

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление 09.03.01 «Информатика и вычислительная техника»

подготовки:

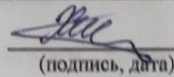
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ
ЗАЩИЩЕННОГО ОБМЕНА ИНФОРМАЦИЕЙ С
ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ MESH-СЕТЕЙ

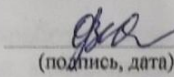
Студент


(подпись, дата)

Ханбекова Елена Валентиновна

29.05.2023

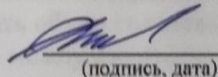
Руководитель


(подпись, дата)

Юрова Ольга Викторовна

10.06.2023

Нормоконтролёр

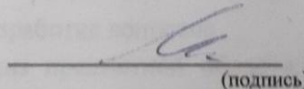

(подпись, дата)

Бычков Андрей Станиславович

10.06.2023

Работа допущена к защите (протокол зас. кафедры от 02.06.2023 г. № 14)

Заведующий кафедрой


(подпись)

М.А. Митрохин

(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

14.06.2023 г. № 16)

Секретарь ГЭК



Е.И. Гурин

(подпись)

(фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

«28» 04 2023 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Ханбекова Елена Валентиновна гр. 19ВВП2 факультета вычислительной техники

2. Тема работы «Программное средство для защищенного обмена информацией с использованием технологии mesh-сетей»

Тема утверждена приказом ПГУ №458/о от "26" апреля 2023г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Юрова О. В.

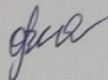
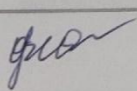
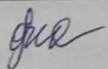
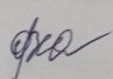
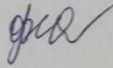
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки – создание мобильного приложения под управлением ОС Android. Приложение должно обеспечивать обмен голосовыми сообщениями в режиме реального времени и защищенную передачу файлов между смартфонами, на которых запущено данное программное средство. Организация соединения между устройствами должна быть организована с использованием технологии mesh-сетей. Данное приложение должно выдавать информацию о своем текущем состоянии, а также поддерживать поиск новых пользователей и создание соединения с ними.

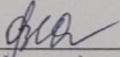
5. Перечень подлежащих разработке вопросов

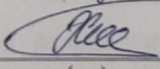
Требуется произвести анализ предметной области: изучить существующие аналоги, изучить актуальные способы создания одноранговых соединений, изучить средства проектирования мобильных приложений, определить архитектуру приложения.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10 %	26.04.23 - 29.04.23	
2.	Выбор архитектуры приложения и технологий, применяемых при проектировании и разработке	15 %	30.04.23 - 04.05.23	
3.	Разработка разметки	20 %	05.05.23 - 09.05.23	
4.	Разработка логики приложения	25 %	10.05.23 - 21.05.23	
5.	Тестирование и отладка	15 %	22.05.23 - 24.05.23	
6.	Выполнение задания по БЖД	5 %	25.05.23 - 27.05.23	
7.	Оформление выпускной работы	10 %	28.05.23 - 31.05.23	

Дата выдачи задания "26" апреля 2023г.

Руководитель бакалаврской работы  Юрова О.В.
(подпись, дата)

Задание к исполнению принял студент  Ханбекова Е.В.
(подпись, дата)

Работу к защите допустить
Декан факультета  13.06.2023 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 120 л., 20 рис., 5 источ., 3 прил.

ANDROID, WI-FI DIRECT, JAVA, ОДНОРАНГОВАЯ СЕТЬ, ШИФРОВАНИЕ

Цель работы – создание программного средства для обмена информацией с использованием технологии *mesh*-сетей, реализующее механизм криптографической защиты информации.

Приложение должно обеспечивать:

- организацию соединения между устройствами с использованием технологии *mesh*-сетей;

- передачу голосовых сообщений в режиме реального времени и файлов между мобильными телефонами, на которых запущено данное программное средство;

- криптографическую защиту информации по алгоритму «Кузнечик».

В ходе работы должны быть реализованы следующие задачи:

- анализ архитектуры *mesh*-сетей;
- разработка приложения;
- отладка и тестирование приложения.

Результатом работы является проект, удовлетворяющий указанным выше требованиям.

В разработке использовался стек технологий, к которому относятся язык программирования *JAVA* и среда разработки *Android Studio* для работы со скриптами.

					ПГУ1.090301.38.001 ПЗ				
		№ докум.	Подпись		Программное средство для защищенного обмена информацией с использованием технологии mesh-сетей Пояснительная записка	Лит.	Лист	Листов	
Разраб	Ханбекова Е.В.								
Провер.	Юрова О.В.						2	120	
Н. Контр.	Бычков А.С.					ФВТ, каф. ВТ гр. 19ВВП2			
Утверд.	Митрохин М.А.								

Содержание

Введение	6
1 Мобильное приложение для обмена голосовыми сообщениями в режиме реального времени и файлами с использованием технологии <i>mesh</i> -сетей	8
1.1 Анализ предметной области	8
1.1.1 Обзор аналогов	10
1.1.2 Выбор среды разработки	12
1.1.3 Выбор коммуникационной технологии	13
1.1.4 Постановка задачи	15
1.1.5 Выводы	16
1.2 Реализация приложения	16
1.2.1 Алгоритм работы приложения	16
1.2.2 Реализация одноранговых соединений с помощью Wi-Fi Direct	17
1.2.3 Реализация алгоритма Push-to-talk	21
1.2.4 Реализация защищенного обмена файлами	24
1.2.5 Реализация криптографической защиты информации	25
1.2.6 Выводы	30
1.3 Тестирование приложения	31
1.3.1 Тестирование приложения	31
1.3.2 Выводы	37
2 Безопасность жизнедеятельности	38
2.1 Требования к помещению	38
2.2 Освещенность рабочего места	39
2.3 Расчет количества светильников в помещении	40

2.4 Требования по организации рабочего места	41
2.5 Выводы	42
Заключение.....	43
Список использованных источников	44
Приложение А.....	45
Приложение Б	49
Приложение В.....	112

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

ГОСТ – государственный стандарт

НПБ – Нормы пожарной безопасности

ПО – программное обеспечение

ПЭВМ – персональная электронно-вычислительная машина

СКЗИ – средство криптографической защиты информации

ЭВМ – электронно-вычислительная машина

BLE (Bluetooth Low Energy) – беспроводная технология Bluetooth с низким энергопотреблением

GPRS (General Packet Radio Service) – пакетная радиосвязь общего пользования

IDE (Integrated Development Enviroment) – интегрированная среда разработки

PTT (Push-to-talk) – стандарт голосовой связи полудуплексного типа, функционирующий с двусторонним радиоинтерфейсом

RSA (Rivest-Shamir-Adleman) – криптографический алгоритм с открытым ключом

SP (Substitution-Permutation) – подстановочно-перестановочная сеть

UML (Unified Modeling Language) – унифицированный язык моделирования

WMN (Wireless Mesh Network) – беспроводная ячеистая сеть

XML (Extensible Markup Language) – расширяемый язык разметки

Введение

В настоящее время каждый человек в течение дня постоянно обменивается информацией с другими людьми, так как у каждого сейчас имеется смартфон, с помощью которого можно звонить или отправлять сообщения. Но сейчас услуги мобильной телефонии и обмена сообщениями теряют свою популярность. Пользователи выбирают мессенджеры, которые используются для частной переписки и публичного оповещения, дополняют новостные ленты онлайн-СМИ, служат для передачи аудио- и видеосообщений. Мессенджеры позволяют быстро обмениваться информацией. Именно поэтому такие приложения стали часто использоваться не только для личной переписки, но и для работы. Однако для функционирования большинства популярных приложений для обмена сообщениями необходим доступ к сети Интернет или услугам провайдера, которые есть далеко не везде, поэтому связь с помощью раций не утратила своей популярности. Так же недостатком сотовой связи является необходимость ожидания ответа абонента. Для оперативной передачи голосовых сообщений могут быть использованы радиосети. Область применения раций достаточно широкая: от активного отдыха до чрезвычайных ситуаций. Радиосети позволяют прослушивать сообщения в режиме реального времени. Современные смартфоны имеют достаточные технические возможности для реализации функционала раций в мобильном приложении. Данная задача может быть решена с помощью реализации технологий *Push-To-Talk* и *mesh*-сетей в мобильном приложении.

Актуальной является проблема конфиденциальности, так как многие мессенджеры являются разработкой зарубежных компаний. Информация, передаваемая с их помощью, может быть перехвачена и отслежена. Поэтому важным становится разработка приложений с определенным уровнем защиты информации. Достичь необходимого уровня защиты информации возможно с помощью шифрования. По мере усиления информатизации различных отраслей экономики РФ соответствующий регулятор вводит требования по защите

каналов связи, причем с применением российских средств криптографической защиты информации. СКЗИ должны использовать отечественные криптоалгоритмы.

					ПГУ1.090301.38.001 ПЗ	Лист
						7
Изм	Лист	№ докум	Подпись	Дата		

1 Мобильное приложение для обмена голосовыми сообщениями в режиме реального времени и файлами с использованием технологии *mesh*-сетей

1.1 Анализ предметной области

Беспроводная *mesh*-сеть (*wireless mesh network, WMN*) — это сеть, включающая в себя различные беспроводные устройства с точками доступа. Каждый клиент в сети также является сервером, передавая данные дальше по сети. Данные сети увеличивают область беспроводного покрытия за счет использования в процессе маршрутизации самих узлов, передающих данные. Так как данная сеть является децентрализованной, передавать данные каждый узел может только соседним узлам. Чаще всего для создания подключения в данных сетях используются интерфейсы *Wi-Fi*, *Wi-Fi Direct* и *Bluetooth*. Это позволяет узлам устанавливать соединения, не имея доступа в Интернет, и даже не пользуясь услуги операторов связи. Архитектура *WMN* представляет собой связанный граф (рисунок 1). Данные передаются от адресата к получателю, проходя через цепочку устройств между ними. Устройства могут рассылать данные либо всем соседним устройствам, либо выбирать конкретные узлы, что уменьшает нагрузку на сеть. Каждый новый клиент сети расширяет ее.

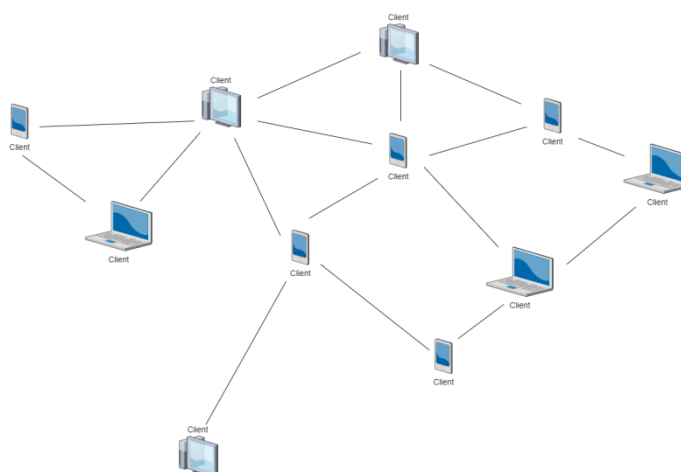


Рисунок 1 – топология *mesh*-сети

Преимущества *mesh*-сети заключаются в следующем:

- масштабируемость сети. Каждое устройство при подключении в сеть автоматически получает всю необходимую информацию о других узлах сети и своей роли в ней;
- самовосстановление сети. В случае выхода из строя какого-то из узлов, сеть способна самостоятельно перенаправить данные по другому пути;
- быстрое и недорогое развертывание. Все узлы в сети являются беспроводными, благодаря чему не требуется дорогостоящая инфраструктура для прокладки кабелей;
- защита информации. *Mesh*-сети в данном вопросе имеют преимущество, так как трафик в таких сетях не может быть перехвачен и проанализирован провайдером;

Так как *mesh*-сети не используют подключение к сети и операторам связи, то трафик не может быть отслежен, но внутри сети необходима защищенная передача сообщений, чтобы при обмене информацией между двумя узлами, данные не были доступны для промежуточных или соседних узлов.

Push-to-talk, или сокращенно *PTT*, — это технология голосовой радиосвязи, основанная на сетях мобильной радиосвязи. Технология реализуется на базе *GPRS* с помощью мобильных операторов или *IP* телефонии. Станции радиосвязи и смартфоны с поддержкой *PTT* используются в корпоративных целях для обеспечения стабильной оперативной связи между пользователями (рисунок 2).

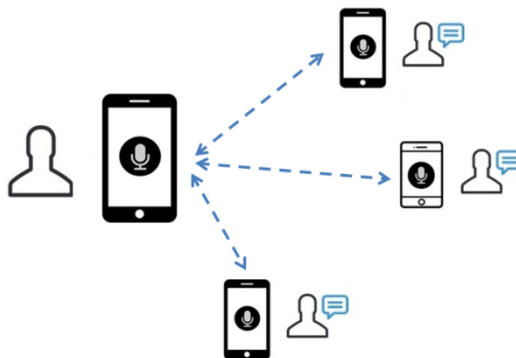


Рисунок 2 – Технология *Push-to-talk*

Для защищенного обмена можно использовать различные методы шифрования. Первый этап, который необходимо проделать, — обмен ключами между будущими сторонами коммуникации. Чтобы быть уверенными в том, что другой участник сети — тот, за кого себя выдает. По этому каналу стороны должны установить общий ключ, а также произвести обмен пробными сообщениями, например подобный рукопожатию протокола *TCP*. Если обе стороны успешно обменялись сообщениями, необходимо сохранить их пары ключей для дальнейшей коммуникации.

1.1.1 Обзор аналогов

На данный момент в мире уже существуют приложения, организующие *mesh*-сети разного назначения. Часто данная технология используется в концепции умного дома, тогда в сети соединяются не только привычные компьютеры и смартфоны, но и бытовые приборы, имеющие возможность подключения к сети. Ниже рассмотрены мобильные приложения, имеющиеся на рынке сейчас.

Berty — это приложение для обмена сообщениями, ориентированное на конфиденциальность, разработанное французской некоммерческой организацией *Berty*.

Функциональные характеристики приложения:

- работает на основе протокола *Berty*;
- подключение к Интернету не требуется благодаря технологии *BLE* и *mDNS*;
- позволяет обмениваться только сообщениями.

Meshenger — это *P2P*-платформа для обмена сообщениями, которая позволяет совершать звонки и видеозвонки между телефонами без Интернета.

Функциональные характеристики приложения:

- для подключения необходимо отсканировать *QR*-код другого пользователя;
- позволяет обмениваться сообщениями, совершать голосовые и

видеозвонки;

- обменивается именем контакта, открытым ключом и *MAC/IP/DNS*-адресом через *QR*-код;

- подключение к сети Интернет не требуется, для соединения используется *Wi-Fi* или *Bluetooth*.

Турфон — мобильное приложение превращает смартфон гида в передатчик, а телефоны туристов — в приемники. Российская компания «Турфон» разработала программное решение, позволяющее организовать потоковое аудиовещание на смартфоны пользователей без подключения к интернету.

Функциональные характеристики приложения:

- подключение к сети Интернет не требуется за счет использования *Wi-Fi*;

- приложение позволяет транслировать звук на неограниченное число устройств с возможностью одновременного вещания на один или несколько разных каналов и с задержкой не более 0,5 секунды.

На основе основных технических характеристик аналогичных приложений была построена таблица (таблица 1.1).

Таблица 1.1 – Сравнение характеристик аналогичных приложений

Название приложения	Преимущества	Недостатки
<i>Berty</i>	Позволяет отправлять текстовые сообщения. Подключение к Интернету не требуется. Соединение по <i>mesh</i> -сети с использованием технологии <i>BLE</i> и <i>mDNS</i> .	Нет возможности обмена голосовыми сообщениями в режиме реального времени.

<i>Meshenger</i>	Позволяет отправлять текстовые сообщения, совершать голосовые и видеозвонки. Подключение к Интернету не требуется. Соединение по <i>mesh</i> -сети.	Необходимо отсканировать QR-код другого пользователя для подключения, то есть необходимо личное присутствие другого пользователя.
Турфон	Организует потоковое аудиовещание с задержкой не более 0,5 секунды. Подключение к Интернету не требуется. Соединение по <i>mesh</i> -сети.	Необходимо отсканировать QR-код организатора для подключения, то есть у сети должен быть организатор, который инициирует связь.

1.1.2 Выбор среды разработки

На данный момент существует огромное количество различных средств разработки приложений на *Android*. На сегодняшний день самыми используемыми средами являются: *Eclipse*, *IntelliJ Idea*, *Android Studio*.

Каждая *IDE* обладает уникальными характеристиками.

Eclipse Java IDE — открытая интегрированная среда разработки на языке программирования *Java*, которая распространяется и поддерживается *Eclipse Foundation*.

Достоинствами *Eclipse* являются:

- наличие документации на русском языке;
- отлично работает на компьютерах с низкой производительностью;
- гибкая настройка под себя благодаря плагинам;
- является бесплатной.

Главным недостатком *Eclipse* является то, что данная среда устарела, и в 2016 году в *Google* отказались от поддержки ее плагинов для *Android*-разработки.

Android Studio — интегрированная среда разработки (IDE) для работы с платформой *Android*, разработанная *Google*. Самым большим преимуществом *Android Studio* является её распространённость на данный момент.

Другими достоинствами *Android Studio* являются:

- наличие удобного *XML* редактора;
- возможность создания проектов разной сложности;
- наличие подробного руководства по использованию *Android Studio*;
- достаточно большая библиотека с готовыми шаблонами и компонентами для разработки ПО;
- встроенный эмулятор;
- весь функционал среды предоставляется бесплатно.

Основным недостатком *Android Studio* является высокое потребление оперативной памяти компьютера.

IntelliJ Idea — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности *Java*, *JavaScript*, *Python*, разработанная компанией *JetBrains*.

Преимуществами *IntelliJ Idea* являются:

- готова к использованию сразу после установки;
- предлагает интеллектуальную помощь в написании кода на многих языках;
- понятный и лаконичный интерфейс;
- поддерживает самые популярные фреймворки тестирования.

Недостатком *IntelliJ Idea* является то, что полный функционал среды доступен только в платной подписке.

Исходя из перечисленных достоинств и недостатков перечисленных сред разработки, было принято решение использовать *Android Studio*.

1.1.3 Выбор коммуникационной технологии

В *mesh*-сетях на данный момент используются несколько технологий создания соединения. Наибольшей популярностью пользуются: *Wi-Fi Direct* и

BLE.

Bluetooth Low Energy (BLE), т.е. *Bluetooth* с низким энергопотреблением, который появился в версии стандарта *Bluetooth* 4.0. Он чаще всего применяется в приложениях, чувствительных к энергопотреблению (например, в устройствах с батарейным питанием) или в устройствах, передающих небольшие объемы данных с большими перерывами между передачами (например, разнообразные сенсоры параметров окружающей среды или управляющие устройства, такие как беспроводные выключатели).

Наиболее важные технические характеристики *BLE* включают в себя:

- используемый частотный диапазон 2.400 - 2.4835 ГГц;
- весь частотный диапазон поделен на 40 каналов по 2 МГц каждый;
- максимальная скорость передачи данных по радиоканалу (начиная с *Bluetooth* версии 5) 2Мбит/с.;
- дальность передачи сильно зависит от физического окружения, а также используемого режима передачи. Например, в режиме большой дальности передачи дальность связи будет выше, а скорость передачи ниже, чем в высокоскоростном режиме. Типичная дальность передачи: 10-30 метров;
- потребление электроэнергии также может изменяться в широких пределах. Оно зависит от реализации устройства, различных параметров протокола и используемого чипсета;
- *BLE* предназначен для передачи данных по каналу с низкой пропускной способностью. Использование *BLE* для приложений с большим объемом часто передаваемых данных существенно увеличивает потребление электроэнергии и сводит на нет основное преимущество *BLE*.

Wi-Fi Direct (Wi-Fi Peer-to-Peer) — стандарт, позволяющий двум и более *Wi-Fi*-устройствам общаться друг с другом без маршрутизаторов и хот-спотов.

- *Wi-Fi Direct* функционирует в стандартном частотном диапазоне — 2,4-5 ГГц;
- он обеспечивает более высокую скорость передачи, чем *Bluetooth*, что

очень ценно для передачи видео в реальном времени или больших объемов данных. В *WiFi Direct* достигается скорость до 250 Мбит/с.;

– диапазон для подключения устройств в *WiFi Direct* достигает 200 метров.

Исходя из этих данных, можно сделать вывод, что технология *Wi-Fi Direct* является более скоростной и покрывает большую площадь, чем *BLE*.

1.1.4 Постановка задачи

Технологии *mesh*-сетей и *Push-to-talk* могут быть использованы в случаях, когда обычные сети не обеспечивают необходимого покрытия, либо не справляются с нагрузкой: на природе, в местах большого скопления людей, в чрезвычайных ситуациях. Услуга ориентирована как на корпоративных клиентов – на курьерские службы, компании занимающиеся перевозкой, строительные организации, правоохранительные органы, военные, так и на массовый рынок.

Задачей данной работы является создание мобильного приложения под управлением ОС *Android*. Пользователь мобильного приложения сможет просмотреть доступные для подключения устройства и подключиться к ним. После успешного подключения к другому устройству пользователь сможет обмениваться голосовыми сообщениями в режиме реального времени, а также файлами по защищенному каналу с другим устройством. Для идентификации в сети пользователю предоставлена возможность редактировать имя своего устройства, отображающееся в сети. На основе данных вариантов использования приложения была построена диаграмма (рисунок 3).

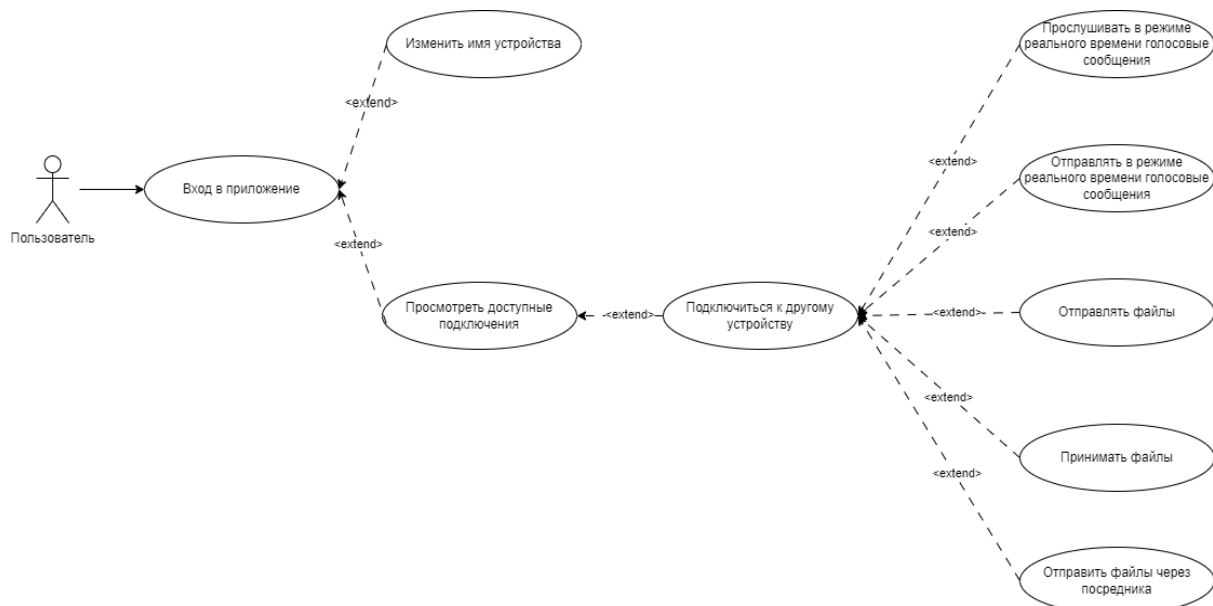


Рисунок 3 – Диаграмма вариантов использования

1.1.5 Выводы

По итогу проведения анализа предметной области и технологий для разработки мобильного приложения под ОС *Android* были выбраны среда разработки *Android Studio*, и язык программирования *JAVA*. В качестве коммуникационной технологии была выбрана технология *Wi-Fi Direct*.

1.2 Реализация приложения

1.2.1 Алгоритм работы приложения

При проектировании приложения для детализации алгоритмической и логической реализации выполняемых приложением функций можно использовать *UML* диаграмму деятельности (рисунок 4).

После запуска приложения начинается поиск возможных подключений. Из списка доступных подключений пользователь может выбрать клиента, к которому он подключится. Приложение осуществляет соединение абонентов. Далее пользователь может либо отправить файл собеседнику, либо обменяться с ним голосовыми сообщениями в режиме реального времени.

