МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Розробка додатку обміну файлами між Windows та Android платформами»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Кузіков Б.О.**

Студента групи ІН – 73-9 — **Наталуха М.Р.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп’ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

“_____” _____20__ г.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-73-9 спеціальності “Комп’ютерні науки” денної форми навчання Наталухи Максима Романовича.

Тема: “Розробка додатку обміну файлами між Windows та Android платформами”

Затверджена наказом по СумДУ

№_____ от _____20__ р.

Зміст пояснювальної записки: 1) Огляд існуючих рішень; 2) постановка завдання й основних вимог; 3) створення протоколу захищеного зв’язку; 4) імплементація протоколу у вигляді Android додатку; 5) імплементація протоколу у вигляді Windows додатку. 6) Опис результатів.

Дата видачі завдання “_____”_____20__ р.

Керівник випускної роботи _____ Кузіков Б.О.

Завдання приняв до виконання _____ Наталуха М.Р.

РЕФЕРАТ

Записка: 61 стор., 20 рис., 8 додатків, 25 джерел.

Об’єкт дослідження — сучасні засоби імплементації мережевих рішень з фокусом на безпеку та децентралізацію.

Мета роботи — розробка протоколу передавання файлів по мережі IPv4 з використанням наскрізного шифрування. Побудова додатків під платформи Windows та Android з повною реалізацією функціоналу протоколу обміну.

Методи дослідження — методи системного аналізу, об`єктно-орієнтованого проектування. .

Результати — розроблено рішення яке дозволяє, на основі наскрізного шифрування, верифікації повідомлень, встановлювати захищений зв’язок в мережі, гарантувати підтвердженні учасників шляхом створення системі довіреностей до зазначених пристроїв використовуючи публічні ключі RSA.

ОБМІН ІНФОРМАЦІЇ, НАСКІЗНЕ ШІФРУВАННЯ, ПУБЛІЧНИЙ КЛЮЧ, ПРИВАТНІЙ КЛЮЧ, СИМЕТРИЧНЕ ШИФРУВАННЯ, ШИФРОТЕКСТ, МЕРЕЖЕВІ ТЕХНОЛОГІЇ, ANDROID ДОДАТОК, СЕРЕДОВИЩЕ РОЗРОБКИ, КОНТРОЛЬНА СУМА.

ЗМІСТ

ВСТУП

В сучасному світі широко поширені засоби передавання інформації, кожен з них має своє призначення та цільову аудиторію. З розвитком мережі Інтернет все більше і більше застосунків орієнтовані на хмарні сховища даних. Це дозволяє світовім лідерам IT збирати інформацію про користувачів, контролювати вміст та надавати платні послуги або показувати рекламні повідомлення користувачам.

Превагами таких рішень є можливість користуватися послугами там де є підключення до глобальної мережі з можливістю легкого синхронізування нових пристроїв та мінімальні затрати локального дискового простору. Недоліками таких застосунків є всі ті обмеження які накладають власники на контент який може зберігатися або передаватися. І в разі недотримання обліковий запис може бути заблокованій і разом з ним всі данні, які були передані даному ресурсу. Гарантувати те, що сервери таких сервісів не можуть бути скомпрометовані, говорити не доводиться, більш того цілодобовий доступ до інформації підвищує ризик безпосередньо компрометації самого облікового запису.

Вирішенням цих всіх недоліків є використання локальних застосунків без прямого доступу в Інтернет. Таких додатків існує менше і працюють вони здебільшого в межах однієї операційної системи. Що призводить до ускладнення процесу обміну у разі відсутності портативних накопичувачів або зв’язку з Інтернетом.

В деяких випадках також є обмеження на обладнання яке може бути використано для передачі даних і це обумовлено пошуком сусідніх пристроїв готових до передачі даних. Метою цієї роботи є створення трохи універсальних компонент, які дозволять імплементувати рішення обміну цифровою інформацією в IPv4 мережі на операційних системах Windows 8.1 і вище та Android 4.4.4 і вище. Мовою на якій будуть написані ці компоненти є Java.

Головною компонентною є тунель зв'язку, захищений наскрізним шифруванням. Два інших компонента це графічні додатки в оточенні яких буде працювати головний модуль під різними операційними системами. Також оточення відповідає за збереження користувацької інформації в кожній з операційних систем.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз існуючих рішень

Значну долю ринку займає додаток Shareit (рис. 1.1), це безкоштовний додаток для передачі файлів, що надає змогу пристроям обмінюватися даними через Wi-Fi протокол. Користувачі можуть використовувати його для передачі файлів будь-якого типу, включаючи фотографії, відео, музику, контакти, додатки та інше. Програма розроблена однойменною компанією SHAREit Technologies Co. Ltd і доступна на 45 мовах, включаючи російську, англійську, іспанську, французьку, арабську і китайську.

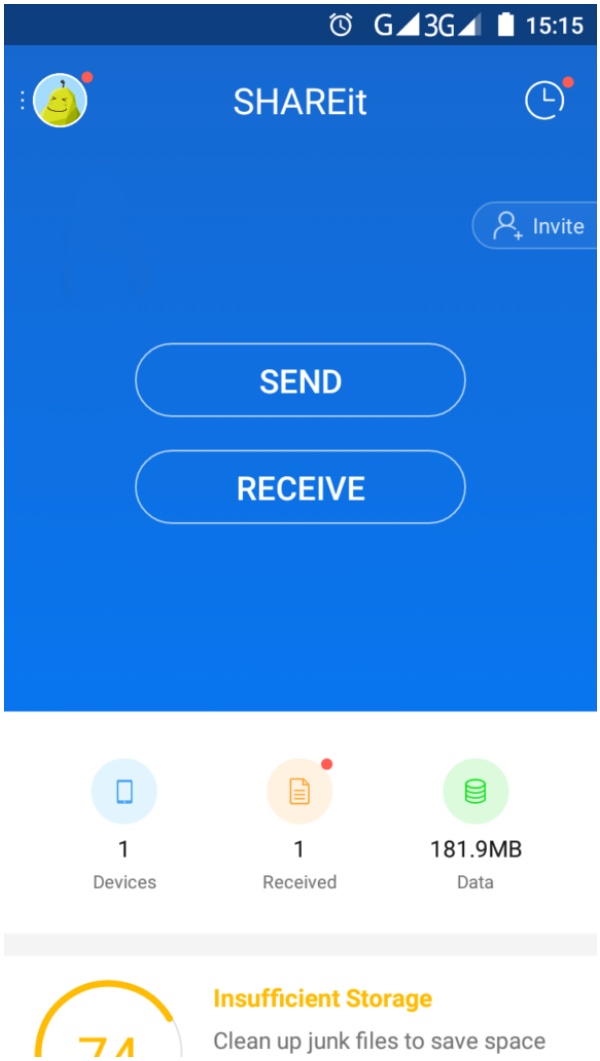

Рисунок 1.1 – Головне меню додатку SHAREit

Головна перевага додатку – більша швидкість передачі даних в порівнянні з Bluetooth протоколом. Трансфер даних безпечний, з огляду на тип підключення, який використовується для передачі файлів. Також важливим є підтримка більшості популярних операційних систем. До появи SHAREit і подібних додатків найкращим єдиним способом для передачі файлів був Bluetooth, або USB-флеш-накопичувач. У разі Bluetooth швидкість передачі даних була незадовільною, а використання USB-накопичувачів мало негативний досвід, пов'язаний зараженням пристроїв, до яких підключені USB-накопичувачі, що містять шкідливі програми.

– Основним недоліками цього додатку є:

– перевантаженість інтерфейсу додатку в останніх версіях, встановлення яких вимагають безпекові виправлення цих версій;

– режим «виживання» сервісів додатку, що негативно впливає на автономність пристрою, тобто якщо закрити додаток він продовжує працювати в фоновому режимі і навіть якщо примусово завершити його сервіси він по при все намагається відновити свою роботу;

– обмеженість використання протоколом Wi-Fi;

– закрита кодова база, як додатку, так і протоколу обміну.

Іншим додатком для обміну файлами і не тільки є Pushbullet (рис. 1.2). Це широкофункціональний додаток, що дозволяє синхронізувати всі власні пристрої в єдину екосистему а допомогою вашого облікового запису Google або Facebook і дозволяє ділитися багатьма речами між телефоном та комп’ютером. Перелік того, що робить Pushbullet:

– обмін файлами та посиланнями;

– дзеркальний перегляд сповіщень телефону на ПК;

– надсилання СМС з ПК;

– віддалений доступ до файлової системи.

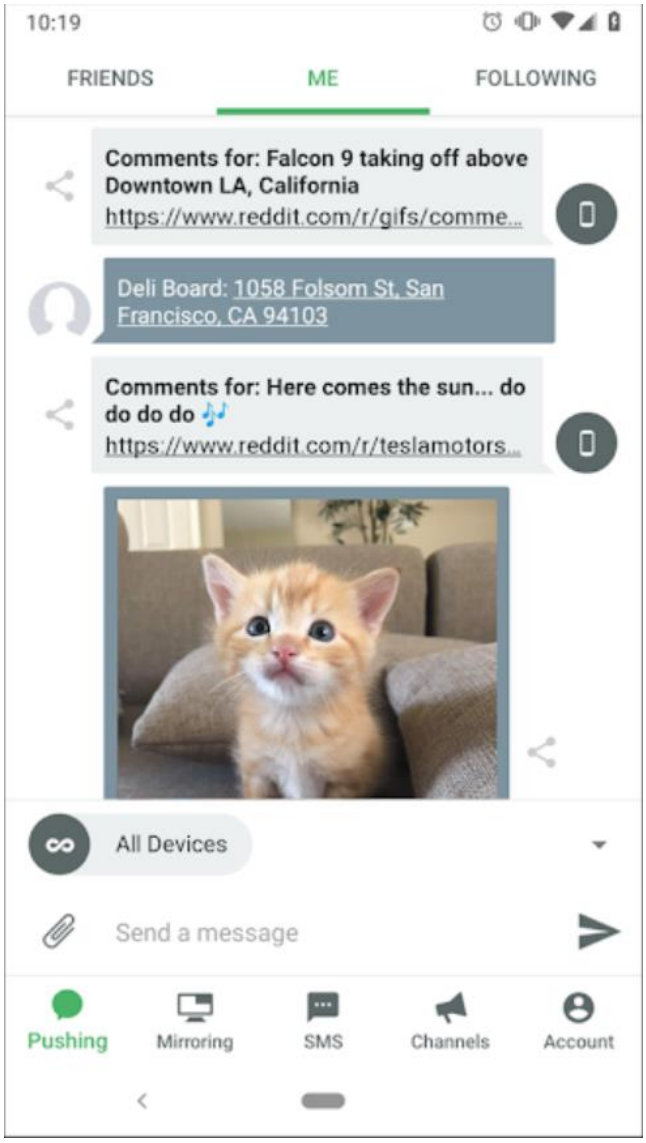

Рисунок 1.2 – Вікно обміну між пристроями в додатку Pushbullet

Головні переваги цього додатку:

– підтримка багатьох операційних систем, у тому числі додаток може бути встановлений як розширення для Chrome-подібних браузерів, але з певними обмеженням по функціоналу;

– легке об’єднання пристроїв за допомогою облікових записів;

– дзеркальне відображення повідомлень мобільних пристроїв;

– доступ до збережених ресурсів за наявності доступу до Інтернету;

– можливість віддалено керувати файловою систему, але тільки в додатках під керуванням операційних систем Windows та Android.

Основні недоліки:

– широкий перелік повноважень які надаються додатку при встановленні;

– перехоплення СМС повідомлень, які синхронізуються через Інтернет;

– відправка відбувається лише при наявності підключення до Інтернету, у разі його відсутності, при наявності локальної мережі додаток не працює;

– всі файли зберігаються на хмарному сховищі, що актуалізує безпекові питання зберігання файлів;

– швидкість передачі файлів обмежена швидкістю підключення до Інтернету;

– відсутність переваг у передачі файлів перед сучасними месенджерами.

Наступне сімейство додатків для обміну повідомленнями, які також можна використовувати для обміну файлами між пристроями є сучасні месенджери такі як Viber (рис. 1.4), Telegram (рис. 1.5) або WhatsApp (рис. 1.3).

Месенджери працюють на більшості операційних систем, Telegram та WhatsApp мають веб версії додатків, що дозволяють без перешкод попрацювати за чужим комп’ютером зберегти результати своєї роботи.

Telegram та Viber підтримують чат «з собою» за замовчуванням, а у WhatsApp такої функції не має, натомість додаток дозволяє створити чат з будь-ким та видалити всіх окрім себе.

Месенджери надають необмежене розміром сховище, але мають певні обмеження на максимальний розмір файлу.

