых видеоконференций (групповых чатов), и соответствующее программное обеспечение (программы, мобильные приложения или веб-сервисы). Наибольшую известность в настоящее время получили мессенджеры WhatsApp, Viber, Skype, Telegram, ICQ, Facebook Messenger, Hangouts (являющийся стандартным сервисом Google) и др. В России отдают предпочтение первым четырем продуктам из этого списка, кроме того личная переписка ведется в рамках социальных сетей, таких как «ВКонтакте» и Facebook, востребованных также и пользователями десктопных систем через веб-сервисы. При этом в среднем каждый абонент пользуется 3 различными системами обмена сообщений [1].

Общий рост внимания к проблемам безопасности, наряду с обострением конкуренции среди мессенджеров, происходящее на фоне скандалов, связанных с продажей пользовательских данных компанией Facebook и блокированием мессенджера Telegram, использованное разработчиком последнего для продвижения идеи о защищенности своей системы, заставляет разработчиков внедрять дополнительные средства защиты, такие как end-to-end шифрование передаваемых данных.

Наряду с мессенджерами, которые изначально разрабатывались и позиционировались как защищенные (Signal Private Messenger, Threema, Wire, Wickr Me, Dust), в 2016 году включено полное шифрование сообщений, голосовых звонков и пересылаемых файлов в мессенджерах WhatsApp и Viber, в ICQ стали шифроваться не только текстовые сообщения, но аудио и видео, а в Facebook Messenger появились «секретные» чаты. В 2018 году систему оконечного шифрования чатов и звонков в рамках «личной беседы» ввел Skype, а «ВКонтакте» для iOS и Android запустил шифрование голосовых и видеозвонков.

«Секретные» чаты поддерживаются и в Telegram, однако в обычном режиме облачных чатов, используемом по умолчанию, данные «видны» серверам Telegram в открытом виде. При этом PR-компания мессенджера строилась на заверениях его владельцев, что никакие спецслужбы не смогут прочесть личные сообщения, и проводилась для англоязычной аудитории под лозунгом «Taking back our right to privacy» («вернем себе право на приватность»). По заявлениям Павла Дурова, «с самого первого дня мы не передали правительствам и третьим лицам ни байта приватных данных». Таким образом, защищенность мессенджера Telegram обеспечивается скорее не техническими мерами, а политикой компании. Однако в августе 2018 года Telegram был вынужден внести изменения в политику конфиденциальности мессенджера в рамках нового европейского закона о защите персональных данных GDPR (General Data Protection Regulation), добавив пункт о возможности раскрытия IP-адреса и номера телефона абонента на основании судебного решения, подтверждающего, что абонента подозревают в терроризме.

Это отражает еще одну тенденцию в обеспечении безопасности мессенджеров, обусловленную усилением регулирования со стороны государства, стремящегося, по возможности, деанонимизировать коммуникации на случай необходимости их контроля. В этом ряду стоит и появившееся в августе 2018 года сообщение об отказе WhatsApp от используемой в настоящее время технологии оконечного шифрования. Новая версия приложения, реализующая ослабленное шифрование в соответствии с требованиями властей США, выйдет в 2019 году.

Мессенджеры используют собственные протоколы передачи данных, при этом шифрование сообщений обычно выполняется на прикладном уровне, а затем зашифрованные данные встраиваются в транспортный протокол. В качестве транспортного протокола, как правило (но необязательно), используется SSL/TLS, ставший де-факто стандартом для всех защищенных web-коммуникаций. Вместе с тем, реализация последнего на мобильных платформах бывает далека от совершенства [2, 3].

Для защиты передаваемых данных мессенджеры, как правило, используют end-to-end (E2E, оконечное или сквозное) шифрование. Это значит, что криптографические ключи генерируются и хранятся на конечном устройстве (устройстве пользователя), а не на серверах системы мгновенных сообщений. Поэтому никто, кроме адресата, даже сервер системы, не сможет прочитать содержимое зашифрованных сообщений. С другой стороны, переписка будет не доступна и самому абоненту, если он перейдет на другое устройство. Поэтому

синхронизация устройств или восстановление в случае потери устройства (то есть получение доступа к архиву переписки) совместно с использованием оконечного шифрования невозможно без депонирования личного ключа вне устройства.

Поскольку оконечное шифрование реализуется на верхних уровнях сетевой архитектуры, адресные данные должны быть доступны в незашифрованном виде в промежуточных узлах (то есть, на серверах системы). А это значит, что метаданные коммуникаций пользователей (кто кому звонил или писал, и когда) остаются открытыми.

Многие мессенджеры в настоящее время реализуют оконечное шифрование на основе протокола Signal, разработанного некоммерческой организацией Open Whisper Systems (OWS) для одноименного мессенджера. Протокол тщательно документирован [4] и имеет реализующие его библиотеки с открытым исходным кодом на языках Java, C и JavaScript. На сегодняшний день протокол Signal используется мессенджерами WhatsApp, Google Allo, Facebook Messenger и Skype. Реализация шифрования Viber использует концепции протокола Signal, но другие криптографические алгоритмы. По заявлениям разработчиков Wire, используемый в этом мессенджере протокол шифрования Proteus также основан на протоколе Signal.

Протокол использует криптографию с открытым ключом на эллиптической кривой Curve25519 или Curve448 для цифровых подписей, согласование ключей на основе модификации протокола Диффи-Хэллмана, шифрование сообщений с помощью симметричного алгоритма AES-256 в режиме CBC и проверку целостности с помощью кода аутентификации HMAC-SHA256. Кроме того, для изменения ключей шифрования в течение сеанса связи используется функция диверсификации ключа (KDF, Key Derivation Function) на основе HMAC-SHA256 и HMAC-SHA512 (HKDF). Все используемые протоколом криптографические примитивы хорошо известны и рекомендованы для применения в системах защиты информации. Безопасность протокола проанализирована в [5].