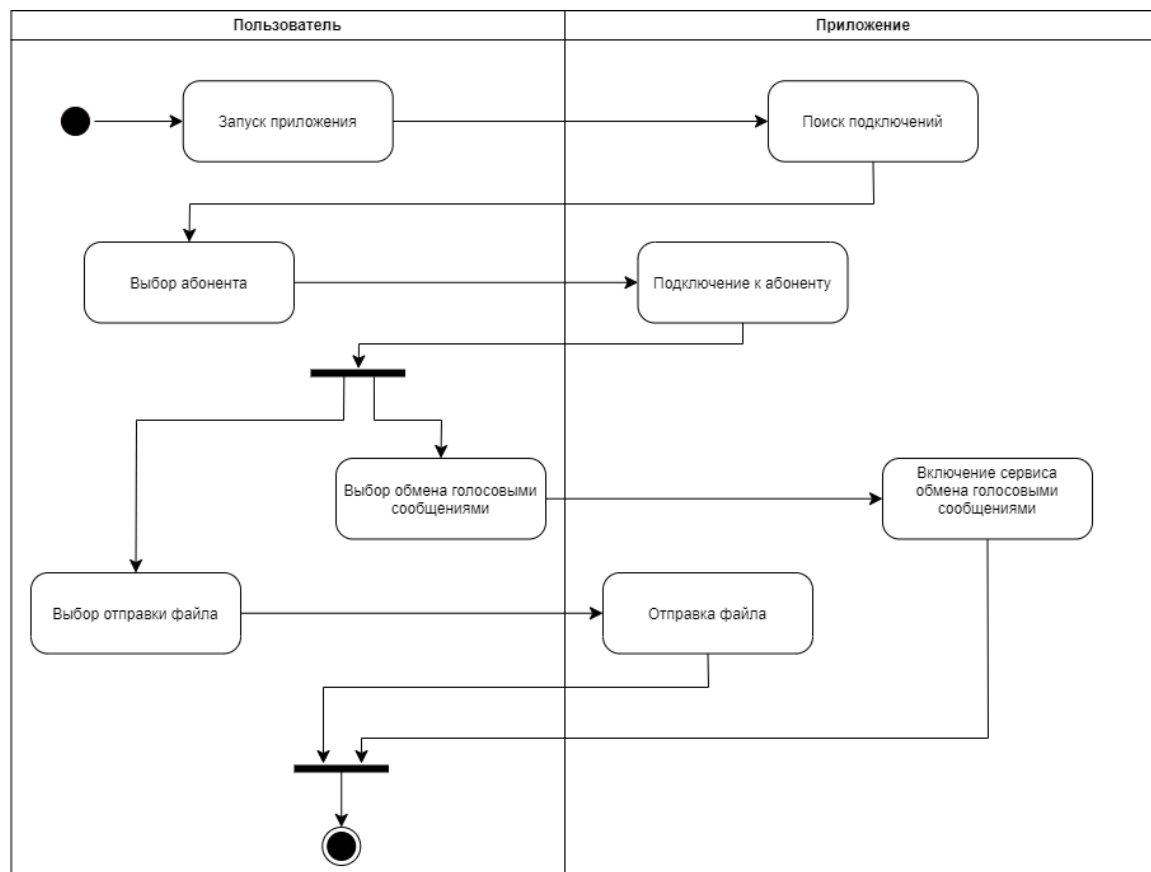


Рисунок 4 – Диаграмма деятельности приложения

1.2.2 Реализация одноранговых соединений с помощью Wi-Fi Direct

Файл манифеста *AndroidManifest.xml* инкапсулирует всю архитектуру Android-приложения, его функциональные возможности и конфигурацию. Чтобы использовать *Wi-Fi Direct* в приложении, необходимо добавить разрешения *ACCESS_FINE_LOCATION*, *CHANGE_WIFI_STATE*, *ACCESS_WIFI_STATE*, *INTERNET* и *NEARBY_WIFI_DEVICES* в манифест приложения [1]. *Wi-Fi Direct* не требует подключения к Интернету, но использует стандартные сокеты *Java*, для которых требуется разрешение *INTERNET*.

```

<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_WIFI_STATE" />

```

```

        <uses-permission
android:name="android.permission.CHANGE_WIFI_STATE" />
        <uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE" />
        <uses-permission
android:name="android.permission.INTERNET" />
        <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />

```

Чтобы использовать *Wi-Fi Direct*, необходимо прослушивать широковещательные намерения, которые сообщают приложению, когда происходят определенные события. Для этого в приложении был создан экземпляр *IntentFilter*, настроенный для прослушивания следующих событий:

- *WIFI_P2P_STATE_CHANGED_ACTION*. Указывает, включен ли *Wi-Fi Direct*;
- *WIFI_P2P_PEERS_CHANGED_ACTION*. Указывает, что доступный список одноранговых узлов изменился;
- *WIFI_P2P_CONNECTION_CHANGED_ACTION*. Указывает, что состояние подключения *Wi-Fi Direct* изменилось.

Для прослушивания изменений состояния *Wi-Fi Direct*, был создан класс *WifiDirectBroadcastReceiver*. В методе *onReceive()* описаны действия для обработки каждого изменения состояния, указанного выше.

При получении события *WIFI_P2P_PEERS_CHANGED_ACTION*, с помощью *requestPeers()* запрашивается список обнаруженных одноранговых узлов. Метод *RequestPeers()* является асинхронным и уведомляет активность, когда список пиров доступен с помощью метода *onPeersAvailable()*, который определен в *WifiP2pManager.PeerListListener* интерфейсе. Метод *onPeersAvailable()* предоставляет *WifiP2pDeviceList*, который выводится на экран, чтобы пользователь мог выбрать устройство для подключения.

```

WifiP2pManager.PeerListListener peerListListener = new
WifiP2pManager.PeerListListener() {

```

```

@Override
public void onPeersAvailable(WifiP2pDeviceList
wifiP2pDeviceList) {
    Log.i(TAG, "onPeersAvailable");
    deviceNames.clear();

    if (wifiP2pDeviceList.getDeviceList().size() > 0)
    {
        deviceList = new
ArrayList(wifiP2pDeviceList.getDeviceList());
        for (WifiP2pDevice device : deviceList) {
            deviceNames.add(device.deviceName);
            Log.i(TAG, "deviceNames содержит: " +
deviceNames);
        }
        ArrayAdapter adapter = new
ArrayAdapter(getApplicationContext(),
android.R.layout.simple_list_item_1, deviceNames);
        deviceListView.setAdapter(adapter);
        Log.i(TAG, "setAdapter");
        if ((file_path[0] != null) && (flagFind ==
false)&&(!iSendFilesDevice.equals("*"))){
            Log.i(TAG, "setAdapter поиск");
            int i = searchItem(iSendFilesDevice);
            Log.i(TAG, "searchItem");
            if (i != 10000) {
                Log.i(TAG, "searchItem нашел");
                flagFind = true;
                wantToTalk = true;
                wantSendFile = "sendfile";
                connectToDevice(i);
            }
        }
    }
}

```

После получения списка возможных абонентов и выбора устройства для подключения, вызывается метод *connect()* для подключения к устройству. Для

вызова этого метода требуется *WifiP2pConfig* объект, содержащий информацию об устройстве, к которому необходимо подключиться. *WifiP2pManager.ActionListener* уведомляет об успешном или неудачном соединении.

Работу с пользовательским интерфейсом и созданием одноранговых соединений реализует класс активности *FirstActivity*.

Класс *FirstActivity.java* включает в себя десять методов. Описание методов данного класса представлено ниже.

- *protected void onCreate()*. Метод, отвечающий за размещение пользовательского интерфейса, инициализацию некоторых полей класса;
- *private void exqListener()*. Метод, отвечающий за обработку взаимодействия пользователя с интерфейсом приложения;
- *void alertDialogDemo()*. Метод, который выводит на экран диалоговое окно;
- *private void initialWork()*. Метод, создающий прослушивателей широковещательных намерений;
- *protected void onResume()*. Метод, отвечающий за регистрацию прослушивателей широковещательных намерений и отключение абонентов;
- *protected void onPause()*. Метод, отключающий регистрацию прослушивателей широковещательных намерений;
- *protected void onDestroy()*. Метод, закрывающий сокет, если он был создан;
- *public void onPeersAvailable(WifiP2pDeviceList wifiP2pDeviceList)*. Метод, предназначенный для получения и отображения на экране доступных абонентов;
- *void connectToDevice(int position)*. Метод, обеспечивающий подключение устройств;
- *public void onConnectionInfoAvailable(WifiP2pInfo wifiP2pInfo)*. Метод, определяющий, кто из абонентов станет сервером, а кто — клиентом.

FirstActivity реализует пользовательский интерфейс (рисунок 5). Интерфейс включает следующие элементы:

- кнопка, отвечающая за запуск поиска доступный для подключения устройств;
- кнопка, открывающая настройки *Wi-Fi Direct* на устройстве, в которых можно изменить имя устройства;
- список доступных устройств;
- текстовое поле, отображающее состояние подключения устройства;
- диалоговое окно, для выбора устройства, с которым можно поделиться файлами.

Как только соединение установлено, устройства могут передавать данные друг другу с помощью сокетов [2]. Для этого были созданы классы *ClientClassAudio* и *ServerClassAudio*.

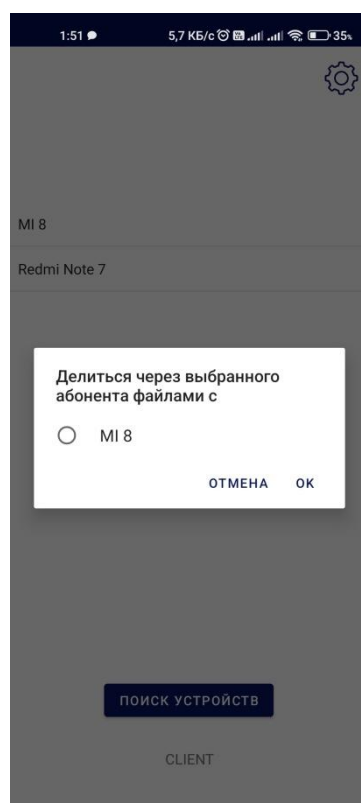


Рисунок 5 – Пользовательский интерфейс *FirstActivity*

1.2.3 Реализация алгоритма *Push-to-talk*

После успешного подключения устройств вызывается активность *ChatActivity*. Данный класс позволяет пользователям обмениваться звуковыми сообщениями в режиме реального времени. Для работы с микрофоном в приложении в файл манифеста было добавлено разрешение *RECORD_AUDIO*. При запуске активности начинает свою работу сервис *AudioStreamingService*. Он осуществляет прослушивание приходящих от сокета данных и их воспроизведение. Данный сервис работает асинхронно, чтобы не блокировать пользовательский поток. Для воспроизведения звука в сервисе был использован класс *AudioTrack* [3].

```
audioTrack = new AudioTrack(
    AudioManager.STREAM_MUSIC,
    SAMPLE_RATE,
    AudioFormat.CHANNEL_OUT_MONO,
    AudioFormat.ENCODING_PCM_16BIT,
    bufferSize,
    AudioTrack.MODE_STREAM);
```

Класс *AudioTrack* управляет звуковым ресурсом и воспроизводит его. Он позволяет передавать аудиобуферы в аудиоприемник для воспроизведения. Это достигается путем «проталкивания» данных в объект *AudioTrack*. Экземпляр *AudioTrack* может работать в двух режимах: статическом или потоковом. В приложении звуковая дорожка не сохраняется в памяти и может быть произвольной длины, так как обмен голосовыми сообщениями осуществляется в режиме реального времени, поэтому *AudioTrack* работает в потоковом режиме. После создания объект *AudioTrack* инициализирует связанный с ним аудиобуфер. Размер этого буфера определяет размер звуковой дорожки для воспроизведения.

```
int bufferSize = AudioRecord.getMinBufferSize(SAMPLE_RATE,
    AudioFormat.CHANNEL_IN_MONO, AudioFormat.ENCODING_PCM_16BIT);
```

Приложение записывает непрерывный поток данных в *AudioTrack*, используя один из методов *write()* и помещает поток в очередь для

воспроизведения. В потоковом режиме данные будут записываться в приемник звука частями, размер которых меньше или равен размеру буфера.

```
audioTrack.write(buffer, 0,buffer.length);
```

За запись и отправку звуковых сообщений отвечает класс *MicRecorder*, который начинает свою работу по нажатию на кнопку записи на экране (рисунок 6).



Рисунок 6 – Кнопка начала записи звукового сообщения

MicRecorder реализует интерфейс *Runnable*, так же для того, чтобы не блокировать пользовательский поток, так как при его длительной блокировке система *Android* решает, что приложение не отвечает и закрывает его.

Кнопка начала записи используется и для приостановки записи звука, при этом изображение на кнопке меняется (рисунок 7).



Рисунок 7 – Кнопка остановки записи звукового сообщения

Для записи звука в *MicRecorder* был использован класс *AudioRecord*, управляющий аудиоресурсами для записи звука с аппаратного обеспечения аудиоввода устройства. Это достигается путем чтения данных из объекта *AudioRecord*.

```
AudioRecord record = new AudioRecord  
(MediaRecorder.AudioSource.VOICE_RECOGNITION, SAMPLE_RATE,  
AudioFormat.CHANNEL_IN_MONO, AudioFormat.ENCODING_PCM_16BIT,  
bufferSize);
```

Приложение производит опрос объекта *AudioRecord* одним из методов *read()*. После создания объект *AudioRecord* инициализирует связанный с ним аудиобуфер, который будет заполнен аудиоданными. Размер этого буфера определяет, как долго *AudioRecord* может записывать звук. Данные считываются с аудиооборудования частями, размер которых меньше общего размера буфера записи.

1.2.4 Реализация защищенного обмена файлами

В пользовательском интерфейсе *ChatActivity* размещена кнопка, отвечающая за открытие класса активности *FileActivity*, который реализует обмен файлами между устройствами. При запуске активности начинает свою работу сервис *FileTransferService*, принимающий файлы с сокета [4], фрагмент данного сервиса представлен ниже.

```
FileOutputStream fos = new FileOutputStream(f);
BufferedInputStream bis = new
BufferedInputStream(SocketHandlerFile.getSocket().getInputStream())
;

while (file_size > 0) {
    int i = bis.read(byteArray);

    DecriptbyteArray = Kuz.Decript(byteArray, myFilekey);
    fos.write(DecriptbyteArray, 0, i);
    Log.d("Recieve", "получил DecriptbyteArray");
    file_size -= i;
}
Log.d("Recieve", "получил file");
```

По нажатию на кнопку «Отправить» открывается окно проводника, в котором пользователь может выбрать файл для отправки. С помощью метода *onActivityResult* программа получает путь отправляемого файла [5]. Далее в отдельном потоке файл анализируется, приложение получает его размер и название. Сначала эти данные отправляются абоненту, а затем побайтно отправляется сам файл.

```

        FileInputStream fis = new
FileInputStream(FileActivity.file.getPath());
        long fileSizeByte;
        fileSizeByte = FileActivity.file.length();
        while (fileSizeByte > 0) {
            int i = fis.read(fileBuffer);
            encryptFileBuffer = Kuz.Encrypt(fileBuffer, result );
            outputStream.write(encryptFileBuffer, 0, i);
            Log.d("Send", "отправил encryptFileBuffer");
            fileSizeByte -= i;
        }
        Log.d("Send", "отправил file");

```

Принимающее устройство сохраняет у себя название файла и определяет директорию, в которой сохранит его, затем побайтно считывает файл.

Когда загрузка и отправка файлов выполнены, потоки отправляют широковещательное сообщение.

```

Intent intent = new Intent(FileActivity.BROADCAST_ACTION);
        intent.putExtra("sessionClosed", true);
        sendBroadcast(intent);

```

Активность обрабатывает эти сообщения и закрывает активность, переходя обратно на экран обмена звуковыми сообщениями.

1.2.5 Реализация криптографической защиты информации

В качестве алгоритма криптографической защиты информации был выбран алгоритм блочного шифрования «Кузнечик», который описан в ГОСТ 34.12–2015.

Основу алгоритма составляет не сеть Фейстеля, как в большинстве блочных шифров, а *SP* – *Substitution-Permutation network*, или подстановочно-перестановочная сеть (рисунок 8). Шифр на основе *SP*-сети получает на вход блок и ключ и совершает несколько чередующихся раундов, состоящих из стадий подстановки и стадий перестановки. В «Кузнечике» каждый раунд включает в себя линейное и нелинейное преобразование плюс операцию наложения итерационного ключа. Всего таких раундов девять и один последний

неполный раунд, в котором выполняется только наложение последнего (десятого) итерационного ключа.

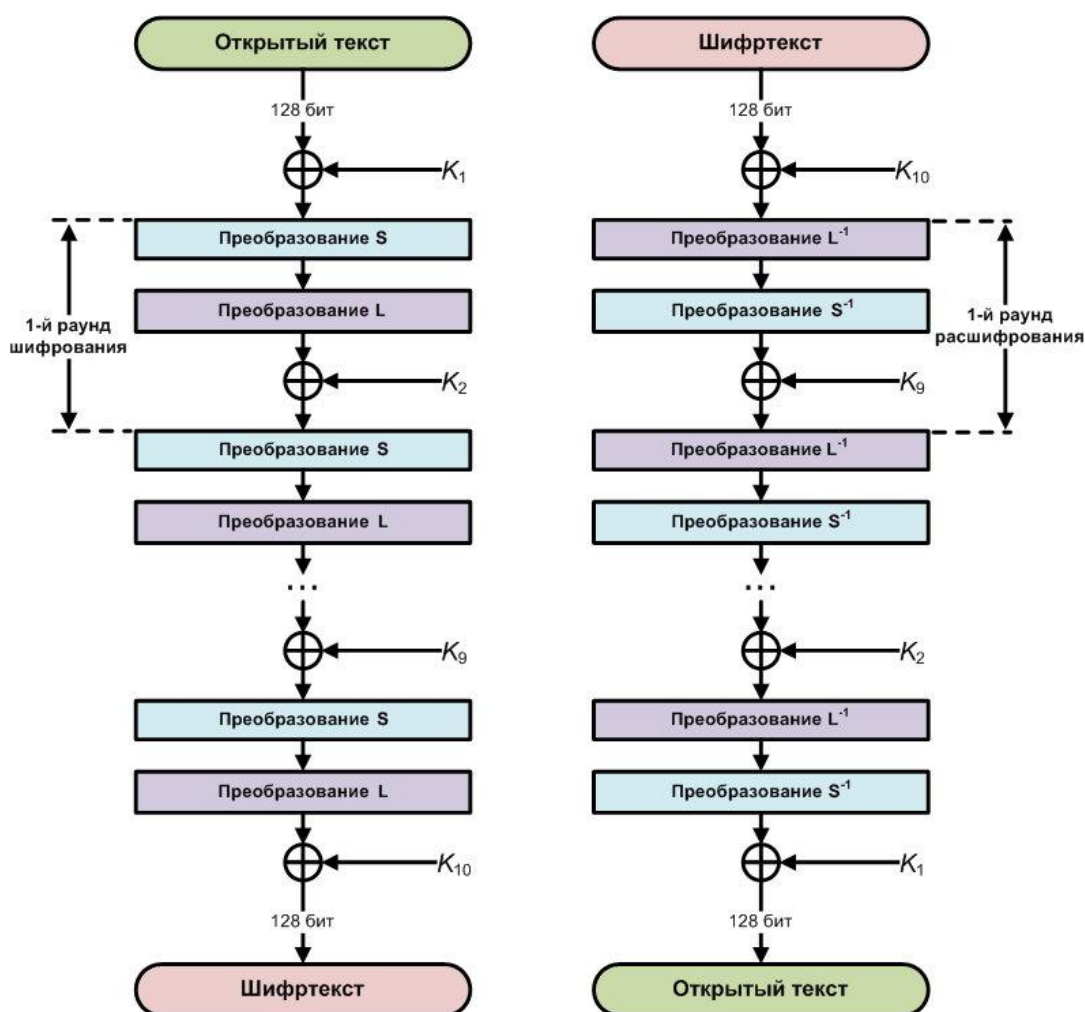


Рисунок 8 - Подстановочно-перестановочная сеть

Итерационные (или раундовые) ключи получаются путем определенных преобразований на основе мастер-ключа, длина которого, как мы уже знаем, составляет 256 бит. Этот процесс начинается с разбиения мастер-ключа пополам, так получается первая пара раундовых ключей. Для генерации каждой последующей пары раундовых ключей применяется восемь итераций сети Фейстеля, в каждой итерации используется константа, которая вычисляется путем применения линейного преобразования алгоритма к значению номера итерации. Получение итерационных ключей показано на рисунке 9.

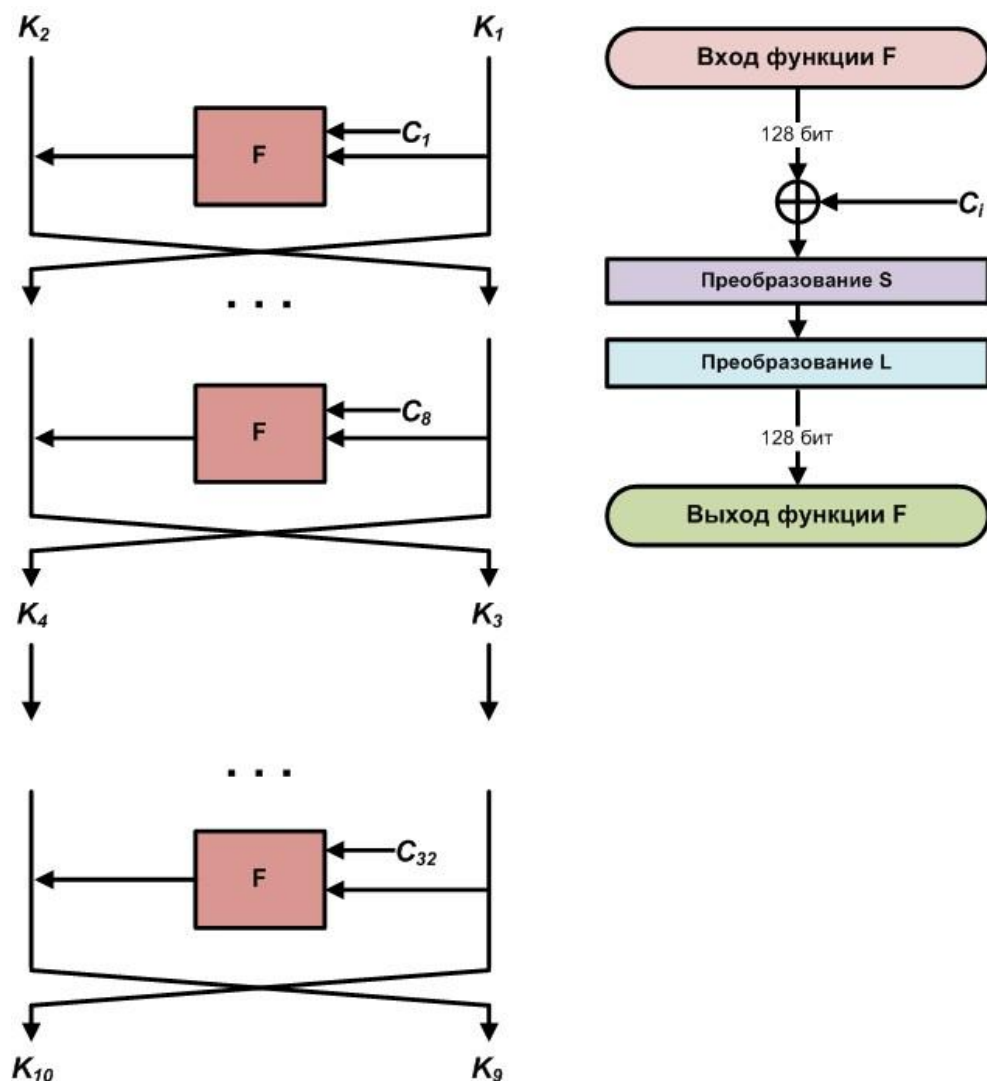


Рисунок 9– Получение итерационных ключей

Данные преобразования описаны в классе *Kuznechik.java*. Для работы с мастер-ключом также был использован алгоритм вычисления хеш-суммы под названием «Стрибог», который содержится в файле *Stribog.java*.

Алгоритм шифрования «Кузнечик» является симметричным, поэтому для шифрования и расшифрования файла используется один и тот же ключ. Передавать симметричные ключи по незащищенному соединению небезопасно. Для решения данной проблемы был использован асимметричный алгоритм шифрования, который использует пары ключей. Один ключ всегда используется для шифрования, а другой – для дешифрования. Смысл заключается в том, чтобы расшифровать смог только получатель, поэтому данные шифруются

публичным ключом, а расшифровываются только приватным. На устройстве, которое отправляет файл генерируются пара ключей. На устройстве, принимающем файл, генерируется один симметричный ключ для шифрования файла. По незащищенному соединению принимающему устройству отправляется публичный ключ. Принимающее устройство шифрует симметричный ключ и отправляет его устройству, передающему файл. Отправляющее устройство расшифровывает приватным ключом ключ для файла, затем шифрует файл и отправляет его абоненту. Схема данного алгоритма представлена на рисунке 10. Для шифрования симметричного ключа использован асимметричный алгоритм шифрования *RSA*. Данный алгоритм относится к асимметричным шифрам. В асимметричных шифрах используются два ключа – открытый и закрытый. Открытые ключи доступны всем желающим и передаются по незащищенному каналу связи. Отправляемое сообщение шифруется открытым ключом. Дешифрируется сообщение при его получении закрытым ключом. *RSA* использует разложение больших чисел на простые множители, что требует большого объема вычислений и эта особенность определяет стойкость данного шифра.

