|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Российский биотехнологический университет (РОСБИОТЕХ)»** | | | | | | | | | | | | | |
|  |  | |  |  | |  |  |  | |  | |  |
| Кафедра | | | Информатика и вычислительная техника пищевых производств | | | | | | | | | | |
|  |  | |  |  | |  |  |  | |  | |  |
| Направление (Специальность) | | |  | | | | | | | | | | |
|  |  | |  |  | |  |  |  | |  | |  |
| Профиль | | |  | | | | | | | | | | |
|  |  | |  |  | |  |  |  | |  | |  |
|  |  | |  |  | |  |  | **К ЗАЩИТЕ** | | | | | |
|  |  | |  |  | |  |  | **(РЕКОМЕНДОВАНО / НЕ РЕКОМЕНДОВАНО)** | | | | | |
|  |  | |  |  | |  |  | зав. кафедрой | | | | | |
|  |  | |  |  | |  |  | к.ф.-м.н., доцент | | | | | |
|  |  | |  |  | |  |  | *(ученая степень, ученое звание)* | | | | | |
|  |  | |  |  | |  |  |  | | |  | Т.А. Санаева | |
|  |  | |  |  | |  |  | *(подпись)* | | |  | *(И.О. Фамилия)* | |
|  |  | |  |  | |  |  | « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ г. | | | | | |
|  |  | |  |  | |  |  |  | |  | |  |
| **КУРСОВАЯ РАБОТА** | | | | | | | | | | | | | |
| *по дисциплине* | | | | | | | | | | | | |
| *«Информационные системы и технологии»* | | | | | | | | | | | | |
|  |  | |  |  | |  |  |  | |  | |  |
| на тему: | |  | | | | | | | | | | | |
|  |  |
|  |  | *(тема курсовой работы)* | | | | | | | | | | | |
|  |  |  |  |  | |  |  |  | |  | |  |
| Обучающийся: | | |  | | « \_\_\_ » \_\_\_\_\_\_\_ 20\_\_ г. | | | |  | | | | |
|  |  |  | *(подпись)* | |  |  |  |  | *(инициалы, фамилия)* | | | | |
|  |  | |  |  | |  |  |  | |  | |  |
|  | | | |  |  | | | группа | | |  |  | |
|  |  | |  |  |  | | |  | | |  | *(шифр группы)* | |
| Руководитель | | |  | | « \_\_\_ » \_\_\_\_\_\_\_ 20\_\_ г. | | | | доц, к.т.н, Т.В. Ящун | | | | |
|  |  | | *(подпись)* | |  |  |  |  | *(уч. степень, уч. звание, инициалы, фамилия)* | | | | |

**СОДЕРЖАНИЕ**

Введение

1. Исследование принципов обеспечения безопасности частной информации на мобильных устройствах

1.1 Безопасность данных, хранящихся на мобильном устройстве

1.2 Информация о местоположении устройства

1.3 Безопасность данных, передающихся между мобильными устройствами

2. Сравнительный анализ существующих программных средств, предназначенных для обмена частной информацией

2.1 Краткие выводы сравнительного анализа

2.2 Подробный анализ приложений

2.2.1 WhatsApp

2.2.2 Viber

2.2.3 Signal

2.2.4 Telegram

2.2.5 Antox

3. Разработка программного средства, построенного на отечественных алгоритмах шифрования

3.1 Требования к эталонному приложению

3.1.1 Архитектура приложения

3.2 Описание программной реализации

3.2.1 Архитектура

3.2.2 Шифрование

Заключение

Список использованных источников

**ВВЕДЕНИЕ**

В ходе выпускной квалификационной работы необходимо исследовать средства обеспечения безопасности на мобильных устройствах и разработать приложение для организации безопасного канала связи между мобильными устройствами по сети Интернет.

В начале рассматриваются общие принципы безопасности информации, хранящейся на мобильном устройстве или передаваемой между ними. Далее в работе сравниваются существующие решения для безопасного обмена данными, оцениваются их плюсы и минусы и приводится сравнительная таблица. В последней главе представляется программная реализация разработанного приложения для безопасной передачи данных между мобильными устройствами по сети Интернет.

Об актуальности данной темы говорит тот факт, что за время написания данной работы самые популярные мессенджеры WhatsApp и Viber обзавелись дополнительными мерами для обеспечения безопасности данных (сквозным шифрованием), которые в теории должны предотвратить возможность доступа к передаваемой информации владельцами данных компаний.

В целом, во времена, когда частная жизнь всё больше становится виртуальной, нет ничего важнее, чем позаботится о безопасности этой жизни. Всё большую часть информации люди хранят и передают с помощью мобильных устройств, поэтому необходимо предоставить средства для обеспечения безопасности этих данных. Многие информационные компании сейчас делают акцент на безопасности данных.

Стоит также упомянуть, что данная работа является продолжением междисциплинарных курсовых работ Пронина Дениса [1] и Колпаченко Сергея [2], в которых были исследованы и реализованы отечественные алгоритмы симметричного и асимметричного шифрования. В этой работе эти алгоритмы соединены вместе в гибридную схему, которая обеспечивает безопасность передаваемых данных. С аналогичной темой автор выступал на Ежегодной межвузовской научно-технической конференции студентов, аспирантов, молодых специалистов имени Е.В. Арменского [3].

алгоритм шифрование мобильное средство

# **1. Исследование принципов обеспечения безопасности частной информации на мобильных устройствах**

# **1.1 Безопасность данных, хранящихся на мобильном устройстве**

Мобильные устройства в современном мире всё чаще используются и для хранения личных данных, и для коммуникации (передачи данных).

По данным международной компании "Лаборатория Касперского" [4], которая специализируется на разработке систем защиты от киберугроз, 88% пользователей хранят на своих мобильных устройствах личную информацию, 48% хранят логины и пароли, и 28% хранят данные, связанные с финансами.

Эта статистика говорит о необходимости защищать информацию, хранящуюся на мобильных устройствах.

Для этого в первую очередь, конечно, необходимо обезопасить своё устройство от вирусов и других вредоносных программ. Согласно некоммерческой правозащитной организации Electronic Frontier Foundation (EFF) [5] они могут появится на девайсе благодаря пользователю, который сам установит вредоносное ПО, или благодаря злоумышленнику, который установит ПО, используя "дыры" в предустановленном ПО либо просто получив физический доступ к мобильному устройству (учитывая размеры современных устройств, такой вариант вполне реален).

В качестве общей защиты от вирусов можно использовать мобильные антивирусы.

В таблице 1 на основе данных из общедоступной универсльной интернет-энциклопедии "Википедия" [6] сравниваются мобильные приложения компаний Avast, Avira Operations, Dr. Web, ESET, Kaspersky Lab, McAfee, Qihoo 360 и Symantec для защиты от вредоносных программ на платформе Android. Все они кроме приведенных в таблице критериев имеют возможность проверки устройства по требованию и защиту устройства в режиме реального времени.