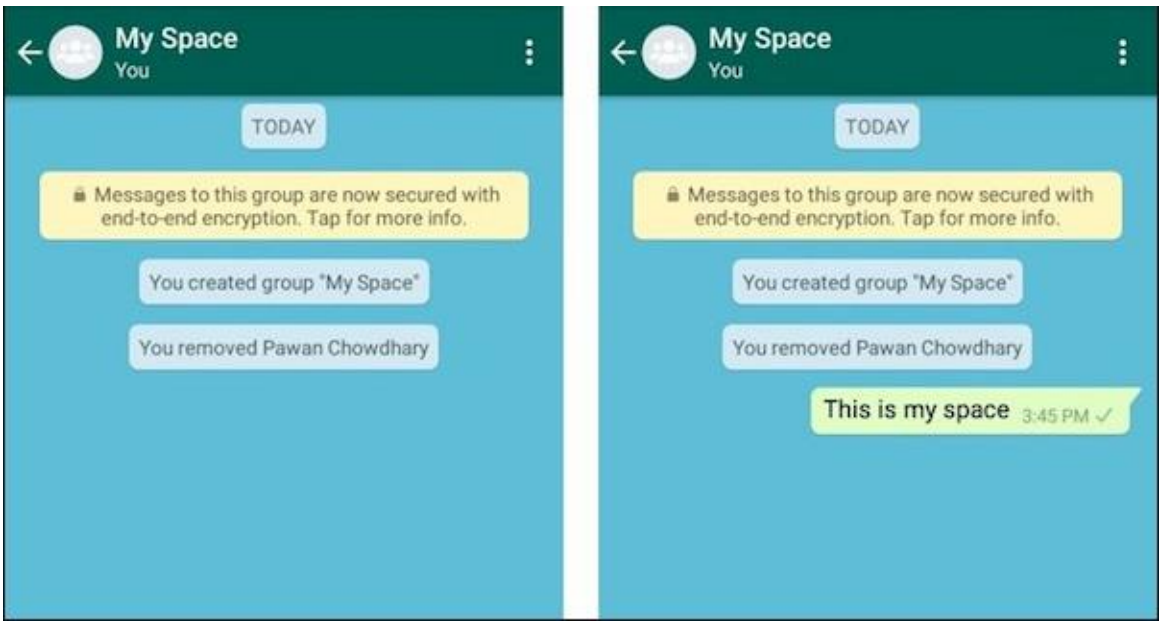

Рисунок 1.3 – Створення чату з собою у WhatsApp

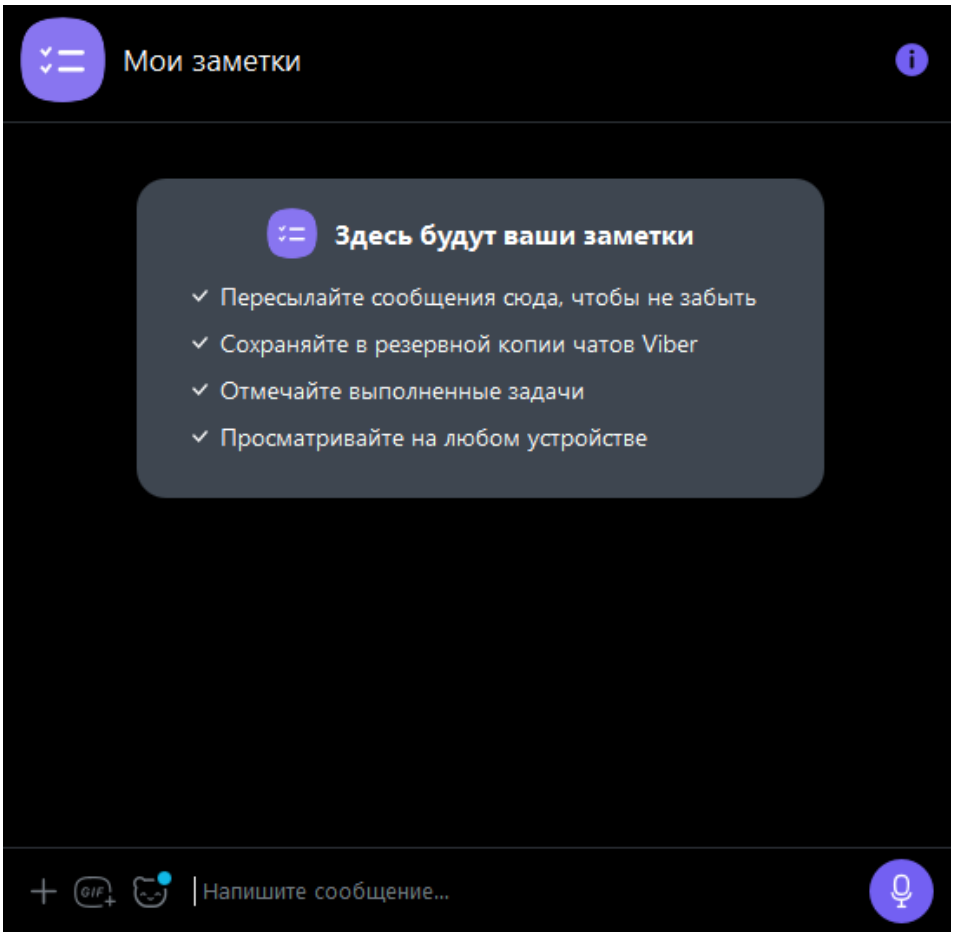

Рисунок 1.4 – Чат з собою у Viber

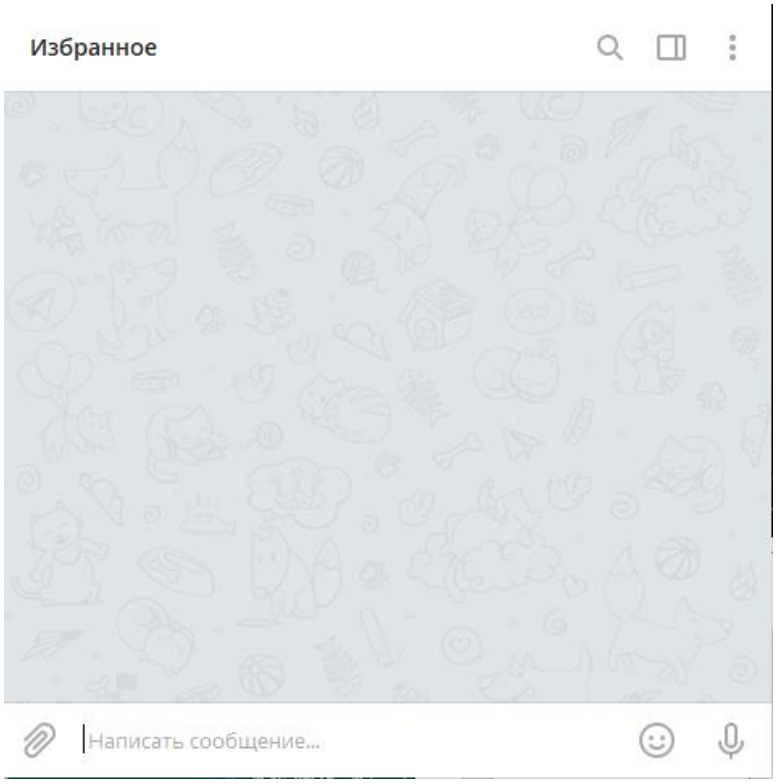

Рисунок 1.5 – Чат з собою у Telegram

Головні переваги цього підходу:

– легкість переходу – у разі використання одно із цих месенджерів не потрібно створювати нові аккаунти;

– доступ до файлових ресурсів всюди, де є доступ до Інтернету;

– є можливість легко поділитися файлами з іншими людьми;

Основні недоліки:

– відправка відбувається лише при наявності підключення до Інтернету;

– всі файли зберігаються на хмарному сховищі, що актуалізує безпекові питання зберігання файлів;

– швидкість передачі файлів обмежена швидкістю підключення до Інтернету;

– Заблокований по будь-яким причинам аккаунт, без повернення втрачає всі збережені дані.

1.2 Постановка задачі

Розробка протоколу обміну має відповідати таким характеристикам:

– Довжина ключа асиметричного шифрування має становити 2048 біт.

– Довжина ключа симетричного шифрування має становити 256 біт.

– Дані на встановлення зв’язку мають бути підписані приватним ключем відправника, зашифрувавши контрольну суму відправленого повідомлення.

– Всі ідентифікуючі данні мають бути передані по протоколу UDP і валідовані.

– Використання всіх наявних мережевих підключень.

– Блокування ниток процесів потоками введення виведення для підвищення енергоефективності.

– Закриття невикористовуваних з’єднань після однієї хвилини простою.

– Повторні запити відправляються першу хвилину кожні 5 секунд, в подальшому – раз в хвилину або коли цільовий пристрій заявить про себе в мережі.

– Кожен пристрій щохвилини розсилає широкомовний пакет, заявляючи свою присутність.

Вимоги до додатку Android:

– Підтримка версій операційної систем починаючи з 4.4.4.

– Повна імплементація розробленого протоколу.

– Обмеження розсилки при вимкненому екрані портативного пристрою.

Вимоги до додатку Windows:

– Підтримка версій операційної систем починаючи з 8.1.

– Повна імплементація розробленого протоколу.

2 ВИБІР МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

2.1 Проектування власного протоколу встановлення захищеного зв’язку

Світу вже давно відомий протокол Діффі — Геллмана, за допомогою якого можно створити захищенний канал зв’язку без попередніх домовленостей. Але від даної технології необхідно відмовитись у зв'язку з тим, що такий протокол вимагає зберігати відправнику значний перелік ключів для кожного отримувача, у разі якщо необхідно створити через деякий час ще одну сессію захищеного звязку. Також генерація нових ключів вимагає витрат процессорного часу на перевірку нових ключів на криптостійкість.

На відміну від протоколу Діффі — Геллмана, існує протокол RSA який побудований на разовій гонерації пари ключів, що дозволяє скоротити час встановлення зв'язку та гарантувати в подальшому простий спосіб створення нової сессії, та підтвердження володіння приватним ключом за допомогою підпису шифрованого пакету.

Основним ідентифікатором пристрою є публічний ключ RSA 2048. Сервіс додатку має відповідати на широкомовні запити з боку інших пристроїв.

Якщо буде отримано пакет з довжиною рівною нулю, тоді додаток має відповісти у юнікод форматі за шаблоном:

{L/W/A/M}{NAME}{md5(Public key)}

Повідомлення складається з трьох частин (виділених фігурними дужками), перша частина – операційна система, виконує лише інформаційну роль, де «L» – Linux, «W» - windows, «A» – android, «M» – MAC OS / IOS. Друга частина це ім’я пристрою яке побачать інші користувачі. Третя частина – це md5 хеш публічного ключа. Коротка відповідь на такий запит обумовлена частим використанням для одержання списку активних девайсів. Хеш md5 використовується через його простоту й достотню довжину що важливо на портативних пристроях, які працюють від акумуляторних джерел живлення, а всі хешовані данні і без того

відомі або можуть стати відомі за запитом отримувачу і використовується лише як скорочення або контрольна сумма.

Запит публічного ключа має формат:

{P}{md5(Public key)}

Де «P» статичний флаг запиту публічного ключа іншого пристрою.

Одержувач має перевірити коректність хешу та відкинути некоректні запити. У разі відповідності відповідь має складатися з флагу «F» та публічного ключа пристрою в форматі:

{F}{Public key RSA 2048}

В свою чергу одержувач має перевірити відповідь на відповідність публічного ключа наявному хешу при його одержанні, це дозволить запобігти спотворенню інформації, яка може відбутися з використанням протоколу UDP.

Запит на встановлення захищеного з’єднання має формат:

{C}{SIGNED(Public key RSA 2048, Enrypted connection data))}{SIGN}

На встановлення захищеного з’єднання вказує флаг «C», друга частина запиту складається з двох частин публічного ключа відправника, та зашифрованого повідомлення публічнім ключем отримувача. В цьому повідомленні міститься номер сесії, ключ симетричного шифрування, часова відмітка, загальний розмір передаваних даних, порт на якому розміщений TCP server socket.Третя частина пакету – це підпис хешу md5 другої частини. Підпис виконується за допомогою приватного ключа відправника.

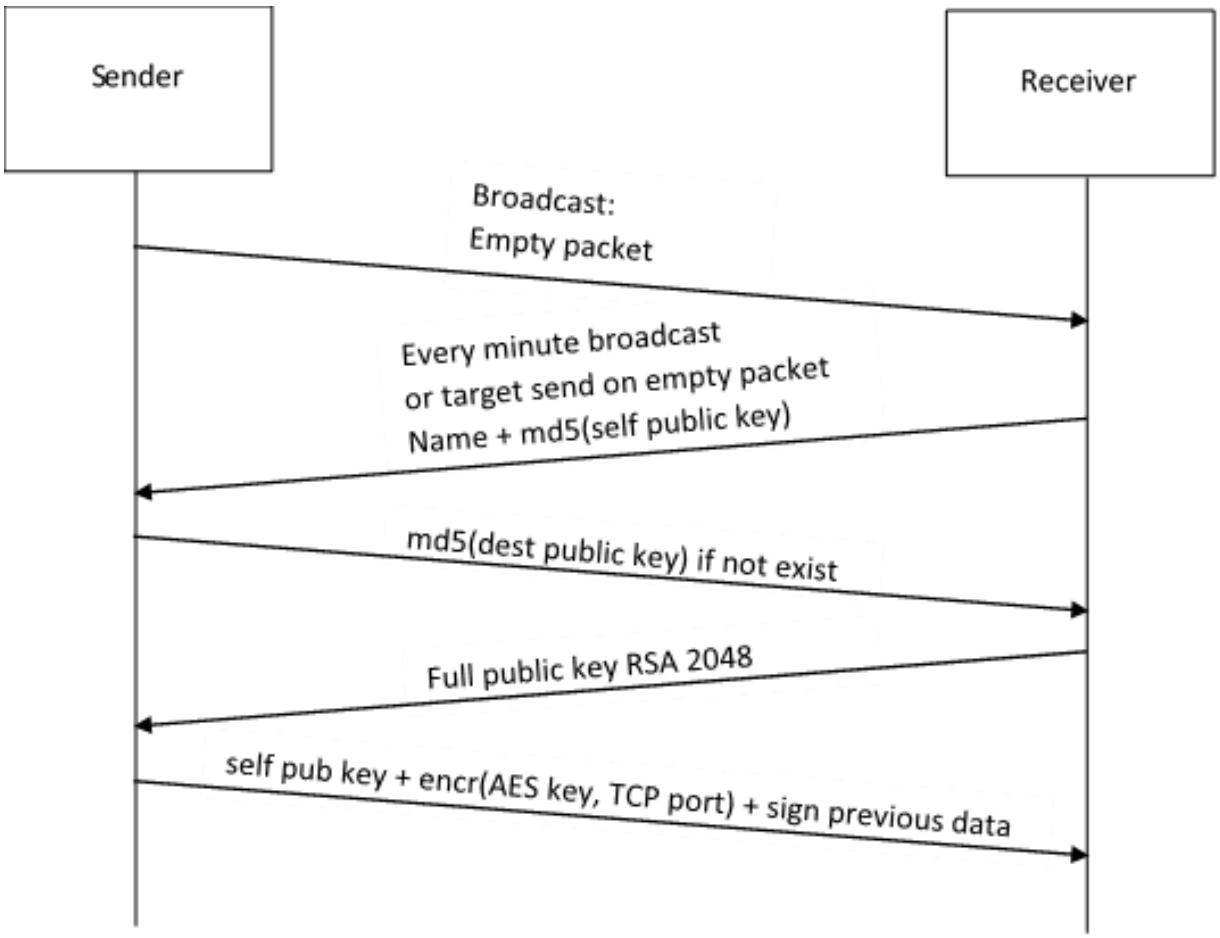

Рисунок 2.1 – UML діаграма створення нового захіщеного зв'язку

Підводячи проміжний підсумок треба підкреслити, що пара ключів асиметричного шифрування використовуються для двох цілей, а саме закрита доставка до отримувача даних, які необхідні для встановлення шифрованого зв’язку та підтвердженням того, що відправник дійсно володіє приватним ключом, як наслідок після проведення перевірок достовірності можна гарантувати, що був отриманих не пошкоджений і не змінений пакет.