Основные особенности протокола Signal обусловлены тем, что вторая сторона коммуникаций может быть недоступна (находиться в автономном режиме, офлайн) в момент отправки сообщения. Поэтому стандартные протоколы аутентификации и обмена ключами (AKE, Authenticated Key Exchange) не могут быть непосредственно применены. Например, в классическом протоколе Диффи-Хэллмана (DH) в формировании общего секретного ключа принимают участие обе стороны, поскольку значение ключа вычисляется на основе секретных

ключей обоих абонентов. При этом абоненты открыто обмениваются значениями, которые можно трактовать как открытые ключи криптосистемы DH.

Для решения проблемы автономности одной из сторон Signal реализует асинхронный протокол передачи X3DH, требуя предварительной отправки на промежуточный сервер партии предварительно вычисленных значений (открытых ключей). Такая отправка производится во время регистрации или позже (рис. 1.2). Когда абонент желает отправить сообщение, он получает необходимые для выполнения AKE-подобного протокола значения получателя с промежуточного сервера (который действует только как буфер) и вычисляет ключ шифрования сообщения.

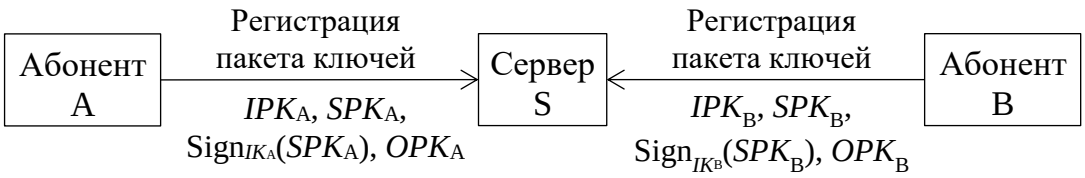

Рисунок 1.2. Первый этап протокола Signal – регистрация ключевой информации пользователей

На этапе регистрации до начала обмена сообщениями каждый абонент формирует 3 типа ключей асимметричной схемы: долговременный ключ *IK* (для подписи), среднесрочный предварительный ключ *SK* и набор одноразовых предварительных ключей *OK*. При этом на сервер отправляются только соответствующие им открытые ключи *IPK*, *SPK*, *OPK* и подпись ($\mathrm{Sign}_{IK}(SPK)$ – среднесрочный открытый ключ *SPK* подписывается долговременным ключом *IK* абонента).

Абонент может периодически (например, раз в неделю или раз в месяц) обновлять свой среднесрочный открытый ключ *SPK* с подписью, а также в любое время загружать новый набор предварительных одноразовых открытых ключей *OPK* (например, когда сервер информирует Боба о том, что их запас на сервере снизился). Долговременный ключ *IPK* является идентификационным и регистрируется абонентом однократно.

Пусть абонент A хочет начать сеанс связи с абонентом B (рис. 1.3). Абонент A запрашивает у сервера и получает пакет ключей абонента B, а затем вычисляет секретный ключ *SK* из нескольких значений протокола DH, полученных на основе значений идентификационных ключей обоих

абонентов IPK_A, IPK_B, среднесрочного открытого ключа получателя SPK_B и краткосрочного открытого ключа EPK_A из вновь сгенерированной отправителем пары ключей асимметричного шифрования.

$DH_1 = DH(IK_A, SPK_B)$

$DH_2 = DH(EPK_A, IK_B)$

$DH_3 = DH(EPK_A, SPK_B)$

$DH_4 = DH(EPK_A, OPK_B)$

$SK = \mathrm{KDF}(DH_1 \parallel DH_2 \parallel DH_3 \parallel DH_4)$

Если на сервере исчерпан запас одноразовых предварительных ключей OPK_B, то значение DH_4 опускается.

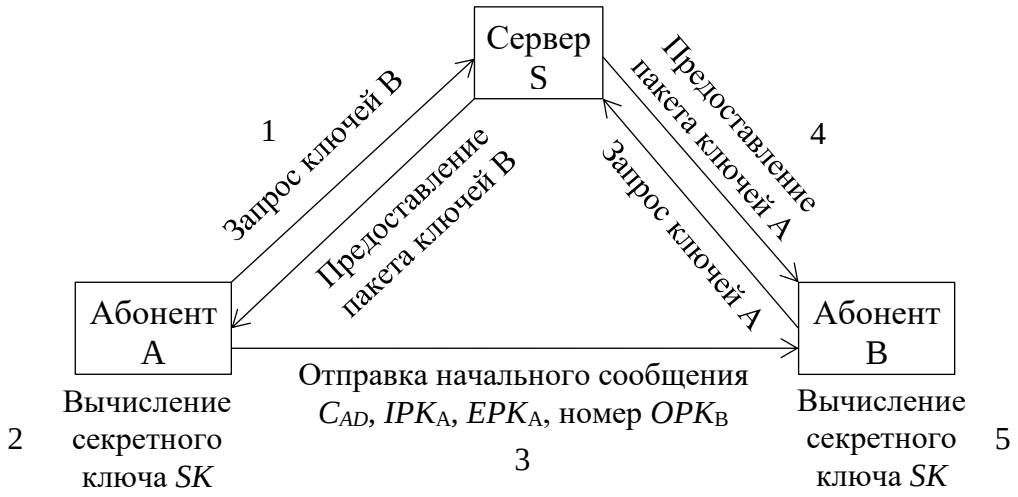

Рисунок 1.3. Второй этап протокола Signal – установка сеанса связи

SK служит основой для формирования цепочки ключей шифрования сообщений. Когда ключ шифрования создан, абонент А шифрует симметричным алгоритмом свое идентификационное сообщение *AD*, составленное из значений идентификационных ключей отправителя и получателя, а также любой дополнительной идентификационной информации – имен абонентов, сертификатов и т.п.