Таблица 1

Сравнительная таблица для мобильных антивирусов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ПО | Брандмауэр | Веб-защита | Управление разрешениями приложений | Антивор (удаленное управление устройством) | Фильтрация звонков и смс |
| Avast Mobile Security | Да | Да | Да | Да | Да |
| Avira Free Android Security <https://en.wikipedia.org/wiki/Avira>НетНетНетДаДа |  |  |  |  |  |
| Dr. Web Mobile Security Suite <https://en.wikipedia.org/wiki/Dr.\_Web>ДаДаНетДаДа |  |  |  |  |  |
| ESET Mobile Security <https://en.wikipedia.org/wiki/NOD32>НетЧастично, только антифишингДаДаДа |  |  |  |  |  |
| Kaspersky Mobile Security <https://en.wikipedia.org/wiki/Kaspersky\_Mobile\_Security>НетДаНетДаДа |  |  |  |  |  |
| McAfee Mobile Security <https://en.wikipedia.org/wiki/McAfee>НетДаДаДаДа |  |  |  |  |  |
| 360 Mobilesafe <https://en.wikipedia.org/w/index.php?title=360\_Mobilesafe&action=edit&redlink=1>ДаДаДаДаДа |  |  |  |  |  |
| Norton Mobile Security <https://en.wikipedia.org/w/index.php?title=Norton\_Mobile\_Security&action=edit&redlink=1>НетДаНетДаДа |  |  |  |  |  |

Уязвимости предустановленного программного обеспечения, к сожалению, может исправить только производитель, пользователь здесь практически бессилен. Единственное, что он может делать, это по возможности своевременно обновлять предустановленное ПО до актуальных версий.

В свою очередь, чтобы ограничить физический доступ к устройству и данным на нем, необходимо установить средства защиты на экран блокировки. На современных устройствах это может быть пароль в классическом понимании этого слова (набор символов), графический пароль (когда для разблокировки надо по точкам нарисовать рисунок) и даже разблокировка по отпечатку пальца владельца. Конечно, всё это не обеспечивает стопроцентной защиты и иногда следует быть всего лишь немного внимательнее по отношению к своим действиям с мобильными устройствами.

# **1.2 Информация о местоположении устройства**

Информации о местоположении человека также является личной и также с некоторой точностью определяется с помощью мобильных устройств, которые излучают сигналы круглосуточно. Есть по крайней мере 4 способа узнать местоположение устройства [5].

Первый способ - определить место по сотовым вышкам. Мобильные операторы могут вычислить, где находится абонент, пока его телефон работает и зарегистрирован в сети. Для этого им всего лишь необходимо определить силу сигнала от мобильного устройства на разных вышках. Точность зависит от многих факторов, например, используемых технологий и количества сотовых вышек в районе абонента, но обычно в городе местоположение устройства таким способом можно определить с точностью до квартала.

Второй способ - использовать IMSI-ловушку. Эти устройства используются правительством и иными лицами, которые могут позволить себе такие технически ложные устройства. Принцип действия таков, что ловушка "притворяется" настоящей сотовой вышкой, мобильные устройства, расположенные близко к ней, подключаются и передают данные через неё. Таким образом, устройства такого рода позволяют следить не только за местоположением, но и за коммуникациями, о чем речь пойдет в разделе 1.3.

Третий способ - отслеживать сигналы Wi-Fi и Bluetooth. Данными передатчиками в настоящее время оснащаются все мобильные устройства вплоть до умных часов. Все передатчики имеют свой уникальный MAC-адрес, который нельзя сменить на большинстве мобильных устройств. Отличие от GSM-сетей заключается в том, что радиус действия Wi-Fi и Bluetooth существенно меньше, то есть, надо уже знать примерное местоположение жертвы, а также MAC-адреса передатчиков, которые установлены в её мобильном устройстве. В плюсы такого способа стоит отнести то, что местоположение можно определить с большей точностью или, например, можно определять, когда жертва входит в здание и когда выходит.

Последний способ - утечка данных о местоположении во время веб-серфинга. Современные мобильные устройства способны сами определить свое местоположение, например, с помощью GPS-сигнала или с помощью сервисов, которые помогают определить местоположение девайса на основе данных от сотовый вышек или Wi-Fi-сетей. Приложения могут получить доступ к этим данным с разрешения пользователя. Некоторые приложения после этого отправляют данные о местоположении пользователя в сеть, благодаря чему у третьих лиц появляется возможность следить за местоположением устройств.

Информация о местах, в которых находятся устройства, может служить не только для того, чтобы непосредственно определить местоположение человека, но, и чтобы определять, какие отношения могут связывать разных людей. Например, с большой вероятностью по этим данным можно определить сослуживцев или пары с близкими отношениями.

К сожалению, обычные люди не в силах как-либо противостоять первым двум способам (возможно только использовать одноразовые телефоны, о чем будет сказано в разделе 1.3), но пользователи всегда могут позаботиться о том, чтобы выключить Wi-Fi и Bluetooth, а также тщательно следить за приложениями, которым они предоставляют доступ к данным о своем местоположении. В этом, к слову, могут помочь антивирусы, рассмотренные в разделе 1.1.

# **1.3 Безопасность данных, передающихся между мобильными устройствами**

Изначально мобильные сети не были спроектированы с защитой от прослушивания. Это означает, что любой человек, имея нужный передатчик, мог прослушать разговор. В наше время ситуация изменилась: в стандарты передачи данных по мобильным сетям были добавлены средства шифрования. К сожалению, эти средства не обеспечивали надежной защиты [5]. Некоторые операторы, а иногда и целые страны при этом даже после добавления шифрования не стали его использовать. Но даже если операторы используют самые последние стандарты с самыми передовыми системами шифрования, некоторые люди все равно имеют доступ к переписке и разговорам по сетям GSM. Ими являются сами операторы, а также правительственные службы.

Например, в России существует СОРМ (система технических средств для обеспечения функций оперативно-розыскных мероприятий). Согласно закону "О связи" такие системы должны быть установлены у каждого операторы. Они позволяют спецслужбам удаленно и без содействия самого оператора прослушивать телефонные разговоры. В особых случаях, для обеспечения безопасности граждан, даже без решения суда.

Также стоит ещё раз упомянуть рассмотренные в разделе 1.2 IMSI-ловушки, которые представляют собой ненастоящие сотовые вышки. Мобильные устройства в свою очередь с ними соединяются и передают данные, к коим и получает доступ владелец ловушки.

Как способ защиты от прослушивания в GSM сетях, а также от отслеживания местоположения по сотовым вышкам можно использовать одноразовые телефоны и SIM-карты, зарегистрированные на другого человека, но здесь существует ряд сложностей и недостатков [5]. Во-первых, смена только телефона или только SIM-карты не поможет, так как у каждого телефона существует собственный уникальный идентификатор IMEI, который связывают с SIM-картой и владельцем этих двух вещей. Во-вторых, даже после смены мобильного устройства и SIM-карты существует масса способов определить владельца телефона: он может появляться в одних и тех же местах, он может использовать несколько мобильных устройств, находящихся постоянно рядом, он может звонить одним и тем же людям, в конце концов, существуют системы распознавания голоса. Но стоит сказать, что это в любом случае сделает атаку на мобильные коммуникации намного труднее.