Нижче наведений код класу Security, що відповідає за всі операції шифрування, які виконуються в додатку і були описані в цьому підрозділі:

```
public class Security {
public static KeyPair generateKeyPair() {
try {
KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
keyPairGenerator.initialize(2048);
```

```
return keyPairGenerator.genKeyPair();
} catch (NoSuchAlgorithmException ignored) {
throw new RuntimeException();
}
}


public static byte[] encrypt(PublicKey publicKey, byte[] data) {
try {
Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.ENCRYPT_MODE, publicKey);
return cipher.doFinal(data);
} catch (GeneralSecurityException e) {
throw new RuntimeException(e);
}
}


public static byte[] decrypt(PrivateKey privateKey, byte[] data) {
try {
Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.DECRYPT_MODE, privateKey);
return cipher.doFinal(data);
} catch (GeneralSecurityException e) {
throw new RuntimeException(e);
}
}


public static SecretKey generateAES() {
try {
KeyGenerator keygen = KeyGenerator.getInstance("AES");
keygen.init(256);
return keygen.generateKey();
} catch (NoSuchAlgorithmException e) {
throw new RuntimeException();
}
}


public static InputStream secureInputStream(InputStream in, SecretKey key) throws
NoSuchPaddingException,      NoSuchAlgorithmException,      InvalidKeyException,
InvalidAlgorithmParameterException {
Cipher cipher = Cipher.getInstance("AES/CFB8/NoPadding");
cipher.init(Cipher.DECRYPT_MODE, key, new IvParameterSpec(new byte[16]));
return new CipherInputStream(in, cipher);
}


public static OutputStream secureOutputStream(OutputStream out, SecretKey key)
throws NoSuchPaddingException, NoSuchAlgorithmException, InvalidKeyException,
InvalidAlgorithmParameterException {
Cipher cipher = Cipher.getInstance("AES/CFB8/NoPadding");
cipher.init(Cipher.ENCRYPT_MODE, key, new IvParameterSpec(new byte[16]));
return new CipherOutputStream(out, cipher);
}
```

```
public static byte[] signatureData(PrivateKey privateKey, byte[] input){
try {
Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.ENCRYPT_MODE, privateKey);
return cipher.doFinal(Security.md5(input).getBytes());
} catch (GeneralSecurityException e) {
return null;
}
}


public static byte[] checkSignature(PublicKey publicKey, byte[] input) throws
GeneralSecurityException{
if(input.length < 256) throw new GeneralSecurityException("Not valid");
byte[] data = clip(input, 0, input.length-257);
byte[] signature = clip(input, input.length-256, input.length-1);
Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.DECRYPT_MODE, publicKey);
if(Arrays.equals(cipher.doFinal(signature),       Security.md5(data).getBytes()))
return data;
throw new GeneralSecurityException("Not valid");
}
}
```

2.2 Проектування додатку Android

В основу покладено принцип проектування KISS (keep it short and simple), тому додаток було розділено на три вікна, а саме список пристроїв в мережі, список файлів на відправу (головне вікно) та список подій.

Оскільки кожне вікно має списки розроблено універсальна адаптивна лінія (рис. 2.2) для кожного з них.

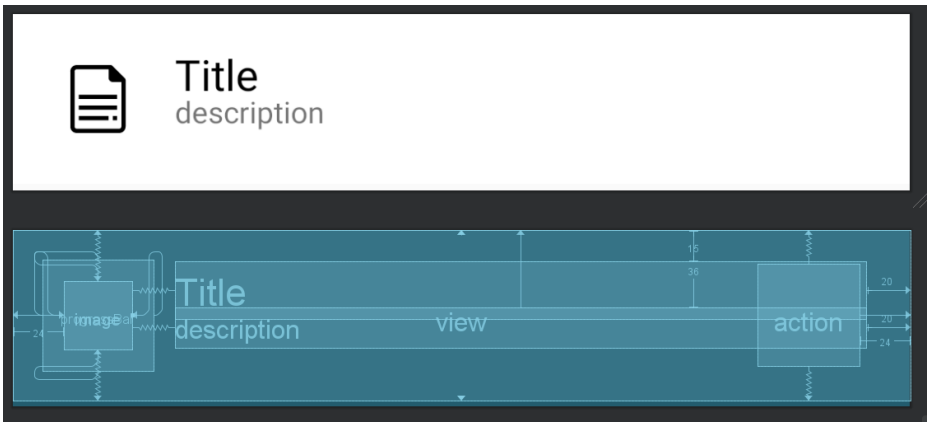

Рисунок 2.2 – Лінія списків

Цей елемент складається з піктограми яка приймає відповідний вигляд залежності від вмісту.

Навколо піктограми розміщений прихований круговий прогресбар який відображає процес відправки або одержання файлу. З іншого боку розмішена кнопка дій яка з’являється при виділенні цього елементу списку. Між цими компонентами розміщено два поля з заголовком та коротким описом.

Приклад використання (рис. 2.3) елемента списку на головному вікні додатку.

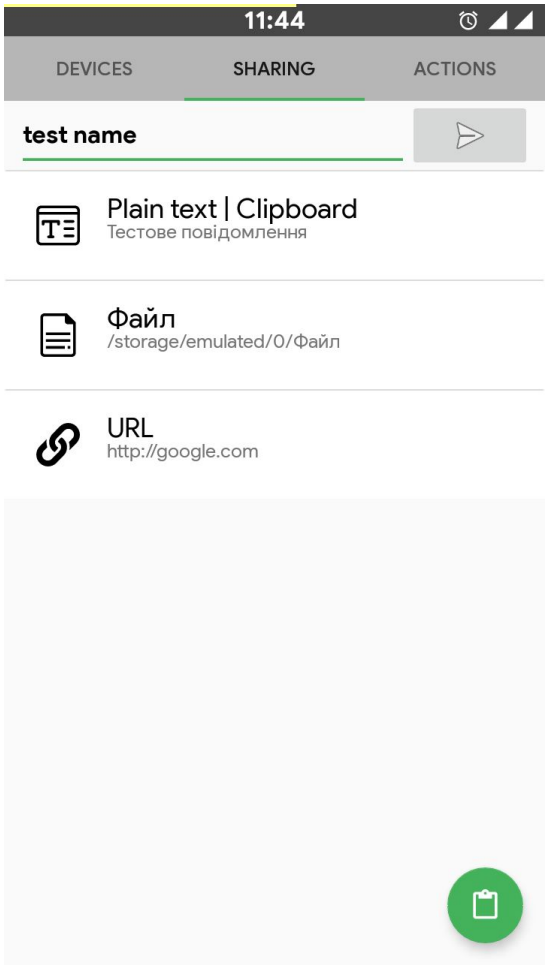

Рисунок 2.3 – Головне вікно додатку

На головному вікні розміщений список елементів готових до відправки, плаваюча кнопка додавання в список вмісту буферу обміну, поле для зміни назви пристрою в мережі та кнопка переходу до вибору одержувачів.

Список одержувачів (рис. 2.4) має аналогічну будову, за виключенням піктограм, які в донному випадку відповідають операційній системі, а їх колір вказує на те, чи внесий цей отримувач в список довірених (якщо так, то блакитний, якщо ні - червоний). Прогресбар в цьому випадку з’являється якщо пристрій онлайн (заповнюється на 100% и приймає зелений колір). Плаваюча кнопка оновлює список пристроїв які є в мережі. У разі виділення декількох

отримувачів плаваюча кнопка змінює зображення (рис. 2.5) на відповідне і у разі натискання відправляє повідомлення всім виділеним одержувачам.

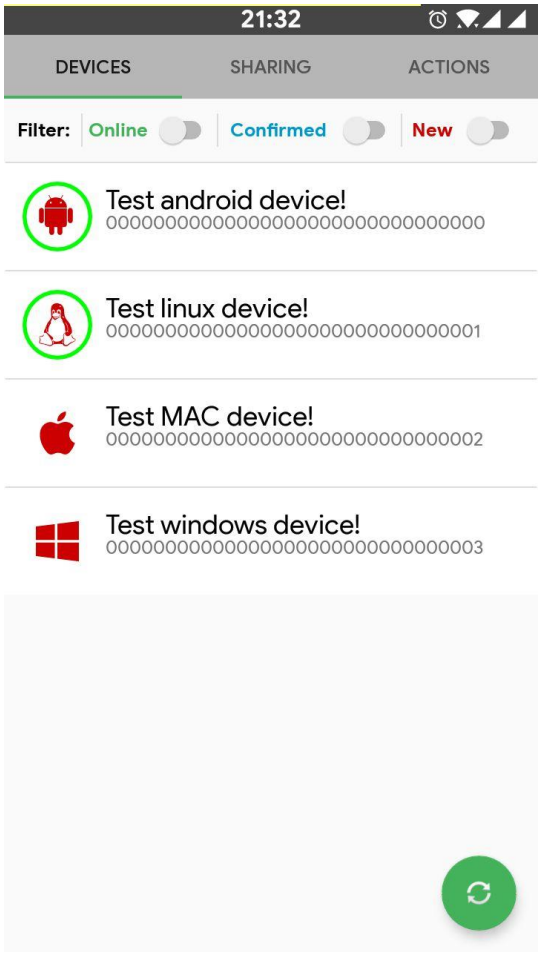

Рисунок 2.4 – Список нових пристроїв в мережі різних ОС

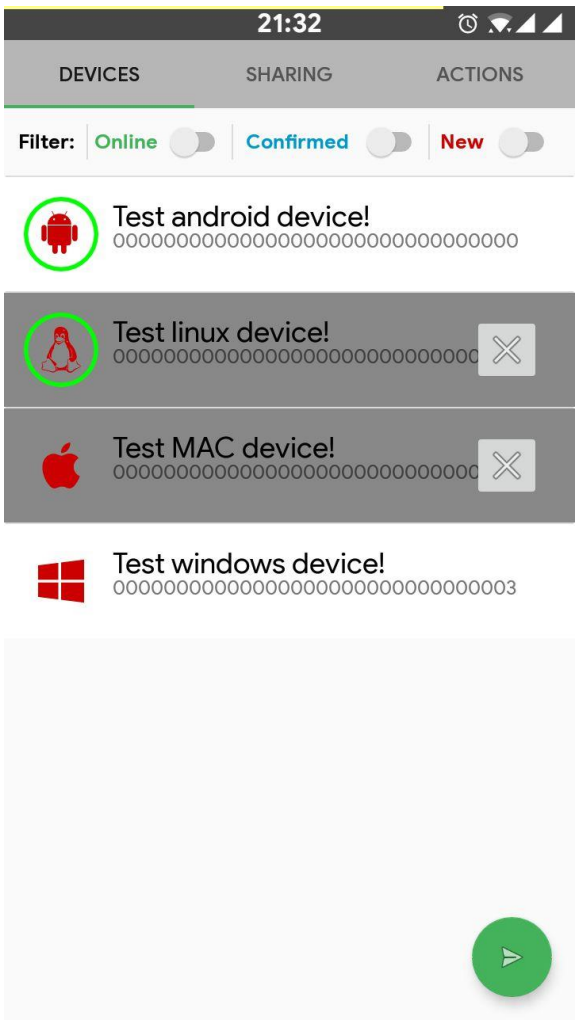

Рисунок 2.5 – Приклад виділення декількох пристроїв

Вікно подій містить аналогічний список, що містить завдання та показує їх статус виконання. В ньому можна переглянути статус відправки або отримання файлів, у разі необхідності відмінити передачу.

Приклад відправлення файлу з Android додатку:

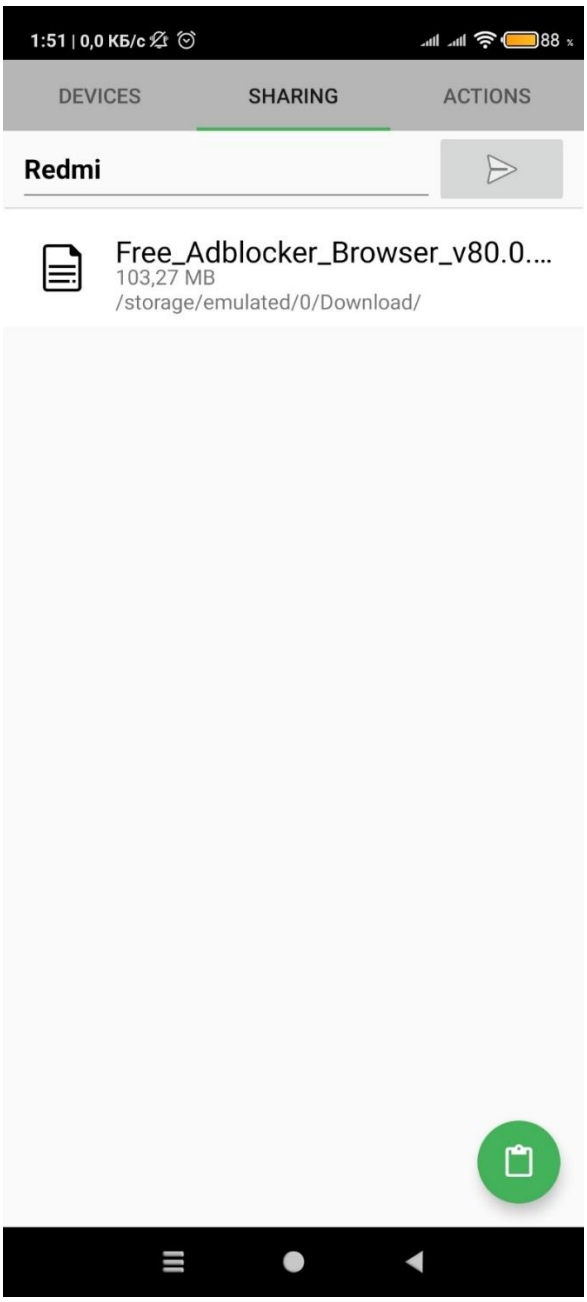

Рисунок 2.6 – Підготовка списку об’єктів для відправлення

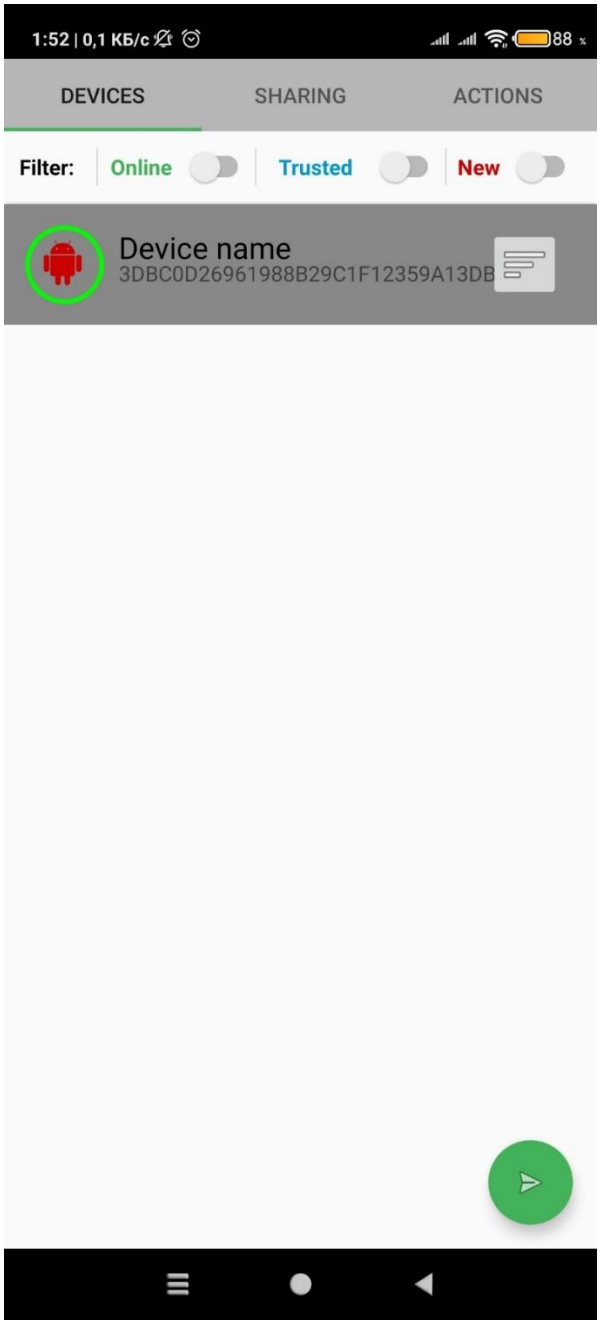

Рисунок 2.7 – Обрання цільових пристроїв

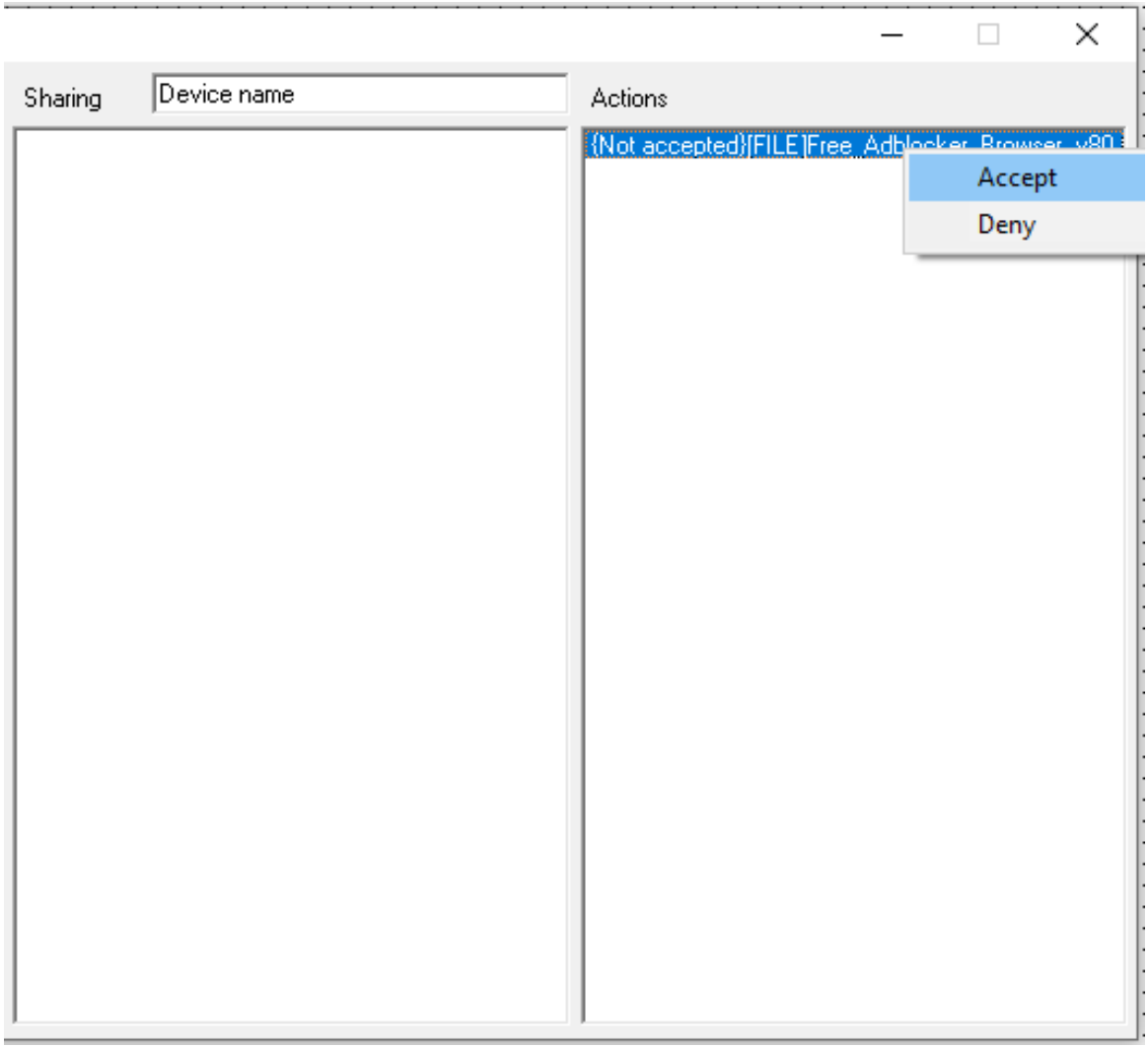

Рисунок 2.8 – Підтвердження відправлення

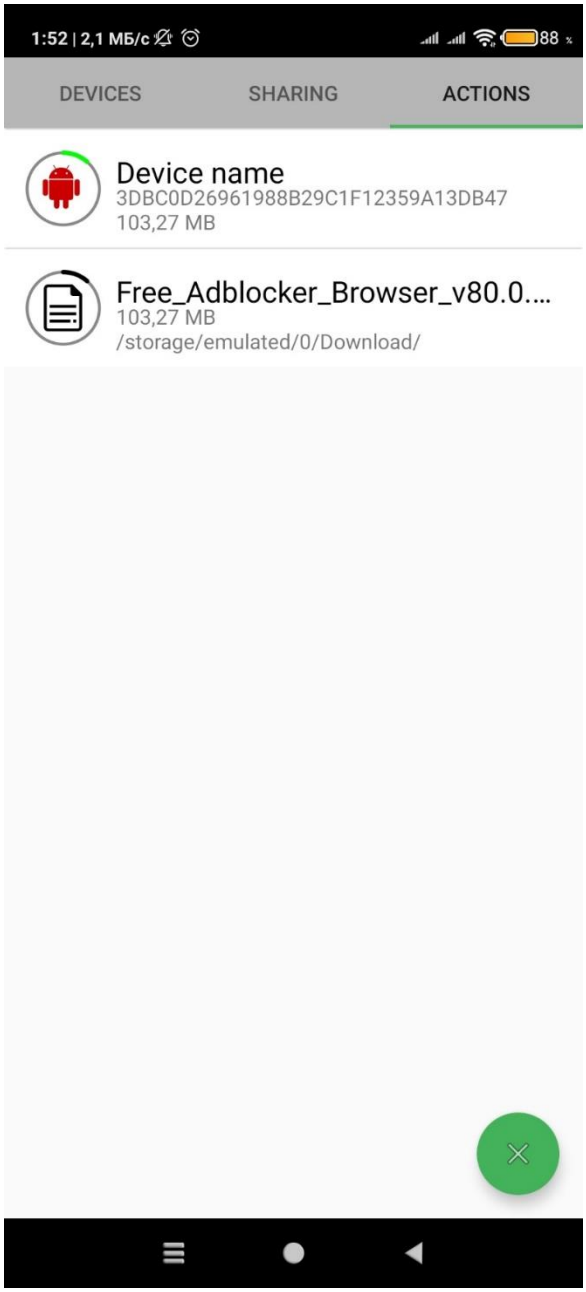

Рисунок 2.9 – Триває передавання

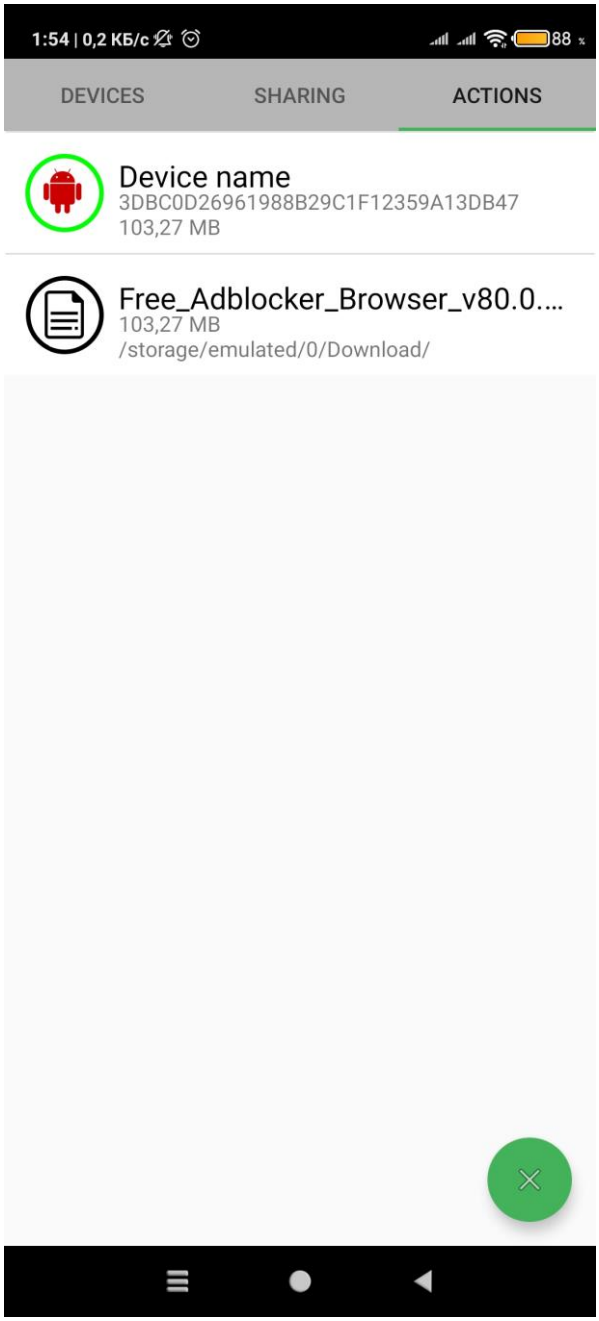

Рисунок 2.10 – Передавання завершено

2.3 Проектування інтерфейсу додатку Windows

Виходячи з досвіду побудови додатку на Android, для спрощення орієнтації в компонентах, було розроблено близький інтерфейс, виконаний на стандартних компонентах операційної системи Windows, де вікно є GUI компонентом яким керує Java код, та передає команді від користувача ядру.