Затем А отправляет получателю В криптограмму C_{AD}, свой идентификационный ключ IPK_A, краткосрочный открытый ключ EPK_A, и информацию о том, какой из предварительных ключей OPK_B был использован для получения ключа шифрования сообщения.

Получив всю эту информацию, абонент В сможет сформировать ключ *SK* для расшифрования сообщения. Если сообщение расшифровано удачно и идентификационная информация корректна, В продолжает использовать цепочку ключей, полученных из *SK*, для шифрования своего сообщения.

Использованные одноразовые ключи OPK_B удаляются сервером, а соответствующие им OK_B – абонентом B.

Следует отметить, что если взаимодействие абонентов с сервером S и друг с другом происходит в недоверенной среде, то возможна реализация угрозы «человек посередине» (MITM, the man in the middle), позволяющей нарушителю подменять ключи при передаче и выдавать себя за любую из сторон коммуникации. Как правило, данная проблема решается с помощью цифровой сертификации и развертывания инфраструктуры открытых ключей PKI, однако протоколом Signal они не описываются, а промежуточный сервер не играет роли центра сертификации.

Защиту от реализации атак «человек посередине» призвано обеспечить использование на нижнем уровне защищенного транспортного протокола (SSL/TLS), однако на практике его реализации в мобильных приложениях зачастую уязвимы для этого типа атак [2, 3]. Кроме того, использование SSL/TLS сохраняет возможность реализации MITM атак на самом сервере (например, в случае контроля со стороны спецслужб и администраторов системы).

Для подтверждения подлинности второй стороны абоненты могут сравнивать полученный идентификационный ключ *IK* через некоторый аутентифицированный канал. Например, они могут сравнивать отпечатки цифровой подписи вручную или путем сканирования QR-кода. Методы создания такого аутентифицированного канала выходят за рамки описания протокола Signal. Проверка подлинности сторон по внешнему каналу может производиться до или после процедуры согласования ключа, однако, как указано в [5] в большинстве реализаций такая проверка может быть произведена лишь после обмена сообщениями.

Еще более остро стоит проблема подтверждение подлинности сервера распределения ключей S во время регистрации, никак не определенное в спецификации протокола Signal. Без обеспечения доверия к серверу существует возможность подмены личности любого абонента. Таким образом, решение вопроса взаимной аутентификации сервера и клиента ложится на конкретную реализацию мессенджера. Как правило, при установке приложение мессенджера привязывается к конкретному телефонному номеру, и клиент аутентифицируется с помощью одноразовых SMS-паролей. Аутентификация сервера может проводиться на нижнем уровне с помощью цифровых сертификатов (как в браузерах), либо с помощью проверки отпечатков открытого ключа, данные о которых могут быть «защиты» в клиентском приложении (как это сделано, например, в Telegram).

Для снижения негативного эффекта от возможной компрометации ключей используется оригинальный Double Ratchet алгоритм, описывающий смену краткосрочного ключа асимметричной схемы *EK/EPK* с каждым ответным сообщением и смену симметричных ключей в порождаемой им ключевой цепочке (для шифрования каждого сообщения используется свой ключ). Согласованное с помощью DH значение секретного ключа *SK* служит основой для получения отправного ключа цепочки (*RK*, root key), ключа цепочки (*CK*, chaining key) и ключа шифрования сообщения (*MK*, message key). Ключ цепочки CK_{j+1} получаются из предыдущего CK_j с помощью конструкции HMAC.

Алгоритм формирования цепочки ключей симметричного шифрования на стороне одного из абонентов (абонента 1) показан на рис. 1.4.

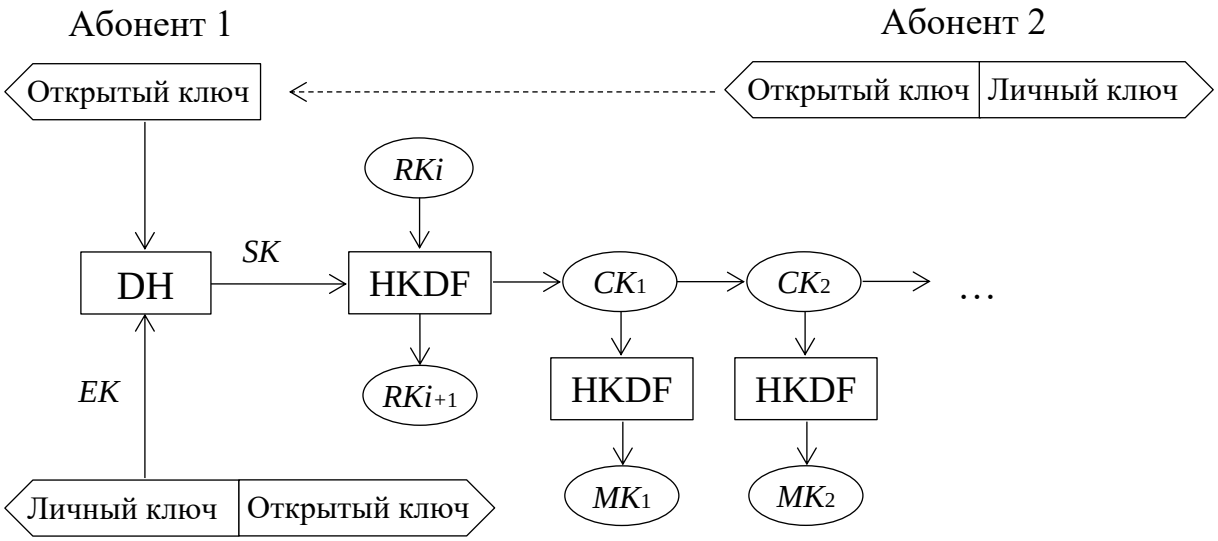

Рисунок 1.4. Формирование цепочек ключей шифрования сообщений

Когда абонент расшифровывает полученное сообщение, он использует для формирования цепочки свое старое значение личного ключа и присланное ему значение открытого ключа другого абонента. В результате ключи расшифрования совпадут с ключами, использованными для шифрования.