Ситуация меняется, если человек начинает использовать приложения для передачи данных по сети Интернет. В данном случае, степень защищенности передающейся информации зависит от средств шифрования, которые использует приложение. Эти средства шифрования могут быть любыми и могут быть надежнее шифрования данных в сетях GSM. Важным условием хорошей криптозащиты приложения является присутствие сквозного шифрования и отсутствие возможности у разработчика и владельца сервера просмотреть переданные данные.

Сделав вывод из всего вышесказанного, я продолжил работу именно над передачей данных в сети Интернет, так как именно эта область в сфере защиты личной информации больше всего зависит от выбора программного обеспечения и предоставляет возможность обеспечить необходимый уровень защиты.

# **2. Сравнительный анализ существующих программных средств, предназначенных для обмена частной информацией**

# **2.1 Краткие выводы сравнительного анализа**

Для анализа мною было выбрано 5 мобильных интернет мессенджеров. WhatsApp и Viber, как наиболее популярные мобильные мессенджеры в России по данным исследовательской компании J’son & Partners Consulting [7], Signal и Telegram, как два наиболее популярных мессенджера по данным Google Play Market, которые позиционируют себя как защищенные, и Antox, так как он имеет распределенную архитектуру. В таблице 2 представлен результат сравнения.

Таблица 2

Сравнительная таблица для интернет-мессенджеров

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мессенджер | Сквозное шифрование | Временные ключи шифрования | Открытый исходный код | Возможность прямого соединения собеседников |
| WhatsApp | Да | Да | Нет | Нет |
| Viber | Да | Да | Нет | Нет |
| Signal | Да | Да | Да | Нет |
| Telegram | Да | Да | Да | Нет |
| Antox | Да | Да | Да | Да |

Хотелось бы отдельно отметить тот факт, что за время написания данной работы самые распространенные мессенджеры в России WhatsApp и Viber по информации представленной на их официальных сайтах получили сквозное шифрование, которого раньше не было.

# **2.2 Подробный анализ приложений**

# **2.1 WhatsApp**

По разным оценка является самым распространенным мобильным приложения для обмена данными по сети Интернет в мире.31-го марта 2016-го года, во время написания данной работы, это приложение обзавелось сквозным шифрованием, что позволяет причислить его к защищенным, хотя изначально оно себя так не позиционировало. Единственным недостатком остается то, что исходные коды этого мессенджера закрыты, и оценить надежность шифрования не представляется возможным. Тем не менее, возможно оценить организацию этого шифрования, которая подробно и ясно изложена в документации [8].

Следует добавить, что реализацией шифрования в мессенджере WhatsApp является некоммерческая организация разработчиков открытого программного обеспечения Open Whisper Systems, которой принадлежит защищенный мессенджер Signal, рассмотренный ниже. Эта организация также известна тем, что все её продукты рекомендовал Эдвард Сноуден [9].

Согласно документации, в основу шифрования лёг Signal Protocol - протокол, который используется в одноименном мессенджере. Все ключи вычисляются с помощью эллиптической кривой Curve25519. При установке генерируются Identity Key Pair - долгосрочная пара ключей, Signed Pre Key - среднесрочная пара ключей, заверенная Identity Key Pair, и One-Time Pre Keys - партия одноразовых ключей. При регистрации пользователя на сервере отправляются публичные части сгенерированных ключей. Секретные ключи остаются только на устройстве.

Для связи двух пользователей по защищенному каналу в начале необходимо установить сессию. После установки сессии нет необходимости её переустанавливать. Это нужно только в случае переустановки приложения или при смене мобильного устройства. При установке сессии происходит следующая последовательность действий:

) Инициатор запрашивает *Identity Key Pair*, *Signed Pre Key* и *One-Time Pre Key* получателя сообщений.

2) Сервер высылает запрошенные ключи, при этом удаляя *One-Time Pre Key*, так как ключи такого типа одноразовые (если *One-Time Pre Keys* получателя были израсходованы и не пополнены, то *One-Time Pre Key* возвращается на сервер).

) Инициатор сохраняет полученные ключи, связывая их с получателем сообщений: *Identity Key* как *Irecipient*, *Signed Pre Key* как *Srecipient* и *One-Time Pre Key* как *Orecipient*.

) Инициатор генерирует пару временных ключей с помощью Curve25519, *Einitiator*.

) Identity Key инициатора пометим как *Iinitiator*.

) После этого генерируется мастер-ключ (ECGH, Elliptic curve Diffie-Hellman - протокол Диффи-Хеллмана на эллиптических кривых) *master\_secret = ECDH (Iinitiator, Srecipient) || ECDH (Einitiator, Irecipient) || ECDH (Einitiator, Srecipient) || ECDH (Einitiator, Orecipient).* Если *One-Time Pre Key* отсутствует, то последнее действие опускается.

) Инициатор использует HKDF (функция формирования ключей) для создания *Root Key* и *Chain Keys* из master\_secret.

В качестве результатов установки сессии получается *Root Key* - 32-байтное значение, используемое для генерации *Chain Key*, что тоже является 32-байтным значением и используется для генерации *Message Key*. *Message Key* - 80-байтное значение, которое используется для шифрования содержания сообщений.

После установки долгосрочной сессии, инициатор может сразу же начинать отправлять сообщения получателю. В начале всех сообщений будет передаваться информация, которая необходима получателю для установки сессии. Чтобы установить сессию, получатель выполнит следующие действия:

1) Получатель вычислит *master-secret*, используя свои секретные ключи и ключи, полученные в сообщениях.

2) Получатель удалит *One-Time Pre Key*, использованный инициатором.

) Получатель использует HKDF для вычисления *Root Key* и *Chain Keys* из *master\_secret*.

Каждое сообщение шифруется новым *Message Key*, который вычисляется следующим образом:

*1) Message Key = HMAC-SHA256 (Chain Key, 0x01).*

2) После первого шага *Chain Key* обновляется по формуле *Chain Key = HMAC-SHA256 (Chain Key, 0x02).*

Таким образом, согласно официальной документации, в мобильном приложении WhatsApp реализована система сквозного шифрования с использованием временных ключей и с возможностью подтверждения личности собеседника, путем сравнения ключей. За счет применения сквозного шифрования владельцы приложения и серверов не должны иметь доступ к перепискам пользователей. Всё это в теории должно обеспечивать высокий уровень безопасности передаваемых личных данных.

# **2.2.2 Viber**

Второй по популярности мессенджер в России [7]. Так же, как и WhatsApp, во время написания данной работы, 19-го апреля 2016-го года с выходом версии 6.0, получил сквозное шифрование для стран Бразилия, Белоруссия, Израиль и Таиланд.

К сожалению, разработчики данного мобильного приложения не раскрывают подробностей криптографической системы. Единственное, что они говорят - теперь есть сквозное шифрование. Таким образом, никто не в силах каким-либо образом оценить степень надежности используемых шифров.