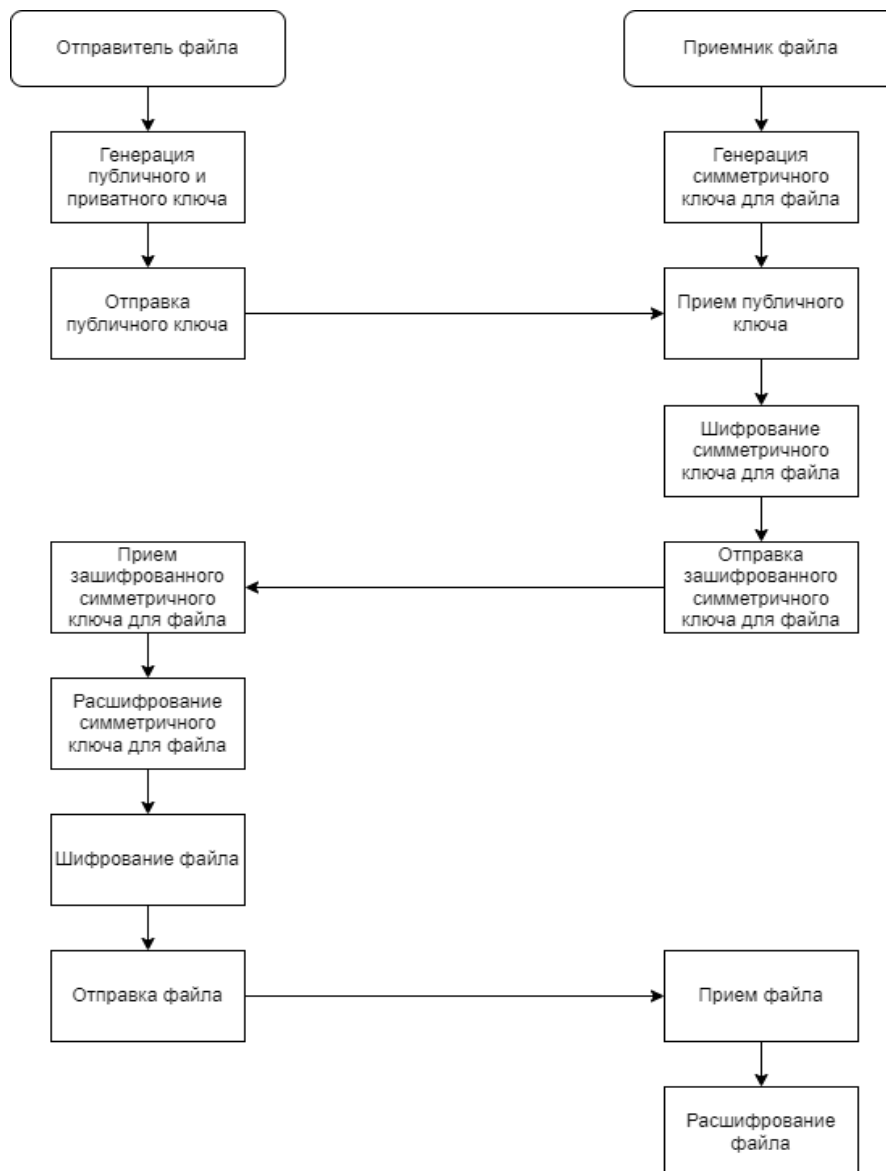


Рисунок 10 – Передача симметричного ключа

Первым этапом асимметричного шифрования является создание получателем шифрограмм пары ключей. Ниже представлена генерация ключей в классе *FileSend*.

```

KeyPairGenerator keyPairGen =
KeyPairGenerator.getInstance("RSA");
keyPairGen.initialize(1024);
KeyPair pair = keyPairGen.generateKeyPair();
PrivateKey privKey = pair.getPrivate();
PublicKey publicKey = pair.getPublic();
  
```

Полученный публичный ключ отправляется по незащищенному каналу

связи абоненту, принимающему файл. Данный абонент генерирует с помощью класса *FileTransferService* симметричный ключ для шифрования файла.

```
String symbols = "qwertyuiopasdfghjklzxcvbnm1234567890";
String random = new Random().ints(32, 0, symbols.length())
    .mapToObj(symbols::charAt)
    .map(Object::toString)
    .collect(Collectors.joining());

byte[] myFilekey = random.getBytes();
```

Ключ для файла шифруется полученным публичным ключом, а затем отправляется абоненту, отправляющему файл.

Устройство, получившее закодированный ключ расшифровывает его приватным ключом.

```
Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.DECRYPT_MODE, privKey);
byte[] result = cipher.doFinal(otherKey);
```

Таким образом, стороны обмениваются симметричным ключом для кодирования файла. После этого происходит шифрование файла алгоритмом «Кузнечик».

```
encryptFileBuffer = Kuz.Encrypt(fileBuffer, result);
```

На принимающем файл устройстве запускается расшифровка файла.

```
DecriptbyteArray = Kuz.Decript(byteArray, myFilekey);
```

1.2.6 Выводы

Было реализовано мобильное приложение, для защищенного обмена информацией с использованием технологии *mesh*-сетей. В данной главе было представлено описание разработки приложения, которое должно стать конечным продуктом данной работы. Разработан пользовательский интерфейс и программный код. Основным языком, использованным при написании приложения, является язык *JAVA*. Реализация приложения сделана в среде разработки *Android Studio*.

1.3 Тестирование приложения

1.3.1 Тестирование приложения

Для проверки корректной работы приложения оно было протестировано на реальных мобильных устройствах, подключенных к среде разработки *Android Studio* для отслеживания появления ошибок. В ходе проверки ошибок и недочетов в работе приложения не было выявлено.

На рисунке 11 представлено главное окно приложения, включающее две кнопки: «Поиск устройств» – начать поиск доступных устройств, кнопка настроек. На главном экране также присутствует текстовое поле для отслеживания состояния устройства.

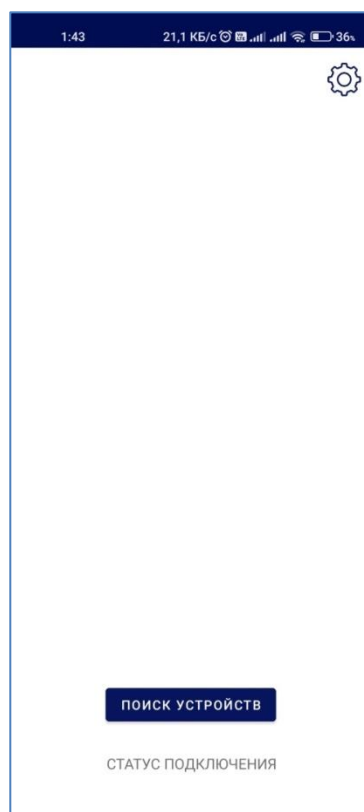


Рисунок 11 – Главный экран приложения

При нажатии на кнопку настроек открывается окно настроек *Wi-Fi Direct* устройства (рисунок 12). После нажатия на главном экране кнопки «Поиск устройств», на экране появляется список доступных устройств (рисунок 13).

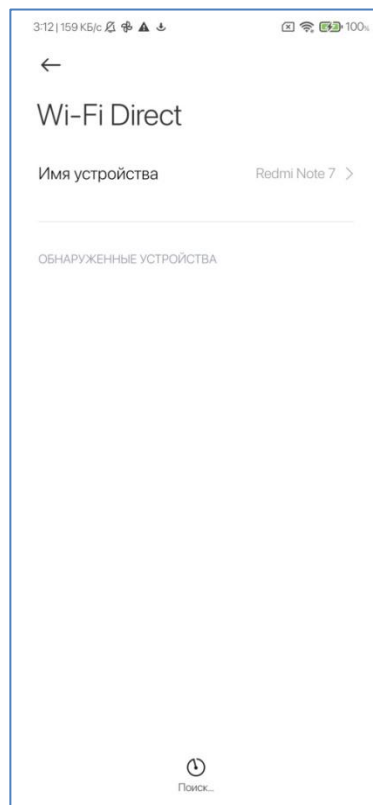


Рисунок 12 – Окно настроек

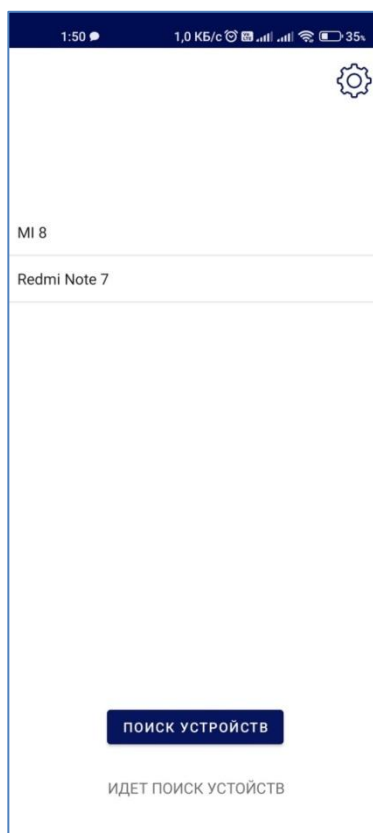


Рисунок 13 – Отображение списка устройств

После нажатия на выбранного из списка абонента, если у данного абонента в списке устройств есть смартфоны, кроме нашего пользователя, то на экране появляется диалоговое окно (рисунок 14), предлагающее выбрать, другое устройство, с которым можно поделиться файлами, если они будут отправлены.

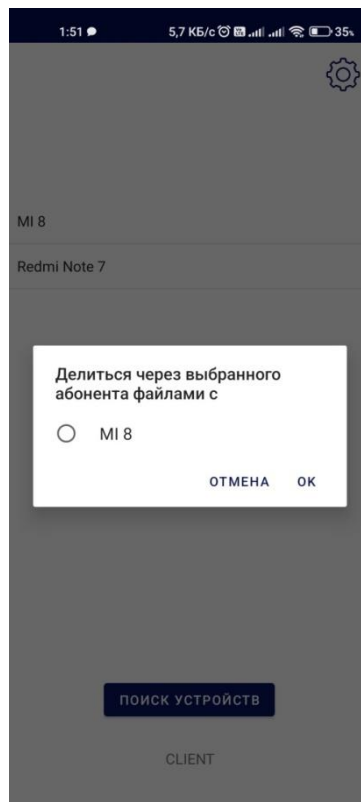


Рисунок 14 – Диалоговое окно

После подключения, устройства переключаются на активность для обмена звуковыми сообщениями (рисунок 15).

На рисунке 16 представлен вид активности после нажатия кнопки для записи звука.

На экране данной активности есть кнопка для отправки файлов, после нажатия на нее приложение переключается на следующую активность, представленную на рисунке 17.

На экране представлена кнопка «Отправить», после нажатия на нее на устройстве открывается файловый менеджер (рисунок 18), в котором необходимо выбрать файл.

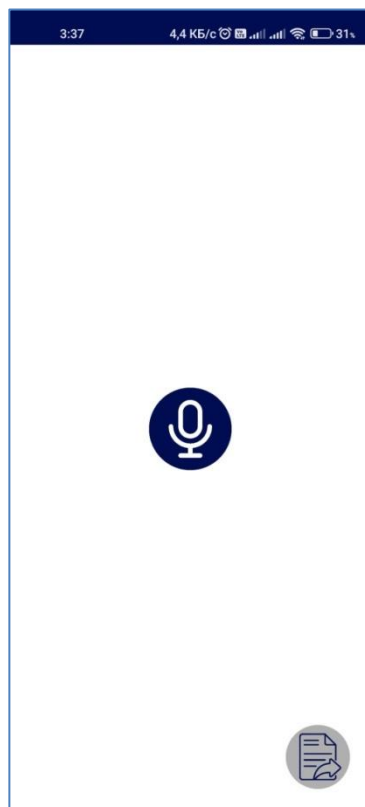


Рисунок 15 – Активность для обмена голосовыми сообщениями

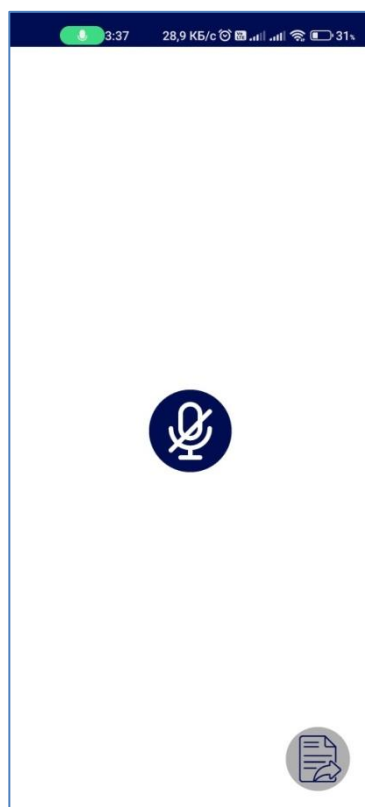


Рисунок 16 – Вид активности после начала записи звука



Рисунок 17 – Активность для обмена файлами

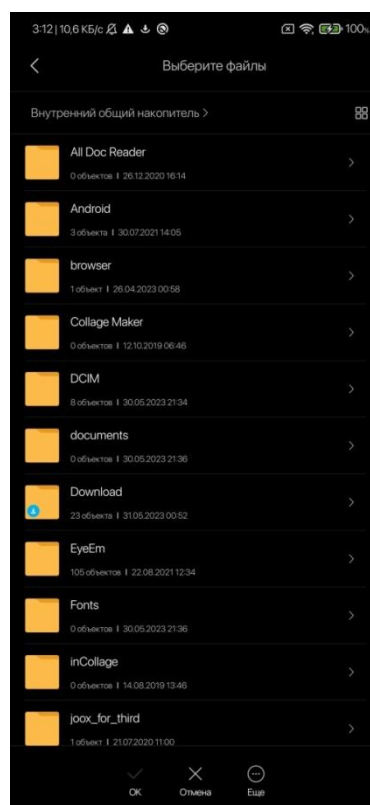


Рисунок 18 – Файловый менеджер

После выбора файла на экране активности появится текстовое поле, отображающее пересылки файла (рисунок 19).



Рисунок 19 – Состояние пересылки файла

После отправки файла приложение закрывает данную активность и возвращается в окно обмена голосовыми сообщениями. Если пользователь вернется на окно первой активности, а его абонентом была выбрана опция поделиться принятыми файлами с другим устройством, то начнется поиск устройства в списке доступных соединений (рисунок 20). Если необходимый абонент будет обнаружен, то на обоих устройствах сразу откроется активность для отправки файлов. После завершения обмена файлами откроется активность обмена звуковыми сообщениями.

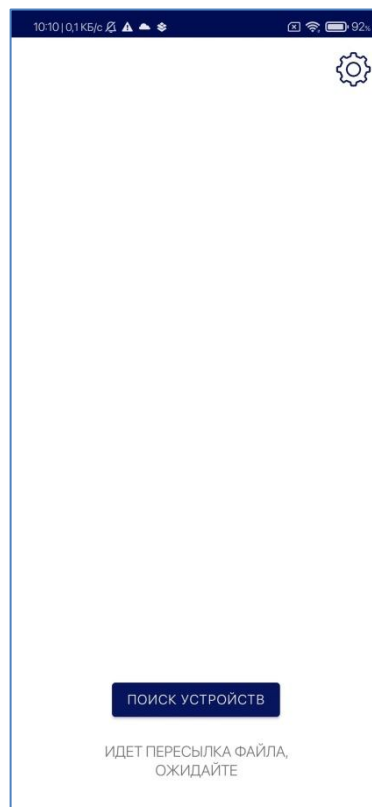


Рисунок 20 – Поиск устройства для пересылки файлов

1.3.2 Выводы

Использование приложения «*RadioMesh*» на физических мобильных устройствах не выявило ошибок и проблем во время работы приложения. Тестирование показало, что приложение работает корректно.

2 Безопасность жизнедеятельности

Рабочее место оператора ЭВМ - это часть пространства, в котором разработчик осуществляет трудовую деятельность, и проводит большую часть рабочего времени. Рабочее место, хорошо приспособленное к трудовой деятельности инженера, обеспечивает ему высокую производительность труда при наименьшем физическом и психическом напряжении.

Согласно ГОСТ 12.2.032– 78 конструкция рабочего места и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. В частности, при организации рабочего места программиста должны быть соблюдены следующие условия:

- оптимальное размещение оборудования, входящего в состав рабочего места;
- достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;
- необходимо естественное и искусственное освещение для выполнения поставленных задач;
- уровень акустического шума не должен превышать допустимого значения.

2.1 Требования к помещению

По взрывопожарной и пожарной опасности помещения разделяют на категории А, Б, В1-В4, Г и Д. Согласно Нормам пожарной безопасности НПБ 105-03, помещения с ЭВМ и ПЭВМ относятся к категории В, которая является пожароопасной, но не предусматривает возможность взрыва. В1 — В4 — пожароопасность. Горючие и трудногорючие жидкости, твердые горючие и трудногорючие вещества и материалы, способные при взаимодействии с водой, кислородом воздуха или друг с другом только гореть, при условии, что помещения, в которых они находятся (обращаются), не относятся к категории А или Б.

Деятельность программиста относится к категории работ IA – легкие физические работы. В соответствии с установленными нормами, температура на рабочем месте должна быть в пределах от 21 до 24 °С в летнее время и 22-25 °С в зимнее. Относительная влажность воздуха должна быть около 55%, температура поверхностей – 20-26 °С. Скорость движения воздуха не должна составлять более 0,1м/с.

В помещениях, в которых работа с использованием ПЭВМ является основной, содержание вредных химических веществ в воздухе не должно превышать предельно допустимых концентраций загрязняющих веществ в атмосферном воздухе населенных мест в соответствии с действующими гигиеническими нормативами.

2.2 Освещенность рабочего места

Рациональное освещение рабочего места является одним из важнейших факторов, влияющих на эффективность трудовой деятельности человека, предупреждающих травматизм и профессиональные заболевания. Правильно организованное освещение создает благоприятные условия труда, повышает работоспособность и производительность труда. Освещение на рабочем месте программиста должно быть таким, чтобы работник мог без напряжения зрения выполнять свою работу. Утомляемость органов зрения зависит от ряда причин: недостаточность освещенности; чрезмерная освещенность; неправильное направление света.

Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение и резь в глазах. Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего. Все эти причины могут привести к несчастному случаю или профзаболеваниям.

Расчет освещенности рабочего места сводится к выбору системы

освещения, определению необходимого числа светильников, их типа и размещения. Процесс работы программиста в таких условиях, когда естественное освещение недостаточно или отсутствует. Искусственное освещение выполняется посредством электрических источников света.

2.3 Расчет количества светильников в помещении

Чтобы найти необходимое для помещения количество светильников можно использовать формулу:

$$N = E_n * S * K_z * Z / (n * \Phi),$$

где: N – число светильников в помещении;

E_n – минимальная освещенность помещения;

S – площадь помещения;

K_z – коэффициент запаса;

Z – коэффициент неравномерности освещения;

n – коэффициент использования светового потока;

Φ – световой поток.

Минимальная освещенность помещения должна быть 300 лк. Коэффициент запаса для сухих помещений равен 1,4. Для расчета будут использованы люминесцентные лампы, имеющие коэффициент неравномерности равным 1,1.

Размер помещения возьмем 5 на 3 метра. Исходя из этих данных, получаем площадь помещения:

$$S = 5 \text{ м} * 3 \text{ м} = 15 \text{ м}^2.$$

После того, как стала известна площадь помещения, можно вычислить его индекс по формуле:

$$i = S / (h_p * (W + L)),$$

где: i – индекс помещения;

h_p – высота светильников над поверхностью стола;

W – ширина помещения;

L – длина помещения.

Размеры комнаты 5 на 3 метра, высота светильников над столом равна 2.1 метров.

$$i = 5 \cdot 3 / (2,1 \cdot (5+3)) = 0,9.$$

Коэффициента отражения потолка R_p – 70%, так как потолок является белой поверхностью. Коэффициент отражения стен R_c равен 50%, так как представляет собой светлую поверхность. По таблице был определен коэффициент использования светового потока n , он равен 0,29. Световой поток для ламп ЛБ40 равен 2800 лм. Подсчитаем количество светильников типа ЛПО с двумя лампами:

$$N = 300 \cdot 15 \cdot 1,4 \cdot 1,1 / (0,29 \cdot 2800 \cdot 2) = 4,27.$$

Для освещения помещения необходимо 5 светильников типа ЛПО.

2.4 Требования по организации рабочего места

При работе в положении сидя рекомендуются следующие параметры рабочего пространства: ширина не менее 700 мм; глубина не менее 400 мм; высота рабочей поверхности стола над полом 700-750 мм.

Оптимальными размерами стола являются: высота 710 мм; длина стола 1300 мм; ширина стола 650 мм. Поверхность для письма должна иметь не менее 40 мм в глубину и не менее 600 мм в ширину.

Под рабочей поверхностью должно быть предусмотрено пространство для ног: высота не менее 600 мм; ширина не менее 500 мм; глубина не менее 400 мм.

Монитор на столе должен располагаться на расстоянии 60-70 см от глаз пользователя, клавиатура – на расстоянии 100-300 мм от края стола, обращенного к пользователю, или на специальной регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Важным элементом рабочего места программиста является кресло. Оно выполняется в соответствии с ГОСТ 21.889-76. При проектировании кресла исходят из того, что при любом рабочем положении программиста его поза

должна быть физиологически правильно обоснованной, т.е. положение частей тела должно быть оптимальным. Для удовлетворения требований физиологии, вытекающих из анализа положения тела человека в положении сидя, конструкция рабочего сидения должна удовлетворять следующим основным требованиям: допускать возможность изменения положения тела, т.е. обеспечивать свободное перемещение корпуса и конечностей тела друг относительно друга; допускать регулирование высоты в зависимости от роста работающего человека, иметь слегка вогнутую поверхность, иметь небольшой наклон назад.

2.5 Выводы

Соблюдение техники безопасности является обязательной мерой охраны здоровья трудящихся. Несоблюдение техники безопасности может угрожать жизни и здоровью работников.

В данной главе были изложены требования к рабочему месту разработчика. Созданные условия должны обеспечивать комфортную работу. Были указаны оптимальные параметры рабочего стола и кресла, рабочей поверхности, а также проведен расчет оптимального освещения производственного помещения. Соблюдение данных условий, позволит сохранить хорошую работоспособность в течение всего рабочего дня, повысит производительность труда программиста.

Заключение

В процессе выполнения выпускной квалификационной работы был проведен анализ актуальности проекта. Были рассмотрены имеющиеся на данный момент аналоги проекта, на их основании был определен основной функционал приложения. В рамках выбора инструментов разработки были проанализированы популярные среды разработки мобильных приложений и выбрана оптимальная среда для выполнения данного проекта. В ходе рассмотрения технологий создания одноранговых соединений был выбран наиболее современный и технологичный вариант.

В ходе работы был изучен принцип разработки мобильных приложений. Были закреплены навыки работы с языком программирования Java, а также навыки работы со средой *Android Studio*.

В результате проделанной работы было разработано мобильное приложение «*RadioMesh*» для защищенного обмена файлами и голосовыми сообщениями в режиме реального времени с использованием технологии mesh-сетей.

В процессе реализации данного проекта были достигнуты все поставленные цели. Итоговое приложение было протестировано и отлажено. Тестирование показало, что приложение работает корректно и обладает всем необходимым функционалом.

Список использованных источников

1. Создание одноранговых соединений с помощью Wi-Fi Direct
[Электронный ресурс] URL: <https://developer.android.com/training/connect-devices-wirelessly/wifi-direct>.
2. Шилдт Герберт Java 8. Руководство для начинающих. - М.: Вильямс, 2015. - 720 с.
3. А. Голощапов. Google Android. Программирование для мобильных устройств. - М.: БХВ-Петербург, 2012. - 448 с.
4. Патрик Нотон. Java. Справочное руководство. Все, что необходимо для программирования на Java. - М.: Бином, 2015. - 448 с.
5. Дон Гриффитс, Дэвид Гриффитс. Head First. Программирование для Android. 3-е изд. – СПб.: Питер, 2018. – 912 с.