Если же абонент хочет послать ответное сообщение, он формирует новую пару кратковременных ключей *EK/EPK* асимметричной схемы, и использует ее для формирования новой цепочки, а открытый ключ *EPK* новой пары отсылает вместе с зашифрованным сообщением.

Если же абонент хочет послать следующее сообщение, не получив ответ на предыдущее, он использует следующий ключ шифрования

сообщения из текущей цепочки, не обновляя ключи асимметричной схемы.

Таким образом, симметричные ключи в цепочках выводятся KDF на основе новых значений DH на каждом этапе, поэтому каждое обновление требует знания свежей информации. Поэтому даже если какой-то среднесрочный или краткосрочный ключ был скомпрометирован, он будет вскоре заменен новым, неизвестным нарушителю значением.

Такой подход позволяет обеспечить специфические свойства безопасности, такие как прямая секретность (forward secrecy) и безопасность после компрометации, или посткомпрометационная секретность (post-compromise security, post-compromise secrecy) [6].

Использование однонаправленных хэш-функций (или HMAC) для формирования цепочки ключей $x \rightarrow H(x) \rightarrow H(H(x)) \rightarrow \ldots$ позволяет обеспечить свойство прямой секретности. Это свойство означает, что если был скомпрометирован текущий ключ, предыдущие значения ключей не могут быть вычислены нарушителем. Это значит, что даже в случае компрометации текущего ключа, нарушитель не сможет расшифровать и прочитать сообщения, отправленные раньше.

Использование же для порождения цепочек обновляющихся согласованных значений DH обеспечивает свойство безопасности после компрометации (post-compromise security). Это свойство позволяет гарантировать безопасность протокола даже в том случае, если секреты одной из сторон ранее были скомпрометированы (рис. 1.5). Можно сказать, что гарантия прямой секретности защищает сеансы от компрометации, произошедшей в последующие моменты времени, а посткомпрометационная секретность защищает сеансы от компрометации, произошедшей ранее.

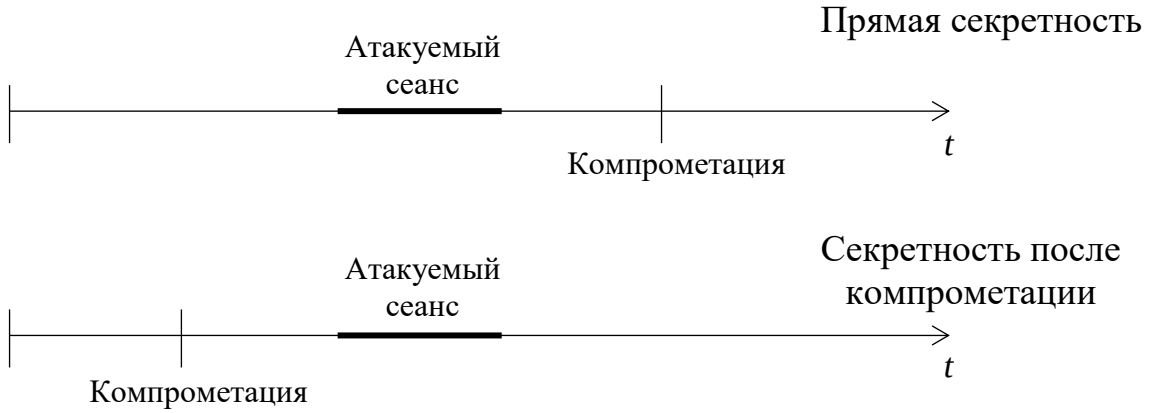

Рисунок 1.5. Сценарии атак, рассматриваемые в рамках будущей секретности и посткомпрометационной секретности

Алгоритм Double Ratchet позволяет учесть недоставленные сообщения за счет включения в каждый заголовок сообщения его номера в цепочке отправки и длину предыдущей цепочки отправки. Это позволяет получателю перейти к соответствующему ключу сообщения при сохранении значений пропущенных ключей на случай, если пропущенные сообщения поступят позже. Вместе с тем, сохранение устаревших ключей несколько ослабляет обеспечиваемое алгоритмом Double Ratchet свойство прямой секретности.

X3DH также обладает свойством будущей секретности.

Авторами исследования [5], опиравшимися в том числе и на анализ кода протокола Signal, не было найдено у него существенных недостатков с точки зрения применения криптографии (при условии создания аутентифицированного канала обмена данными между абонентами и с сервером). Ошибки в коде протокола, связанные с нарушением граничных значений в протоколе (например, использование нулевых значений ключа), а также общие классы ошибок программного обеспечения рассмотрены в [7].

Реализации мессенджеров содержат наряду с кодом библиотек протокола Signal значительный объем стороннего кода. Вместе с тем, каждое применение протокола имеет свои особенности, что может существенно сказаться на надежности оконечного шифрования. При этом анализ большинства реализаций затруднен в связи с закрытостью их кода (WhatsApp, Viber, Skype, Wickr, Threema). Например, известны проблемы с реализацией протокола в системе WhatsApp, которое позволяет серверу принудительно менять ключи пользователя, что в свою очередь может привести к раскрытию недоставленных сообщений [8]. Подобная уязвимость отсутствует в мессенджере Signal. Несовершенна и защита групповых чатов WhatsApp, несмотря на использование сквозного шифрования [9-11].

Кроме того, протокол Signal был в дальнейшем дополнен алгоритмом управления сеансами пользователей для обеспечения синхронизации переписки на разных устройствах, управления параллельными сеансами и восстановления из резервных копий. В этом подпротоколе полномочия сервера существенно расширены, а компрометация хотя бы одного устройства ставит под угрозу безопасность всей коммуникации. Документация к алгоритму управления сеансами лишь указывает на то, что связь между

устройствами и серверами должна быть зашифрована и аутентифицирована, по-прежнему не оговаривая способов решения этих задач.