К плюсам данного приложения можно отнести наличие "скрытых чатов". Пользователь может скрыть некоторые чаты и доступ к ним можно будет получить только после ввода PIN-кода. Это позволяет ограничить физический доступ к конфиденциальной переписке, например, в ситуации, когда мобильным устройством завладел злоумышленник.

# **2.2.3 Signal**

История этого мобильного приложения началась в 2010-ом году с создания двух приложений: TextSecure для защищенной переписки и RedPhone для осуществления защищенных голосовых вызовов по сети Интернет. С самого начала разработчики позиционировали своё программное обеспечение как конфиденциальное и способное защитить личные данные от атак посредника путём сквозного шифрования. В 2013-ом году TextSecure и RedPhone были объединены в одно приложение - Signal. Одноименное приложение анонсировали для iOS.

Разработчиком приложений выступает компания Open Whisper Systems. Она имеет свой репозиторий на портале GitHub [10], куда выкладывает все исходные коды своих проектов. Это делается для того, чтобы сообщество независимых разработчиков могло инспектировать код на наличие уязвимостей. Также можно собрать приложение из исходных кодов и проверить, отличается ли оно от того, что распространяется с помощью официальных магазинов приложений.

Для шифрования используется Signal Protocol, общие принципы которого описаны выше в разделе 2.2.1 Раньше этот протокол носил название TextSecure Protocol по понятным причинам. Его особенность заключается в том, что он совмещает в себе положительные стороны протоколов OTR и SCIMP, а также избавлен от их недостатков.

OTR (Off-the-Record Messaging Protocol) - протокол шифрования данных при обмене мгновенными сообщениями, представленный в 2004-ом году. К плюсам данного протокола можно отнести то, что в нем используются временные ключи шифрования. Это позволяет быть уверенным в том, что при потери секретных ключей прошлая переписка не будет скомпрометирована. Также стоит отметить то, что сообщения не подписываются автором или адресатом, и при перехвате сообщения злоумышленником будет невозможно доказать причастность кого-либо к этим сообщениям. К сожалению, есть у этого протокола и минусы, которые несколько затрудняют его использование на мобильных устройствах. Согласно протоколу [11] для начала обмена данными два пользователя должны обменяться ключами. Для этого им надо находится в самом приложении. В добавление, с каждым новым сообщением отправляется новый ключ, но, чтобы выработать новый ключ, надо, чтобы собеседник подтвердил предыдущий. Другими словами, у каждого сообщения будет свой временный ключ шифрования только в том случае, если пользователи будут обмениваться по одному сообщению строго друг за другом.

SCIMP (Silent Circle Instant Message Protocol) - протокол шифрования, разработанный компанией Silent Circle и использующийся в одноименном приложении. Продукты этой компании также рекомендует [12] Эдвард Сноуден. Согласно этому протоколу, каждый новый ключ получается итерацией из предыдущего. Такая система обеспечивает новый ключ шифрования для каждого нового сообщения вне зависимости от того, в какой последовательности и что отправляется. Это же и является недостатком данного протокола. Во-первых, при использовании асинхронных процессов в мобильных операционных системах может так получится, что сначала будет доставлено сообщение под номером 6, а только после него сообщение под номером 5, что будет неправильным. Во-вторых, все ключи получаются связанными и, таким образом, проблема с одним временным ключом будет влиять на всю сессию.

Signal Protocol избавлен от недостатков благодаря тому, что все необходимые ключи: и долговременные, и одноразовые - хранятся на сервере. Это позволяет добиться легкости использования и быстроты действия.

В целом, мессенджер Signal обладает всеми плюсами рассмотренного выше WhatsApp, при этом имеет открытый исходный код. Открытость исходного кода в свою очередь позволяет говорить о действительно надежных методах шифрования, проверенных независимыми экспертами.

# **2.2.4 Telegram**

Данное приложение создано Павлом Дуровым, основателем социальной сети "ВКонтакте". С самого создания разработчики говорили о том, что отличительной чертой данного мессенджера является наличие шифрования. Чаты в этой программе делятся на обычные и секретные. Далее я буду рассматривать секретные чаты, так как только они в данной программе поддерживают сквозное шифрование.

Для шифрования данных разработчики используют протокол MTProto. Рассмотрим протокол шифрования данных в секретных чатах, который приведенный в документации [13]:

) Инициатор запускает метод *messages. getDhConfig*, чтобы сгенерировать параметры для протокола Диффи-Хеллмана: простое число *p* и элемент высокого порядка *g*.

2) Инициатор вычисляет случайное 2048-битное число *a*, вычисляет *g\_a: = pow (g, a) mod dh\_prime* и запускает *messages. requestEncryption*, чтобы отправить запрос получателю.

) Получатель принимает запрос, обновляет параметры для протокола Диффи-Хеллмана, вычисляет случайное 2048-битное число *b* по тем же правилам, что инициатор вычислил *a*.

) После принятия запроса получатель вычисляет конечный общий ключ по формуле *key = (pow (g\_a, b) mod dh\_prime),* затем вычисляет *g\_b: = pow (g, b) mod dh\_prime* и запускает *messages. acceptEncryption*.

) После получение *g\_b* инициатор также вычисляет конечный общий ключ по формуле *key = (pow (g\_b, a) mod dh\_prime).*

) Далее с помощью SHA-1 (алгоритм криптографического хеширования) вычисляются ключи шифрования:

a) Вычисляется *msg\_key* путем выбора 128 бит из SHA-1 от текста сообщения;

*b) x=0*;

*c) sha1\_a = SHA1 (msg\_key + substr (key, x, 32));*

*d) sha1\_b = SHA1 (substr (key, 32+x, 16) + msg\_key + substr (key, 48+x, 16));*

*e) sha1\_с = SHA1 (substr (key, 64+x, 32) + msg\_key);*

*f) sha1\_d = SHA1 (msg\_key + substr (key, 96+x, 32));*

*g) aes\_key = substr (sha1\_a, 0,8) + substr (sha1\_b, 8,12) + substr (sha1\_c, 4,12);*

*h) aes\_iv = substr (sha1\_a, 8,12) + substr (sha1\_b, 0,8) + substr (sha1\_c, 16,4) + substr (sha1\_d, 0,8).*

Данные шифруются 256-битным ключом *aes\_key* и 256-битным вектором инициализации *aes\_iv*, используя алгоритм шифрования AES-256 в режиме расширения неопределённого искажения (infinite garble extension, IGE).

) Отпечаток ключа шифрования *key\_fingerprint* и *msg\_key* добавляются в начало массива байтов, после чего запускается *messages. sendEncrypted*.

8) Для расшифровки алгоритм из пункта 6 воспроизводится в обратном порядке.

Новые ключи в данном протоколе формируются в двух случаях: если старыми были зашифрованы 100 сообщений или если старым ключам больше недели и с помощью них зашифровали хотя бы одно сообщение. Для формирования новых ключей существует отдельный протокол, описанный на сайте разработчиков [14] и мало чем отличающийся от приведённого выше.

Стоит отметить полезную особенность секретных чатов - в их настройках можно указать время автоматического удаления сообщений. Например, поставить таймер на 5 секунд и тогда каждое сообщение будет автоматически удаляться через 5 секунд после доставки.