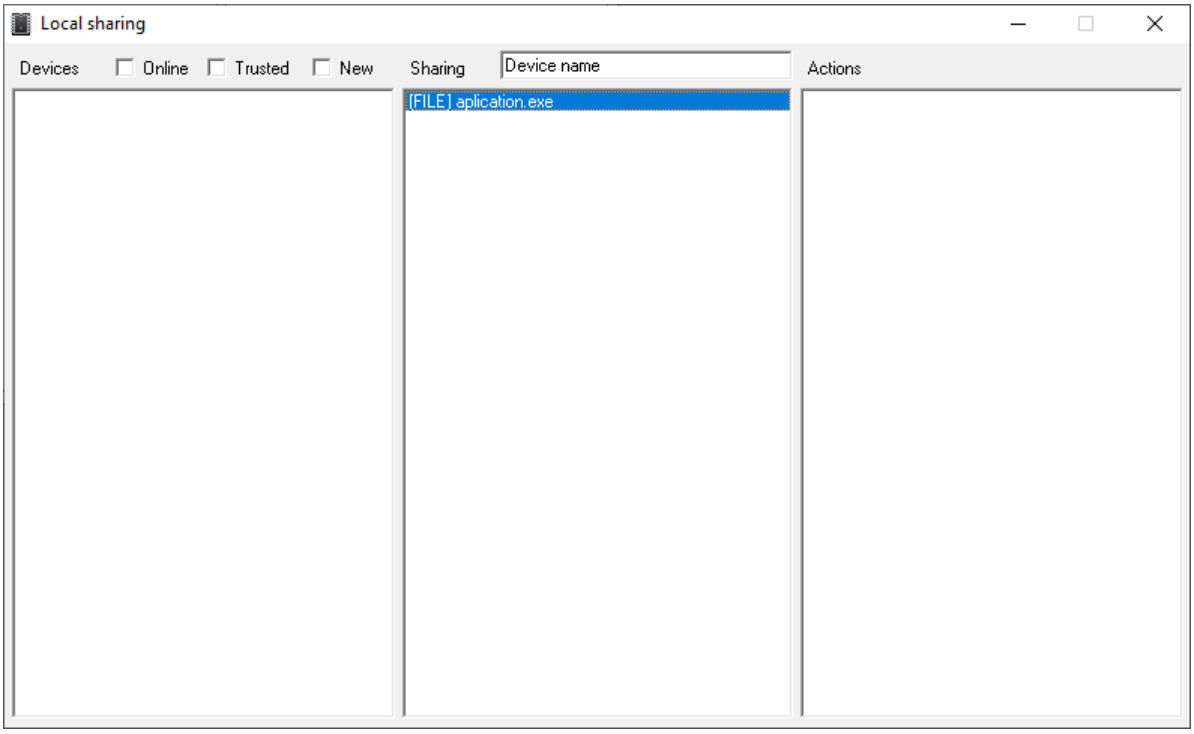

Рисунок 2.11 – Підготовка списку файлів

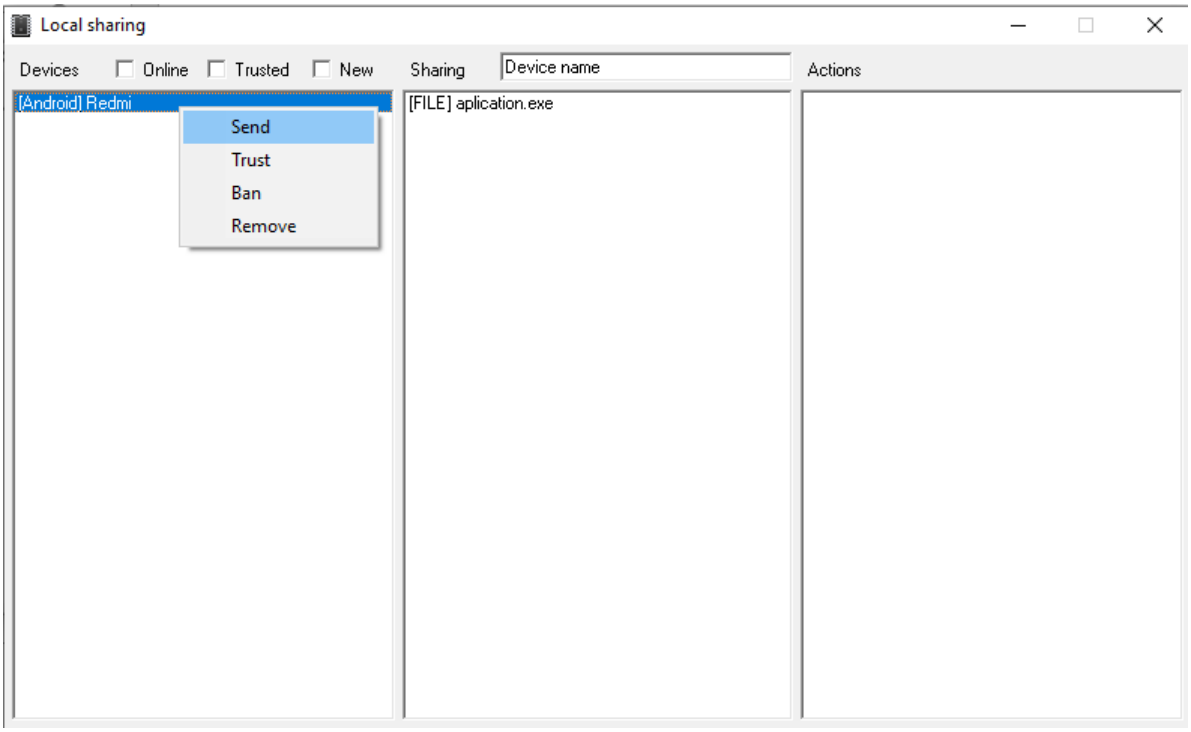

Рисунок 2.12 – Обрання цільового пристрою

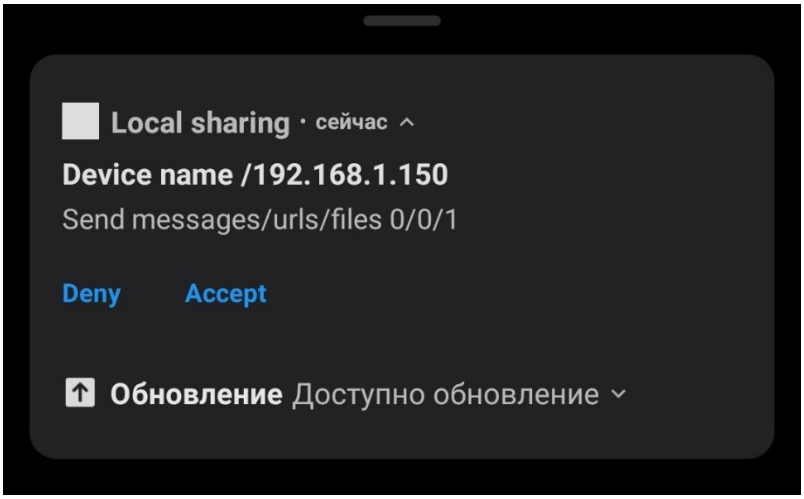

Рисунок 2.13 – Підтвердження пересилки

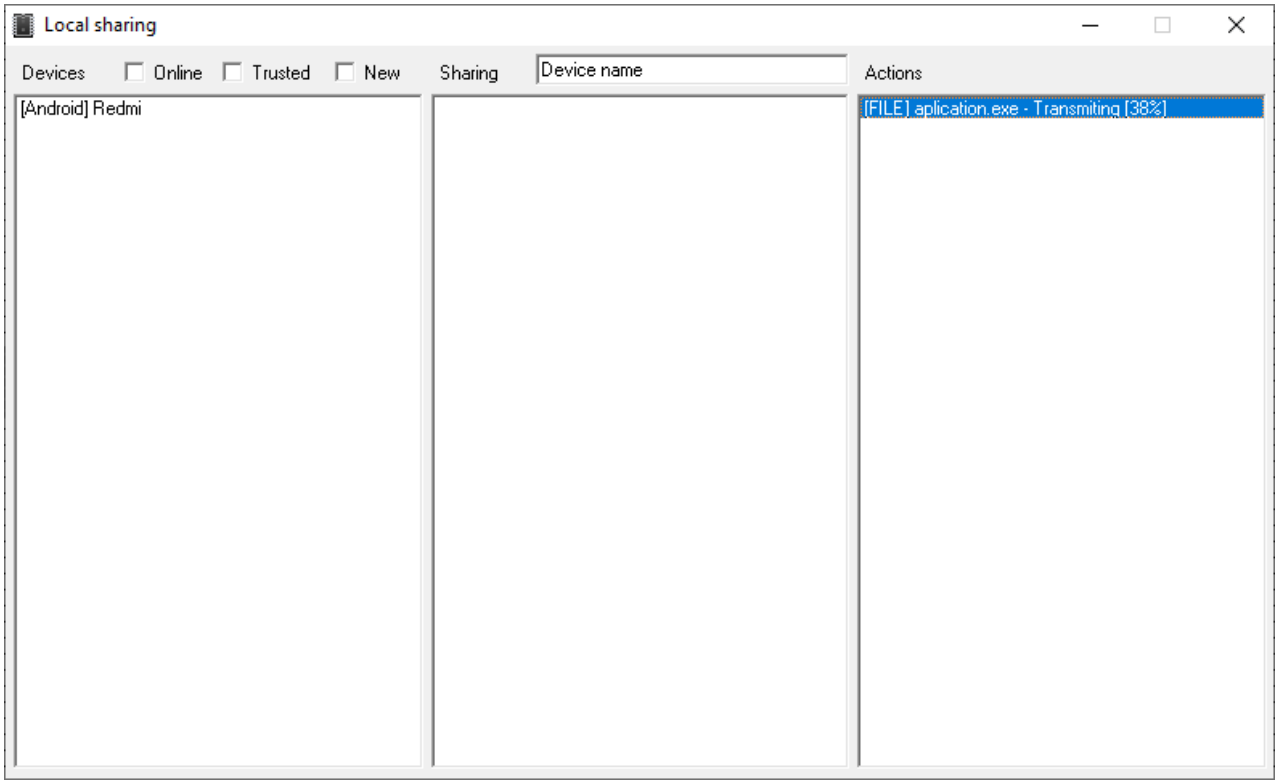

Рисунок 2.14 – Триває передавання

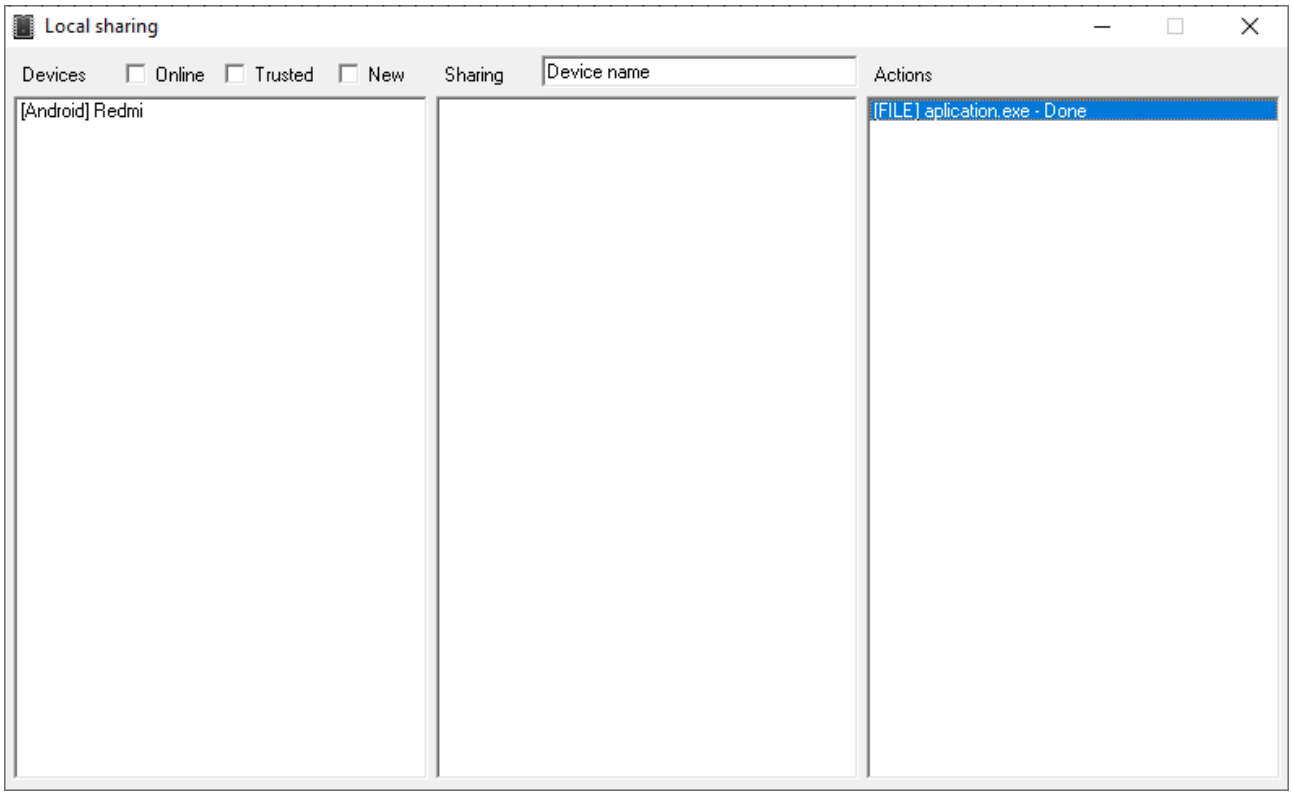

Рисунок 2.15 – Передавання завершено

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

Програмна реалізація додатку відбувається в у запропонованому Google середовищі Android Studio. Це середовище має широкий спектр інструментів для розробки Android-додатків. Для зберігання даних в додатку використано SQLlite, як легка та функціональна БД. Це сховище має містити відомості стосовно власних ключів шифрування, переліку файлів, що накоплені на відправу, користувачів та їх публічні ключі, список подій та їх статус. База даних необхідна вразі завершення сервісу додатку, який містить всі ці дані, окрім публічних ключів інших користувачів.

Приклад коду абстрактного класу списку BaseFragment:

```
public abstract class BaseFragment {
protected List<Line> lines = new LinkedList<>();
protected MainActivity MAIN;
protected View root;
protected ViewGroup viewGroup;

public BaseFragment(View root, MainActivity MAIN) {
this.MAIN = MAIN;
this.root = root;
this.viewGroup = root.findViewById(getLayout());
}

protected abstract int getLayout();

abstract Set<? extends CanBuildLine> getServiceList();

public boolean update(){
Set<? extends CanBuildLine> list = getServiceList();

try {
int i = 0;
if(list.size() == 0) throw new RuntimeException();
for (CanBuildLine line : list)
while (line != lines.get(i++).getParent())
if (i == lines.size()) throw new RuntimeException();
for(Line line : lines){
if(list.contains(line.getParent())) line.getParent().updateLine(line, this);
else removeLine(line);
}
} catch (Exception ex){
for (Line line : lines) line.remove();
lines.clear();
for (CanBuildLine buildLine : list) lines.add(buildLine.buildLine(viewGroup,
this));
}
```

```
return true;
}

public void addLine(CanBuildLine line){
lines.add(line.buildLine(viewGroup, this));
}

public boolean removeLine(Line line){
if(lines.contains(line) && SharingService.Pipe.remove(line.getParent()))
return lines.remove(line.remove());
return false;
}

public void OnIcon(Line line){}

public void OnSelect(Line line){}

public void OnUnSelect(Line line){}

public void OnMenuSelect(Line line, int index){}

public void OnPanelSelect(){}

public void OnFabClick(){}

public List<String> getMenu(Line line){return new ArrayList<>(); }

public List<Line> getLines() {
return lines;
}

public List<Line> getSelected(){
List<Line> ll = new ArrayList<>();
for(Line l: lines) if (l.isSelected()) ll.add(l);
return ll;
}
}
```

Цей код спадкується іншими класами, що відповідають за списки пристрої, накопичених файли та події, які реалізовують його абстрактні методи відповідно до потреб кожного з списків.

Розробка додатку під операційну систему Windows відбувається за рахунок перевикористання модуля транспортування шляхом компілювання в jar модуль з графічнім оточенням Windows.

ВИСНОВКИ

В результаті виконання дипломного проекту було створено і детально описано протокол встановлення зашифрованого з’єднання в основу якого покладено наскрізне шифрування, який вдалося зробити стійким, безпечним та ефективним, в мережі IPv4 використовуючи протоколи TCP та UDP з урахуванням переваг кожного з них. А саме пошук пристроїв по всіх наявних мережевих підключеннях без значних витрат на пошук сусідів, аналогічно ARP запитам. Та у разі необхідності створення сесії передачі даних відкривається тривале TCP підключення.