Дополнительно источником проблем безопасности может быть некорректная реализация процедур, не имеющих прямого отношения к протоколу. Так, например, клиентская часть мессенджера Signal в процессе миграции от расширения для Chrome до полноценного десктопного клиента экспортирует сообщения пользователя в незашифрованные текстовые файлы. При этом никаких предупреждений не выводится, а созданные файлы остаются на диске даже после завершения апгрейда [12].

В отличие от уже упомянутого WhatsApp, мессенджер Telegram использует собственный протокол MTProto, безопасность которого, несмотря на заявления разработчиков, является спорным вопросом [13]. Telegram позиционируется как система с открытым кодом, однако в свободном доступе отсутствует исходный код серверной части, кроме того, используемые решения не столь подробно документированы, как в Signal.

Протокол определяет 3 уровня: высокоуровневый слой, определяющий взаимодействие приложения с API, криптографический слой и компонент доставки, ответственный за выбор способа передачи сообщений (транспортного протокола).

Криптопротокол MTProto за время своего существования претерпел несколько изменений, и в настоящее время использует асимметричную криптосистему RSA-2048, согласование ключа Диффи-Хэллмана, симметричное шифрование с помощью алгоритма AES-256 в режиме IGE и хэш-функцию SHA-256.

На начальном этапе производится аутентификация клиента на основе обмена случайными значениями и регистрация сформированного им открытого ключа асимметричной пары на сервере (рис. 1.6). Аутентификация устройства клиента производится с помощью отправки SMS-кода на телефонный номер, указанный при регистрации. Если в дальнейшем пользователь осуществит вход с другого устройства, ему будет вновь выслан SMS-код.

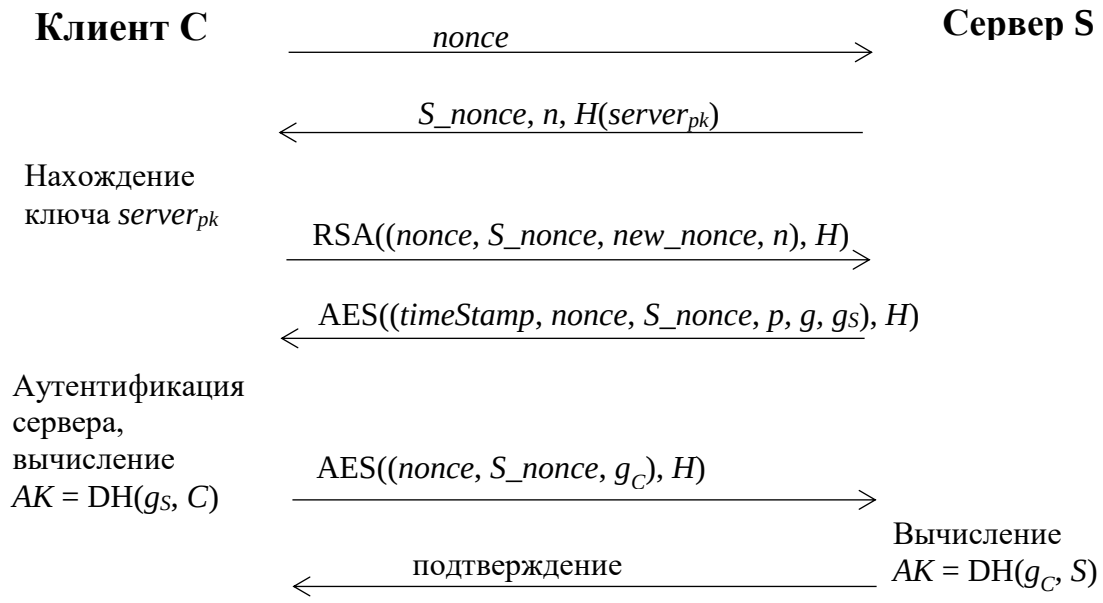

Рисунок 1.6. Регистрация клиента в протоколе MTProto

1. Клиент C отправляет запрос на сервер S со случайной строкой *nonce*.
2. S отправляет в ответ другую случайную последовательность *S_nonce*, целое число *n* (64 бита) и отпечаток открытого RSA-ключа сервера.
3. Клиент C раскладывает *n* на два простых числа *p* и *q*, $p < q$, и выбирает из набора публичных ключей сервера, хранящийся на устройстве, открытый ключ $server_{pk}$, который должен соответствовать пришедшему с сервера отпечатку.
4. C выбирает другую случайную строку *new_nonce*, отличающуюся от предыдущих строк *nonce* и *S_nonce*. Эта строка в открытом виде не передавалась. Затем клиент собирает набор из трех случайных строк, чисел *n*, *p* и *q*, вычисляет его хэш *H* и шифрует все это по алгоритму RSA с помощью ключа $server_{pk}$ и отправляет на сервер S.
5. В ответ сервер S передает свою метку времени, значения *nonce* и *S_nonce* (что позволяет клиенту аутентифицировать сервер), а также параметры криптосистемы Диффи-Хэллмана *g*, *p* и свой открытый DH ключ g_S, и хэш содержимого. Полученный пакет дополняется случайными значениями и передается в зашифрованном виде. Для шифрования используется алгоритм AES-256 на временном ключе и с вектором инициализации, полученным на основе *S_nonce* и *new_nonce* (рис. 1.7).
6. Клиент проверяет присланные значения, а затем выбирает секретное число *C* и вычисляет с помощью криптосистемы Диффи-Хэллмана общий секретный ключ $AK = \mathrm{DH}(g_S, C)$ и свой открытый ключ g_C.

Затем g_C, значения *nonce* и *S_nonce* и хэш всего пакета шифруются AES и отправляются на сервер S.