Исходные тексты клиентской части данного приложения представлены на портале GitHub [15]. Таким образом, клиентскую часть также могут инспектировать независимые эксперты. Исходные коды серверной части остаются закрытыми и публиковаться не собираются.

Из всего вышесказанного можно сделать вывод о том, что секретные чаты в мессенджере Telegram предоставляют возможность организовать действительно защищенный канал связи.

# **2.2.5 Antox**

Это мессенджер с открытым исходным кодом [16], который работает на базе протокола Tox. Главной особенностью данного протокола является его распределенность, то есть, пользователи соединяются напрямую (peer-to-peer) друг с другом, минуя какие-либо централизованные серверы. Сам протокол устроен так, что при каждом соединении используется сквозное шифрование. В общих словах, протокол разрабатывался как раз для осуществления обмена данными (мгновенными сообщениями, мультимедиа файлами, видеозвонками) между пользователями таким образом, чтобы никто не мог шпионить за этими данными. Мессенджер Antox является лишь реализацией данного протокола для операционной системы Android, поэтому далее я буду говорить только о самом протоколе.

Первая реализация протокола появилась в 2013-ом году на портале GitHub [17]. Клиенты для работы с данным протоколом начали появляться позже и независимо от разработчиков самого протокола. Вообще, все разработки, связанные с данным протоколом, осуществляются независимыми программистами, а не какими-либо компаниями. Это связано с позиционированием протокола как свободное программное обеспечение, защищенное от бэкдоров (уязвимостей, специально созданных разработчиками для доступа к личной информации клиентов) и шпионажа за пользователями. Вся коммуникация, отправка сообщений и обновление статусов пользователей поддерживается за счет распределенных хеш-таблиц (Distributed Hash Table, DHT) и осуществляется без центрального сервера. По такому же принципу работают, например, сети BitTorrent.

Для шифрования трафика используется библиотека NaCl. Создателем этой библиотеки является Даниэль Бернштейн (Daniel J. Bernstein), больше известный как создатель qmil, Salsa20, Poly1305 и Curve25519.

При первом запуске клиента вызывается функция *crypto\_box\_keypair ()* [18], которая генерирует пару ключей по 32 байта каждый с помощью эллиптической кривой Curve25519. Публичный ключ при этом становится идентификационным номером пользователя (*client\_id*). Чтобы добавить к себе контакт, надо добавить в список контактов его публичный ключ. Для шифрования данных используется функция *crypto\_box ():*

1) Общий ключ вырабатывается из ключа получателя, секретного ключа отправителя и случайного 24-байтного числа;

2) После этого данные шифруются алгоритмом поточного шифрования Salsa20, который использует такие операции как сложение 32-битных чисел, побитовое сложение по модулю 2 и сдвиги битов;

3) Чтобы пометить принадлежность пакетов используется алгоритм Poly1305.

Из минусов данного мессенджера и протокола стоит отметить большие объёмы загружаемых и отправляемых данных, даже если приложение почти не используется. Это связано с распределенной структурой протокола и использованием распределенных хеш-таблиц. Но и благодаря этому же достигается высокая степень защиты от атак посредника.

# **3. Разработка программного средства, построенного на отечественных алгоритмах шифрования**

В качестве результата проведенного исследования автор предоставляет описание мобильного приложения для безопасного обмена данными, а также пример его программной реализации.

# **3.1 Требования к эталонному приложению**

# **1.1 Архитектура приложения**

Большинство провайдеров услуг по обмену информацией по сети Интернет предлагает передавать данные через их серверы. То есть, они предоставляют клиент-серверную архитектуру. Такую ситуацию можно видеть и в программах, рассмотренных в разделе 2. Если говорить о безопасности передачи данных, то данная структура изначально представляет угрозу, если пользователь не доверяет разработчикам приложения, так как эта информация может сохраняться, изменяться, передаваться третьим лицам.

Эту проблему можно решить, организовав соединение типа точка-точка (peer-to-peer) между абонентами-пользователями, как это сделано в протоколе Tox. Такое соединение позволяет передавать данные напрямую между абонентами, что значительно снижает вероятность доступа к этому трафику третьими лицами. Вопрос лишь в том, как обменяться данными для подключения друг к другу, ведь когда у приложения клиент-серверная архитектура, клиенты просто подключаются к серверу, который всегда расположен по одному и тому же адресу.

Есть разные способы обмена контактной информацией между клиентами: можно как передать данные в реальном мире, так и отправить электронным письмом. Подойдет для этого и сигнальный сервер, который будет реализовывать обмен контактными адресами абонентов. Наличие такого сервера опять же позволяет провести на коммуникации разного рода хакерские атаки, но уже не столь значительные как при клиент-серверной архитектуре. С другой стороны, при применении такого решения пользователи не будут передавать большие объёмы информации, как это реализовано в протоколе Tox.

При изучении вопроса соединения двух абонентов напрямую через мобильные средства связи более детально возникают трудности, связанные с применением NAT’ов мобильными оператора при предоставлении доступа к сети Интернет.

Network Address Translation (NAT) - механизм, который преобразовывает IP-адреса пакетов в сетях TCP/IP. Он позволяет существенно сэкономить на количестве выдаваемых IP-адресов, что очень важно, пока используется IPv4. Все мобильные сети построены с помощью NAT’ов и устройства, которые подключены к таким сетям имеют локальный IP-адрес и глобальный.

Всего существует 4 типа построения NAT [19]:

1) Cone NAT.

Внутренний IP-адрес (192.168.0.5: 1234) проецируется на внешний (1.1.1.3: 5678). Пакет с адреса 192.168.0.5: 1234 будет отправлен через 1.1.1.3: 5678. Пакет из внешней сети, посланный на адрес 1.1.1.3: 5678, будет перенаправлен на 192.168.0.5: 1234.

2) Address-Restricted cone NAT.

Внутренний IP-адрес (192.168.0.5: 1234) проецируется на внешний (1.1.1.3: 5678). Пакет с адреса 192.168.0.5: 1234 будет отправлен через 1.1.1.3: 5678. Пакет из внешней сети (1.1.1.50: 3456), отправленный на 1.1.1.3: 5678 будет доставлен на 192.168.0.5: 1234 только в том случае, если с адреса 192.168.0.5: 1234 уже посылались пакеты в эту внешнюю сеть (1.1.1.50).

) Port-Restricted cone NAT.

Внутренний IP-адрес (192.168.0.5: 1234) проецируется на внешний (1.1.1.3: 5678). Пакет с адреса 192.168.0.5: 1234 будет отправлен через 1.1.1.3: 5678. Пакет из внешней сети (1.1.1.50: 3456), отправленный на 1.1.1.3: 5678 будет доставлен на 192.168.0.5: 1234 только в том случае, если с адреса 192.168.0.5: 1234 уже посылались пакеты в эту внешнюю сеть по этому же порту (1.1.1.50: 3456).

) Symmetric NAT.