					<i>ПГУ1.090301.38.001 ПЗ</i>	Лист
Изм	Лист	№ докум	Подпись	Дата		44

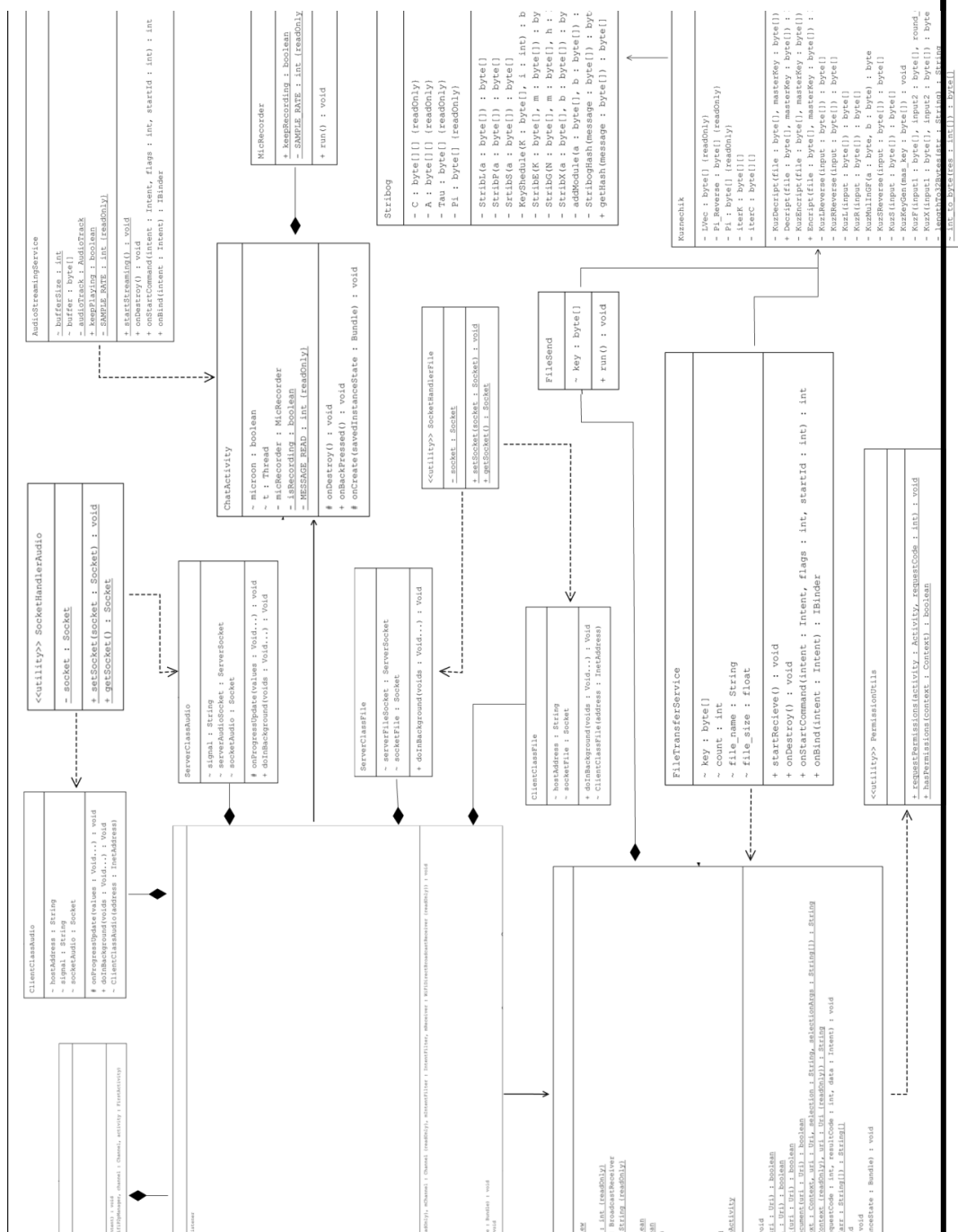
Приложение А

(Обязательное)

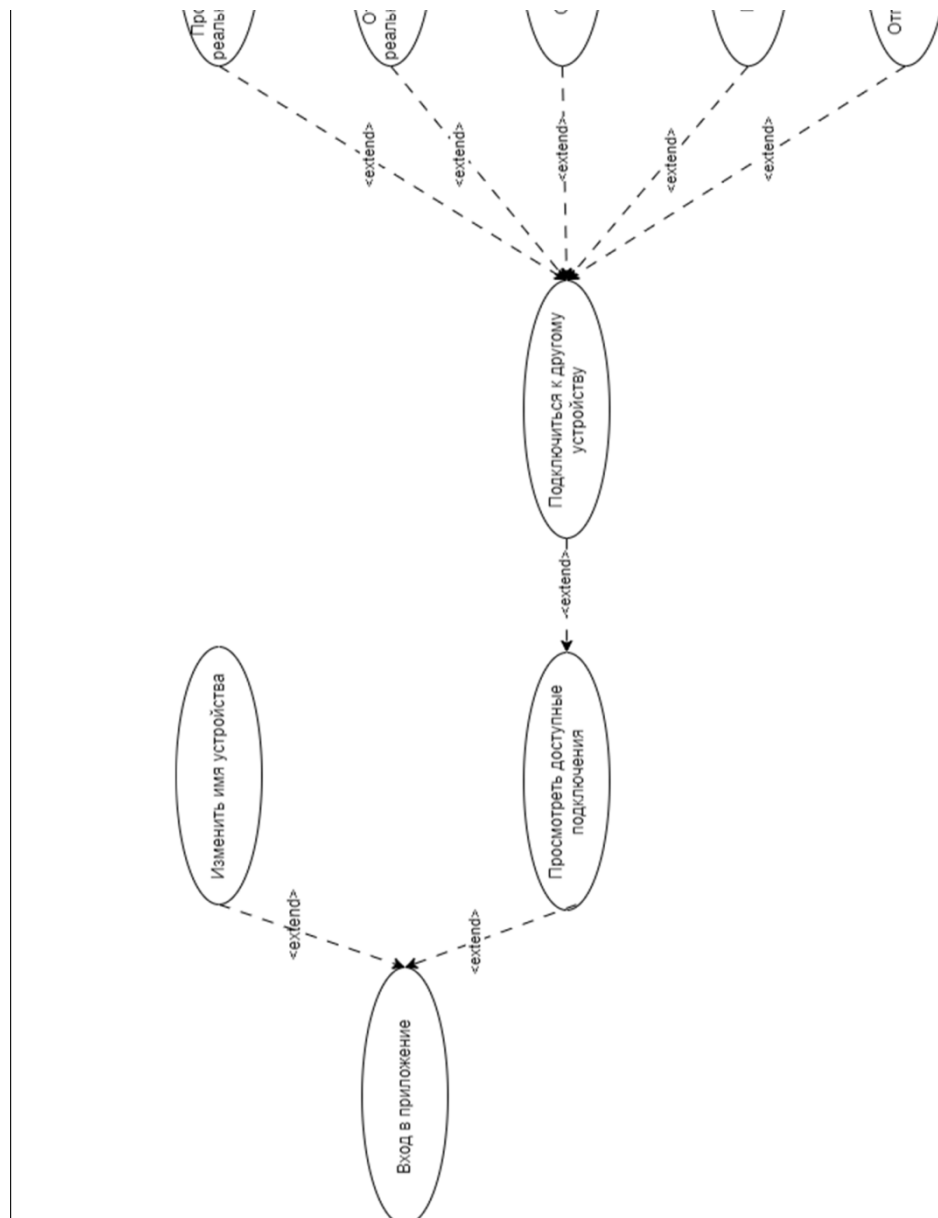
Программное средство для защищенного обмена информацией с
использованием технологии *mesh*-сетей
UML-диаграммы

					ПГУ1.090301.38.001 ПЗ	Лист
						45
Изм	Лист	№ докум	Подпись	Дата		

Приложение А.1 – Диаграмма классов



Приложение А.2 – Диаграмма прецедентов



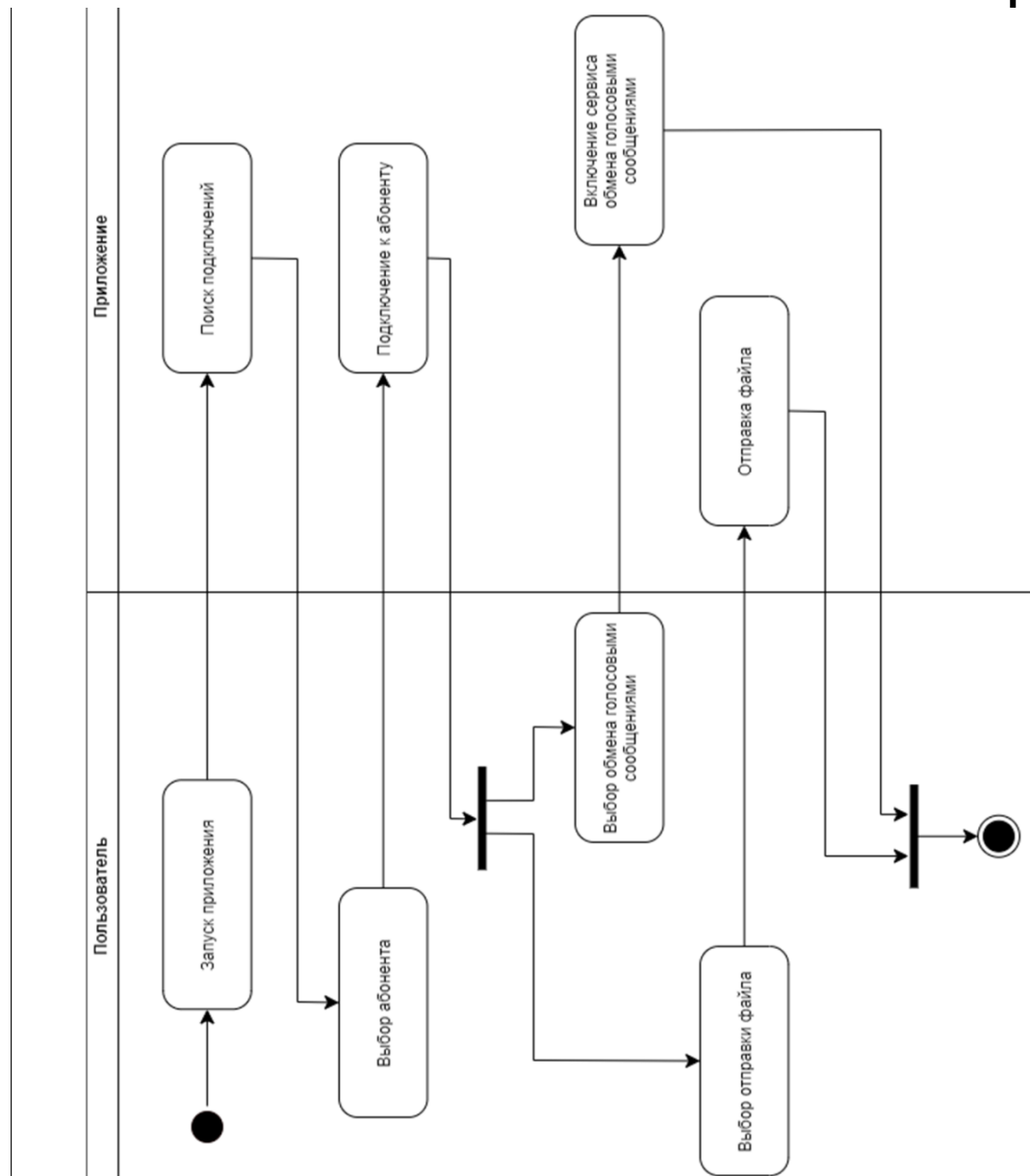
Приложение А.3 – Диаграмма деятельности

Изм	Лист	№ докум	Подпись	Дата

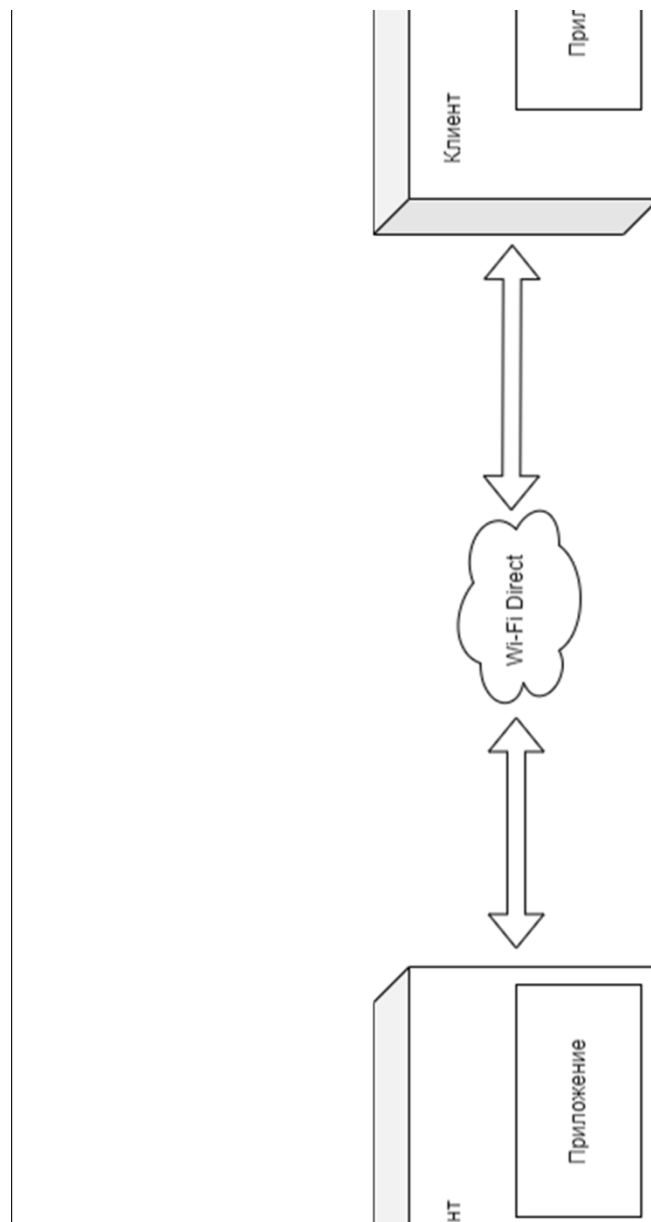
ПГУ1.090301.38.001 ПЗ

Лист

47



Приложение А.4 – Диаграмма последовательности



Приложение Б

(Обязательное)

Программное средство для защищенного обмена информацией с
использованием технологии *mesh*-сетей

Листинг программы

Файл FirstActivity.java

```
package com.example.meshandroid;

import static java.lang.Thread.sleep;
import android.Manifest;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.wifi.WifiManager;
import android.net.wifi.p2p.WifiP2pGroup;
import android.net.wifi.p2p.WifiP2pManager;
import android.net.wifi.p2p.WifiP2pConfig;
import android.net.wifi.p2p.WifiP2pDevice;
import android.net.wifi.p2p.WifiP2pDeviceList;
import android.net.wifi.p2p.WifiP2pInfo;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.IntentFilter;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.provider.Settings;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;

public class FirstActivity extends AppCompatActivity {

    private final String TAG = this.getClass().toString();
```

					ПГУ1.090301.38.001 ПЗ	Лист
Изм	Лист	№ докум	Подпись	Дата		50

```

WifiManager wifiManager;
WifiP2pManager mManager;
WifiP2pManager.Channel mChannel;
BroadcastReceiver mReceiver;
IntentFilter mIntentFilter;
static boolean isServer = false;
public static String myNameWiFiDirect = "***";
static TextView connectionStatus;
ArrayList<WifiP2pDevice> deviceList;
ArrayList deviceNames;
ArrayList neighboringDeviceList;
ListView deviceListView;
Button btn_discover;
ImageButton btn_settings;
ServerClassAudio serverClassAudio;
ServerClassFile serverClassFile;
ClientClassAudio clientClassAudio;
ClientClassFile clientClassFile;
static InetAddress groupOwnerAddressForFile;
String shareFilesDevice = "*";
static String iSendFilesDevice = "*";
static String[] file_path = new String[100];
static int counFiles = 0 ;
boolean wantToTalk;
boolean flagFind = false;
boolean flagConnect = false;
static String wantSendFile = "notsendfile";
public static final int PORT_AUDIO_USED = 9584;
public static int PORT_FILE_USED = 3876;
private static Context mContext;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activty_main);
    Log.i(TAG, "onCreate");
    wantToTalk = false;
    deviceListView = (ListView) findViewById(R.id.device_list);
    initialWork();
    exqListener();
    neighboringDeviceList = new ArrayList();
    deviceNames = new ArrayList();
}

@Override
protected void onRestoreInstanceState(@NonNull Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    deviceNames.clear();
    Log.i(TAG, "deviceList first clear");
}

private void exqListener() {

```

```

        Log.i(TAG, "exqListener");
        Button btn_discover = (Button)
findViewById(R.id.btn_discover);
        ImageButton settings_btn = (ImageButton)
findViewById(R.id.btn_settings);
        btn_discover.setOnClickListener(new View.OnClickListener()
{

            @SuppressWarnings("MissingPermission")
            @Override
            public void onClick(View view) {

                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
{
                    mManager.discoverPeers(mChannel, new
WifiP2pManager.ActionListener() {
                        // @RequiresApi(api = Build.VERSION_CODES.O)
                        @Override
                        public void onSuccess() {
                            connectionStatus.setText("ИДЕТ ПОИСК
УСТОЙСТВ");
                        }

                        // @RequiresApi(api =
Build.VERSION_CODES.O)
                        @Override
                        public void onFailure(int i) {
                            connectionStatus.setText("ПОИСК
НЕВОЗМОЖЕН, ВКЛЮЧИТЕ WI-FI И GPS");
                        }
                    });
                }
            }
        });

        settings_btn.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                startActivity(new
Intent(Settings.ACTION_WIFI_IP_SETTINGS));
            }
        });
        Log.i(TAG, "file_path " + file_path[0]);
        Log.i(TAG, "iSendFilesDevice " + iSendFilesDevice);
        if ((file_path[0] !=
null)&&(!iSendFilesDevice.equals("*"))) {
            Log.i(TAG, "file_path[0] != null");
            disconnect(mManager, mChannel, mIntentFilter,
(WiFiDirectBroadcastReceiver) mReceiver);
            btn_discover.setClickable(false);
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {

```

```

        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            return;
        }
        mManager.discoverPeers(mChannel, new
WifiP2pManager.ActionListener() {
            //@RequiresApi(api = Build.VERSION_CODES.O)
            @Override
            public void onSuccess() {
                connectionStatus.setText("ИДЕТ ПОИСК
УСТРОЙСТВ");

                Log.i(TAG, "discoverPeers onSuccess");
                connectionStatus.setText("ИДЕТ ПЕРЕСЫЛКА
ФАЙЛА, ОЖИДАЙТЕ");
            }

            @Override
            public void onFailure(int i) {
                connectionStatus.setText("ПОИСК НЕВОЗМОЖЕН,
ВКЛЮЧИТЕ WI-FI И GPS");
                Log.i(TAG, "discoverPeers onFailure");
            }
        });
    }

    // Connect to selected device
    deviceListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView,
View view, int i, long l) {
            wantToTalk = true;
            connectToDevice(i);
        }
    });
}

public int searchItem(String textToSearch) {
    int i = 10000;
    Log.i (TAG, "textToSearch " + textToSearch);
    if (deviceNames.contains(textToSearch)) {
        Log.i(TAG, "deviceList.contains " + textToSearch);
        i = deviceNames.indexOf(textToSearch);
    }
    return i;
}

void alertDialogDemo() {
    if (neighboringDeviceList.indexOf(myNameWiFiDirect) != -1)
    {
        neighboringDeviceList.remove(myNameWiFiDirect);
    }
}

```

```

    }
    ArrayList<String> MemberList = new ArrayList<>();

    for (int i = 0; i < neighboringDeviceList.size(); i++) {
        MemberList.add((String) neighboringDeviceList.get(i));
    }
    AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(FirstActivity.this);
    // set dialog message
    alertDialogBuilder
        .setTitle("Делиться через выбранного абонента
файлами с ")
        //.setCancelable(false)
        .setSingleChoiceItems(MemberList.toArray(new
String[0]), -1, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int
item) {
                shareFilesDevice = MemberList.get(item);
            }
        })
        .setPositiveButton("OK", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int
id) {
            }
        })
        .setNegativeButton("ОТМЕЧА",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface
dialog, int id) {
                    dialog.cancel();
                }
            });
    // create alert dialog
    AlertDialog alertDialog = alertDialogBuilder.create();
    // show it
    alertDialog.show();
}

private void initialWork() {
    Log.i(TAG, "initialWork");
    // Read in UI by ID defined in the XML
    btn_discover = findViewById(R.id.btn_discover);
    deviceListView = findViewById(R.id.device_list);

    connectionStatus = findViewById(R.id.connectionStatus);

    ///wifi manager
    wifiManager = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE); //
initialise the wifi manager

```

```

        mManager = (WifiP2pManager)
getSystemService(Context.WIFI_P2P_SERVICE);
        mChannel = mManager.initialize(this, getMainLooper(),
null);
        mReceiver = new WifiDirectBroadcastReceiver(mManager,
mChannel, this);

        //intents
        mIntentFilter = new IntentFilter();

mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);

mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);

mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);

mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);
    }

    @Override
    protected void onResume() {
        Log.i(TAG, "onResume");
        super.onResume();
        registerReceiver(mReceiver, mIntentFilter);
        connectionStatus.setText("СТАТУС ПОДКЛЮЧЕНИЯ ");
        if (mManager != null && mChannel != null) {
            if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                return;
            }
            mManager.requestGroupInfo(mChannel, new
WifiP2pManager.GroupInfoListener() {
                @Override
                public void onGroupInfoAvailable(WifiP2pGroup
group) {
                    if (group != null && mManager != null &&
mChannel != null) {
                        mManager.removeGroup(mChannel, new
WifiP2pManager.ActionListener() {
                            @Override
                            public void onSuccess() {
                                Log.d(TAG, "removeGroup onSuccess -
");
                            }
                        }
                    }
                }
            }
        }
    }

```

```

        public void onFailure(int reason) {
            Log.d(TAG, "removeGroup onFailure -
" + reason);
        }
    });
}
});
}
}

@Override
protected void onPause() {
    Log.i(TAG, "onPause");
    super.onPause();
    unregisterReceiver(mReceiver);
}

@Override
protected void onDestroy() {
    Log.i(TAG, "onDestroy");
    connectionStatus.setText(" ");
    ///закрыть все инпуты и аутпуты и сокеты
    if (isServer) {
        try {
            serverClassAudio.serverAudioSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            serverClassAudio.socketAudio.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            serverClassFile.serverFileSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            serverClassFile.socketFile.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    } else {
        try {
            clientClassAudio.socketAudio.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {

```

```

        clientClassFile.socketFile.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
super.onDestroy();
}

// Получение имен устройств после сканирования и вывод их на
экран
WifiP2pManager.PeerListListener peerListListener = new
WifiP2pManager.PeerListListener() {

    @Override
    public void onPeersAvailable(WifiP2pDeviceList
wifiP2pDeviceList) {
        Log.i(TAG, "onPeersAvailable");
        deviceNames.clear();

        if (wifiP2pDeviceList.getDeviceList().size() > 0) {
            deviceList = new
ArrayList(wifiP2pDeviceList.getDeviceList());
            for (WifiP2pDevice device : deviceList) {
                deviceNames.add(device.deviceName);
                Log.i(TAG, "deviceNames содержит: " +
deviceNames);
            }
            ArrayAdapter adapter = new
ArrayAdapter(getApplicationContext(),
android.R.layout.simple_list_item_1, deviceNames);
            deviceListView.setAdapter(adapter);
            Log.i(TAG, "setAdapter");
            if ((file_path[0] != null) && (flagFind ==
false) && (!iSendFilesDevice.equals("*"))){
                Log.i(TAG, "setAdapter поиск");
                int i = searchItem(iSendFilesDevice);
                Log.i(TAG, "searchItem");
                if (i != 10000) {
                    Log.i(TAG, "searchItem нашел");
                    flagFind = true;
                    wantToTalk = true;
                    wantSendFile = "sendfile";
                    connectToDevice(i);
                }
            }
        }
    }
};

// Соединение с выбранным устройством
void connectToDevice(int position) {

```

```

        if (flagConnect == false) {
            Log.i(TAG, "connectToDevice Start ");
            connectionStatus.setText("ИДЕТ ПОДКЛЮЧЕНИЕ");

final WifiP2pDevice toConnect = deviceList.get(position);
            WifiP2pConfig config = new WifiP2pConfig();
            config.deviceAddress = toConnect.deviceAddress;
            config.groupOwnerIntent = 15;
            if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                Log.i(TAG, "checkSelfPermission connectToDevice");
                return;
            }
            mManager.connect(mChannel, config, new
WifiP2pManager.ActionListener() {
                @Override
                public void onSuccess() {
                    flagConnect = true;
                    Log.i(TAG, "Успешно подключено: " +
toConnect.deviceName);
                    connectionStatus.setText("ПОДКЛЮЧЕНО К " +
toConnect.deviceName);
                }
                @Override
                public void onFailure(int reason) {
                    Log.i(TAG, "Ошибка подключения. Код: " +
reason);
                    connectionStatus.setText("ОШИБКА ПОДКЛЮЧЕНИЯ");
                }
            });
        }

        }

        public void disconnect(final WifiP2pManager mManager, final
WifiP2pManager.Channel mChannel, IntentFilter mIntentFilter, final
WifiDirectBroadcastReceiver mReceiver) {
            Log.i("", "disconnectDevice Start ");
            registerReceiver(mReceiver, mIntentFilter);
            connectionStatus.setText("СТАТУС ПОДКЛЮЧЕНИЯ ");
            if (mManager != null && mChannel != null) {
                if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                    return;
                }
                mManager.requestGroupInfo(mChannel, new
WifiP2pManager.GroupInfoListener() {
                    @Override

```

```

        public void onGroupInfoAvailable(WifiP2pGroup
group) {
            if (group != null && mManager != null &&
mChannel != null) {
                mManager.removeGroup(mChannel, new
WifiP2pManager.ActionListener() {
                    @Override
                    public void onSuccess() {
                        Log.d("", "removeGroup onSuccess -
");
                    }

                    @Override
                    public void onFailure(int reason) {
                        Log.d("", "removeGroup onFailure -"
+ reason);
                    }
                });
            }
        }

        WifiP2pManager.ConnectionInfoListener connectionInfoListener =
new WifiP2pManager.ConnectionInfoListener() {
            @Override
            public void onConnectionInfoAvailable(WifiP2pInfo
wifiP2pInfo) {
                Log.i("", "Решает, кто будет клиент, а кто хост ");
                final InetAddress groupOwnerAddress =
wifiP2pInfo.groupOwnerAddress;
                groupOwnerAddressForFile = groupOwnerAddress;
                if (wifiP2pInfo.groupFormed &&
wifiP2pInfo.isGroupOwner) {
                    isServer = true;
                    new ServerClassAudio().execute((Void[]) null);
//start thread
                    Log.i("", "isServer = true");

                } else if (wifiP2pInfo.groupFormed) {

                    isServer = false;
                    new
ClientClassAudio(groupOwnerAddress).execute((Void[]) null);
                    Log.i("", "isServer = false");
                }
            }
        };

```

```

    public class ServerClassAudio extends AsyncTask<Void, Void,
Void> {
        Socket socketAudio;
        ServerSocket serverAudioSocket;
        String signal = "";
        @Override
        public Void doInBackground(Void... voids) {
            Log.i("", "ServerClass start ");
            try {
                serverAudioSocket = new
ServerSocket(PORT_AUDIO_USED);
                socketAudio = serverAudioSocket.accept();

                SocketHandlerAudio.setSocket(socketAudio);
                Log.i("", "setSocket(socketAudio)");

                Log.i(TAG, "wantToTalk" + wantToTalk);
                if (wantToTalk == true) {
                    ObjectOutputStream signalobjectOutputStream =
new ObjectOutputStream(socketAudio.getOutputStream());

                    signalobjectOutputStream.writeObject(wantSendFile);
                    Log.i(TAG, "передал wantSendFile");

                    DataOutputStream newPortFileOutputStream = new
DataOutputStream(socketAudio.getOutputStream());

                    newPortFileOutputStream.writeInt(FirstActivity.PORT_FILE_USED);
                    newPortFileOutputStream.flush();
                    Log.d(TAG, "отправил newPort " +
FirstActivity.PORT_FILE_USED);

                } else {
                    try {
                        ObjectInputStream signalobjectInputStream =
new ObjectInputStream(socketAudio.getInputStream());
                        try {
                            Object object =
signalobjectInputStream.readObject();
                            signal = (String) object;
                            wantSendFile = signal;
                            Log.i(TAG, "Принял signal: " + signal);
                        } catch (ClassNotFoundException e) {
                            e.printStackTrace();
                        }
                    }
                    DataInputStream newPortInputStream = new
DataInputStream(socketAudio.getInputStream());
                    FirstActivity.PORT_FILE_USED =
newPortInputStream.readInt();
                    Log.d(TAG, "получил newPort " +
FirstActivity.PORT_FILE_USED);
                } catch (IOException e) {