7. Полученные S значения достаточны для формирования сервером общего секретного ключа *AK*. Если все присланные клиентом значения правильны, сервер посылает ответ с подтверждением (или с отказом в противном случае).

Общий секретный ключ *AK* теперь может использоваться для шифрования обмена данными между клиентом и сервером при использовании облачных чатов (без оконечного шифрования).

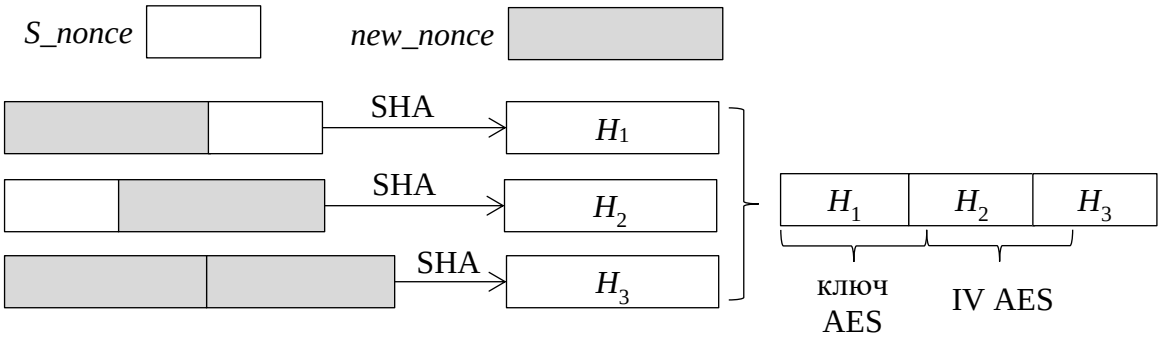

Рисунок 1.7. Формирование временного ключа и вектора инициализации шифра AES во время регистрации MTProto

Оконечное шифрование в режиме секретных чатов выполняется следующим образом. При инициализации секретного чата пользователи A и B выполняют стандартный протокол Диффи-Хэллмана согласования общего ключа *AK* без предварительной аутентификации сторон. Согласование общего ключа периодически проводится заново. Как уже отмечалось, поскольку согласование ключа производится без аутентификации канала, это делает возможным реализацию классической атаки «человек посередине». Пользователям предлагается перед обменом сообщениями проводить визуальную сверку отпечатков *AK* в виде графического кода.

Само шифрование сообщений пользователя производится по схеме, представленной на рис. 1.8 (заливкой здесь отмечены секретные параметры шифрования). Дополнительная информация, добавляемая к сообщению при шифровании, включает исходящий и входящий номер сообщения, случайные значения соли и дополнения, общую длину и другие значения. В процедуре шифрования, используемой в облачных чатах, добавляемая к сообщению дополнительная информация несколько

отличается, общая же структура процедуры шифрования остается неизменной.

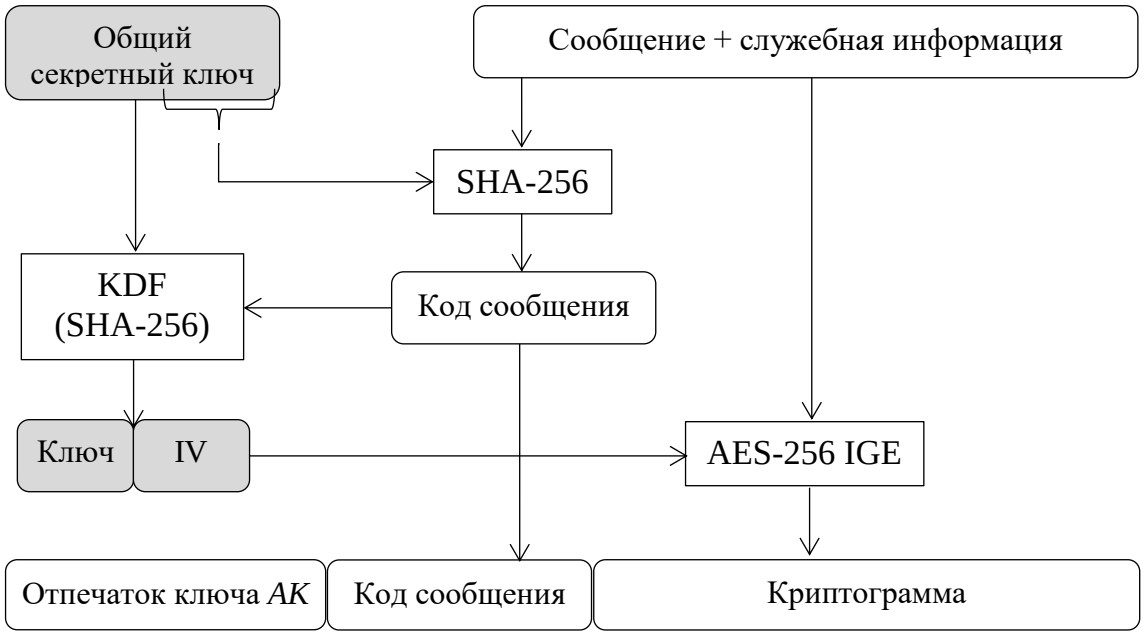

Рисунок 1.8. Общая схема шифрования сообщений в MTProto

На основе шифруемой информации (то есть самого сообщения с добавленной к нему дополнительной информацией) и части общего секретного ключа рассчитывается код сообщения, служащий для проверки целостности, а также позволяющий сформировать различные ключи шифрования для разных сообщений. Код сообщения передается вместе с криптограммой в открытом виде. Для вывода ключей шифрования используется функция на основе SHA-256, построенная по принципу, сходному с KDF, представленной на рис. 1.7.

Поскольку отпечаток общего секретного ключа отсылается с каждым сообщением, при использовании оконечного шифрования абонент в любой момент может удостовериться в подлинности второй стороны коммуникации с помощью сверки графических кодов, однако это придется делать вручную.