При отправлении каждого пакета данных внутренний адрес будет меняться на внешний. Но если у предыдущих типов внешний адрес: порт были всегда одинаковы, то при так типа NAT’а, порт у каждого пакета будет разный. То есть, обратно информацию посылать будет нельзя.

К сожалению, Symmetric NAT не позволяет соединить напрямую абонентов, поэтому пользователи с таким типом сети в любом случае будут общаться с помощью сервера. Что касается остальных типов NAT, то все ограничения обходятся с помощью STUN-серверов, которые позволяют узнать нужные IP-адреса, порты и соединиться напрямую.

Стоит также помнить о том, что при передаче информации, пользователь может менять своё местоположение. При этом его телефон будет отключаться от одной сотовой вышки и подключатся к другой, что в свою очередь повлечет за собой смену и внешнего, и внутреннего IP-адресов пользователя. Это в свою очередь повлечет за собой разрыв прямого соединения.

Принимая во внимание всё вышесказанное, предлагается такая архитектура приложения. Мобильный клиент с помощью STUN-сервера контролирует свои IP-адреса, отправляет их на сервер приложения и обновляет при смене. Таким образом возможно будет решить проблему с разорванным соединением, а также будет отображаться статус собеседника (в сети, не в сети). Сервер в свою очередь хранит идентификаторы пользователей и их корректные IP-адреса. При создании соединения двух пользователей, сервер должен отправить им адреса друг друга, чтобы они смогли соединиться напрямую, а также должен обновлять адреса для поддержания соединения.

Дополнительным плюсом такой архитектуры является отсутствие необходимости передавать все данные сервером. То есть, затраты на сервер будут существенно меньше, при этом приложение проще масштабировать.

# **3.1.2 Шифрование**

В качестве алгоритма шифрования предлагается использовать гибридную схему, которая имеет много общих черт со всеми перечисленными протоколами в разделе 2.

Изначально существовало два типа криптографических систем. Первый тип - симметричные криптосистемы (симметричное шифрование). Это система, в которой и шифрование, и расшифровка происходят с применением одного и того же ключа, который должен обеими сторонами держаться в строгом секрете. К плюсам данных систем можно отнести скорость работы, большую изученность. Главным недостатком данных систем является сложность обмена ключами. Второй тип систем, который появился значительно позже - криптографические системы с открытым ключом (асимметричное шифрование). В таких системах есть два ключа: открытый и закрытый. Открытым ключом шифруется сообщение, он может лежать в открытом доступе, закрытый же ключ должен находится в секрете, так как им сообщения можно расшифровать. Главным плюсом данных систем является отсутствие необходимости обмениваться единственным секретным ключом. Недостатками же являются больший размер ключа при той же криптостойкости, большая требовательность к ресурсам системы и низкая скорость шифрования.

На данный момент существуют гибридные схемы шифрования, которые представляют собой смесь симметричного шифрования и асимметричного.

Подобные схемы лишены главного недостатка симметричного шифрования - необходимости обмена общим ключом. При этом они имеют такую же скорость шифрования.

На рисунке 1 можно увидеть основной принцип действия таких схем.



Рис. 1. Гибридная схема шифрования

В целом же всё происходит таким образом. Допустим, Алиса хочет передать Бобу сообщение. У Алисы и Боба есть по паре ключей: открытый и закрытый. Алиса берет открытый ключ Боба, и генерирует с помощью него сеансовый ключ. После этого Алиса с помощью какого-либо алгоритма симметричного шифрования и выработанного сеансового ключа шифрует сообщение и передает его Бобу. Боб с помощью сеансового ключа и своего секретного ключа расшифровывает сообщение.

В качестве алгоритма асимметричного шифрования был выбран алгоритм, описанный в стандарте ГОСТ Р 34.10-2012 [20]. Данный алгоритм основан на эллиптических кривых. Считается, что задача дискретного логарифма на эллиптической кривой является более трудной по сравнению с задачей дискретного логарифма в конечных полях.

Для максимальной стойкости алгоритма необходимо выбрать подходящую эллиптическую кривую. Кривая описывается уравнением *y2 = x3 + ax + b*. Для вычисления криптостойкости необходимо знать порядок группы точек эллиптической кривой *p* (простое число). Также необходимо подобрать точку *P (xp,yp),* принадлежащую кривой с большим порядком *q* для того, чтобы, умножая эту точку на числа, которые меньше её порядка, получать разные точки. Если выполнено условие *4a3 + 27b2 = 0*, то кривая называется сингулярной и такую кривую нельзя использовать в эллиптической криптографии. Методы решения логарифма на эллиптической кривой имеют сложность . Учитывая то, что в программной реализации используется *q = 2384*, можно говорить о сложности, равной 2192.

В качестве алгоритма симметричного шифрования был выбран блочный шифр под названием "Кузнечик", описанный в стандарте ГОСТ Р 34.12-2015 [21]. В этом алгоритме блоки имеют длину 128 бит, а ключ - 256 бит. Для генерации ключа используется сеть Фейстеля.

При шифровании информации происходит 10 раундов с преобразованиями. В каждом раунде сначала 128-битный входной вектор побитно складывается с раундовым ключом, после этого к каждому 8-битному подвектору применяется фиксированная подстановка, а после этого применяется линейный регистр сдвига, который движется 16 раз и реализуется над полем Галуа.

В университете Канады Concordia Institute for Information Systems Engineering (CIISE) исследовали [22] данный алгоритм и пришли к выводу, что он имеет временную сложность равную 2140.

Таким образом, применяя гибридный алгоритм, составленный из двух, описанных выше, на практике, получается надежная криптосистема с большой стойкостью.

Также необходимо отметить то, что шифроваться должны как каналы между пользователями, так и канал соединения пользователя с сервером. Для этого у сервера тоже должны быть открытый и закрытый ключи.

Сам процесс шифрования и расшифровки представляет собой следующее. Открытый ключ представляет собой точку на эллиптической кривой. При передаче данных о точке координаты x и y "склеиваются", образуя длинную строку открытого ключа. Закрытый ключ представляет собой большое целое число.

Перед отправкой сообщения собеседнику генерируется случайное большое число *k*. Затем оно умножается на *P* (точка с большим порядком) и на *Y* (открытый ключ собеседника). То есть, *U = kP* и *V = kY*. Для применения блочного алгоритма шифрования программа берет *xv* координату точки *V* по модулю 2256. Таким образом, мастер-ключ будет вычисляется так *K = xv (mod 2256).* При этом *U* является сеансовым ключом, которые отправляется вместе с сообщением (шифротектом).

Собеседник получает *U* и шифротекст. Для расшифровки ему необходимо вычислить ключ из *U*. Это возможно сделать, умножив закрытый ключ *x* на сеансовый ключ *U*: *xU = xkP = kxP = kY = V*. Координата *xv* взятая по модулю 2256 и будет являться общим ключом для расшифровки сообщения.

# **3.2 Описание программной реализации**

Кроме исследования и определения общих принципов построения приложения для безопасного обмена частной информацией, также был разработан пример программной реализации для операционной системы Android. Эта операционная система основана на ядре Linux и собственной реализации виртуальной машины Java от Google. Первая версия появилась в 2008 году и с тех пор ОС Android быстро завоевала рынок.