```

```

        e.printStackTrace();
    }
}
if ((signal.equals("sendfile")) || (wantSendFile ==
"sendfile")) {
    startActivity(new
Intent(getApplicationContext(), FileActivity.class));
}else{
    if (wantToTalk == false) {
        try {

            ObjectOutputStream objectOutputStream =
new ObjectOutputStream(socketAudio.getOutputStream());

            objectOutputStream.writeObject(deviceNames);
            Log.i(TAG, "передал
neighboringDeviceList");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (wantToTalk == true) {
        try {
            ObjectInputStream objectInputStream =
new ObjectInputStream(socketAudio.getInputStream());
            try {
                Object object =
objectInputStream.readObject();
                neighboringDeviceList =
(ArrayList<String>) object;
                Log.i(TAG, "Принял
neighboringDeviceList: ");
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        Log.i(TAG, "передача publishProgress");
        publishProgress();
        if (isCancelled()) Log.i(TAG, "task уже
завершен");
        if (neighboringDeviceList.size() > 1) {
            sleep(10000);
        }
    }
    if (wantToTalk == true) {
        try {
            ObjectOutputStream objectOutputStream =
new ObjectOutputStream(socketAudio.getOutputStream());

            objectOutputStream.writeObject(shareFilesDevice);

```

```

        Log.i(TAG, "передал shareFilesDevice");
        startActivity(new
Intent(getApplicationContext(), ChatActivity.class));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (wantToTalk == false) {
        try {
            ObjectInputStream objectInputStream =
new ObjectInputStream(socketAudio.getInputStream());
            try {
                Object object =
objectInputStream.readObject();
                iSendFilesDevice = (String) object;
                Log.i(TAG, "Принял
shareFilesDevice" + iSendFilesDevice);
                startActivity(new
Intent(getApplicationContext(), ChatActivity.class));
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
} catch (IOException e) {
    e.printStackTrace();
}
} catch (InterruptedException e) {
    e.printStackTrace();
}
Log.i(TAG, " return null" );
return null;
}

protected void onProgressUpdate(Void... values) {
    super.onProgressUpdate();
    if (neighboringDeviceList.size() > 1) {
        alertDialogDemo();
    }
}

}

public class ClientClassAudio extends AsyncTask<Void, Void,
Void> {
    Socket socketAudio;
    String signal = "";
    String hostAddress;

    ClientClassAudio(InetAddress address) {

```

```

        this.socketAudio = new Socket();
        this.hostAddress = address.getHostAddress();
    }

    @Override
    public Void doInBackground(Void... voids) {
        Log.i("", "ClientClass start ");
        try {
            socketAudio.connect(new
            InetAddress(hostAddress, PORT_AUDIO_USED), 50000);
            SocketHandlerAudio.setSocket(socketAudio);
            Log.i("", "setSocket(socketAudio) ");

            if (file_path[0] != null) {
                startActivity(new
                Intent(getApplicationContext(), FileActivity.class));
            } else {
                Log.i(TAG, "wantToTalk" + wantToTalk);
                if (wantToTalk == true) {
                    ObjectOutputStream signalObjectOutputStream
                    = new ObjectOutputStream(socketAudio.getOutputStream());

                    signalObjectOutputStream.writeObject(wantSendFile);
                    Log.i(TAG, "передал wantSendFile");

                    DataOutputStream newPortFileOutputStream =
                    new DataOutputStream(socketAudio.getOutputStream());

                    newPortFileOutputStream.writeInt(FirstActivity.PORT_FILE_USED);
                    newPortFileOutputStream.flush();
                    Log.d(TAG, "отправил newPort " +
                    FirstActivity.PORT_FILE_USED);
                } else {
                    try {
                        ObjectInputStream
                        signalObjectInputStream = new
                        ObjectInputStream(socketAudio.getInputStream());

                        try {
                            Object object =
                            signalObjectInputStream.readObject();
                            signal = (String) object;
                            wantSendFile = signal;
                            Log.i(TAG, "Принял signal: " +
                            signal);
                        } catch (ClassNotFoundException e) {
                            e.printStackTrace();
                        }

                        DataInputStream newPortInputStream =
                        new DataInputStream(socketAudio.getInputStream());
                        FirstActivity.PORT_FILE_USED =
                        newPortInputStream.readInt();
                    }
                }
            }
        }
    }

```

```

                                Log.d(TAG, "получил newPort " +
FirstActivity.PORT_FILE_USED);
                                } catch (IOException e) {
                                    e.printStackTrace();
                                }
                            }
                            Log.i(TAG, "ППроверка signal: " + signal);
                            if ((signal.equals("sendfile")) ||
(wantSendFile == "sendfile")) {
                                Log.i(TAG, "ППроверка signal2: " + signal);
                                startActivity(new
Intent(getApplicationContext(), FileActivity.class));
                            } else {
                                if (wantToTalk == false) {
                                    try {
                                        ObjectOutputStream
objectOutputStream = new
ObjectOutputStream(socketAudio.getOutputStream());

                                objectOutputStream.writeObject(deviceNames);
                                    Log.i(TAG, "передал
neighboringDeviceList");
                                    } catch (IOException e) {
                                        e.printStackTrace();
                                    }
                                }
                                if (wantToTalk == true) {
                                    try {
                                        ObjectInputStream objectInputStream
= new ObjectInputStream(socketAudio.getInputStream());
                                        try {
                                            Object object =
objectInputStream.readObject();
                                            neighboringDeviceList =
(ArrayList<String>) object;
                                            Log.i(TAG, "Принял
neighboringDeviceList: ");
                                        } catch (ClassNotFoundException e)
{
                                            e.printStackTrace();
                                        }
                                    } catch (IOException e) {
                                        e.printStackTrace();
                                    }
                                }

                                Log.i(TAG, "передача publishProgress");
                                publishProgress();

                                if (isCancelled()) Log.i(TAG, "task уже
завершен");

                                if (neighboringDeviceList.size() > 1) {

```

```

        sleep(10000);
    }
}
if (wantToTalk == true) {
    try {

ObjectOutputStream objectOutputStream = new
ObjectOutputStream(socketAudio.getOutputStream());

objectOutputStream.writeObject (shareFilesDevice);
        Log.i (TAG, "передал
shareFilesDevice");

        startActivity(new
Intent (getApplicationContext(), ChatActivity.class));
    } catch (IOException e) {
        e.printStackTrace();
    }
}
if (wantToTalk == false) {
    try {
        ObjectInputStream objectInputStream
= new ObjectInputStream(socketAudio.getInputStream());
        try {
            Object object =
objectInputStream.readObject();

            iSendFilesDevice = (String)
object;

            Log.i (TAG, "Принял
shareFilesDevice " + iSendFilesDevice);
            startActivity(new
Intent (getApplicationContext(), ChatActivity.class));
        } catch (ClassNotFoundException e)
{
            e.printStackTrace();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
    }
} catch (IOException e) {
    e.printStackTrace();
} catch (InterruptedException e) {
    e.printStackTrace();
}
return null;
}

protected void onProgressUpdate(Void... values) {
    super.onProgressUpdate();
}

```

```

        if (neighboringDeviceList.size() > 1) {
            alertDialogDemo();
        }
    }
}

```

Файл ChatActivity.java

```

package com.example.meshandroid;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ImageButton;
import java.io.IOException;
import java.net.Socket;

public class ChatActivity extends AppCompatActivity {

    private static final int MESSAGE_READ = 1;
    private static boolean isRecording = false;
    private MicRecorder micRecorder;
    Thread t;
    boolean microon = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Log.e("Chat_activity", "onCreate");
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_chat);

        ImageButton micro_btn = (ImageButton)
        findViewById(R.id.btn_micro);
        ImageButton filesend_btn = (ImageButton)
        findViewById(R.id.btn_filesend);

        Log.e("OUTPUT_SOCKET", "SUCCESS");
        startService(new Intent(this,
        AudioStreamingService.class));
        Log.e("OUTPUT_SOCKET", "SUCCESS after");

        micro_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (microon == false) {
                    // stream audio
                    microon = true;
                    micro_btn.setImageResource(R.drawable.microon);
                    micRecorder = new MicRecorder();
                }
            }
        });
    }
}

```

```

        t = new Thread(micRecorder);
        if (micRecorder != null) {
            MicRecorder.keepRecording = true;
        }
        t.start();
    } else if (microon == true) {
        microon = false;
    }

    micro_btn.setImageResource(R.drawable.microoff);
    if (micRecorder != null) {
        MicRecorder.keepRecording = false;
    }
}

});
filesend_btn.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v) {
        Log.e("Chat_Activity", "playAudio = false");
        stopService(new Intent(getApplicationContext(),
        AudioStreamingService.class));
        startActivity(new Intent(getApplicationContext(),
        FileActivity.class));
    }
});

@Override
public void onBackPressed() {
    Socket socket = SocketHandlerAudio.getSocket();
    try {
        socket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    stopService(new Intent(getApplicationContext(),
    FileTransferService.class));
    startActivity(new Intent(getApplicationContext(),
    FirstActivity.class));
    finish();
}

@Override
protected void onDestroy() {
    super.onDestroy();
    if (micRecorder != null) {
        MicRecorder.keepRecording = false;
    }
}
}

```

Файл FileActivity.java

```
package com.example.meshandroid;

import static
com.example.meshandroid.FirstActivity.groupOwnerAddressForFile;
import static com.example.meshandroid.FirstActivity.isServer;
import static com.example.meshandroid.FirstActivity.wantSendFile;

import android.content.BroadcastReceiver;
import android.content.ContentUris;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.database.Cursor;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.provider.DocumentsContract;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.io.File;
import java.io.IOException;
import java.net.Socket;
import java.util.Arrays;

public class FileActivity extends AppCompatActivity {
    public static FileActivity fileActivity;
    private FileSend fileSend;
    Thread t;
    static File file;
    static String fileName;
    static Float file_size;
    static int count;
    boolean iSendFile;
    static boolean pressButton;
    static boolean closeRecieve;
    static boolean closeAct;

    public static final String BROADCAST_ACTION =
"action_session_closed";
    private BroadcastReceiver broadcastReceiver;
    private static final int PERMISSION_STORAGE = 101;
    Button send_btn;
```

```

static TextView fileStatus;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_file);
    fileActivity = this;
    closeRecieve = false;
    closeAct = false;
    iSendFile = false;
    pressButton = false;
    fileStatus = findViewById(R.id.FileStatus);
    send_btn = (Button) findViewById(R.id.button_send);

    if (PermissionUtils.hasPermissions(this)) {
        send_btn.setEnabled(true); //разрешение есть
    } else {
        send_btn.setEnabled(false); //разрешение не
предоставлено
        PermissionUtils.requestPermissions(FileActivity.this,
PERMISSION_STORAGE);
    }

    if (wantSendFile == "sendfile") {
        count = 0;
        while (FirstActivity.file_path[count] != null) {
            count++;
        }
        Log.e("File_Activity", "count = " + count);
        connectsend();
    } else {
        if (isServer == true) {
            new ServerClassFile().execute((Void[]) null);
//start thread
        } else {
            new
ClientClassFile(groupOwnerAddressForFile).execute((Void[]) null);
//start thread
        }
        count = 1;
        Log.e("File_Activity", "SUCCESS");
        startService(new Intent(this,
FileTransferService.class));
        Log.e("File_Activity", "SUCCESS after");

        send_btn.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View view) {
        iSendFile = true;
        pressButton = true;
    }
});
    }
}

```

```

        stopService(new Intent(getApplicationContext(),
FileTransferService.class));
        Intent i = new
Intent().setType("file/*").setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(Intent.createChooser(i,
"Выбор файла"), 123);
        fileStatus.setText("ОТПРАВКА ФАЙЛА");
    }
});
// Обработчик получения сообщений от сервиса сессии
broadcastReceiver = new BroadcastReceiver() {
    // действия при получении сообщений - тут мы
получаем сообщение о том, что сессия завершилась
    public void onReceive(Context context, Intent
intent) {
        boolean sessionClosed =
intent.getBooleanExtra("sessionClosed", false);
        if (iSendFile == true) {
            unregisterReceiver(broadcastReceiver);
            onBackPressed();
        } else {
            if (count - 1 != 0) {
                unregisterReceiver(broadcastReceiver);
                Socket socket =
SocketHandlerFile.getSocket();
                connectreseive();
            } else {
                unregisterReceiver(broadcastReceiver);
                onBackPressed();
            }
        }
    }
};
// создаем фильтр для BroadcastReceiver
IntentFilter intFilter = new
IntentFilter(BROADCAST_ACTION);
// регистрируем (включаем) BroadcastReceiver
registerReceiver(broadcastReceiver, intFilter);
}
}

public void connectreseive() {
    if (isServer == true) {
        new ServerClassFile().execute((Void[]) null); //start
thread
    } else {
        new
ClientClassFile(groupOwnerAddressForFile).execute((Void[]) null);
//start thread
    }
    startService(new Intent(this, FileTransferService.class));
}

```

```

        broadcastReceiver = new BroadcastReceiver() {
            // действия при получении сообщений - тут мы получаем
            // сообщение о том, что сессия завершилась
            public void onReceive(Context context, Intent intent) {
                boolean sessionClosed =
                intent.getBooleanExtra("sessionClosed", false);
                if (isSendFile == true) {
                    unregisterReceiver(broadcastReceiver);
                    onBackPressed();
                } else {
                    if (count - 1 != 0) {
                        unregisterReceiver(broadcastReceiver);
                        Socket socket =
                        SocketHandlerFile.getSocket();
                        connectreseive();
                    } else {
                        unregisterReceiver(broadcastReceiver);
                        onBackPressed();
                    }
                }
            }
        };
        // создаем фильтр для BroadcastReceiver
        IntentFilter intFilter = new
        IntentFilter(BROADCAST_ACTION);
        // регистрируем (включаем) BroadcastReceiver
        registerReceiver(broadcastReceiver, intFilter);
    }

    public void connectsend() {
        if (isServer == true) {
            new ServerClassFile().execute((Void[]) null); //start
thread
        } else {
            new
            ClientClassFile(groupOwnerAddressForFile).execute((Void[]) null);
            //start thread
        }

        int i = 0;
        send_btn.setClickable(false);
        fileName =
        FirstActivity.file_path[i].substring(FirstActivity.file_path[i].las
tIndexOf("/") + 1);
        Log.d("File Path", FirstActivity.file_path[i]);
        file = new File(FirstActivity.file_path[i]);
        if (file.canRead()) {
            Log.d("File ", "canRead");
        }
        file_size = Float.parseFloat(String.valueOf(file.length() /
1024));
    }

```

```

        Log.d("file size.....", Float.toString(file_size) +
        "Kb");

        send_btn.setClickable(false);
        fileStatus.setText("ОТПРАВКА ФАЙЛА");
        t = new Thread(new FileSend());
        t.start();
        broadcastReceiver = new BroadcastReceiver() {
            // действия при получении сообщений - тут мы получаем
            // сообщение о том, что сессия завершилась
            public void onReceive(Context context, Intent intent) {
                boolean sessionClosed =
                intent.getBooleanExtra("sessionClosed", false);
                count--;
                Log.d("count = ", String.valueOf(count));
                FirstActivity.file_path =
                removeFirstElement(FirstActivity.file_path);
                Log.d("File Path", FirstActivity.file_path[0]);
                unregisterReceiver(broadcastReceiver);
                Socket socket = SocketHandlerFile.getSocket();
                try {
                    socket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                if (count == 0) {
                    stopService(new Intent(getApplicationContext(),
                    FileTransferService.class));
                    startActivity(new
                    Intent(getApplicationContext(), ChatActivity.class));
                    finish();
                } else {
                    connectsend();
                }
            }
        };
        // создаем фильтр для BroadcastReceiver
        IntentFilter intFilter = new
        IntentFilter(BROADCAST_ACTION);
        // регистрируем (включаем) BroadcastReceiver
        registerReceiver(broadcastReceiver, intFilter);
    }

    public static String[] removeFirstElement(String[] arr) {
        return Arrays.copyOfRange(arr, 1, arr.length);
    }

    @Override
    protected void onActivityResult(int requestCode, int
    resultCode, Intent data) {
        if (requestCode == PERMISSION_STORAGE) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {

```

```

        if (PermissionUtils.hasPermissions(this)) {
            send_btn.setEnabled(true); //разрешение есть
        } else {
            send_btn.setEnabled(false); //разрешения нет
        }
    }
}
if (requestCode == 123) {
    Uri selectedImageUri = data.getData();
    final String filePath =
getPath(getApplicationContext(), selectedImageUri);
    fileName = filePath.substring(filePath.lastIndexOf("/")
+ 1);

    Log.d("File Path", filePath);
    file = new File(filePath);
    if (file.canRead()) {
        Log.d("File ", "canRead");
    }
    file_size =
Float.parseFloat(String.valueOf(file.length() / 1024));
    Log.d("file size.....", Float.toString(file_size)
+ "Kb");

    t = new Thread(new FileSend());
    t.start();
}
super.onActivityResult(requestCode, resultCode, data);
}

public static String getPath(final Context context, final Uri
uri) {

    final boolean isKitKat = Build.VERSION.SDK_INT >=
Build.VERSION_CODES.KITKAT;
    // DocumentProvider
    if (isKitKat && DocumentsContract.isDocumentUri(context,
uri)) {
        // ExternalStorageProvider
        if (isExternalStorageDocument(uri)) {
            final String docId =
DocumentsContract.getDocumentId(uri);
            final String[] split = docId.split(":");
            final String type = split[0];

            if ("primary".equalsIgnoreCase(type)) {
                return
Environment.getExternalStorageDirectory() + "/" + split[1];
            }
        }
        // DownloadsProvider
        else if (isDownloadsDocument(uri)) {

```

```

        final String id =
DocumentsContract.getDocumentId(uri);
        final Uri contentUri = ContentUris.withAppendedId(
Uri.parse("content://downloads/public_downloads"), Long.valueOf(id)
);
        return getDataColumn(context, contentUri, null,
null);
    }
    // MediaProvider
    else if (isMediaDocument(uri)) {
        final String docId =
DocumentsContract.getDocumentId(uri);
        final String[] split = docId.split(":");
        final String type = split[0];

        Uri contentUri = null;
        if ("image".equals(type)) {
            contentUri =
MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
        } else if ("video".equals(type)) {
            contentUri =
MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
        } else if ("audio".equals(type)) {
            contentUri =
MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
        }
        final String selection = "_id=?";
        final String[] selectionArgs = new String[]{
            split[1]
        };
        return getDataColumn(context, contentUri,
selection, selectionArgs);
    }
}
// MediaStore (and general)
else if ("content".equalsIgnoreCase(uri.getScheme())) {
    // Return the remote address
    if (isGooglePhotosUri(uri))
        return uri.getLastPathSegment();
    return getDataColumn(context, uri, null, null);
}
// File
else if ("file".equalsIgnoreCase(uri.getScheme())) {
    return uri.getPath();
}
return null;
}

public static String getDataColumn(Context context, Uri uri,
String selection, String[] selectionArgs) {
    Cursor cursor = null;

```

```

        final String column = "_data";
        final String[] projection = {
            column
        };
        try {
            cursor = context.getContentResolver().query(uri,
projection, selection, selectionArgs, null);
            if (cursor != null && cursor.moveToFirst()) {
                final int index =
cursor.getColumnIndexOrThrow(column);
                return cursor.getString(index);
            }
        } finally {
            if (cursor != null)
                cursor.close();
        }
        return null;
    }
    public static boolean isExternalStorageDocument(Uri uri) {
        return
"com.android.externalstorage.documents".equals(uri.getAuthority());
    }
    public static boolean isDownloadsDocument(Uri uri) {
        return
"com.android.providers.downloads.documents".equals(uri.getAuthority
());
    }
    public static boolean isMediaDocument(Uri uri) {
        return
"com.android.providers.media.documents".equals(uri.getAuthority());
    }
    public static boolean isGooglePhotosUri(Uri uri) {
        return
"com.google.android.apps.photos.content".equals(uri.getAuthority())
;
    }
    @Override
    public void onBackPressed() {
        closeRecieve = true;
        Socket socket = SocketHandlerFile.getSocket();
        try {
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        stopService(new Intent(getApplicationContext(),
FileTransferService.class));
        startActivity(new Intent(getApplicationContext(),
ChatActivity.class));
        finish();
    }
    @Override

```

```

        protected void onDestroy() {
            super.onDestroy();
            stopService(new Intent(getApplicationContext(),
FileTransferService.class));
        }
    }
}

```

Файл AudioStreamingService.java

```

package com.example.meshandroid;

import android.app.Service;
import android.content.Intent;
import android.media.AudioFormat;
import android.media.AudioManager;
import android.media.AudioTrack;
import android.os.Build;
import android.os.IBinder;
import android.util.Log;
import androidx.annotation.RequiresApi;
import java.io.IOException;
import java.io.InputStream;

public class AudioStreamingService extends Service {
    private static final int SAMPLE_RATE = 16000;
    public static boolean keepPlaying = true;
    private static AudioTrack audioTrack;
    byte[] buffer;
    static int bufferSize;

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet
implemented");
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    public int onStartCommand(Intent intent, int flags, int
startId) {
        startStreaming();
        Log.e("OUTPUT_SOCKET", "startStreaming");
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public void onDestroy() {
        Log.e("PLAY", "onDestroy");
        keepPlaying = false;
        if (audioTrack != null)
            audioTrack.release();
    }
}

```

```

    }
    static public void startStreaming() {
        keepPlaying = true;
        bufferSize = 1280;
        AudioTrack.getMinBufferSize(SAMPLE_RATE,
AudioFormat.CHANNEL_OUT_MONO,
        AudioFormat.ENCODING_PCM_16BIT);

if (bufferSize == AudioTrack.ERROR || bufferSize ==
AudioTrack.ERROR_BAD_VALUE) {
    bufferSize = SAMPLE_RATE * 2;
}
Log.e("PLAY", "buffersize = " + bufferSize);

    audioTrack = new AudioTrack(
        AudioManager.STREAM_MUSIC,
        SAMPLE_RATE,
        AudioFormat.CHANNEL_OUT_MONO,
        AudioFormat.ENCODING_PCM_16BIT,
        bufferSize,
        AudioTrack.MODE_STREAM);

    audioTrack.play();

    Log.v("PLAY", "Audio streaming started");

    byte[] buffer = new byte[bufferSize];
    int offset = 0;
    Runnable audioPlayerRunnable = new Runnable() {
        @Override
        public void run() {
            try {
                Log.e("PLAY", "InputStream");
                InputStream inputStream =
SocketHandlerAudio.getSocket().getInputStream();
                int bytes_read = inputStream.read(buffer, 0,
bufferSize);

                while (keepPlaying && (bytes_read != -1)) {
                    audioTrack.write(buffer, 0, buffer.length);
                    bytes_read = inputStream.read(buffer, 0,
bufferSize);

                    Log.e("PLAY", "inputStream.read");

                }
                audioTrack.release();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (NullPointerException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

    };
    Thread t = new Thread(audioPlayerRunnable);
    t.start();
}
}

```

Файл ClientClassFile.java

```

package com.example.meshandroid;

import static com.example.meshandroid.FirstActivity.PORT_FILE_USED;
import static java.lang.Thread.sleep;
import android.os.AsyncTask;
import android.util.Log;
import java.io.IOException;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.Socket;

public class ClientClassFile extends AsyncTask<Void, Void, Void> {
    Socket socketFile;
    String hostAddress;
    ClientClassFile(InetAddress address) {
        this.socketFile = new Socket();
        this.hostAddress = address.getHostAddress();
    }

    @Override
    public Void doInBackground(Void... voids) {

        Log.i("", "ClientClass start ");
        try {
            sleep(2000);
            Log.i("", "PORT_FILE_USED = " + PORT_FILE_USED);
            socketFile.connect(new InetSocketAddress(hostAddress,
PORT_FILE_USED), 50000);
            SocketHandlerFile.setSocket(socketFile);
            Log.i("", "setSocket(socketFile) ");
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return null;
    }
}

```

Файл FileSend.java

```

package com.example.meshandroid;

```

```

import static com.example.meshandroid.FileActivity.fileActivity;
import static java.lang.Thread.sleep;
import android.content.Intent;
import android.util.Log;
import java.io.BufferedOutputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import javax.crypto.Cipher;

public class FileSend implements Runnable {
    byte[] key;

    @Override
    public void run() {
        try {

            KeyPairGenerator keyPairGen =
KeyPairGenerator.getInstance("RSA");
            keyPairGen.initialize(1024);
            KeyPair pair = keyPairGen.generateKeyPair();
            PrivateKey privKey = pair.getPrivate();
            PublicKey publicKey = pair.getPublic();

            Kuznechik Kuz = new Kuznechik();
            if (FirstActivity.file_path[0] != null) {
                sleep(5000);
            }
            Log.d("Send", "FileSend начал");
            FirstActivity.PORT_FILE_USED = (int) ((Math.random() *
(9900 - 3500)) + 3500);

            ObjectOutputStream keyObjectOutputStream = new
ObjectOutputStream(SocketHandlerFile.getSocket().getOutputStream());
;
            keyObjectOutputStream.writeObject(publicKey);
            keyObjectOutputStream.flush();
            Log.d("Send", "отправил key" + publicKey);

            ObjectInputStream otherKeyInputStream = new
ObjectInputStream(SocketHandlerFile.getSocket().getInputStream());
            Object keyobject = otherKeyInputStream.readObject();

```

```

byte[] otherKey = (byte[]) keyobject;
Log.d("Send", "получил otherKey " + otherKey);

Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.DECRYPT_MODE, privKey);
byte[] result = cipher.doFinal(otherKey);
Log.d("Send", "расшифровал otherKey " + result);

final DataOutputStream countFileOutputStream = new
DataOutputStream(SocketHandlerFile.getSocket().getOutputStream());
countFileOutputStream.writeInt(FileActivity.count);
countFileOutputStream.flush();
Log.d("Send", "отправил count " + FileActivity.count);

final DataOutputStream newPortFileOutputStream = new
DataOutputStream(SocketHandlerFile.getSocket().getOutputStream());
newPortFileOutputStream.writeInt(FirstActivity.PORT_FILE_USED);
newPortFileOutputStream.flush();
Log.d("Send", "отправил newPort " + FirstActivity.
PORT_FILE_USED);

final DataOutputStream sizeFileOutputStream = new
DataOutputStream(SocketHandlerFile.getSocket().getOutputStream());
sizeFileOutputStream.writeFloat(FileActivity.file_size);
sizeFileOutputStream.flush();
Log.d("Send", "отправил file_size " +
FileActivity.file_size);

ObjectOutputStream nameFileObjectOutputStream = new
ObjectOutputStream(SocketHandlerFile.getSocket().getOutputStream())
;

nameFileObjectOutputStream.writeObject(FileActivity.fileName);
nameFileObjectOutputStream.flush();
Log.d("Send", "отправил file_name " +
FileActivity.fileName);

final BufferedOutputStream outputStream = new
BufferedOutputStream(SocketHandlerFile.getSocket().getOutputStream(
));

final byte[] fileBuffer = new byte[8192];
byte[] encryptFileBuffer = new byte[8192];
if (FileActivity.file.exists()) {
    Log.d("Send", "файл существует");
}
FileActivity.file.createNewFile();
Log.d("Send", " file: " + FileActivity.file.getPath());
FileInputStream fis = new
FileInputStream(FileActivity.file.getPath());