Следует отметить, что сообщения об обнаружении уязвимостей в Telegram (например, [14]) появляются регулярно, и не менее регулярно они опровергаются владельцами мессенджера, заявляющими о его исключительной безопасности.

Знание особенностей оконечного шифрования позволяет прийти к пониманию существующих ограничений и возможных проблем безопасности. Так, основной проблемой является реализация

аутентификации сторон коммуникации. Единственным действительно надежным способом связывания открытого ключа с абонентом, а значит, и обеспечения аутентификации стороны при децентрализованном (без участия сервера) распределении ключей в асимметричных схемах является использование цифровой сертификации. Последнее невозможно вне инфраструктуры открытого ключа (PKI). Вынужденный отказ от использования цифровых сертификатов приводит к перекладыванию решения проблемы аутентификации сторон общения на самих абонентов, которые должны осуществлять сверку отпечатков открытых ключей или их графических представлений каждый раз, когда производится процедура согласования общего ключа (в начале сеанса и в случае его изменения в дальнейшем). Это не самый удобный вариант аутентификации, поскольку:

- для снижения вероятности вскрытия шифра ключ может меняться достаточно часто;
- не все реализации мессенджеров осуществляют уведомление пользователей о смене ключа;
- пользователь должен быть достаточно информирован, чтобы осознавать необходимость сверки ключей, и обладать хорошей дисциплиной, чтобы производить ее регулярно;
- если сверка производится визуально (как в Telegram) велика вероятность ошибки для внешне «похожих» графических кодов.

Еще одна серьезная проблема связана с синхронизацией переписки для разных устройств при использовании оконечного шифрования. Системы обмена сообщениями решают эту проблему по-разному. Например, Telegram и Skype отказались от синхронизации переписки для секретных чатов (переписка будет доступна только на том устройстве, на котором начата); синхронизация выполняется только для обычных облачных чатов, не использующих оконечного шифрования. Другие системы поддерживают возможность синхронизации и при использовании оконечного шифрования, например, протокол Signal содержит соответствующий алгоритм. Однако остаются вопросы к реализации этой функции. Например, Signal (как и многие другие мессенджеры) позволяет открыть параллельную сессию (и получить полный доступ к переписке) на другом устройстве, просто отсканировав QR-код, что дает пространство для злоупотреблений даже при кратковременном доступе нарушителя к устройству [15, 16]. Для синхронизации переписки на разных устройствах WhatsApp и Viber поддерживают механизм восстановления сообщений из облака, которые сохраняются в нем в незашифрованном виде [17]. Таким образом,

появляется еще один возможный источник информации о переписке, защищенность которого обеспечивается средствами, отличными от сквозного шифрования. Несмотря на то, что создание резервных копий, как правило, может быть отключено в приложении, они могут создаваться автоматически также и самим устройством, и не всегда контролируются пользователем.

Еще одна проблема – возможность сбора метаинформации о контактах и действиях абонентов самой системой обмена сообщениями или третьими лицами (проблема защиты социального графа). Внимание к этому вопросу было привлечено во время скандала, связанного с компанией Facebook, которая занималась широкомасштабным сбором и продажей информации о пользователях. Как правило, возможность такого сбора оговаривается в пользовательском соглашении мессенджера. Например, в пользовательском соглашении WhatsApp – продукта, принадлежащего компании Facebook, указано, что мессенджер собирает информацию о деятельности абонентов (например, как абоненты используют услуги, как взаимодействуют с другими пользователями с помощью сервисов системы и т. п.), журналы веб-сайта, информацию о конкретном устройстве при установке, доступе или использовании сервиса системы (такие как модель телефона, его операционная система и информация из браузера, IP-адреса и мобильной сети, включая номер телефона). В то же время разработчики мессенджера Signal утверждают, что сервер системы сохраняет лишь номер телефона, указанный абонентом при регистрации, и дату последнего входа в систему (с точностью до дня). С другой стороны, как уже отмечалось выше, при использовании оконечного шифрования адресная информация передается в открытом виде и может быть доступна нарушителю.

Как видно, само по себе использование оконечного шифрования не является гарантией безусловной защищенности, как это представляют многочисленные публикации, сводя вопрос безопасности к тому, включено ли в мессенджере оконечное шифрование по умолчанию или нет. В [18] предложены следующие критерии для оценки безопасности мессенджеров и проведен анализ нескольких наиболее известных систем:

- степень централизации управления и архитектуры системы (возможность работы без поддержки серверной части системы, возможность прямых peer-to-peer соединений абонентов без участия сервера);
- возможность анонимной регистрации и использования;
- поддержка оконечного шифрования;
- поддержка синхронизации секретных чатов;

- уведомление о необходимости проверки отпечатков ключей собеседников в секретных чатах;
- запрет скриншотов в режиме секретного чата;
- поддержка групповых секретных чатов;
- уведомление о необходимости проверки отпечатков ключей собеседников в групповых секретных чатах;
- защита социального графа.

Укажем, не раскрывая подробно, еще ряд моментов, которые могут служить источником проблем безопасности при использовании систем обмена сообщениями на мобильных платформах.

Критическим с точки зрения безопасности является получение нарушителем физического доступа к устройству, в связи с этим встают вопросы о надежности удаления информации (старых сообщений), создании локальных резервных копий переписки (автоматическом сохранении сообщений на устройстве) и защите локального хранилища.

Создаваемый системой аккаунт пользователя, как правило, привязывается к номеру телефона, на который в ходе регистрации или при входе с другого устройства отправляется код подтверждения посредством SMS. Известная уязвимость в протоколе сотовой связи SS7 позволяет осуществлять перехват SMS-сообщений, как следствие, нарушитель может зайти в аккаунт абонента, зная лишь номер его телефона. Последнее позволяет, как минимум, отправлять сообщения от имени абонента, а в ряде случаев и получить доступ к его переписке [19].