Простий опис протоколу дозволить впровадити його як плагін до інших файлових менеджерах різних операційних систем, що дозволить легко синхронізувати все більше пристроїв. Також Windows додаток може використовуватися під керуванням GNU/Linux використовуючи емулятор «Wine».

У випадку цього дипломного проекту було імплементовано цей протокол у вигляді додатків для операційних систем Windows та Android. Додаток має простий та зрозумілий інтерфейс, мінімум екранів, які надають простоту та легкість використання. Основними принципами розробки стали «Single Responsibility» та «Keep it short and simple».

На основі публічних ключі розроблена система довіреностей, що дозволяє легко обмінюватися файлами між власними пристроями без зайвих підтверджень. Також було створено систему фільтрів, що дозволяє розділити всі пристрої в мережі.

Широке використання захищених протоколів дозволяє надати користувачу упевненість в безпечності його даних, хоча б поки всі системи знаходяться за локальним брандмауером.

СПИСОК ЛІТЕРАТУРИ

1. Гринберг, UX-дизайн. Идея – Гринберг С., Карпендейл Ш., Маркардт Н., Бакстон Б.. – СПб.: Издательство "Питер", 2014. - 272 с.
2. Берд, Руководство по разработке Android приложений, Барри – Барри Берд. - М.: Диалектика – Вильямс, 2014. - 521 с.
3. Гарнаев, Андрей Программирование на Java – Андрей Гарнаев , Сергей Гарнаев. – Москва: СПб. [и др.] : Питер, 2017. - 718 с.
4. Гонсалвес, Энтони Изучаем Android – Энтони Гонсалвес. – М.: Питер, 2016. - 640 с.
5. Гупта, Арун Java SE 7 / Арун Гупта. – М.: Вильямс, 2014. - 336 с.
6. Монахов, В. Язык программирования – В. Монахов. – М.: БХВ-Петербург, 2012. - 720 с.
7. Савитч, Уолтер Язык Java. Курс программирования под Android –Уолтер Савитч. - М.: Вильямс, 2015. - 928 с.
8. Хабибуллин, Ильдар Самоучитель Java – Ильдар Хабибуллин. – М.: БХВ-Петербург, 2014. - 768 с.
9. Шилдт, Герберт Java 8. Руководство для начинающих – Герберт Шилдт. - М.: Вильямс, 2015. - 720 с.
10. Эккель, Брюс Философия Java – Брюс Эккель. - М.: Питер, 2016. - 809 с.
11. А. Молдовян Криптография / А. Молдовян, Н. Молдовян, Б. Советов. - М.: СПб: Лань, 2019. - 224 с.
12. А. Щербаков Прикладная криптография. Использование и синтез криптографических интерфейсов / А. Щербаков, А. Домашев. - М.: Русская Редакция, 2010. - 406 с.
13. А.А. Малюк Введение в защиту информации в автоматизированных системах. Учебное пособие / А.А. Малюк. - М.: Горячая линия - Телеком, 2013. - 148 с.

14. А.В. Соколов Защита информации в распределенных корпоративных сетях и системах / А.В. Соколов, В.Ф. Шаньгин. - М.: ДМК Пресс, 2010. - 656 с.
15. А.В. Спесивцев Защита информации в персональных ЭВМ / А.В. Спесивцев, В.А. Вегнер, А.Ю. Крутяков. - М.: Радио и связь, 2013. - 192 с.
16. В.А. Галатенко Стандарты информационной безопасности / В.А. Галатенко. - М.: Интуит. ру Интернет-Университет Информационных Технологий, 2019. - 328 с.
17. В.Ф. Шаньгин Защита компьютерной информации / В.Ф. Шаньгин. - М.: ДМК Пресс, 2015. - 544 с.
18. Дж.К. Фостер Разработка средств безопасности и эксплойтов / Дж.К. Фостер, В. Лю. - М.: Питер, 2017. - 432 с.
19. М. Ховард Защищенный код / М. Ховард, М. Леблан. - М.: Русская редакция; Издание 2-е, испр., 2010. - 704 с.
20. Мельников Защита информации в компьютерных системах / Мельников, Викторович Виталий. - М.: Финансы и статистика; Электроинформ, 2014. - 368 с.
21. A. Lee, NIST Special Publication 800-21, Guideline for Implementing Cryptography in the Federal Government, National Institute of Standards and Technology, 1999 - 86 p.
22. A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, New York, 1997 - 132 p.
23. Elonka Dunin, Klaus Schmeh, Code Breaking and Cryptograms – A Pracitical Guide. Robinson 2020 - 155 p.
24. O. A. Logachev, V. V Yashchenko, Boolean Functions in Coding Theory and Cryptography. American Mathematical Society, Providence RI 2012 - 240 p.
25. Craig P. Bauer, Secret History – The Story of Cryptology. CRC Press, Boca Raton 2013 - 213 p.

ДОДАТОК А

Вихідний код додатку, файл MainActivity.java

```
package com.example.main;

public class MainActivity extends AppCompatActivity {
private Handler handler;
public ViewPager viewPager;
private DevicesFragment devicesFragment;
private SharingFragment sharingFragment;
private ActionsFragment actionsFragment;
private FloatingActionButton fab;
public static LayoutInflater inflater;

@SuppressLint("HandlerLeak")
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(new Bundle());
setContentView(R.layout.activity_main);

System.out.println(CONFIG.DIR_DESTINATION);
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN) {
ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE,
Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);
} else ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);

inflater = LayoutInflater.from(MainActivity.this);
handler = new Handler() {
@Override
public void handleMessage(@NonNull Message msg) {
switch (msg.what) {
case -10:
showMessage("Permission denied");
break;
case -1:
if(!update()) handler.sendEmptyMessageDelayed(-1,50);
break;
case 0:
if (viewPager.getCurrentItem() != 0) break;
handler.removeMessages(0);
handler.sendEmptyMessageDelayed(0,30000);
Network.sendQueryOnlineDevices();
break;
}
}
};

stayAliveService();

onSharedIntent();
fab = findViewById(R.id.fab);
SectionsPagerAdapter sectionsPagerAdapter = new SectionsPagerAdapter(this,
getSupportFragmentManager());
viewPager = findViewById(R.id.view_pager);
```

```
viewPager.setAdapter(sectionsPagerAdapter);
((TabLayout) findViewById(R.id.tabs)).setupWithViewPager(viewPager);
viewPager.setCurrentItem(1);
viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
@Override
public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {
}

@Override
public void onPageSelected(int position) {
switch (position) {
case 0:
handler.sendEmptyMessage(0);
if(devicesFragment != null) devicesFragment.OnPanelSelect();
else fab.setImageResource(android.R.drawable.ic_popup_sync);
break;
case 1:
handler.removeMessages(0);
fab.setImageDrawable(getPasteIcon());
if(sharingFragment != null) sharingFragment.OnPanelSelect();
break;
case 2:
handler.removeMessages(0);
fab.setImageResource(android.R.drawable.ic_menu_close_clear_cancel);
if(actionsFragment != null) actionsFragment.OnPanelSelect();
break;
}
update();
}

@Override
public void onPageScrollStateChanged(int state) {
}
});

fab.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
stayAliveService();
switch (viewPager.getCurrentItem()) {
case 0: devicesFragment.OnFabClick(); break;
case 1: sharingFragment.OnFabClick(); break;
case 2: actionsFragment.OnFabClick(); break;
}
}
});
}

@Override
protected void onStart() {
super.onStart();
stayAliveService();
update();
handler.sendEmptyMessageDelayed(0, 1000);
}

@Override
```

```
protected void onStop() {
super.onStop();
handler.removeMessages(0);
}

void addToList(String path) {
try {
SharingService.Pipe.addUnit(UnitTypes.FILE, path);
} catch (FileNotFoundException e) {
Toast.makeText(this, "File " + new File(path).getName() + " already added to
list", Toast.LENGTH_SHORT).show();
}
}

private void onSharedIntent() {
Intent receivedIntent = getIntent();
String receivedAction = receivedIntent.getAction();
if (receivedAction != null) switch (receivedAction) {
case Intent.ACTION_SEND:
Uri uri = receivedIntent.getParcelableExtra(Intent.EXTRA_STREAM);
if (uri != null) addToList(getRealPathFromURI(uri));
break;
case Intent.ACTION_SEND_MULTIPLE:
ArrayList<Uri> uris =
receivedIntent.getParcelableArrayListExtra(Intent.EXTRA_STREAM);
if (uris != null) for (Uri uri2 : uris) addToList(getRealPathFromURI(uri2));
break;
case Intent.ACTION_MAIN:
break;
}
}

private void stayAliveService() {
if (!SharingService.isAlive())
startService(new Intent(MainActivity.this, SharingService.class));
SharingService.Pipe.setHandlerUI(handler);
}

@SuppressWarnings("ConstantConditions")
public String getClipboard() {
try {
ClipboardManager clipboard = (ClipboardManager)
getSystemService(CLIPBOARD_SERVICE);
return clipboard.getPrimaryClip().getItemAt(0).getText().toString();
} catch (NullPointerException ignore) { return ""; }
}

@SuppressLint("UseCompatLoadingForDrawables")
Drawable getPasteIcon() {
TypedArray a = getTheme().obtainStyledAttributes(R.style.AppTheme, new
int[]{R.attr.actionModePasteDrawable});
int attributeResourceId = a.getResourceId(0, 0);
return getResources().getDrawable(attributeResourceId);
}

public void showMessage(String msg, boolean fast) {
Snackbar.make(viewPager, msg,
fast?Snackbar.LENGTH_SHORT:Snackbar.LENGTH_LONG).show();
```

```
}

public void showMessage(String msg) {
showMessage(msg, false);
}

public boolean update() {
try{switch (viewPager.getCurrentItem()) {
case 0: return devicesFragment.update();
case 1: return sharingFragment.update();
case 2: return actionsFragment.update();
}}catch (NullPointerException e){return false;}
return true;
}

public String getRealPathFromURI(Uri contentUri) {
String path = contentUri.getPath();
Cursor cursor = this.getContentResolver().query(contentUri, null, null, null,
null);
if(cursor != null && path != null){
int idx;
if(path.startsWith("/external/image") || path.startsWith("/internal/image"))
idx = cursor.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
else if(path.startsWith("/external/video") ||
path.startsWith("/internal/video"))
idx = cursor.getColumnIndex(MediaStore.Video.VideoColumns.DATA);
else if(path.startsWith("/external/audio") ||
path.startsWith("/internal/audio"))
idx = cursor.getColumnIndex(MediaStore.Audio.AudioColumns.DATA);
else return contentUri.getPath();
if(cursor.moveToFirst()) {
String result = cursor.getString(idx);
cursor.close();
return result;
}
}
return path;
}

public void setDevicesFragment(DevicesFragment devicesFragment) {
this.devicesFragment = devicesFragment;
}

public void setSharingFragment(SharingFragment sharingFragment) {
this.sharingFragment = sharingFragment;
}

public void setActionsFragment(ActionsFragment actionsFragment) {
this.actionsFragment = actionsFragment;
}

public void setFabIcon(@DrawableRes int res){
fab.setImageResource(res);
}

public static void toast(String text){
Toast.makeText(inflater.getContext(), text, Toast.LENGTH_SHORT).show();
}
```

```
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[],
int[] grantResults) {
if (requestCode != 1 || grantResults.length <= 0 || grantResults[0] !=
PackageManager.PERMISSION_GRANTED)
Toast.makeText(MainActivity.this, "Permission denied to work your storage",
Toast.LENGTH_SHORT).show();
}
}
```