```

```

        long fileSizeByte;
        fileSizeByte = FileActivity.file.length();
        while (fileSizeByte > 0) {
            int i = fis.read(fileBuffer);
            encriptFileBuffer = Kuz.Encript(fileBuffer,
result);

            outputStream.write(encriptFileBuffer, 0, i);
            Log.d("Send", "отправил encriptFileBuffer");
            fileSizeByte -= i;
        }
        Log.d("Send", "отправил file");
        FileActivity.fileStatus.setText("ФАЙЛ ОТПРАВЛЕН");

        keyObjectOutputStream.close();
        countFileOutputStream.close();
        newPortFileOutputStream.close();
        outputStream.flush();
        fis.close();
        sizeFileOutputStream.close();
        nameFileObjectOutputStream.close();
        outputStream.close();

        sleep(2000);
        Intent intent = new
Intent(FileActivity.BROADCAST_ACTION);
        intent.putExtra("sessionClosed", true);
        fileActivity.sendBroadcast(intent);

    } catch (FileNotFoundException e) {
        System.err.println("File not found!");
    } catch (IOException e) {
        System.err.println("IOException");
    } catch (Exception e) {

    }
}
}

```

Файл FileSend.java

```

package com.example.meshandroid;

import static com.example.meshandroid.FileActivity.pressButton;
import static java.lang.Thread.sleep;
import android.app.Service;
import android.content.Intent;
import android.os.Environment;
import android.os.IBinder;
import android.util.Log;
import androidx.annotation.Nullable;
import java.io.BufferedInputStream;

```

```

import java.io.DataInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.InvalidKeyException;

import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PublicKey;
import java.util.Random;
import java.util.stream.Collectors;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

public class FileTransferService extends Service {
    float file_size;
    String file_name;
    int count;
    byte[] key;

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int
startId) {
        startRecieve();
        Log.e("Recieve", "start listening");
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public void onDestroy() {
        Log.e("Recieve", "onDestroy");
    }

    public void startRecieve() {
        Runnable recieveFile = new Runnable() {
            @Override
            public void run() {
                try {
                    String symbols =
"54643sfawgeweqwretrety5643jukuy1231231";

```

```

        String random = new Random().ints(32, 0,
symbols.length())
                .mapToObj(symbols::charAt)
                .map(Object::toString)
                .collect(Collectors.joining());
        byte[] myFilekey = random.getBytes();
        Log.d("Recieve", "мой ключ " + myFilekey);
        KeyPairGenerator keyPairGen =
KeyPairGenerator.getInstance("RSA");
        keyPairGen.initialize(1024);
        PublicKey publicKey;
        Kuznechik Kuz = new Kuznechik();
        Log.d("Recieve", "начал ");
        sleep(10000);
        Log.d("Recieve", "подождал ");

        if (!SocketHandlerFile.getSocket().isClosed()
&& (pressButton != true)) {
                ObjectInputStream keyobjectInputStream =
new
ObjectInputStream(SocketHandlerFile.getSocket().getInputStream());
                Object keyobject =
keyobjectInputStream.readObject();
                publicKey = (PublicKey) keyobject;
                Log.d("Recieve", "получил key " +
publicKey);

                Cipher cipher = Cipher.getInstance("RSA");
                cipher.init(Cipher.ENCRYPT_MODE,
publicKey);

                byte[] data = cipher.doFinal(myFilekey);

                ObjectOutputStream keyObjectOutputStream =
new
ObjectOutputStream(SocketHandlerFile.getSocket().getOutputStream())
;
                keyObjectOutputStream.writeObject(data);
                keyObjectOutputStream.flush();
                Log.d("Send", "отправил key " + data);

                DataInputStream countFilesInputStream = new
DataInputStream(SocketHandlerFile.getSocket().getInputStream());
                FileActivity.count =
countFilesInputStream.readInt();
                FileActivity.fileStatus.setText("ЗАГРУЗКА
ФАЙЛА");

                Log.d("Recieve", "получил count " +
FileActivity.count);

                DataInputStream newPortInputStream = new
DataInputStream(SocketHandlerFile.getSocket().getInputStream());

```

```

        FirstActivity.PORT_FILE_USED =
newPortInputStream.readInt();
        Log.d("Recieve", "получил newPort " +
FirstActivity.PORT_FILE_USED);

        DataInputStream sizeFileInputStream = new
DataInputStream(SocketHandlerFile.getSocket().getInputStream());
        file_size = sizeFileInputStream.readFloat()
* 1024;
        Log.d("Recieve", "получил file_size " +
file_size);

ObjectInputStream objectInputStream = new
ObjectInputStream(SocketHandlerFile.getSocket().getInputStream());
Object object =
objectInputStream.readObject();
        file_name = (String) object;
        Log.d("Recieve", "получил file_name " +
file_name);

        byte[] byteArray = new byte[8192];
        byte[] DecriptbyteArray = new byte[8192];
        File externalAppDir = null;
        try {
            externalAppDir = new
File(Environment.getExternalStorageDirectory() + "/Android/data/" +
getPackageName());
            if (!externalAppDir.exists()) {
                Log.d("Recieve", "директории не
существует ");
                Log.d("Recieve", "создал директорию
" + externalAppDir.getPath());
                Files.createDirectory(Paths.get(externalAppDir.getPath()));
            }
            catch (IOException e) {
                Log.d("Recieve", "ошибка директории ");
                e.printStackTrace();
            }

            File f = new File(externalAppDir + "/" +
file_name);
            Log.d("Recieve", "путь " + f.getPath());
            Log.d("Recieve", "count " +
FirstActivity.counFiles);

            FirstActivity.file_path[FirstActivity.counFiles] = f.getPath();
            FirstActivity.counFiles++;
            f.createNewFile();

```

```

        FileOutputStream fos = new
FileOutputStream(f);

        BufferedInputStream bis = new
BufferedInputStream(SocketHandlerFile.getSocket().getInputStream())
;
        while (file_size > 0) {
            int i = bis.read(byteArray);
            DecriptbyteArray =
Kuz.Decript(byteArray, myFilekey);
            fos.write(DecriptbyteArray, 0, i);
            Log.d("Recieve", "получил
DecriptbyteArray");
            file_size -= i;
        }
        Log.d("Recieve", "получил file");
        FileActivity.fileStatus.setText("ФАЙЛ
ЗАГРУЖЕН");

        count++;

        fos.close();
        bis.close();
        sizeFileInputStream.close();
        objectInputStream.close();

        sleep(2000);
        Intent intent = new
Intent(FileActivity.BROADCAST_ACTION);
        intent.putExtra("sessionClosed", true);
        sendBroadcast(intent);
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    catch (NullPointerException e) {
        e.printStackTrace();
    }
    catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    catch (InterruptedException e) {
        e.printStackTrace();
    }
    catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    catch (NoSuchPaddingException e) {
        e.printStackTrace();
    }
    catch (IllegalBlockSizeException e) {
        e.printStackTrace();
    }
    catch (BadPaddingException e) {
        e.printStackTrace();
    }
    catch (InvalidKeyException e) {
        e.printStackTrace();
    }
}

};

```

```

        Thread t = new Thread(recieveFile);
        t.start();
    }
}

```

Файл Kuznechik.java

```

package com.example.meshandroid;

import android.util.Log;
import java.util.Arrays;

class Kuznechik{
    static byte[] int_to_byte(int[] res) {
        byte[] ut = new byte[res.length];
        for (int i = 0; i < res.length; i++) {
            ut[i] = (byte) res[i];
        }
        return ut;
    }

    private String lengthTo32Bytes(String str) {
        Stribog hash = new Stribog();
        byte[]res = hash.getHash(str.getBytes());
        for (int i = 0; i < 10000; i++)
            res = hash.getHash(res);
        String out = "";
        for (byte b : res) {
            out += Integer.toHexString(b);
        }
        return out;
    }

    private byte[][] iterC = new byte[32][];
    private byte[][] iterK = new byte[10][];

    private final byte[] Pi = int_to_byte(new int[]{
        252, 238, 221, 17, 207, 110, 49, 22, 251, 196, 250,
218, 35, 197, 4, 77,
        233, 119, 240, 219, 147, 46, 153, 186, 23, 54, 241,
187, 20, 205, 95, 193,
        249, 24, 101, 90, 226, 92, 239, 33, 129, 28, 60, 66,
139, 1, 142, 79,
        5, 132, 2, 174, 227, 106, 143, 160, 6, 11, 237, 152,
127, 212, 211, 31,
        235, 52, 44, 81, 234, 200, 72, 171, 242, 42, 104, 162,
253, 58, 206, 204,
        181, 112, 14, 86, 8, 12, 118, 18, 191, 114, 19, 71,
156, 183, 93, 135,
        21, 161, 150, 41, 16, 123, 154, 199, 243, 145, 120,
111, 157, 158, 178, 177,

```

```

        50, 117, 25, 61, 255, 53, 138, 126, 109, 84, 198, 128,
195, 189, 13, 87,
        223, 245, 36, 169, 62, 168, 67, 201, 215, 121, 214,
246, 124, 34, 185, 3,
        224, 15, 236, 222, 122, 148, 176, 188, 220, 232, 40,
80, 78, 51, 10, 74,
        167, 151, 96, 115, 30, 0, 98, 68, 26, 184, 56, 130,
100, 159, 38, 65,
        173, 69, 70, 146, 39, 94, 85, 47, 140, 163, 165, 125,
105, 213, 149, 59,
        7, 88, 179, 64, 134, 172, 29, 247, 48, 55, 107, 228,
136, 217, 231, 137,
        225, 27, 131, 73, 76, 63, 248, 254, 141, 83, 170, 144,
202, 216, 133, 97,
        32, 113, 103, 164, 45, 43, 9, 91, 203, 155, 37, 208,
190, 229, 108, 82,

89, 166, 116, 210, 230, 244, 180, 192, 209, 102, 175, 194, 57, 75,
99, 182
    });

    private final byte[] Pi_Reverse = int_to_byte(new int[]{
        165, 45, 50, 143, 14, 48, 56, 192, 84, 230, 158, 57,
85, 126, 82, 145,
        100, 3, 87, 90, 28, 96, 7, 24, 33, 114, 168, 209, 41,
198, 164, 63,
        224, 39, 141, 12, 130, 234, 174, 180, 154, 99, 73, 229,
66, 228, 21, 183,
        200, 6, 112, 157, 65, 117, 25, 201, 170, 252, 77, 191,
42, 115, 132, 213,
        195, 175, 43, 134, 167, 177, 178, 91, 70, 211, 159,
253, 212, 15, 156, 47,
        155, 67, 239, 217, 121, 182, 83, 127, 193, 240, 35,
231, 37, 94, 181, 30,
        162, 223, 166, 254, 172, 34, 249, 226, 74, 188, 53,
202, 238, 120, 5, 107,
        81, 225, 89, 163, 242, 113, 86, 17, 106, 137, 148, 101,
140, 187, 119, 60,
        123, 40, 171, 210, 49, 222, 196, 95, 204, 207, 118, 44,
184, 216, 46, 54,
        219, 105, 179, 20, 149, 190, 98, 161, 59, 22, 102, 233,
92, 108, 109, 173,
        55, 97, 75, 185, 227, 186, 241, 160, 133, 131, 218, 71,
197, 176, 51, 250,
        150, 111, 110, 194, 246, 80, 255, 93, 169, 142, 23, 27,
151, 125, 236, 88,
        247, 31, 251, 124, 9, 13, 122, 103, 69, 135, 220, 232,
79, 29, 78, 4,
        235, 248, 243, 62, 61, 189, 138, 136, 221, 205, 11, 19,
152, 2, 147, 128,
        144, 208, 36, 52, 203, 237, 244, 206, 153, 16, 68, 64,
146, 58, 1, 38,

```

```

        18, 26, 72, 104, 245, 129, 139, 199, 214, 32, 10, 8, 0,
76, 215, 116
    });

    private final byte[] LVec = int_to_byte(new int[]{148, 32, 133,
16, 194, 192, 1, 251, 1, 192, 194, 16, 133, 32, 148, 1});

    private byte[] KuzX(byte[] input1, byte[] input2)
    {
        byte[] output = new byte[16];
        for (int i = 0; i < 16; i++) {
            output[i] = (byte) (input1[i] ^ input2[i]);
        }
        return output;
    }

    private byte[][] KuzF(byte[] input1, byte[] input2, byte[]
round_C) {
        byte[] state = new byte[16];
        state = KuzX(input1, round_C);
        state = KuzS(state);
        state = KuzL(state);
        return new byte[][]{KuzX(state, input2),
Arrays.copyOf(input1, input1.length)};
    }

    private void KuzKeyGen(byte[] mas_key) {

        byte[][] iterNum = new byte[32][];
        for (int i = 0; i < 32; i++) {
            iterNum[i] = new byte[]{0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, (byte) (i + 1)};
            iterC[i] = KuzL(iterNum[i]);
        }

        byte[] A = new byte[16];
        for (int i = 0; i < 16; i++) {
            A[i] = mas_key[i];
        }
        byte[] B = new byte[16];
        int j = 0;
        for (int i = 16; i < 32; i++) {
            B[j] = mas_key[i];
            j++;
        }
        j = 0;
        iterK[0] = B;
        iterK[1] = A;

        byte[] C = new byte[16];
        byte[] D = new byte[16];
        byte[][] tmp;

```

```

    for (int i = 0; i < 4; i++) {
        tmp = KuzF(A, B, iterC[0 + 8 * i]);

        C = tmp[0];
        D = tmp[1];

        tmp = KuzF(C, D, iterC[1 + 8 * i]);
        A = tmp[0];
        B = tmp[1];
        tmp = KuzF(A, B, iterC[2 + 8 * i]);
        C = tmp[0];
        D = tmp[1];
        tmp = KuzF(C, D, iterC[3 + 8 * i]);
        A = tmp[0];
        B = tmp[1];
        tmp = KuzF(A, B, iterC[4 + 8 * i]);
        C = tmp[0];
        D = tmp[1];
        tmp = KuzF(C, D, iterC[5 + 8 * i]);
        A = tmp[0];
        B = tmp[1];
        tmp = KuzF(A, B, iterC[6 + 8 * i]);
        C = tmp[0];
        D = tmp[1];
        tmp = KuzF(C, D, iterC[7 + 8 * i]);
        A = tmp[0];
        B = tmp[1];
        iterK[2 * i + 2] = A;
        iterK[2 * i + 3] = B;
    }

}

private byte[] KuzS(byte[] input)
{
    byte[] output = new byte[16];
    for (int i = 0; i < 16; i++) {
        int T = input[i];
        if (T < 0) {
            T += 256;
        }
        output[i] = Pi[T];
    }
    return output;
}

private byte[] KuzSReverse(byte[] input)
{
    byte[] output = new byte[16];
    for (int i = 0; i < 16; i++) {
        int T = input[i];
        if (T < 0) {

```

```

        T += 256;
    }
    output[i] = Pi_Reverse[T];
}
return output;
}

private byte KuzMulInGF(byte a, byte b) {
    byte p = 0;
    byte counter;
    byte hi_bit_set;
    for (counter = 0; counter < 8 && a != 0 && b != 0;
counter++) {
        if ((b & 1) != 0) {
            p ^= a;
        }
        hi_bit_set = (byte) (a & 0x80);
        a <<= 1;
        if (hi_bit_set != 0) {
            a ^= 0xc3;
        }
        b >>= 1;
    }
    return p;
}

private byte[] KuzR(byte[] input) {
    byte a_15 = 0;
    byte[] state = new byte[16];
    for (int i = 0; i <= 15; i++) {
        a_15 ^= KuzMulInGF(input[i], LVec[i]);
    }
    for (int i = 15; i > 0; i--) {
        state[i] = input[i - 1];
    }
    state[0] = a_15;
    return state;
}

private byte[] KuzL(byte[] input) {
    byte[] state = input;
    for (int i = 0; i < 16; i++) {
        state = KuzR(state);
    }
    return state;
}

private byte[] KuzRReverse(byte[] input) {
    byte a_15 = input[0];
    byte[] state = new byte[16];
    for (int i = 0; i < 15; i++) {
        state[i] = input[i + 1];
    }

```

```

    }
    for (int i = 15; i >= 0; i--) {
        a_15 ^= KuzMulInGF(state[i], LVec[i]);
    }
    state[15] = a_15;
    return state;
}

private byte[] KuzLReverse(byte[] input) {
    byte[] state = input;
    for (int i = 0; i < 16; i++) {
        state = KuzRReverse(state);
    }
    return state;
}

public byte[] Encrypt(byte[] file, byte[] masterKey) {
    return KuzEncrypt(file, masterKey);
}

private byte[] KuzEncrypt(byte[] file, byte[] masterKey) {
    Log.d("kuznechik", "KuzEncrypt");
    masterKey = (lengthTo32Bytes(new
String(masterKey))).getBytes();
    KuzKeyGen(masterKey);
    int NumOfBlocks;
    int NumberOfNull;
    byte[] OriginText = file;
    byte[] encrText = new byte[0];
    if ((file.length % 16) == 0) {
        NumOfBlocks = file.length / 16;
        encrText = Arrays.copyOf(encrText, file.length);
    } else {
        NumOfBlocks = (file.length / 16) + 1;
        NumberOfNull = NumOfBlocks * 16 - file.length;
        int Startlength = file.length;
        OriginText = Arrays.copyOf(OriginText,
OriginText.length + NumberOfNull);
        encrText = Arrays.copyOf(encrText, OriginText.length);
        if (NumberOfNull == 1) {
            OriginText[OriginText.length - 1] = (byte) 0x80;
        } else {
            for (int i = OriginText.length - 1; i >= 0; i--) {
                if (i == OriginText.length - 1) {
                    OriginText[OriginText.length - 1] = (byte)
(byte) -127;
                } else if (OriginText[i] != 0) {
                    OriginText[i + 1] = 0x01;
                    break;
                }
            }
        }
    }
    for (int i = 0; i < NumOfBlocks; i++)

```

```

    {
        byte[] block = new byte[16];
        for (int j = 0; j < 16; j++) {
            block[j] = OriginText[i * 16 + j];
        }
        for (int j = 0; j < 9; j++) {
            block = KuzX(block, iterK[j]);
            block = KuzS(block);
            block = KuzL(block);
        }
        block = KuzX(block, iterK[9]);
        for (int j = 0; j < 16; j++) {
            encrText[i * 16 + j] = block[j];
        }
    }
    return encrText;
}

public byte[] Decrypt(byte[] file, byte[] masterKey) {
    return KuzDecrypt(file, masterKey);
}

private byte[] KuzDecrypt(byte[] file, byte[] masterKey) {
    Log.d("kuznechik", "KuzDecrypt");
    masterKey = (lengthTo32Bytes(new
String(masterKey))).getBytes();
    KuzKeyGen(masterKey);
    int NumOfBlocks = file.length / 16;
    byte[] OriginText = file;
    byte[] decrText = new byte[file.length];
    for (int i = 0; i < NumOfBlocks; i++) {
        byte[] block = new byte[16];
        for (int j = 0; j < 16; j++) {
            block[j] = OriginText[i * 16 + j];
        }
        block = KuzX(block, iterK[9]);
        for (int j = 8; j >= 0; j--) {
            block = KuzLReverse(block);
            block = KuzSReverse(block);
            block = KuzX(block, iterK[j]);
        }
        for (int j = 0; j < 16; j++) {
            decrText[i * 16 + j] = block[j];
        }
        if (i == NumOfBlocks - 1 && (decrText[decrText.length -
1] == (byte) -127 || decrText[decrText.length - 1] == 0x80)) {
            if (decrText[decrText.length - 1] == (byte) -127) {
                int Zeros = 0;
                for (int j = decrText.length - 1; j > 0; j--) {
                    if (decrText[j] == (byte) -127 ||
decrText[j] == 0x01 || decrText[j] == 0) {
                        Zeros++;
                    } else {
                        break;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    decrText = Arrays.copyOf(decrText,
decrText.length - Zeros);
    }
    if (decrText[decrText.length - 1] == 0x80) {
        decrText = Arrays.copyOf(decrText,
decrText.length - 1);
    }
}
}
return decrText;
}
}

```

Файл MicRecorder.java

```

package com.example.meshandroid;
import android.annotation.SuppressLint;
import android.media.AudioFormat;
import android.media.AudioRecord;
import android.media.MediaRecorder;
import android.util.Log;
import java.io.IOException;
import java.io.OutputStream;

public class MicRecorder implements Runnable {
    private static final int SAMPLE_RATE = 16000;
    public static volatile boolean keepRecording = true;
    @Override
    public void run() {

        android.os.Process.setThreadPriority(android.os.Process.THREAD_PRIORITY_AUDIO);
        int bufferSize = AudioRecord.getMinBufferSize(SAMPLE_RATE,
        AudioFormat.CHANNEL_IN_MONO, AudioFormat.ENCODING_PCM_16BIT);
        Log.e("AUDIO", "bufferSize = " + bufferSize);
        if (bufferSize == AudioRecord.ERROR || bufferSize ==
        AudioRecord.ERROR_BAD_VALUE) {
            bufferSize = SAMPLE_RATE * 2;
        }
        try {
            final OutputStream outputStream =
        SocketHandlerAudio.getSocket().getOutputStream();
            final byte[] audioBuffer = new byte[bufferSize];
            @SuppressWarnings("MissingPermission") AudioRecord record =
        new AudioRecord(MediaRecorder.AudioSource.VOICE_RECOGNITION,
        SAMPLE_RATE, AudioFormat.CHANNEL_IN_MONO,
        AudioFormat.ENCODING_PCM_16BIT, bufferSize);
            if (record.getState() != AudioRecord.STATE_INITIALIZED)
        {

```

```

        Log.e("AUDIO", "Audio Record can't initialize!");
        return;
    }
    record.startRecording();
    Log.e("AUDIO", "STARTED RECORDING");

    while (keepRecording) {
        int numberOfBytes = record.read(audioBuffer, 0,
audioBuffer.length);
        Runnable writeToOutputStream = new Runnable() {
            @Override
            public void run() {
                try {
                    outputStream.write(audioBuffer);
                    Log.e("AUDIO", "outputStream.write ");
                    outputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        };
        Log.e("AUDIO", "writeToOutputStream");
        Thread thread = new Thread(writeToOutputStream);
        thread.start();
    }
    record.stop();
    record.release();
    Log.e("AUDIO", "Streaming stopped");
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

Файл PermissionUtils.java

```

package com.example.meshandroid;

import android.Manifest;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.os.Environment;
import android.provider.Settings;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

public class PermissionUtils {

```

```

        public static boolean hasPermissions(Context context) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
                return Environment.isExternalStorageManager();
            } else if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)
        {
            return ContextCompat.checkSelfPermission(context,
Manifest.permission.WRITE_EXTERNAL_STORAGE)
                == PackageManager.PERMISSION_GRANTED;
        } else {
            return true;
        }
    }

    public static void requestPermissions(Activity activity, int
requestCode) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            try {
                Intent intent = new
Intent(Settings.ACTION_MANAGE_APP_ALL_FILES_ACCESS_PERMISSION);

intent.addCategory("android.intent.category.DEFAULT");

intent.setData(Uri.parse(String.format("package:%s",
activity.getPackageName())));
                activity.startActivityForResult(intent,
requestCode);
            } catch (Exception e) {
                Intent intent = new Intent();

intent.setAction(Settings.ACTION_MANAGE_ALL_FILES_ACCESS_PERMISSION
);
                activity.startActivityForResult(intent,
requestCode);
            }
        } else {
            ActivityCompat.requestPermissions(activity,
                new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
                requestCode);
        }
    }
}

```

Файл ServerClassFile.java

```

package com.example.meshandroid;

import static com.example.meshandroid.FirstActivity.PORT_FILE_USED;
import android.os.AsyncTask;
import android.util.Log;
import java.io.IOException;

```

```

import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;

public class ServerClassFile extends AsyncTask<Void, Void, Void> {

    Socket socketFile;
    ServerSocket serverFileSocket;

    @Override
    public Void doInBackground(Void... voids) {
        Log.i("", "ServerClass start ");
        try {
            serverFileSocket = new ServerSocket();
            serverFileSocket.setReuseAddress(true);
            serverFileSocket.bind(new
InetSocketAddress(PORT_FILE_USED));
            Log.i("", "PORT_FILE_USED = " + PORT_FILE_USED);
            socketFile = serverFileSocket.accept();
            SocketHandlerFile.setSocket(socketFile);
            Log.i("", "setSocket(socketFile)");
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}

```

Файл SocketHandlerAudio.java

```

package com.example.meshandroid;

import java.net.Socket;

public class SocketHandlerAudio {
    private static Socket socket;

    public static synchronized Socket getSocket(){
        return socket;
    }

    public static synchronized void setSocket(Socket socket){
        SocketHandlerAudio.socket = socket;
    }
}

```

Файл SocketHandlerFile.java

```

package com.example.meshandroid;

import java.net.Socket;

```

```

public class SocketHandlerFile {
    private static Socket socket;

    public static synchronized Socket getSocket(){
        return socket;
    }

    public static synchronized void setSocket(Socket socket){
        SocketHandlerFile.socket = socket;
    }
}

```

Файл Stribog.java