По данным международной исследовательской и консалтинговой компании International Data Corporation (IDC) [23] на август 2015-го года 82.8% смартфонов работает на мобильной операционной системе Android. Это позволяет сказать, что данная операционная система является безоговорочным лидером рынка, и, если разработчик приложения желает охватить как можно большее количество пользователей, то в первую очередь стоит разработать приложение именно для Android.

Касаемо минимальной версии Android, которую поддерживало бы приложение, была выбрана 4.1 Согласно официальной статистики разработчиков операционной системы на 2 мая 2016 года [24], при выборе данной версии разработанное приложение будет поддерживать 95.7% устройств (включая и смартфоны, и планшеты).

Так как программа была написана для ОС Android, выбор языка программирования пал на Java. Среда разработки программного обеспечения - Android Studio 1.5.1 Build #AI-141.2456560, JRE: 1.8.0\_77-b03 amd64.

Общий принцип действия приложения такой:

) Пользователь заходит в приложение и видит кнопки управления соединением, выбранный контакт для соединения, свой открытый ключ, который он может скопировать и передать кому-либо и окно чата (рис.2).



Рис. 2. Начальный экран приложения

2) Чтобы соединиться с контактом Denis Smartphone необходимо нажать кнопку "START". Это же действие должен выполнить собеседник на своем мобильном устройстве. После нажатия кнопки на сигнальный сервер посылаются данные об IP-адресе устройства со специальной строкой, которая состоит из открытых ключей двух собеседников, расставленных в порядке возрастания. Это нужно для того, чтобы контактные данные собеседников "нашли" друг друга на сигнальном сервере (рис.3, рис.4).



Рис. 3. Ожидание соединения на устройстве 1



Рис. 4. Ожидание соединения на устройстве 2

) После этого на одном из устройств необходимо нажать кнопку "CALL". После этого соединение будет установлено (рис.5, рис.6, рис.7, рис.8, рис.9, рис.10).



Рис. 5. Соединение установлено на устройстве 1



Рис. 6. Соединение установлено на устройстве 2



Рис. 7. Тест чата на устройстве 1



Рис. 8. Тест чата на устройстве 2



Рис. 9. Повторный тест чата на устройстве 1



Рис. 10. Повторный тест чата на устройстве 2

) Чтобы разорвать соединение необходимо нажать на клавишу "HANG UP" (рис. 11).



Рис. 11. Соединение разорвано

5) Для добавления контакта необходимо нажать на клавишу "ADD". После этого необходимо ввести имя контакта (рис. 12) и его открытый ключ (рис. 13).



Рис. 12. Добавление имени контакта



Рис. 13. Добавление открытого ключа контакта

К сожалению, в данном приложении не был реализован механизм постоянной поддержки связи за счет сигнального сервера при смене IP-адреса, а также не было применено шифрование для канала клиент-сервер. То есть, информация об IP-адресах передается без шифрования, шифруется только канал для передачи сообщений между клиентами.

# **3.2.1 Архитектура**

Из-за рассмотренных в разделе 3.1.1 сложностей при реализации предложенной архитектуры было принято решение использовать проект WebRTC. Это cамый бурно развивающийся peer-to-peer проект последнего времени с открытым исходным кодом [25], предназначенный для потоковой передачи данных между браузерами, который развивают такие гиганты IT-индустрии как Google, Mozila и Ericsson.

Так как это проект с открытым исходным кодом, его портировали [26] на все остальные платформы, в том числе и на операционную систему Android. Это позволило мне использовать существующие библиотеки для организации peer-to-peer связи.

Использовать данные библиотеки удобно из-за того, что в стандартах для клиентских приложений всё описано и при желании их можно будет реализовать и на других платформах. Единственное, что не прописано в стандарте - серверная часть. В качестве сигнального сервера может быть любая машина на любой технологии.

В качестве STUN-сервера используется сигнальный сервер от Google stun. l. google.com: 19302. Так как это STUN сервер, то, если абонент будет за Symmetric NAT, он не сможет воспользоваться данным приложением. Тем не менее, при желании можно добавить адрес TURN сервера, который будет использоваться для передачи данных, если не будет возможности соединить абонентов напрямую.

В качестве сигнального сервера, я использую проект Greta и его API для Android с открытым исходным кодом [27]. Данный проект существует именно для поддержания децентрализованной передачи информации. Когда два пользователя хотят связаться, на сигнальном сервере создается "комната", название которой состоит из идентификационных номеров абонентов, расположенных в порядке возрастания. С помощью этой "комнаты" абоненты обмениваются информацией друг о друге и уже соединяются напрямую, минуя какие-либо другие сервер.

Также стоит упомянуть о структуре самого приложения для ОС Android. При первой установке создается база данных SQLite. Такие базы данных часто используются в мобильных приложениях, так как являются встраиваемыми. При установке создается таблица с контактами. Первая запись этой таблицы используется для хранения собственных ключей пользователя. Остальные записи включают в себя имя контакта и его идентификационный номер (открытый ключ). Для поддержания работоспособности базы данных, её создания и обновления был разработан класс *DBHelper. java*.

История переписки не сохраняется во внешней памяти. Она существует только пока работает приложение. После выхода, переписка удаляется.

# **3.2.2 Шифрование**

Как и было сказано выше, при установке приложения создаются закрытый и открытый ключи пользователя. Чтобы соединиться с собеседником необходимо, чтобы он присутствовал в списке контактов, где должен храниться его открытый ключ. Открытый ключ представляет собой точку на эллиптической кривой. При передаче данных о точке координаты *x* и *y* "склеиваются", образуя длинную строку открытого ключа. Закрытый ключ представляет собой большое целое число.

Инициатор пишет сообщение и нажимает кнопку отправить. При этом приложение шифрует сообщение и прикрепляет к началу шифротекста временный, сеансовый ключ. Получатель принимает сообщение и с помощью временного ключа расшифровывает его.

При выборе эллиптической кривой использовался документ [28], опубликованный Национальным Институтом Стандартов и Технологий (National Institute of Standards and Technology, NIST) в США, который рекомендует некоторые кривые к использованию в криптографии. В приложении используется кривая Curve P-384.

Для работы с большими числами при программировании на языке Java используется встроенный класс *BigInteger*. Также для работы с эллиптическими кривыми были разработаны классы *ECPoint. java* и *ECCrypto. java*. *ECPoint. java* используется для реализации операций на эллиптических кривых. В свою очередь *ECCrypto. java* используется для реализации шифрования на эллиптических кривых.

При осуществлении блочного шифрования используется таблица, которая служит для реализации операции многочленов в конечно поле из восьми элементов. Это таблица хранится отдельно в ресурсных файлах и содержит 65 536 шестнадцатеричных числа. Для развертки мастер-ключа используются 32 итерационные константы. Они были рассчитаны заранее и хранятся в коллекции *ArrayList<int [] >*.