ДОДАТОК Б

Вихідний код додатку, файл SharingService.java

```
package com.example.main;

public class SharingService extends Service implements Runnable {
private static Storage storage;
private static Thread thread;
private static Set<Unit> unitSet = new LinkedHashSet<>();
private static Set<Device> deviceSet = new LinkedHashSet<>();
private static Set<Action> actionSet = new LinkedHashSet<>();

private Handler notificationHandler;
int notificationID = 0;

private static Pattern url =
Pattern.compile("^(http://www\\.|https://www\\.|http://|https://)?[a-z0-9]+([\\-
.][a-z0-9]+)*\\.[a-z]{2,5}(:[0-9]{1,5})?(/.*)?$");

public static Set<? extends CanBuildLine> getUnitSet(){ return new
LinkedHashSet<>(unitSet); }

public static Set<? extends CanBuildLine> getDeviceSet(){ return new
LinkedHashSet<>(deviceSet); }

public static Set<? extends CanBuildLine> getActionSet(){ return new
LinkedHashSet<>(actionSet); }

static boolean isAlive(){
return thread != null && thread.isAlive();
}

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

@Override
public void onStart(Intent intent, int startId) {
super.onStart(intent, startId);
try {
if(intent.hasExtra("confirm")) {
Device device = null;
InetAddress address =
InetAddress.getByAddress(intent.getByteArrayExtra("address"));
byte[] bPublicKey= intent.getByteArrayExtra("public");
PublicKey publicKey = Security.deserializePublicKey(bPublicKey);
String md5 = Security.md5(bPublicKey);
for(Device d: deviceSet) if(d.getMd5().equalsIgnoreCase(md5)) {device = d;
break;}
if(device == null) device = new Device("Unknown", md5, DeviceTypes.Unix);
device.setPublicKey(publicKey);
device.setInetAddress(address);
createInputAction(new SenderContainer(intent.getByteArrayExtra("container")),
device);
}
```

```
((NotificationManager)
getSystemService(NOTIFICATION_SERVICE)).cancel(intent.getIntExtra("id", 0));
} catch (Exception ignored) { }
}

@SuppressLint("HandlerLeak")
@Override
public void onCreate() {
super.onCreate();
storage = new Storage(this);
selfTest();
if (thread == null || !thread.isAlive()) {
thread = new Thread(this);
thread.start();
}
createNotificationChannel();

notificationHandler = new Handler(){
@Override
public void handleMessage(@NonNull Message msg) {
if (!(msg.obj instanceof SenderContainer)) return;
SenderContainer container = (SenderContainer) msg.obj;
Intent intent = new Intent(SharingService.this, SharingService.class);
intent.putExtra("container", container.toByte());
intent.putExtra("id", notificationID);
intent.putExtra("public",
Security.serializePublicKey(container.getPublicKey()));
intent.putExtra("address", container.getInetAddress().getAddress());
intent.putExtra("confirm", true);
PendingIntent acceptPendingIntent =
PendingIntent.getService(SharingService.this, 10, intent,
PendingIntent.FLAG_CANCEL_CURRENT);
Intent intent2 = new Intent(SharingService.this, SharingService.class);
intent.putExtra("id", notificationID);
PendingIntent denyPendingIntent = PendingIntent.getService(SharingService.this,
20, intent2, PendingIntent.FLAG_CANCEL_CURRENT);

String name = "Unknown";
String md5 =
Security.md5(Security.serializePublicKey(container.getPublicKey()));
for(Device device : deviceSet) if(device.getMd5().equalsIgnoreCase(md5)){
name = device.getName() + " " + device.getInetAddress();
break;
}
NotificationCompat.Builder builder = new
NotificationCompat.Builder(SharingService.this, "New files")
.setSmallIcon(R.mipmap.ic_launcher)
.setContentTitle(name)
.setContentText("Send messages/urls/files
"+container.getCountStrings()+"/"+container.getCountUrls()+"/"+container.getCoun
tFiles())
.setPriority(NotificationCompat.PRIORITY_HIGH)
//.addAction(android.R.drawable.ic_menu_delete, "Ban sender", banPendingIntent)
.addAction(android.R.drawable.ic_menu_delete, "Deny", denyPendingIntent)
.addAction(android.R.drawable.ic_input_add, "Accept", acceptPendingIntent)
.setAutoCancel(true);
NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
```

```
notificationManager.notify(notificationID++, builder.build());

}

};
}

private void createNotificationChannel() {
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
int importance = NotificationManager.IMPORTANCE_DEFAULT;
NotificationChannel channel = new NotificationChannel("New files", "New files",
importance);
NotificationManager notificationManager =
getSystemService(NotificationManager.class);
notificationManager.createNotificationChannel(channel);
}
}

@Override
public void onDestroy() {
super.onDestroy();
thread.interrupt();
}

@SuppressWarnings("InfiniteLoopStatement")
@Override
public void run() {
while (true) {
try {
listenUDP();
Thread.sleep(100);
} catch (Exception e) { e.printStackTrace();}
}
}

public static class Pipe {

private static android.os.Handler handlerUI;

public static void forceUpdate(){
handlerUI.sendEmptyMessage(-1);
}

public static void setHandlerUI(android.os.Handler handlerUI) {
Pipe.handlerUI = handlerUI;
}

public static Unit addUnit(UnitTypes type, String data) throws
FileNotFoundException {
Unit unit = new Unit(type, data);
if (unitSet.add(unit)) {
handlerUI.sendEmptyMessage(-1);
return unit;
}
else return null;
}

public static Unit addUnit(String data) {
```

```
try {
if(url.matcher(data).matches()) return addUnit(UnitTypes.URL, data);
else {
File f = new File(data);
if(f.exists() && !f.isDirectory()) return addUnit(UnitTypes.FILE, data);
else return addUnit(UnitTypes.TEXT, data);
}
} catch (FileNotFoundException e){return null;}
}

public static Device addDevice(String name, String md5, DeviceTypes type) {
Device tmp = new Device(name, md5, type);
if(!deviceSet.add(tmp)) for (Device one : deviceSet) if (one.equals(tmp)){
tmp = one.update(name, type);
break;
}
return tmp;
}

public static boolean remove(Object object) {
if (object instanceof Action) return actionSet.remove(object);
else if (object instanceof Device) return deviceSet.remove(object);
else if (object instanceof Unit) return unitSet.remove(object);
else return false;
}

public static void sendEmptyMessage(int i) {
handlerUI.sendEmptyMessage(i);
}
}

/**
* empty message - query devices - response: [(OS)L/W/A/M][char-NAME][md5(Public
key)]
* [P][md5(Public key)] query public key - response [F][Public key RSA 2048]
* [C][encrypted (md5(Public key) + port + AES 512 key)] - create tcp tunnel -
request to try connect to tcp socket
*/
public void listenUDP(){
try {
DatagramSocket socket = new DatagramSocket(26885);
byte[] buffer = new byte[65536];
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
global:
while (true) {
socket.receive(packet);
if(packet.getLength() == 0){
Network.sendUDPMessage(packet.getAddress(),26885, "A" +
storage.getSelfDeviceName() +
md5(serializePublicKey(storage.getSelfKeyPair().getPublic())));
continue;
}
String response = new String(buffer, 0, packet.getLength());
if(checkFlag(response, 'W', 'L', 'A', 'M')){
String md5 = response.substring(packet.getLength()-32);
if(md5.equals(Security.md5(Security.serializePublicKey(storage.getSelfKeyPair().
getPublic())))) continue;
for(Device device : deviceSet) if (device.getMd5().equals(md5)){
```

```
device.setOnline(true);
device.setInetAddress(packet.getAddress());
Pipe.forceUpdate();
continue global;
}

Device device = Pipe.addDevice(
response.substring(1, packet.getLength()-32),
md5,
DeviceTypes.convert(response.charAt(0))
);
device.setInetAddress(packet.getAddress());
Network.sendUDPMessage(packet.getAddress(), 26885, "P"+md5);

} else if(checkFlag(response, 'P')){

if(packet.getLength() < 33) return;
byte[] serializedPublicKey =
serializePublicKey(storage.getSelfKeyPair().getPublic());
if(md5(serializedPublicKey).equalsIgnoreCase(response.substring(1,33)))
Network.sendUDPMessage(packet.getAddress(),26885, addAll("F".getBytes(),
serializedPublicKey));

} else if(checkFlag(response, 'F')){

setDeviceNetworkData(clip(buffer, 1, packet.getLength()-1),
packet.getAddress());

} else if(checkFlag(response, 'C')){
System.out.println("C detect " + packet.getLength());
beginSession(clip(buffer,1, packet.getLength()-1), packet.getAddress());
}
}
} catch (IOException e) {
e.printStackTrace();
}
}

public static void beginSession(Action action) {

SenderContainer container = new SenderContainer()
.setSessionID(action.getSessionID())
.setTotalSize(action.totalSize())
.setSecretKey(action.getSecurityKey())
.setPort(action.getPort())
.setCountFiles(action.count(UnitTypes.FILE))
.setCountStrings(action.count(UnitTypes.TEXT))
.setCountUrls(action.count(UnitTypes.URL));
byte[] data = addAll(
Security.serializePublicKey(storage.getSelfKeyPair().getPublic()),
Security.encrypt(action.getDevice().getPublicKey(), container.toByte())
);
byte[] sign = Security.signatureData(storage.getSelfKeyPair().getPrivate(),
data);
byte[] udpdata = addAll("C".getBytes(), data, sign);
Network.sendUDPMessage(action.getDevice().getInetAddress(),26885, udpdata);
}
```

```
private void beginSession(byte[] data, InetAddress address) {
try {
PublicKey destPublicKey = Security.deserializePublicKey(clip(data, 0,
data.length-513));
byte[] validData = Security.checkSignature(destPublicKey, clip(data, 0,
data.length-1));
byte[] bContainer = clip(validData, validData.length-256, validData.length-1);
byte[] b = Security.decrypt(storage.getSelfKeyPair().getPrivate(), bContainer);

SenderContainer container = new SenderContainer(b);
container.setPublicKey(destPublicKey);
container.setInetAddress(address);
boolean isTrust = false;
for(Device device : deviceSet) if(device.getPublicKey().equals(destPublicKey) &&
device.isTrusted()) {
isTrust = true;
createInputAction(container, device);
break;
}
if(!isTrust) notificationHandler.sendMessage(Message.obtain(notificationHandler,
0, container));
} catch (Exception ignored){ignored.printStackTrace();}
}

private void createInputAction(SenderContainer container, Device device) {
actionSet.add(new Action(device, container));
Pipe.forceUpdate();
}

private void setDeviceNetworkData(byte[] publicKey, InetAddress address) throws
IOException {
String md5 = Security.md5(publicKey);
for(Device device : deviceSet)
if(device.getMd5().equalsIgnoreCase(md5)){
device.setPublicKey(Security.deserializePublicKey(publicKey));
device.setInetAddress(address);
device.setOnline(true);
Pipe.forceUpdate();
}
}

public static void setDevicesOnline(boolean isOnline) {
for(Device device: deviceSet) device.setOnline(isOnline);
}

private static boolean checkFlag(String str, char... flags){
for (char flag : flags) if (str.charAt(0) == flag) return true;
return false;
}

public static void trustDevice(Device device, boolean val){
device.setTrust(val);
}

public static void banDevice(Device device, boolean val){
device.setBan(val);
if(val) device.setTrust(false);
}
```

```
public static void createAction(Set<Device> devices){
for(Device device : devices) actionSet.add(new Action(unitSet, device));
unitSet = new LinkedHashSet<>();
Pipe.forceUpdate();
}

public static class SenderContainer {
private InetAddress inetAddress;
private PublicKey publicKey;
private long timestamp = System.currentTimeMillis() / 1000L;
private long sessionID;
private long totalSize;
private SecretKey secretKey;
private int port;
private int countFiles;
private int countStrings;
private int countUrls;

SenderContainer(){}

byte[] toByte(){
byte[] connectionData = null;
ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
DataOutputStream dataOutputStream = new DataOutputStream(byteArrayOutputStream);
try {
dataOutputStream.writeByte(100);
dataOutputStream.writeInt(countFiles);
dataOutputStream.writeInt(countStrings);
dataOutputStream.writeInt(countUrls);
dataOutputStream.writeLong(timestamp);
dataOutputStream.writeLong(sessionID);
dataOutputStream.writeLong(totalSize);
dataOutputStream.write(Security.serializeSecretKey(secretKey));
dataOutputStream.writeInt(port);
dataOutputStream.flush();
connectionData = byteArrayOutputStream.toByteArray();
dataOutputStream.close();
byteArrayOutputStream.close();
} catch (IOException ignored) {}
return connectionData;
}

SenderContainer(byte[] in){
try {
ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(in);
DataInputStream dataInputStream = new DataInputStream(byteArrayInputStream);
dataInputStream.readByte();
countFiles = dataInputStream.readInt();
countStrings = dataInputStream.readInt();
countUrls = dataInputStream.readInt();
timestamp = dataInputStream.readLong();
sessionID = dataInputStream.readLong();
totalSize = dataInputStream.readLong();
byte[] bSecretKey = new byte[32];
dataInputStream.read(bSecretKey);
secretKey = Security.deserializeSecretKey(bSecretKey);
port = dataInputStream.readInt();
```

```
dataInputStream.close();
byteArrayInputStream.close();
}catch (Exception ignored){}
}

public long getTimestamp() {
return timestamp;
}

public SenderContainer setTimestamp(long timestamp) {
this.timestamp = timestamp;
return this;
}

public long getSessionID() {
return sessionID;
}

public SenderContainer setSessionID(long sessionID) {
this.sessionID = sessionID;
return this;
}

public long getTotalSize() {
return totalSize;
}

public SenderContainer setTotalSize(long totalSize) {
this.totalSize = totalSize;
return this;
}

public SecretKey getSecretKey() {
return secretKey;
}

public SenderContainer setSecretKey(SecretKey secretKey) {
this.secretKey = secretKey;
return this;
}

public int getPort() {
return port;
}

public SenderContainer setPort(int port) {
this.port = port;
return this;
}

public int getCountFiles() {
return countFiles;
}

public SenderContainer setCountFiles(int countFiles) {
this.countFiles = countFiles;
return this;
```

```
}

public int getCountStrings() {
return countStrings;
}

public SenderContainer setCountStrings(int countStrings) {
this.countStrings = countStrings;
return this;
}

public int getCountUrls() {
return countUrls;
}

public SenderContainer setCountUrls(int countUrls) {
this.countUrls = countUrls;
return this;
}

public PublicKey getPublicKey() {
return publicKey;
}

public void setPublicKey(PublicKey publicKey) {
this.publicKey = publicKey;
}

public InetAddress getInetAddress() {
return inetAddress;
}

public void setInetAddress(InetAddress inetAddress) {
this.inetAddress = inetAddress;
}

public String toString(){
return "Receive{" +
"timestamp=" + timestamp +
", port=" + port +
", sessionID=" + sessionID +
", totalSize=" + totalSize +
", countFiles=" + countFiles +
", countStrings=" + countStrings +
", countUrls=" + countUrls +
'}';
}
}

private void selfTest(){
byte[] b = new byte[230];
for (int i = 0; i < b.length; i++) b[i] = (byte) new Random().nextInt();
byte[] e = Security.encrypt(storage.getSelfKeyPair().getPublic(), b);
byte[] b2 = Security.decrypt(storage.getSelfKeyPair().getPrivate(), e);
if(!Arrays.equals(b,b2)) throw new RuntimeException("BAD keypair!!!!!!!");
}
}
```

ДОДАТОК В

Вихідний код додатку, файл Network.java

```
package com.example.main;

public class Network {

public static boolean sendUDPMessage(InetAddress address, int port, @NotNull
String message){
try {
DatagramSocket socket = new DatagramSocket();
socket.setBroadcast(true);
byte[] sendData = message.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
address, port);
socket.send(sendPacket);
} catch (IOException e) { return false; }
return true;
}

public static boolean sendUDPMessage(InetAddress address, int port, @NotNull
byte[] message){
try {
DatagramSocket socket = new DatagramSocket();
socket.setBroadcast(true);
DatagramPacket sendPacket = new DatagramPacket(message, message.length, address,
port);
socket.send(sendPacket);
} catch (IOException e) { return false; }
return true;
}

public static Set<InetAddress> getBroadcastAddresses() {
Set<InetAddress> broadcastAddresses = new HashSet<>();
StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
StrictMode.setThreadPolicy(policy);
try {
for (Enumeration<NetworkInterface> interfaces =
NetworkInterface.getNetworkInterfaces(); interfaces.hasMoreElements();){
NetworkInterface networkInterface = interfaces.nextElement();
List<InterfaceAddress> interfaceAddresses =
networkInterface.getInterfaceAddresses();
for(InterfaceAddress address:interfaceAddresses)
if(address.getBroadcast() != null)
broadcastAddresses.add(address.getBroadcast());
}
} catch (SocketException ex) { ex.printStackTrace(); }
return broadcastAddresses;
}

public static void sendQueryOnlineDevices(){
SharingService.setDevicesOnline(false);
for (InetAddress address : getBroadcastAddresses())
Network.sendUDPMessage(address, 26885,"");
}
```

```
public static void sendSelfOnline(Storage storage){
for (InetAddress address : getBroadcastAddresses())
Network.sendUDPMessage(address, 26885, "A" + storage.getSelfDeviceName() +
Security.md5(Security.serializePublicKey(storage.getSelfKeyPair().getPublic())))
;
}

}
```