```

package com.example.meshandroid;

import static com.example.meshandroid.Kuznechik.int_to_byte;
import java.util.Arrays;

public class Stribog {
    private final byte[] Pi =int_to_byte(new int[]{
        252, 238, 221, 17, 207, 110, 49, 22, 251, 196, 250,
218, 35, 197, 4, 77,
        233, 119, 240, 219, 147, 46, 153, 186, 23, 54, 241,
187, 20, 205, 95, 193,
        249, 24, 101, 90, 226, 92, 239, 33, 129, 28, 60, 66,
139, 1, 142, 79,
        5, 132, 2, 174, 227, 106, 143, 160, 6, 11, 237, 152,
127, 212, 211, 31,
        235, 52, 44, 81, 234, 200, 72, 171, 242, 42, 104, 162,
253, 58, 206, 204,
        181, 112, 14, 86, 8, 12, 118, 18, 191, 114, 19, 71,
156, 183, 93, 135,
        21, 161, 150, 41, 16, 123, 154, 199, 243, 145, 120,
111, 157, 158, 178, 177,
        50, 117, 25, 61, 255, 53, 138, 126, 109, 84, 198, 128,
195, 189, 13, 87,
        223, 245, 36, 169, 62, 168, 67, 201, 215, 121, 214,
246, 124, 34, 185, 3,
        224, 15, 236, 222, 122, 148, 176, 188, 220, 232, 40,
80, 78, 51, 10, 74,
        167, 151, 96, 115, 30, 0, 98, 68, 26, 184, 56, 130,
100, 159, 38, 65,
        173, 69, 70, 146, 39, 94, 85, 47, 140, 163, 165, 125,
105, 213, 149, 59,
        7, 88, 179, 64, 134, 172, 29, 247, 48, 55, 107, 228,
136, 217, 231, 137,
        225, 27, 131, 73, 76, 63, 248, 254, 141, 83, 170, 144,
202, 216, 133, 97,
        32, 113, 103, 164, 45, 43, 9, 91, 203, 155, 37, 208,
190, 229, 108, 82,

```

```

89, 166, 116, 210, 230, 244, 180, 192, 209, 102, 175,
194, 57, 75, 99, 182
});

private final byte[] Tau = int_to_byte(new int[]{
    0, 8, 16, 24, 32, 40, 48, 56,
    1, 9, 17, 25, 33, 41, 49, 57,
    2, 10, 18, 26, 34, 42, 50, 58,
    3, 11, 19, 27, 35, 43, 51, 59,
    4, 12, 20, 28, 36, 44, 52, 60,
    5, 13, 21, 29, 37, 45, 53, 61,
    6, 14, 22, 30, 38, 46, 54, 62,
    7, 15, 23, 31, 39, 47, 55, 63
});

private final byte[][] A = {
    {(byte) 0x8e, (byte) 0x20, (byte) 0xfa, (byte) 0xa7,
    (byte) 0x2b, (byte) 0xa0, (byte) 0xb4, (byte) 0x70}, {(byte) 0x47,
    (byte) 0x10, (byte) 0x7d, (byte) 0xdd, (byte) 0x9b, (byte) 0x50,
    (byte) 0x5a, (byte) 0x38},
    {(byte) 0xad, (byte) 0x08, (byte) 0xb0, (byte) 0xe0,
    (byte) 0xc3, (byte) 0x28, (byte) 0x2d, (byte) 0x1c}, {(byte) 0xd8,
    (byte) 0x04, (byte) 0x58, (byte) 0x70, (byte) 0xef, (byte)
    0x14, (byte) 0x98, (byte) 0x0e},
    {(byte) 0x6c, (byte) 0x02, (byte) 0x2c, (byte) 0x38,
    (byte) 0xf9, (byte) 0x0a, (byte) 0x4c, (byte) 0x07}, {(byte) 0x36,
    (byte) 0x01, (byte) 0x16, (byte) 0x1c, (byte) 0xf2, (byte) 0x05,
    (byte) 0x26, (byte) 0x8d},
    {(byte) 0x1b, (byte) 0x8e, (byte) 0x0b, (byte) 0x0e,
    (byte) 0x79, (byte) 0x8c, (byte) 0x13, (byte) 0xc8}, {(byte) 0x83,
    (byte) 0x47, (byte) 0x8b, (byte) 0x07, (byte) 0xb2, (byte) 0x46,
    (byte) 0x87, (byte) 0x64},
    {(byte) 0xa0, (byte) 0x11, (byte) 0xd3, (byte) 0x80,
    (byte) 0x81, (byte) 0x8e, (byte) 0x8f, (byte) 0x40}, {(byte) 0x50,
    (byte) 0x86, (byte) 0xe7, (byte) 0x40, (byte) 0xce, (byte) 0x47,
    (byte) 0xc9, (byte) 0x20},
    {(byte) 0x28, (byte) 0x43, (byte) 0xfd, (byte) 0x20,
    (byte) 0x67, (byte) 0xad, (byte) 0xea, (byte) 0x10}, {(byte) 0x14,
    (byte) 0xaf, (byte) 0xf0, (byte) 0x10, (byte) 0xbd, (byte) 0xd8,
    (byte) 0x75, (byte) 0x08},
    {(byte) 0x0a, (byte) 0xd9, (byte) 0x78, (byte) 0x08,
    (byte) 0xd0, (byte) 0x6c, (byte) 0xb4, (byte) 0x04}, {(byte) 0x05,
    (byte) 0xe2, (byte) 0x3c, (byte) 0x04, (byte) 0x68, (byte) 0x36,
    (byte) 0x5a, (byte) 0x02},
    {(byte) 0x8c, (byte) 0x71, (byte) 0x1e, (byte) 0x02,
    (byte) 0x34, (byte) 0x1b, (byte) 0x2d, (byte) 0x01}, {(byte) 0x46,
    (byte) 0xb6, (byte) 0x0f, (byte) 0x01, (byte) 0x1a, (byte) 0x83,
    (byte) 0x98, (byte) 0x8e},
    {(byte) 0x90, (byte) 0xda, (byte) 0xb5, (byte) 0x2a,
    (byte) 0x38, (byte) 0x7a, (byte) 0xe7, (byte) 0x6f}, {(byte) 0x48,
    (byte) 0x6d, (byte) 0xd4, (byte) 0x15, (byte) 0x1c, (byte) 0x3d,
    (byte) 0xfd, (byte) 0xb9},

```

```

        { (byte) 0x24, (byte) 0xb8, (byte) 0x6a, (byte) 0x84,
(byte) 0x0e, (byte) 0x90, (byte) 0xf0, (byte) 0xd2}, {(byte) 0x12,
(byte) 0x5c, (byte) 0x35, (byte) 0x42, (byte) 0x07, (byte) 0x48,
(byte) 0x78, (byte) 0x69},
        { (byte) 0x09, (byte) 0x2e, (byte) 0x94, (byte) 0x21,
(byte) 0x8d, (byte) 0x24, (byte) 0x3c, (byte) 0xba}, {(byte) 0x8a,
(byte) 0x17, (byte) 0x4a, (byte) 0x9e, (byte) 0xc8, (byte) 0x12,
(byte) 0x1e, (byte) 0x5d},
        { (byte) 0x45, (byte) 0x85, (byte) 0x25, (byte) 0x4f,
(byte) 0x64, (byte) 0x09, (byte) 0x0f, (byte) 0xa0}, {(byte) 0xac,
(byte) 0xcc, (byte) 0x9c, (byte) 0xa9, (byte) 0x32, (byte) 0x8a,
(byte) 0x89, (byte) 0x50},
        { (byte) 0x9d, (byte) 0x4d, (byte) 0xf0, (byte) 0x5d,
(byte) 0x5f, (byte) 0x66, (byte) 0x14, (byte) 0x51}, {(byte) 0xc0,
(byte) 0xa8, (byte) 0x78, (byte) 0xa0, (byte) 0xa1, (byte) 0x33,
(byte) 0x0a, (byte) 0xa6},
        { (byte) 0x60, (byte) 0x54, (byte) 0x3c, (byte) 0x50,
(byte) 0xde, (byte) 0x97, (byte) 0x05, (byte) 0x53}, {(byte) 0x30,
(byte) 0x2a, (byte) 0x1e, (byte) 0x28, (byte) 0x6f, (byte) 0xc5,
(byte) 0x8c, (byte) 0xa7},
        { (byte) 0x18, (byte) 0x15, (byte) 0x0f, (byte) 0x14,
(byte) 0xb9, (byte) 0xec, (byte) 0x46, (byte) 0xdd}, {(byte) 0x0c,
(byte) 0x84, (byte) 0x89, (byte) 0x0a, (byte) 0xd2, (byte) 0x76,
(byte) 0x23, (byte) 0xe0},
        { (byte) 0x06, (byte) 0x42, (byte) 0xca, (byte) 0x05,
(byte) 0x69, (byte) 0x3b, (byte) 0x9f, (byte) 0x70}, {(byte) 0x03,
(byte) 0x21, (byte) 0x65, (byte) 0x8c, (byte) 0xba, (byte) 0x93,
(byte) 0xc1, (byte) 0x38},
        { (byte) 0x86, (byte) 0x27, (byte) 0x5d, (byte) 0xf0,
(byte) 0x9c, (byte) 0xe8, (byte) 0xaa, (byte) 0xa8}, {(byte) 0x43,
(byte) 0x9d, (byte) 0xa0, (byte) 0x78, (byte) 0x4e, (byte) 0x74,
(byte) 0x55, (byte) 0x54},
        { (byte) 0xaf, (byte) 0xc0, (byte) 0x50, (byte) 0x3c,
(byte) 0x27, (byte) 0x3a, (byte) 0xa4, (byte) 0x2a}, {(byte) 0xd9,
(byte) 0x60, (byte) 0x28, (byte) 0x1e, (byte) 0x9d, (byte) 0x1d,
(byte) 0x52, (byte) 0x15},
        { (byte) 0xe2, (byte) 0x30, (byte) 0x14, (byte) 0x0f,
(byte) 0xc0, (byte) 0x80, (byte) 0x29, (byte) 0x84}, {(byte) 0x71,
(byte) 0x18, (byte) 0x0a, (byte) 0x89, (byte) 0x60, (byte) 0x40,
(byte) 0x9a, (byte) 0x42},
        { (byte) 0xb6, (byte) 0x0c, (byte) 0x05, (byte) 0xca,
(byte) 0x30, (byte) 0x20, (byte) 0x4d, (byte) 0x21}, {(byte) 0x5b,
(byte) 0x06, (byte) 0x8c, (byte) 0x65, (byte) 0x18, (byte) 0x10,
(byte) 0xa8, (byte) 0x9e},
        { (byte) 0x45, (byte) 0x6c, (byte) 0x34, (byte) 0x88,
(byte) 0x7a, (byte) 0x38, (byte) 0x05, (byte) 0xb9}, {(byte) 0xac,
(byte) 0x36, (byte) 0x1a, (byte) 0x44, (byte) 0x3d, (byte) 0x1c,
(byte) 0x8c, (byte) 0xd2},
        { (byte) 0x56, (byte) 0x1b, (byte) 0x0d, (byte) 0x22,
(byte) 0x90, (byte) 0x0e, (byte) 0x46, (byte) 0x69}, {(byte) 0x2b,
(byte) 0x83, (byte) 0x88, (byte) 0x11, (byte) 0x48, (byte) 0x07,
(byte) 0x23, (byte) 0xba},

```

```

        { (byte) 0x9b, (byte) 0xcf, (byte) 0x44, (byte) 0x86,
(byte) 0x24, (byte) 0x8d, (byte) 0x9f, (byte) 0x5d}, {(byte) 0xc3,
(byte) 0xe9, (byte) 0x22, (byte) 0x43, (byte) 0x12, (byte) 0xc8,
(byte) 0xc1, (byte) 0xa0},
        { (byte) 0xef, (byte) 0xfa, (byte) 0x11, (byte) 0xaf,
(byte) 0x09, (byte) 0x64, (byte) 0xee, (byte) 0x50}, {(byte) 0xf9,
(byte) 0x7d, (byte) 0x86, (byte) 0xd9, (byte) 0x8a, (byte) 0x32,
(byte) 0x77, (byte) 0x28},
        { (byte) 0xe4, (byte) 0xfa, (byte) 0x20, (byte) 0x54,
(byte) 0xa8, (byte) 0x0b, (byte) 0x32, (byte) 0x9c}, {(byte) 0x72,
(byte) 0x7d, (byte) 0x10, (byte) 0x2a, (byte) 0x54, (byte) 0x8b,
(byte) 0x19, (byte) 0x4e},
        { (byte) 0x39, (byte) 0xb0, (byte) 0x08, (byte) 0x15,
(byte) 0x2a, (byte) 0xcb, (byte) 0x82, (byte) 0x27}, {(byte) 0x92,
(byte) 0x58, (byte) 0x04, (byte) 0x84, (byte) 0x15, (byte) 0xeb,
(byte) 0x41, (byte) 0x9d},
        { (byte) 0x49, (byte) 0x2c, (byte) 0x02, (byte) 0x42,
(byte) 0x84, (byte) 0xfb, (byte) 0xae, (byte) 0xc0}, {(byte) 0xaa,
(byte) 0x16, (byte) 0x01, (byte) 0x21, (byte) 0x42, (byte) 0xf3,
(byte) 0x57, (byte) 0x60},
        { (byte) 0x55, (byte) 0x0b, (byte) 0x8e, (byte) 0x9e,
(byte) 0x21, (byte) 0xf7, (byte) 0xa5, (byte) 0x30}, {(byte) 0xa4,
(byte) 0x8b, (byte) 0x47, (byte) 0x4f, (byte) 0x9e, (byte) 0xf5,
(byte) 0xdc, (byte) 0x18},
        { (byte) 0x70, (byte) 0xa6, (byte) 0xa5, (byte) 0x6e,
(byte) 0x24, (byte) 0x40, (byte) 0x59, (byte) 0x8e}, {(byte) 0x38,
(byte) 0x53, (byte) 0xdc, (byte) 0x37, (byte) 0x12, (byte) 0x20,
(byte) 0xa2, (byte) 0x47},
        { (byte) 0x1c, (byte) 0xa7, (byte) 0x6e, (byte) 0x95,
(byte) 0x09, (byte) 0x10, (byte) 0x51, (byte) 0xad}, {(byte) 0x0e,
(byte) 0xdd, (byte) 0x37, (byte) 0xc4, (byte) 0x8a, (byte) 0x08,
(byte) 0xa6, (byte) 0xd8},
        { (byte) 0x07, (byte) 0xe0, (byte) 0x95, (byte) 0x62,
(byte) 0x45, (byte) 0x04, (byte) 0x53, (byte) 0x6c}, {(byte) 0x8d,
(byte) 0x70, (byte) 0xc4, (byte) 0x31, (byte) 0xac, (byte) 0x02,
(byte) 0xa7, (byte) 0x36},
        { (byte) 0xc8, (byte) 0x38, (byte) 0x62, (byte) 0x96,
(byte) 0x56, (byte) 0x01, (byte) 0xdd, (byte) 0x1b}, {(byte) 0x64,
(byte) 0x1c, (byte) 0x31, (byte) 0x4b, (byte) 0x2b, (byte) 0x8e,
(byte) 0xe0, (byte) 0x83}
    };

```

```

private final byte [][] C = {
    {
        (byte) 0xb1, (byte) 0x08, (byte) 0x5b, (byte)
0xda, (byte) 0x1e, (byte) 0xca, (byte) 0xda, (byte) 0xe9, (byte)
0xeb, (byte) 0xcb, (byte) 0x2f, (byte) 0x81, (byte) 0xc0, (byte)
0x65, (byte) 0x7c, (byte) 0x1f,
        (byte) 0x2f, (byte) 0x6a, (byte) 0x76, (byte)
0x43, (byte) 0x2e, (byte) 0x45, (byte) 0xd0, (byte) 0x16, (byte)
0x71, (byte) 0x4e, (byte) 0xb8, (byte) 0x8d, (byte) 0x75, (byte)
0x85, (byte) 0xc4, (byte) 0xfc,
    }
}

```

```

        (byte) 0x4b, (byte) 0x7c, (byte) 0xe0, (byte)
0x91, (byte) 0x92, (byte) 0x67, (byte) 0x69, (byte) 0x01, (byte)
0xa2, (byte) 0x42, (byte) 0x2a, (byte) 0x08, (byte) 0xa4, (byte)
0x60, (byte) 0xd3, (byte) 0x15,
        (byte) 0x05, (byte) 0x76, (byte) 0x74, (byte)
0x36, (byte) 0xcc, (byte) 0x74, (byte) 0x4d, (byte) 0x23, (byte)
0xdd, (byte) 0x80, (byte) 0x65, (byte) 0x59, (byte) 0xf2, (byte)
0xa6, (byte) 0x45, (byte) 0x07
    },
    {
        (byte) 0x6f, (byte) 0xa3, (byte) 0xb5, (byte)
0x8a, (byte) 0xa9, (byte) 0x9d, (byte) 0x2f, (byte) 0x1a, (byte)
0x4f, (byte) 0xe3, (byte) 0x9d, (byte) 0x46, (byte) 0x0f, (byte)
0x70, (byte) 0xb5, (byte) 0xd7,
        (byte) 0xf3, (byte) 0xfe, (byte) 0xea, (byte)
0x72, (byte) 0x0a, (byte) 0x23, (byte) 0x2b, (byte) 0x98, (byte)
0x61, (byte) 0xd5, (byte) 0x5e, (byte) 0x0f, (byte) 0x16, (byte)
0xb5, (byte) 0x01, (byte) 0x31,
        (byte) 0x9a, (byte) 0xb5, (byte) 0x17, (byte)
0x6b, (byte) 0x12, (byte) 0xd6, (byte) 0x99, (byte) 0x58, (byte)
0x5c, (byte) 0xb5, (byte) 0x61, (byte) 0xc2, (byte) 0xdb, (byte)
0x0a, (byte) 0xa7, (byte) 0xca,
        (byte) 0x55, (byte) 0xdd, (byte) 0xa2, (byte)
0x1b, (byte) 0xd7, (byte) 0xcb, (byte) 0xcd, (byte) 0x56, (byte)
0xe6, (byte) 0x79, (byte) 0x04, (byte) 0x70, (byte) 0x21, (byte)
0xb1, (byte) 0x9b, (byte) 0xb7
    },
    {
        (byte) 0xf5, (byte) 0x74, (byte) 0xdc, (byte)
0xac, (byte) 0x2b, (byte) 0xce, (byte) 0x2f, (byte) 0xc7, (byte)
0x0a, (byte) 0x39, (byte) 0xfc, (byte) 0x28, (byte) 0x6a, (byte)
0x3d, (byte) 0x84, (byte) 0x35,
        (byte) 0x06, (byte) 0xf1, (byte) 0x5e, (byte)
0x5f, (byte) 0x52, (byte) 0x9c, (byte) 0x1f, (byte) 0x8b, (byte)
0xf2, (byte) 0xea, (byte) 0x75, (byte) 0x14, (byte) 0xb1, (byte)
0x29, (byte) 0x7b, (byte) 0x7b,
        (byte) 0xd3, (byte) 0xe2, (byte) 0x0f, (byte)
0xe4, (byte) 0x90, (byte) 0x35, (byte) 0x9e, (byte) 0xb1, (byte)
0xc1, (byte) 0xc9, (byte) 0x3a, (byte) 0x37, (byte) 0x60, (byte)
0x62, (byte) 0xdb, (byte) 0x09,
        (byte) 0xc2, (byte) 0xb6, (byte) 0xf4, (byte)
0x43, (byte) 0x86, (byte) 0x7a, (byte) 0xdb, (byte) 0x31, (byte)
0x99, (byte) 0x1e, (byte) 0x96, (byte) 0xf5, (byte) 0x0a, (byte)
0xba, (byte) 0x0a, (byte) 0xb2
    },
    {
        (byte) 0xef, (byte) 0x1f, (byte) 0xdf, (byte)
0xb3, (byte) 0xe8, (byte) 0x15, (byte) 0x66, (byte) 0xd2, (byte)
0xf9, (byte) 0x48, (byte) 0xe1, (byte) 0xa0, (byte) 0x5d, (byte)
0x71, (byte) 0xe4, (byte) 0xdd,
        (byte) 0x48, (byte) 0x8e, (byte) 0x85, (byte)
0x7e, (byte) 0x33, (byte) 0x5c, (byte) 0x3c, (byte) 0x7d, (byte)

```

```

0x9d, (byte) 0x72, (byte) 0x1c, (byte) 0xad, (byte) 0x68, (byte)
0x5e, (byte) 0x35, (byte) 0x3f,
        (byte) 0xa9, (byte) 0xd7, (byte) 0x2c, (byte)
0x82, (byte) 0xed, (byte) 0x03, (byte) 0xd6, (byte) 0x75, (byte)
0xd8, (byte) 0xb7, (byte) 0x13, (byte) 0x33, (byte) 0x93, (byte)
0x52, (byte) 0x03, (byte) 0xbe,
        (byte) 0x34, (byte) 0x53, (byte) 0xea, (byte)
0xa1, (byte) 0x93, (byte) 0xe8, (byte) 0x37, (byte) 0xf1, (byte)
0x22, (byte) 0x0c, (byte) 0xbe, (byte) 0xbc, (byte) 0x84, (byte)
0xe3, (byte) 0xd1, (byte) 0x2e
    },
    {
        (byte) 0x4b, (byte) 0xea, (byte) 0x6b, (byte)
0xac, (byte) 0xad, (byte) 0x47, (byte) 0x47, (byte) 0x99, (byte)
0x9a, (byte) 0x3f, (byte) 0x41, (byte) 0x0c, (byte) 0x6c, (byte)
0xa9, (byte) 0x23, (byte) 0x63,
        (byte) 0x7f, (byte) 0x15, (byte) 0x1c, (byte)
0x1f, (byte) 0x16, (byte) 0x86, (byte) 0x10, (byte) 0x4a, (byte)
0x35, (byte) 0x9e, (byte) 0x35, (byte) 0xd7, (byte) 0x80, (byte)
0x0f, (byte) 0xff, (byte) 0xbd,
        (byte) 0xbf, (byte) 0xcd, (byte) 0x17, (byte)
0x47, (byte) 0x25, (byte) 0x3a, (byte) 0xf5, (byte) 0xa3, (byte)
0xdf, (byte) 0xff, (byte) 0x00, (byte) 0xb7, (byte) 0x23, (byte)
0x27, (byte) 0x1a, (byte) 0x16,
        (byte) 0x7a, (byte) 0x56, (byte) 0xa2, (byte)
0x7e, (byte) 0xa9, (byte) 0xea, (byte) 0x63, (byte) 0xf5, (byte)
0x60, (byte) 0x17, (byte) 0x58, (byte) 0xfd, (byte) 0x7c, (byte)
0x6c, (byte) 0xfe, (byte) 0x57
    },
    {
        (byte) 0xae, (byte) 0x4f, (byte) 0xae, (byte)
0xae, (byte) 0x1d, (byte) 0x3a, (byte) 0xd3, (byte) 0xd9, (byte)
0x6f, (byte) 0xa4, (byte) 0xc3, (byte) 0x3b, (byte) 0x7a, (byte)
0x30, (byte) 0x39, (byte) 0xc0,
        (byte) 0x2d, (byte) 0x66, (byte) 0xc4, (byte)
0xf9, (byte) 0x51, (byte) 0x42, (byte) 0xa4, (byte) 0x6c, (byte)
0x18, (byte) 0x7f, (byte) 0x9a, (byte) 0xb4, (byte) 0x9a, (byte)
0xf0, (byte) 0x8e, (byte) 0xc6,
        (byte) 0xcf, (byte) 0xfa, (byte) 0xa6, (byte)
0xb7, (byte) 0x1c, (byte) 0x9a, (byte) 0xb7, (byte) 0xb4, (byte)
0x0a, (byte) 0xf2, (byte) 0x1f, (byte) 0x66, (byte) 0xc2, (byte)
0xbe, (byte) 0xc6, (byte) 0xb6,
        (byte) 0xbf, (byte) 0x71, (byte) 0xc5, (byte)
0x72, (byte) 0x36, (byte) 0x90, (byte) 0x4f, (byte) 0x35, (byte)
0xfa, (byte) 0x68, (byte) 0x40, (byte) 0x7a, (byte) 0x46, (byte)
0x64, (byte) 0x7d, (byte) 0x6e
    },
    {
        (byte) 0xf4, (byte) 0xc7, (byte) 0x0e, (byte)
0x16, (byte) 0xee, (byte) 0xaa, (byte) 0xc5, (byte) 0xec, (byte)
0x51, (byte) 0xac, (byte) 0x86, (byte) 0xfe, (byte) 0xbf, (byte)
0x24, (byte) 0x09, (byte) 0x54,

```