# **ЗАКЛЮЧЕНИЕ**

В ходе выпускной квалификационной работы были исследованы средства обеспечения безопасности частной информации на мобильных устройствах и разработано средство для безопасной передачи данных между мобильными устройствами по сети Интернет.

В первой главе были рассмотрены общие принципы безопасности данных на мобильных устройствах, мобильные антивирусы и представлены некоторые советы для улучшения безопасности личных данных. Во второй главе речь шла о приложениях для обмена данными по сети Интернет с помощью мобильных устройств. Все мессенджеры были рассмотрены с точки зрения обеспечения безопасности передаваемой информации и у каждого решения есть свою плюсы и минусы. В последней главе были представлены как результат этой работы эталонная реализация защищенного канала связи между мобильными устройствами, а также реальное приложение, которое разрабатывалось по архитектуре эталонной.

В качестве дальнейших исследований возможна доработка существующей программной реализации до эталонной, а также доработка и самой эталонной архитектуры. В современном мире появляются всё новые способы атак, поэтому и средства защиты должны развиваться.

Используя это исследование, любой человек, который не доверяет сторонним решениям и разработчикам, способен при необходимости построить свой собственный защищенный канал для передачи информации между мобильными устройствами.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Пронин Д.С. Реализация схемы асимметричного шифрования. НИУ ВШЭ, 2015.

2. Колпаченко С.Ю. Реализация перспективного отечественного алгоритма блочного шифрования. НИУ ВШЭ, 2015.

. Пронин Д.С. Исследование мобильных средств обеспечения безопасности частной информации // Ежегодная научно-техническая конференция студентов, аспирантов и молодых специалистов МИЭМ НИУ ВШЭ: Материалы конференции. М.: МИЭМ НИУ ВШЭ, 2016. C.304.

4. Kaspersky Lab. "Consumer Security Risks Survey", 2015. URL: https: // press. kaspersky.com/files/2015/08/Kaspersky\_Lab\_Consumer\_Security\_Risks\_Survey\_2015\_ENG. pdf <https://press.kaspersky.com/files/2015/08/Kaspersky\_Lab\_Consumer\_Security\_Risks\_Survey\_2015\_ENG.pdf> (дата обращения: 20.05.2016).

. Electronic Frontier Foundation. "The Problem with Mobile Phones", 2015. URL: https: // ssd. eff.org/en/module/problem-mobile-phones <https://ssd.eff.org/en/module/problem-mobile-phones> (дата обращения: 20.05.2016).

6. Wikipedia. "Comparison of antivirus software", 2015. URL: https: // en. wikipedia.org/wiki/Comparison\_of\_antivirus\_software <https://en.wikipedia.org/wiki/Comparison\_of\_antivirus\_software> (дата обращения: 20.05.2016).

7. Эксперты назвали самый популярный мессенджер в России // РБК. 2016. URL: <http://www.rbc.ru/technology\_and\_media/18/01/2016/569cddd29a794722c534df2c> (дата обращения: 22.05.2016).

8. WhatsApp Encryption Overview. Technical white paper. 2016. URL: https: // www.whatsapp.com/security/WhatsApp-Security-Whitepaper. pdf <https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf> (дата обращения: 23.05.2016).

9. Dell Cameron. Edward Snowden tells you what encrypted messaging apps you should use // The Daily Dot. 2015. URL: http://www.dailydot.com/politics/edward-snowden-signal-encryption-privacy-messaging/ (дата обращения: 22.05.2016).

10. Open Whisper Systems // GitHub. URL: https: // github.com/WhisperSystems <https://github.com/WhisperSystems> (дата обращения: 23.05.2016).

. Advanced cryptographic ratcheting. 2013. URL: https: // whispersystems.org/blog/advanced-ratcheting/ <https://whispersystems.org/blog/advanced-ratcheting/> (дата обращения: 23.05.2016).

12. Баранский С. Продукты, которые рекомендует Эдвард Сноуден вместо опасных, по его мнению, Facebook, Google и Dropbox. URL: <http://lifehacker.ru/2014/10/13/snowden-new-yorker/> (дата обращения: 24.05.2016).

13. Secret chats, end-to-end encryption URL: https: // core. telegram.org/api/end-to-end <https://core.telegram.org/api/end-to-end> (дата обращения: 24.05.2016).

. Re-keying protocol. URL: https: // core. telegram.org/api/end-to-end/pfs <https://core.telegram.org/api/end-to-end/pfs> (дата обращения: 24.05.2016).

. Telegram // GitHub. URL: https: // github.com/DrKLO/Telegram <https://github.com/DrKLO/Telegram> (дата обращения: 24.05.2016).

. Antox // GitHub. URL: https: // github.com/Antox/Antox <https://github.com/Antox/Antox> (дата обращения: 24.05.2016).

. Toxcore // GitHub. URL: https: // github.com/irungentoo/toxcore <https://github.com/irungentoo/toxcore> (дата обращения: 24.05.2016).

. Crypto. URL: https: // github.com/irungentoo/toxcore/blob/master/docs/updates/Crypto. md <https://github.com/irungentoo/toxcore/blob/master/docs/updates/Crypto.md> (дата обращения: 24.05.2016).

19. Типы Network Address Translation (NAT). 2009. URL: <http://aoz.com.ua/2009/01/26/nat-types/> (дата обращения: 24.05.2016).

. ГОСТ Р 34.10-2012. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. М.: Стандартинформ, 2013.

. ГОСТ Р 34.12-2015. Информационная технология. Криптографическая защита информации. Блочный шифры. М.: Стандартинформ, 2015.

22. Riham AlTawy and Amr M. Youssef. A Meet in the Middle Attack on Reduced Round Kuznyechik. Concordia University. URL: https: // eprint. iacr.org/2015/096. pdf <https://eprint.iacr.org/2015/096.pdf> (дата обращения: 24.05.2016).

. Smartphone OS Market Share, 2015 Q2. URL: http://www.idc.com/prodserv/smartphone-os-market-share. jsp <http://www.idc.com/prodserv/smartphone-os-market-share.jsp> (дата обращения: 18.05.2016).

24. Dashboards. URL: https: // developer. android.com/about/dashboards/index.html <https://developer.android.com/about/dashboards/index.html> (дата обращения: 19.05.2016).

. WebRTC. Source code. URL: https: // chromium. googlesource.com/external/webrtc <https://chromium.googlesource.com/external/webrtc> (дата обращения: 24.05.2016).

. WebRTC-Two-Androids-Demo // GitHub. URL: https: // github.com/leonardogcsoares/WebRTC-Two-Androids-Demo <https://github.com/leonardogcsoares/WebRTC-Two-Androids-Demo> (дата обращения: 24.05.2016).

. Signaling-API-for-Greta // GitHub. URL: https: // github.com/leonardogcsoares/Signaling-API-for-Greta <https://github.com/leonardogcsoares/Signaling-API-for-Greta> (дата обращения: 24.05.2016).

. Recommended elliptic curves for federal government use. 1999. URL: http://csrc. nist.gov/groups/ST/toolkit/documents/dss/NISTReCur. pdf <http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf> (дата обращения: 24.05.2016).