ДОДАТОК Г

Вихідний код додатку, файл Action.java

```
package com.example.main.domain;

public class Action implements CanBuildLine {
public enum Status {HASHING, WAIT, TRANSMITTING, DONE}
Status status = Status.WAIT;
long sessionID = new Random().nextLong();
Set<Unit> units;
Device device;
public float progress = 0;
long totalsize = -1;
int port;
SecretKey securityKey = Security.generateAES();
Map<Unit, NetworkContainer> container = new LinkedHashMap<>();
List<NetworkContainer> networkContainers;

public void newContainer(NetworkContainer networkContainer) {
Unit unit = new Unit(networkContainer);
units.add(unit);
this.container.put(unit, networkContainer);
}

public Action(Device device, SharingService.SenderContainer container) {
this.device = device;
port = container.getPort();
sessionID = container.getSessionID();
totalsize = container.getTotalSize();
securityKey = container.getSecretKey();
units = new LinkedHashSet<>();
int size =
container.getCountFiles()+container.getCountStrings()+container.getCountUrls();
networkContainers = new ArrayList<>(size);
for (int i = 0; i < size; i++) networkContainers.add(null);
new Receiver(device.getInetAddress(), port, networkContainers, this);
status = Status.TRANSMITTING;
}

public Action(Set<Unit> units, Device device) {
this.units = units;
this.device = device;
for(Unit unit: units) {
unit.isInput = true;
if (!unit.done) {
unit.subscribe(this);
status = Status.HASHING;
}
}
if(!status.equals(Status.HASHING)) md5Notify();
}

public SecretKey getSecurityKey() {
return securityKey;
}
```

```
public int getPort() {
return port;
}

public Status getStatus() {
return status;
}

public boolean isShowThis() {
return showThis;
}

public void setShowThis(boolean showThis) {
this.showThis = showThis;
}

boolean showThis = false;


@Override
public Line buildLine(ViewGroup vg, BaseFragment executor) {
Line line = new Line(vg, this);
updateLine(line, executor);
line.show();
return line;
}

@Override
public void updateLine(Line line, BaseFragment executor) {
line.setText(device.getName(),
device.getMd5()+"\r\n"+humanReadableByteCount(totalsize))
.setImage(TypesUtil.getTypeRes(device.getType(), device.isTrusted()))
.setExecutor(executor)
.selectable(true)
.setOnline(true)
.update();
}

public synchronized void md5Notify() {
for (Unit unit : units) if(!unit.done) return;
status = Status.TRANSMITTING;

Sender sender = new Sender(device.getInetAddress(), buildContainer());
sender.beginTCP(this);
}

List<NetworkContainer> buildContainer(){
List<NetworkContainer> networkContainers = new ArrayList<>();
for (Unit unit: units) {
switch (unit.type){
case TEXT: networkContainers.add(new NetworkContainerMessage(unit.data)); break;
case FILE: networkContainers.add(new NetworkContainerFile(unit.data, unit.md5));
break;
case URL:  networkContainers.add(new NetworkContainerURL(unit.data)); break;
}
container.put(unit, networkContainers.get(networkContainers.size()-1));
}
return networkContainers;
```

```
}

public Set<? extends CanBuildLine> getUnits() {
return units;
}

public Device getDevice() {
return device;
}

public long totalSize(){
if(totalsize == -1) {
totalsize = 0;
for (Unit unit: units) totalsize += unit.size;
}
return totalsize;
}

public int count(UnitTypes unitType){
int count = 0;
for (Unit unit: units) if(unit.type.equals(unitType)) count++;
return count;
}
public void serverRun(int port){
System.out.println("run server");
this.port = port;
SharingService.beginSession(this);
}

public void serverStop(boolean fail){
if (fail) status = Status.WAIT;
else status = Status.DONE;
}

public long getSessionID() {
return sessionID;
}

public void updateProgress(){
double progress = 0;
long totalsize = 0;
for(Map.Entry<Unit, NetworkContainer> entry : container.entrySet()){
float unitProgress = entry.getValue().getProgress();
progress += unitProgress*entry.getValue().getSize();
totalsize += entry.getValue().getSize()/100;
if(showThis) {
if(entry.getKey().status.equals(Status.DONE)) continue;
if(entry.getValue().getStatus().equals(com.example.main.pipe.Status.DELIVERED)){
entry.getKey().setStatus(Status.DONE); continue;}
if(isError(entry.getValue().getStatus())) entry.getKey().progress = -1;
else entry.getKey().progress = unitProgress*100;
}
}
if(totalsize == 0) return;
this.progress = (float) (progress / totalsize);
SharingService.Pipe.forceUpdate();
}
}
```

ДОДАТОК Д

Вихідний код додатку, файл Device.java

```
package com.example.main.domain;

public class Device implements CanBuildLine {
private String name;
private String md5;
private DeviceTypes type;
private boolean isOnline;
private boolean isTrust;
private boolean isBan;
private boolean lostOnline = false;
private PublicKey publicKey;
private InetAddress inetAddress;

public PublicKey getPublicKey() {
return publicKey;
}

public InetAddress getInetAddress() {
return inetAddress;
}

public Device(String name, String md5, DeviceTypes type) {
this.name = name;
this.md5 = md5;
this.type = type;
}

public void setPublicKey(PublicKey publicKey) {
this.publicKey = publicKey;
}

public void setInetAddress(InetAddress inetAddress) {
this.inetAddress = inetAddress;
}

@Override
public Line buildLine(ViewGroup vg, BaseFragment executor) {
Line line = new Line(vg, this);
updateLine(line, executor);
line.show();
return line;
}

@Override
public void updateLine(Line line, BaseFragment executor) {
line.setText(name, md5)
.setImage(TypesUtil.getTypeRes(type, isTrusted()))
.setExecutor(executor)
.selectable(true)
.setOnline(true);
}

@Override
```

```
public int hashCode() {
return md5.hashCode();
}

@Override
public boolean equals(@Nullable Object obj) {
if(!(obj instanceof Device)) return false;
return md5.equals(((Device) obj).md5);
}

public void setOnline(boolean online) {
lostOnline = !online && !isOnline;
isOnline = online;
}

public Device update(String name, DeviceTypes type) {
this.name = name;
this.type = type;
return this;
}

public String getName() {
return name;
}

public boolean isTrusted() {
return isTrust;
}

public void setTrust(boolean trust) {
isTrust = trust;
}

public void setBan(boolean ban) {
isBan = ban;
}

public boolean isBaned() {
return isBan;
}

public boolean isLostOnline() {
return lostOnline;
}

public DeviceTypes getType() {
return type;
}

public String getMd5() {
return md5;
}
}
```

ДОДАТОК Е

Вихідний код додатку, файл Unit.java

```
package com.example.main.domain;

public class Unit implements CanBuildLine {
UnitTypes type;
String data;
String md5;
boolean done;
List<Action> listeners = new LinkedList<>();
long size;
boolean isInput;
NetworkContainer container;
Action.Status status = Action.Status.WAIT;
float progress;
public Unit(NetworkContainer container){
this.container = container;
switch (container.getType()){
case TEXT: type = UnitTypes.TEXT;
isInput = true;
md5 = container.getMd5();
size = container.getSize();
data = container.toString();
status = Action.Status.DONE;
return;
case URL: type = UnitTypes.URL;
isInput = true;
md5 = container.getMd5();
size = container.getSize();
data = container.toString();
status = Action.Status.DONE;
return;
case FILE: type = UnitTypes.FILE;
data = ((NetworkContainerFile) container).getFilename();
break;
}
isInput = true;
md5 = container.getMd5();
size = container.getSize();
status = Action.Status.TRANSMITTING;
}

public Unit(UnitTypes type, String data) throws FileNotFoundException {
if (type == UnitTypes.FILE){
File f = new File(data);
if(!f.exists() || f.isDirectory()) throw new FileNotFoundException();
size = f.length();
new AsyncFileHasher(f){
public void callback(String md5) {
Unit.this.md5 = md5;
if(md5 == null) {
SharingService.Pipe.remove(Unit.this);
SharingService.Pipe.sendEmptyMessage(-10);
}
else SharingService.Pipe.forceUpdate();
```

```
done = true;
for (Action action : listeners) action.md5Notify();
}
};
} else {done = true; size = data.length();}
this.type = type;
this.data = data;
}

public Line buildLine(ViewGroup vg, BaseFragment executor){
Line line = new Line(vg, this);
updateLine(line, executor);
line.show();
return line;
}

public void updateLine(Line line, BaseFragment executor) {
String title;
if(type == UnitTypes.FILE) title = new File(data).getName();
else if(type == UnitTypes.URL) title = "URL";
else title = "Plain text | Clipboard";
if(isInput)               line.setText(title,               (type               ==
UnitTypes.FILE?humanReadableByteCount(size)+"\r\n":"")+data)
.setImage(TypesUtil.getTypeRes(type))
.setExecutor(executor)
.selectable(true)
.setVisibleProgressBar(true)
.setProgress(progress);
else                  line.setText(title,                  (type                  ==
UnitTypes.FILE?humanReadableByteCount(size)+"\r\n":"")+data).setExecutor(executo
r).selectable(true);
}

@Override
public int hashCode() {
return data.hashCode();
}

@Override
public boolean equals(@Nullable Object obj) {
if(!(obj instanceof Unit)) return false;
return data.equals(((Unit) obj).data);
}

public void subscribe(Action action){
listeners.add(action);
}
public void setStatus(Action.Status status) {
this.status = status;
}
public String getData() {
return data;
}
public UnitTypes getType() {
return type;
}
}
```

ДОДАТОК Ж

Вихідний код додатку, файл TypesUtil.java

```
package com.example.main.domain.types;

import com.example.main.R;

public class TypesUtil {

public static int getTypeRes(UnitTypes type) {
switch (type) {
case URL:  return R.drawable.ic_link;
case FILE: return R.drawable.ic_file;
case TEXT: return R.drawable.ic_text;
default: return -1;
}
}

public static int getTypeRes(DeviceTypes type, boolean isTrusted){
if(isTrusted) switch (type){
case Android: return R.drawable.ic_android_trusted;
case MAC:     return R.drawable.ic_mac_trusted;
case Win:     return R.drawable.ic_windows_trusted;
case Unix:    return R.drawable.ic_linux_trusted;
}
switch (type) {
case Android: return R.drawable.ic_android_new;
case Win:     return R.drawable.ic_windows_new;
case MAC:     return R.drawable.ic_mac_new;
case Unix:    return R.drawable.ic_linux_new;
default:      return -1;
}
}
}
```

ДОДАТОК И

Вихідний код додатку, файл Utils.java

```
package com.example.main;

public class Utils {
public static byte[] addAll(byte[]... array) {
int size = 0;
for (byte[] b : array) size += b.length;
byte[] joined = new byte[size];
size = 0;
for (byte[] b : array) System.arraycopy(b, 0, joined, (size += b.length) -
b.length, b.length);
return joined;
}

public static byte[] int2byte(int val){
return new byte[]{(byte) ((val >> 8) & 0xFF), (byte) (val & 0xFF)};
}

public static int byte2int(byte... val){
return val[0] << 8 | val[1] ;
}

public static int randRange(int min, int max){
return (int) (Math.random() * (max - min + 1) + min);
}

public static <T> T[] clip(T[] arr, int start, int end){
T[] clip =
(T[])java.lang.reflect.Array.newInstance(arr.getClass().getComponentType(), end-
start+1);
System.arraycopy(arr, start, clip,0, end-start+1);
return clip;
}

public static byte[] clip(byte[] arr, int start, int end){
byte[] clip = new byte[end-start+1];
System.arraycopy(arr, start, clip,0, end-start+1);
return clip;
}

public static int[] clip(int[] arr, int start, int end){
int[] clip = new int[end-start+1];
System.arraycopy(arr, start, clip,0, end-start+1);
return clip;
}

public static boolean getBit(byte b, int position) {
return ((b >> position) & 1) == 1;
}

public static String humanReadableByteCount(long bytes) {
int unit = 1024;
if (bytes < unit) return bytes + " B";
int exp = (int) (Math.log(bytes) / Math.log(unit));
```

```
String pre = ("KMGTPE").charAt(exp-1) + "";
return String.format("%.2f %sB", bytes / Math.pow(unit, exp), pre);
}

public static byte[] longToBytes(long x) {
ByteBuffer buffer = ByteBuffer.allocate(8);
buffer.putLong(x);
return buffer.array();
}

public static long bytesToLong(byte[] bytes) {
ByteBuffer buffer = ByteBuffer.allocate(8);
buffer.put(bytes);
buffer.flip();
return buffer.getLong();
}

public static String md5toHexString(byte[] bytes) {
StringBuilder result = new StringBuilder();
for (byte aByte : bytes)
result.append(Integer.toString((aByte & 0xff) + 0x100, 16).substring(1));
return result.toString();
}

public static byte[] md5toByteArray(String md5){
byte[] result = new byte[16];
for (int i = 0; i < 16; i++) result[i] = (byte) Integer.parseInt(md5.substring(i*2,
i*2+2), 16);
return result;
}

public static byte[] combine(byte[]... arrays){
int size = 0;
for (byte[] array : arrays) size += array.length;
byte[] result = new byte[size];
size = 0;
for (byte[] array: arrays)
for (int i = 0; i < array.length; i++, size++)
result[size] = array[i];
return result;
}
public static String md5(String str){
try {MessageDigest messageDigest = MessageDigest.getInstance("MD5");
return     md5toHexString(messageDigest.digest(str.getBytes(Charset.forName("UTF-
8"))));
} catch (NoSuchAlgorithmException e) {
e.printStackTrace();
return null;
}
}
static  void  checkReadBuffer(byte[]  buffer,  int  minSize,  int  id)  throws
SmallBufferException {
if(buffer  ==  null  ||  buffer.length  <  minSize  +  2)  throw  new
SmallBufferException(minSize+2);
buffer[0] = (byte) (id>>>8);
buffer[1] = (byte) (id);
}
}
```