```

        (byte) 0x39, (byte) 0x9e, (byte) 0xc6, (byte)
0xc7, (byte) 0xe6, (byte) 0xbf, (byte) 0x87, (byte) 0xc9, (byte)
0xd3, (byte) 0x47, (byte) 0x3e, (byte) 0x33, (byte) 0x19, (byte)
0x7a, (byte) 0x93, (byte) 0xc9,
        (byte) 0x09, (byte) 0x92, (byte) 0xab, (byte)
0xc5, (byte) 0x2d, (byte) 0x82, (byte) 0x2c, (byte) 0x37, (byte)
0x06, (byte) 0x47, (byte) 0x69, (byte) 0x83, (byte) 0x28, (byte)
0x4a, (byte) 0x05, (byte) 0x04,
        (byte) 0x35, (byte) 0x17, (byte) 0x45, (byte)
0x4c, (byte) 0xa2, (byte) 0x3c, (byte) 0x4a, (byte) 0xf3, (byte)
0x88, (byte) 0x86, (byte) 0x56, (byte) 0x4d, (byte) 0x3a, (byte)
0x14, (byte) 0xd4, (byte) 0x93
    },
    {
        (byte) 0x9b, (byte) 0x1f, (byte) 0x5b, (byte)
0x42, (byte) 0x4d, (byte) 0x93, (byte) 0xc9, (byte) 0xa7, (byte)
0x03, (byte) 0xe7, (byte) 0xaa, (byte) 0x02, (byte) 0x0c, (byte)
0x6e, (byte) 0x41, (byte) 0x41,
        (byte) 0x4e, (byte) 0xb7, (byte) 0xf8, (byte)
0x71, (byte) 0x9c, (byte) 0x36, (byte) 0xde, (byte) 0x1e, (byte)
0x89, (byte) 0xb4, (byte) 0x44, (byte) 0x3b, (byte) 0x4d, (byte)
0xdb, (byte) 0xc4, (byte) 0x9a,
        (byte) 0xf4, (byte) 0x89, (byte) 0x2b, (byte)
0xcb, (byte) 0x92, (byte) 0x9b, (byte) 0x06, (byte) 0x90, (byte)
0x69, (byte) 0xd1, (byte) 0x8d, (byte) 0x2b, (byte) 0xd1, (byte)
0xa5, (byte) 0xc4, (byte) 0x2f,
        (byte) 0x36, (byte) 0xac, (byte) 0xc2, (byte)
0x35, (byte) 0x59, (byte) 0x51, (byte) 0xa8, (byte) 0xd9, (byte)
0xa4, (byte) 0x7f, (byte) 0x0d, (byte) 0xd4, (byte) 0xbf, (byte)
0x02, (byte) 0xe7, (byte) 0x1e
    },
    {
        (byte) 0x37, (byte) 0x8f, (byte) 0x5a, (byte)
0x54, (byte) 0x16, (byte) 0x31, (byte) 0x22, (byte) 0x9b, (byte)
0x94, (byte) 0x4c, (byte) 0x9a, (byte) 0xd8, (byte) 0xec, (byte)
0x16, (byte) 0x5f, (byte) 0xde,
        (byte) 0x3a, (byte) 0x7d, (byte) 0x3a, (byte)
0x1b, (byte) 0x25, (byte) 0x89, (byte) 0x42, (byte) 0x24, (byte)
0x3c, (byte) 0xd9, (byte) 0x55, (byte) 0xb7, (byte) 0xe0, (byte)
0x0d, (byte) 0x09, (byte) 0x84,
        (byte) 0x80, (byte) 0x0a, (byte) 0x44, (byte)
0x0b, (byte) 0xdb, (byte) 0xb2, (byte) 0xce, (byte) 0xb1, (byte)
0x7b, (byte) 0x2b, (byte) 0x8a, (byte) 0x9a, (byte) 0xa6, (byte)
0x07, (byte) 0x9c, (byte) 0x54,
        (byte) 0x0e, (byte) 0x38, (byte) 0xdc, (byte)
0x92, (byte) 0xcb, (byte) 0x1f, (byte) 0x2a, (byte) 0x60, (byte)
0x72, (byte) 0x61, (byte) 0x44, (byte) 0x51, (byte) 0x83, (byte)
0x23, (byte) 0x5a, (byte) 0xdb
    },
    {
        (byte) 0xab, (byte) 0xbe, (byte) 0xde, (byte)
0xa6, (byte) 0x80, (byte) 0x05, (byte) 0x6f, (byte) 0x52, (byte)

```

```

0x38, (byte) 0x2a, (byte) 0xe5, (byte) 0x48, (byte) 0xb2, (byte)
0xe4, (byte) 0xf3, (byte) 0xf3,
    (byte) 0x89, (byte) 0x41, (byte) 0xe7, (byte)
0x1c, (byte) 0xff, (byte) 0x8a, (byte) 0x78, (byte) 0xdb, (byte)
0x1f, (byte) 0xff, (byte) 0xe1, (byte) 0x8a, (byte) 0x1b, (byte)
0x33, (byte) 0x61, (byte) 0x03,
    (byte) 0x9f, (byte) 0xe7, (byte) 0x67, (byte)
0x02, (byte) 0xaf, (byte) 0x69, (byte) 0x33, (byte) 0x4b, (byte)
0x7a, (byte) 0x1e, (byte) 0x6c, (byte) 0x30, (byte) 0x3b, (byte)
0x76, (byte) 0x52, (byte) 0xf4,
    (byte) 0x36, (byte) 0x98, (byte) 0xfa, (byte)
0xd1, (byte) 0x15, (byte) 0x3b, (byte) 0xb6, (byte) 0xc3, (byte)
0x74, (byte) 0xb4, (byte) 0xc7, (byte) 0xfb, (byte) 0x98, (byte)
0x45, (byte) 0x9c, (byte) 0xed
    },
    {
        (byte) 0x7b, (byte) 0xcd, (byte) 0x9e, (byte)
0xd0, (byte) 0xef, (byte) 0xc8, (byte) 0x89, (byte) 0xfb, (byte)
0x30, (byte) 0x02, (byte) 0xc6, (byte) 0xcd, (byte) 0x63, (byte)
0x5a, (byte) 0xfe, (byte) 0x94,
        (byte) 0xd8, (byte) 0xfa, (byte) 0x6b, (byte)
0xbb, (byte) 0xeb, (byte) 0xab, (byte) 0x07, (byte) 0x61, (byte)
0x20, (byte) 0x01, (byte) 0x80, (byte) 0x21, (byte) 0x14, (byte)
0x84, (byte) 0x66, (byte) 0x79,
        (byte) 0x8a, (byte) 0x1d, (byte) 0x71, (byte)
0xef, (byte) 0xea, (byte) 0x48, (byte) 0xb9, (byte) 0xca, (byte)
0xef, (byte) 0xba, (byte) 0xcd, (byte) 0x1d, (byte) 0x7d, (byte)
0x47, (byte) 0x6e, (byte) 0x98,
        (byte) 0xde, (byte) 0xa2, (byte) 0x59, (byte)
0x4a, (byte) 0xc0, (byte) 0x6f, (byte) 0xd8, (byte) 0x5d, (byte)
0x6b, (byte) 0xca, (byte) 0xa4, (byte) 0xcd, (byte) 0x81, (byte)
0xf3, (byte) 0x2d, (byte) 0x1b
    },
    {
        (byte) 0x37, (byte) 0x8e, (byte) 0xe7, (byte)
0x67, (byte) 0xf1, (byte) 0x16, (byte) 0x31, (byte) 0xba, (byte)
0xd2, (byte) 0x13, (byte) 0x80, (byte) 0xb0, (byte) 0x04, (byte)
0x49, (byte) 0xb1, (byte) 0x7a,
        (byte) 0xcd, (byte) 0xa4, (byte) 0x3c, (byte)
0x32, (byte) 0xbc, (byte) 0xdf, (byte) 0x1d, (byte) 0x77, (byte)
0xf8, (byte) 0x20, (byte) 0x12, (byte) 0xd4, (byte) 0x30, (byte)
0x21, (byte) 0x9f, (byte) 0x9b,
        (byte) 0x5d, (byte) 0x80, (byte) 0xef, (byte)
0x9d, (byte) 0x18, (byte) 0x91, (byte) 0xcc, (byte) 0x86, (byte)
0xe7, (byte) 0x1d, (byte) 0xa4, (byte) 0xaa, (byte) 0x88, (byte)
0xe1, (byte) 0x28, (byte) 0x52,
        (byte) 0xfa, (byte) 0xf4, (byte) 0x17, (byte)
0xd5, (byte) 0xd9, (byte) 0xb2, (byte) 0x1b, (byte) 0x99, (byte)
0x48, (byte) 0xbc, (byte) 0x92, (byte) 0x4a, (byte) 0xf1, (byte)
0x1b, (byte) 0xd7, (byte) 0x20
    }
};

```

```

public byte[] getHash(byte[] message) {
    return StribogHash(message);
}

private byte[] StribogHash(byte[] message)
{
    byte[] h = new byte[64];
    Arrays.fill(h, (byte)0x01);

    byte[] N = new byte[64];
    Arrays.fill(N, (byte)0x00);

    byte[] Sigma = new byte[64];
    Arrays.fill(Sigma, (byte)0x00);

    byte[] v_0 = new byte[64];
    Arrays.fill(v_0, (byte)0x00);

    byte[] v_512 = new byte[64];
    Arrays.fill(v_512, (byte)0x00);
    v_512[63] = (byte)0x02;

    int length = message.length * 8;
    int inc = 0;

    while(length >= 512)
    {
        inc++;
        byte[] tmp = Arrays.copyOfRange(message, message.length
- 64*(inc), message.length - (inc-1)*64);
        h = StribG(N, tmp, h);
        N = addModule(N, v_512);
        Sigma = addModule(Sigma, tmp);
        length -=512;
    }

    message = Arrays.copyOf(message, message.length - 64*inc);

    if(message.length < 64)
    {
        byte[] tmpMessage = new byte[64];
        Arrays.fill(tmpMessage, (byte) 0x00);
        tmpMessage[64 - message.length - 1] = (byte) 0x01;

        for(int i = 64 - message.length; i < 64; i++)
            tmpMessage[i] = message[i + message.length - 64];

        message = tmpMessage;
    }
}

```

```

    h = StribG(N, message, h);

    byte[] NMessage = new byte[64];
    Arrays.fill(NMessage, (byte) 0x00);

    inc = 0;
    while(length > 0)
    {
        NMessage[63 - inc] = (byte) (length & 0xff);
        length >>= 8;
        inc++;
    }

    N = addModule(N, NMessage);
    Sigma = addModule(Sigma, message);

    h = StribG(v_0, N, h);
    h = StribG(v_0, Sigma, h);

    return Arrays.copyOf(h, 32);
}

private byte[] addModule(byte[] a, byte[] b)
{
    byte[] result = new byte[64];
    int t = 0;

    for(int i = 63; i >= 0; i--)
    {
        t = (a[i] & 0xff) + (b[i] & 0xff) + (t >> 8);
        result[i] = (byte) (t & 0xff);
    }
    return result;
}

private byte[] StribX(byte[] a, byte[] b)
{
    byte[] result = new byte[a.length];
    for(int i = 0; i < a.length; i++)
        result[i] = (byte) (a[i] ^ b[i]);
    return result;
}

private byte[] StribG(byte [] N, byte[] m, byte[] h)
{
    byte[] K;
    K = StribX(h, N);
    K = SrtibS(K);
    K = StribP(K);
    K = StribL(K);

```

```

        byte[] t = StribE(K, m);
        t = StribX(h, t);
        byte[] result = StribX(t, m);

        return result;
    }

private byte[] StribE(byte[] K, byte[] m)
{
    byte[] state;

    state = StribX(K, m);
    for(int i = 0; i < 12; i++)
    {
        state = SrtibS(state);
        state = StribP(state);
        state = StribL(state);

        K = KeyShedule(K, i);
        state = StribX(state, K);
    }

    return state;
}

private byte[] KeyShedule(byte[] K, int i)
{
    K = StribX(K, C[i]);

    K = SrtibS(K);
    K = StribP(K);
    K = StribL(K);

    return K;
}

private byte[] SrtibS(byte[] a)
{
    byte[] result = new byte[64];

    for(int i = 0; i < 64; i++)
        result[i] = Pi[(a[i] & 0xFF)];

    return result;
}

private byte[] StribP(byte[] a)
{
    byte[] result = new byte[64];

    for(int i = 0; i < 64; i++)

```

```

        result[i] = a[Tau[i]];

    return result;
}

private byte[] StribL(byte[] a)
{
    byte[] result = new byte[64];
    for(int i = 0; i < 8; i++)
    {
        for(int k = 0; k < 8; k++)
        {
            if((a[i*8+k] & (0x80)) != 0)
            {
                result[8*i] ^= A[k*8+0][0];
                result[8*i+1] ^= A[k*8+0][1];
                result[8*i+2] ^= A[k*8+0][2];
                result[8*i+3] ^= A[k*8+0][3];
                result[8*i+4] ^= A[k*8+0][4];
                result[8*i+5] ^= A[k*8+0][5];
                result[8*i+6] ^= A[k*8+0][6];
                result[8*i+7] ^= A[k*8+0][7];
            }
            if((a[i*8+k] & (0x40)) != 0)
            {
                result[8*i] ^= A[k*8+1][0];
                result[8*i+1] ^= A[k*8+1][1];
                result[8*i+2] ^= A[k*8+1][2];
                result[8*i+3] ^= A[k*8+1][3];
                result[8*i+4] ^= A[k*8+1][4];
                result[8*i+5] ^= A[k*8+1][5];
                result[8*i+6] ^= A[k*8+1][6];
                result[8*i+7] ^= A[k*8+1][7];
            }
            if((a[i*8+k] & (0x20)) != 0)
            {
                result[8*i] ^= A[k*8+2][0];
                result[8*i+1] ^= A[k*8+2][1];
                result[8*i+2] ^= A[k*8+2][2];
                result[8*i+3] ^= A[k*8+2][3];
                result[8*i+4] ^= A[k*8+2][4];
                result[8*i+5] ^= A[k*8+2][5];
                result[8*i+6] ^= A[k*8+2][6];
                result[8*i+7] ^= A[k*8+2][7];
            }
            if((a[i*8+k] & (0x10)) != 0)
            {
                result[8*i] ^= A[k*8+3][0];
                result[8*i+1] ^= A[k*8+3][1];
                result[8*i+2] ^= A[k*8+3][2];
                result[8*i+3] ^= A[k*8+3][3];
                result[8*i+4] ^= A[k*8+3][4];
            }
        }
    }
}

```

```

        result[8*i+5] ^= A[k*8+3][5];
        result[8*i+6] ^= A[k*8+3][6];
        result[8*i+7] ^= A[k*8+3][7];
    }
    if((a[i*8+k] & (0x8)) != 0)
    {
        result[8*i] ^= A[k*8+4][0];
        result[8*i+1] ^= A[k*8+4][1];
        result[8*i+2] ^= A[k*8+4][2];
        result[8*i+3] ^= A[k*8+4][3];
        result[8*i+4] ^= A[k*8+4][4];
        result[8*i+5] ^= A[k*8+4][5];
        result[8*i+6] ^= A[k*8+4][6];
        result[8*i+7] ^= A[k*8+4][7];
    }
    if((a[i*8+k] & (0x4)) != 0)
    {
        result[8*i] ^= A[k*8+5][0];
        result[8*i+1] ^= A[k*8+5][1];
        result[8*i+2] ^= A[k*8+5][2];
        result[8*i+3] ^= A[k*8+5][3];
        result[8*i+4] ^= A[k*8+5][4];
        result[8*i+5] ^= A[k*8+5][5];
        result[8*i+6] ^= A[k*8+5][6];
        result[8*i+7] ^= A[k*8+5][7];
    }
    if((a[i*8+k] & (0x2)) != 0)
    {
        result[8*i] ^= A[k*8+6][0];
        result[8*i+1] ^= A[k*8+6][1];
        result[8*i+2] ^= A[k*8+6][2];
        result[8*i+3] ^= A[k*8+6][3];
        result[8*i+4] ^= A[k*8+6][4];
        result[8*i+5] ^= A[k*8+6][5];
        result[8*i+6] ^= A[k*8+6][6];
        result[8*i+7] ^= A[k*8+6][7];
    }
    if((a[i*8+k] & (0x1)) != 0)
    {
        result[8*i] ^= A[k*8+7][0];
        result[8*i+1] ^= A[k*8+7][1];
        result[8*i+2] ^= A[k*8+7][2];
        result[8*i+3] ^= A[k*8+7][3];
        result[8*i+4] ^= A[k*8+7][4];
        result[8*i+5] ^= A[k*8+7][5];
        result[8*i+6] ^= A[k*8+7][6];
        result[8*i+7] ^= A[k*8+7][7];
    }
    }
    }
    return result;
}

```

```
}
```

Файл WiFiDirectBroadcastReceiver.java

```
package com.example.meshandroid;

import android.Manifest;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.NetworkInfo;
import android.net.wifi.p2p.WifiP2pDevice;
import android.net.wifi.p2p.WifiP2pManager;
import android.util.Log;

import androidx.core.app.ActivityCompat;

import com.example.meshandroid.FirstActivity;

public class WiFiDirectBroadcastReceiver extends BroadcastReceiver
{
    private final String LOG_TAG = this.toString();
    private WifiP2pManager mManager;
    private WifiP2pManager.Channel mChannel;
    private FirstActivity mActivity;
    public WiFiDirectBroadcastReceiver(WifiP2pManager manager,
    WifiP2pManager.Channel channel, FirstActivity activity) {
        super();
        this.mManager = manager;
        this.mChannel = channel;
        this.mActivity = activity;
    }
    // Handle wifi statuses here
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        //Log.i(LOG_TAG, "Попытка поиска...");
        // Wifi enabled or disabled?
        if
(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {

            int state =
intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, -1);
            if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {
                Log.i(LOG_TAG, "Wifi Direct is enabled");
            } else {
                Log.i(LOG_TAG, "Wifi Direct is not enabled");
            }
            // Handle device status changes (a new device is
            // available, an existing device is no longer available etc.)

```

```

        } else if
(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {
            if (mManager != null) {
                if (ActivityCompat.checkSelfPermission(mActivity,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                    Log.i(LOG_TAG, "нет разрешений
(WiFiDirectBroadcastReceiver)");
                    return;
                }
                mManager.requestPeers(mChannel,
mActivity.peerListListener);
            }
            // Handle connection status changes (connecting to a
new device, losing an existing connection etc.)
        } else if
(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action))
{
            if (mManager == null) {
                return;
            }
            NetworkInfo networkInfo = (NetworkInfo)
intent.getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);
            if (networkInfo.isConnected()) {
                // we are connected with the other device, request
connection
                // info to find group owner IP
                mManager.requestConnectionInfo(mChannel,
mActivity.connectionInfoListener);
            }

        } else if
(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION.equals(action))
{
            Log.d("FirstActivity.myNameWiFiDirect= ",
FirstActivity.myNameWiFiDirect);
            WifiP2pDevice myDevice = (WifiP2pDevice)
intent.getParcelableExtra(WifiP2pManager.EXTRA_WIFI_P2P_DEVICE);
            FirstActivity.myNameWiFiDirect = myDevice.deviceName;
            Log.d("Wifi Direct: My Device",
FirstActivity.myNameWiFiDirect);

            // Respond to this device's wifi state changing
        }
    }
}

```

Приложение В

(Обязательное)

Программное средство для защищенного обмена информацией с
использованием технологии *mesh*-сетей

Презентация

					ПГУ1.090301.38.001 ПЗ	Лист
						112
Изм	Лист	№ докум	Подпись	Дата		



ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ЗАЩИЩЕННОГО ОБМЕНА ИНФОРМАЦИЕЙ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ MESH-СЕТЕЙ

Автор работы: студент группы 19ВВП2 Ханбекова Е.В.
Научный руководитель: к.т.н., доцент каф. ВТ ПГУ Юрова О.В.

Пенза 2023



АКТУАЛЬНОСТЬ ПРОЕКТА

Мессенджеры позволяют быстро обмениваться информацией. Однако для функционирования большинства популярных приложений для обмена сообщениями необходим доступ к сети Интернет или услугам провайдера, которые есть далеко не везде, поэтому связь с помощью раций не утратила своей популярности.

Для оперативной передачи голосовых сообщений могут быть использованы радиосети. Область применения раций достаточно широкая: от работы и активного отдыха до чрезвычайных ситуаций. Но приобретение радиооборудования требует материальных вложений.



Изм	Лист	№ докум	Подпись	Дата

ПГУ1.090301.38.001 ПЗ

Лист

113

АКТУАЛЬНОСТЬ ПРОЕКТА

Современные смартфоны имеют достаточные технические возможности для реализации функционала раций в мобильном приложении. Данная задача может быть решена с помощью реализации технологий Push-To-Talk и mesh-сетей в мобильном приложении.

Актуальной является проблема конфиденциальности, так как многие мессенджеры являются разработкой зарубежных компаний. Информация, передаваемая с их помощью, может быть перехвачена и отслежена. Поэтому важным становится разработка приложений с определенным уровнем защиты информации. Достичь необходимого уровня защиты информации возможно с помощью шифрования.

ЦЕЛЬ И ЗАДАЧИ ПРОЕКТА

Цель:

Создание приложения под управлением ОС Android для обмена информацией с использованием технологии mesh-сетей, реализующее механизм криптографической защиты информации.

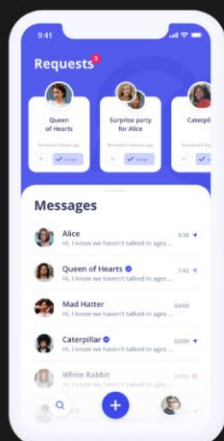
Задачи:

Приложение должно обеспечивать:

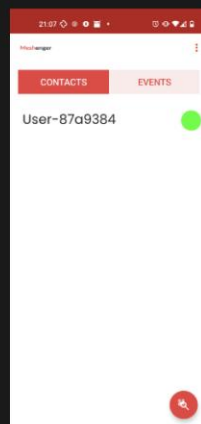
- Организацию соединения между устройствами с использованием технологии mesh-сетей;
- Передачу голосовых сообщений в режиме реального времени и файлов между мобильными телефонами, на которых запущено данное программное средство;
- Криптографическую защиту информации по алгоритму «Кузнечик».

АНАЛОГИ

BERTY



MESSENGER



ТУРФОН



СТЕК ТЕХНОЛОГИЙ

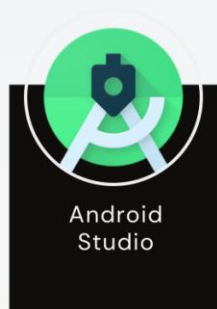


ДИАГРАММА ПРЕЦЕДЕНТОВ

```
graph LR; User[Пользователь] --> UC1(Вход в приложение); UC1 -.->|<extend>| UC2(Изменить имя устройства); UC1 -.->|<extend>| UC3(Просмотреть доступные подключения); UC3 -.->|<extend>| UC4(Подключиться к другому устройству); UC4 -.->|<extend>| UC5(Прослушивать в режиме реального времени голосовые сообщения); UC4 -.->|<extend>| UC6(Отправлять в режиме реального времени голосовые сообщения); UC4 -.->|<extend>| UC7(Отправлять файлы); UC4 -.->|<extend>| UC8(Принимать файлы); UC4 -.->|<extend>| UC9(Отправить файлы через посредника);
```

The diagram illustrates the use cases for a VoIP application. It starts with a user (Пользователь) who enters the application (Вход в приложение). From there, the user can extend the functionality to change the device name (Изменить имя устройства) or view available connections (Просмотреть доступные подключения). From the connections view, the user can extend to connect to another device (Подключиться к другому устройству). This connection step further extends into several actions: real-time voice message listening (Прослушивать в режиме реального времени голосовые сообщения), real-time voice message sending (Отправлять в режиме реального времени голосовые сообщения), file sending (Отправлять файлы), file receiving (Принимать файлы), and sending files through a mediator (Отправить файлы через посредника).

[illegible]

ДЕМОНСТРАЦИЯ ПРИЛОЖЕНИЯ



ГЛАВНЫЙ ЭКРАН



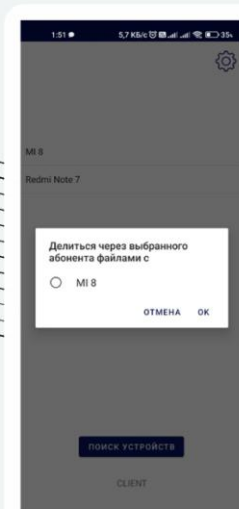
НАСТРОЙКИ



ДЕМОНСТРАЦИЯ ПРИЛОЖЕНИЯ



ПОДКЛЮЧЕНИЕ



ОКНО РАЦИИ



ОКНО РАЦИИ



Изм	Лист	№ докум	Подпись	Дата
-----	------	---------	---------	------

ПГУ1.090301.38.001 ПЗ

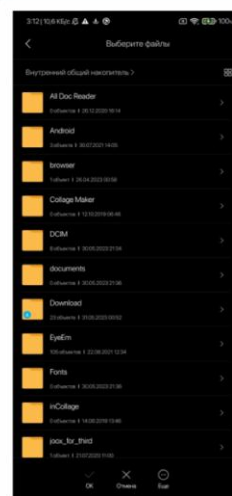
Лист

117

ДЕМОНСТРАЦИЯ ПРИЛОЖЕНИЯ

ОКНО ОТПРАВКИ ФАЙЛОВ

ФАЙЛОВЫЙ МЕНЕДЖЕР



ЗАКЛЮЧЕНИЕ

В процессе выполнения проекта был проведен анализ актуальности проекта. Были рассмотрены имеющиеся на данный момент аналоги проекта.

В результате проделанной работы было разработано мобильное приложение «RadioMesh» для защищенного обмена файлами и голосовыми сообщениями в режиме реального времени с использованием технологии mesh-сетей.

В процессе реализации данного проекта были достигнуты все поставленные цели. Итоговое приложение было протестировано и отлажено. Тестирование показало, что приложение работает корректно и обладает всем необходимым функционалом.

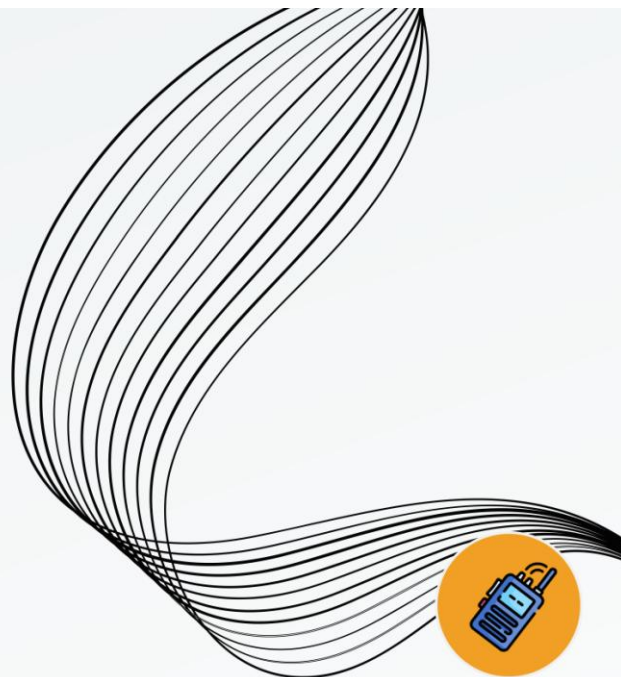
Изм	Лист	№ докум	Подпись	Дата

ПГУ1.090301.38.001 ПЗ

Лист

118

**THANK'S FOR
WATCHING**



					ПГУ1.090301.38.001 ПЗ	Лист
Изм	Лист	№ докум	Подпись	Дата		119