Получение несанкционированного доступа к пользовательским данным из другого приложения или с помощью вредоносного кода. Данная угроза носит достаточно общий характер, однако она особенно актуальна для мобильных платформ, операционные системы которых имеют ограниченную модель безопасности.

Использование небезопасных клиентов. Если абонент загружает клиентское программное обеспечение не из официального магазина, а по прямой ссылке, велика вероятность использования поддельного клиента с неконтролируемыми возможностями.

Таким образом, наличие криптографических функций является важным, но далеко не единственным фактором обеспечения безопасности мобильных систем, и мессенджеров в частности.

Литература:

1. Названы любимые мессенджеры россиян //CNews. Издание о высоких технологиях, 28.02.2018 [Электронный ресурс]. URL: http://www.cnews.ru/news/top/2018-02-

28_whatsapp_stal_samym_populyarnym_messendzheram_v (дата обращения: 31.10.2018).
2. Dennis Fisher. Исследователи обнаружили проблемы с SSL в WhatsApp, 24.02.2014 [Электронный ресурс]. URL: https://threatpost.ru/issledovateli-obnaruzhili-problemy-s-ssl-v-whatsapp/1626/ (дата обращения: 31.10.2018).
3. Карев Антон. Грандиозный факап. Разбираем уязвимости проверки сертификатов SSL и TLS в небраузерном софте // Хакер, 08.03.2018 [Электронный ресурс]. URL: https://xakep.ru/2018/03/08/ssl-tls-fuckup/, https://justpaste.it/1i089 (дата обращения: 31.10.2018).
4. Signal. Technical Information [Электронный ресурс]. URL: https://signal.org/docs/ (дата обращения: 31.10.2018).
5. Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowlingy, Luke Garratt, Douglas Stebila. A Formal Security Analysis of the Signal Messaging Protocol [Электронный ресурс]. URL: https://eprint.iacr.org/2016/1013.pdf (дата обращения: 31.10.2018).
6. Katriel Cohn-Gordon, Cas Cremers, Luke Garratt. On Post-Compromise Security. – Department of Computer Science, University of Oxford, 25.10.2016 [Электронный ресурс]. URL: https://eprint.iacr.org/2016/221.pdf (дата обращения: 01.11.2018).
7. JP Aumasson, Markus Vervier. Hunting for vulnerabilities in Signal - Hitbsecconf2017, Amsterdam [Электронный ресурс]. URL: https://conference.hitb.org/hitbsecconf2017ams/materials/D2T1 - Markus Vervier - Hunting for Vulnerabilities in Signal.pdf (дата обращения: 01.11.2018).
8. Нефедова Мария. В WhatsApp найдена уязвимость, позволяющая читать чужие сообщения, 13.01.2017 [Электронный ресурс]. URL: https://xakep.ru/2017/01/13/whatsapp-retransmission-problem/ (дата обращения: 01.11.2018).
9. Andy Greenberg. WhatsApp security flaws could allow snoops to slide into group chats, 10.01.2018 [Электронный ресурс]. URL: https://www.wired.com/story/whatsapp-security-flaws-encryption-group-chats/ (дата обращения: 01.11.2018).
10. Уязвимость в WhatsApp позволяет получить доступ к данным участников групповых чатов, 04.04.2018 [Электронный ресурс]. URL: https://www.securitylab.ru/news/492456.php (дата обращения: 01.11.2018).
11. Уязвимость в WhatsApp позволяет изменять сообщения в групповых чатах. 09.08.2018 [Электронный ресурс]. URL: https://www.securitylab.ru/news/494962.php (дата обращения: 01.11.2018).
12. Самые защищенные и безопасные мессенджеры – обзор, 25.10.2018 [Электронный ресурс]. URL: http://www.voipoffice.ru/tags/zaschischennye_messendzhery/ (дата обращения: 01.11.2018).
13. Jakob Bjerre Jakobsen. A practical cryptanalysis of the Telegram messaging protocol. Master's thesis. − Aarhus University. Department of computer science, September 2015 [Электронный ресурс]. URL: https://nourbakhsh.ir/wp-content/uploads/2015/11/ jakobsen-master-thesis-telegram.pdf (дата обращения: 02.11.2018).
14. Hayk Saribekyan, Akaki Margvelashvili. Анализ безопасности Telegram, 11.01.2018 [Электронный ресурс]. URL: https://www.securitylab.ru/analytics/490726.php (дата обращения: 02.11.2018).

15. Калошин М. Как могут читать вашу/чужую переписку в WhatsApp, 01.10.2015 [Электронный ресурс]. URL: https://www.kaloshin.me/2015/10/how-to-read-whatsapp-messages-from-another-device.html (дата обращения: 02.11.2018).
16. Нефёдова М. Атака QRLJacking доказывает небезопасность авторизации с использованием SQRL, 02.08.2016 [Электронный ресурс]. URL: https://xakep.ru/2016/08/02/qrljacking/ (дата обращения: 02.11.2018).
17. Bill Budington, Gennie Gebhart. Where WhatsApp Went Wrong: EFF's Four Biggest Security Concerns, 26.01.2017 [Электронный ресурс]. URL: https://www.eff.org/deeplinks/2016/10/where-whatsapp-went-wrong-effs-four-biggest-security-concerns (дата обращения: 04.11.2018).
18. Губарева В., Колисниченко Д. Шифруйся грамотно! Выбираем мессенджер для безопасной и приватной переписки // Хакер, 03.07.2018 [Электронный ресурс]. URL: https://xakep.ru/2018/07/03/messengers/, https://teletype.in/@ht_vbok/HJGdsTYGQ (дата обращения: 04.11.2018).
19. Как взломать Telegram и WhatsApp: спецслужбы не нужны, 06.05.2016 [Электронный ресурс]. URL: https://habr.com/company/pt/blog/283052/ (дата обращения: 04.11.2018).