Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Natívna Android aplikácia na bezpečnú správu súborov

Diplomová práca

2021 **Bc. Michal Olenčin**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Natívna Android aplikácia na bezpečnú správu súborov

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Ing. Ján Perháč, PhD.
Konzultant:	

Košice 2021 **Bc. Michal Olenčin**

Abstrakt v SJ

Práca je venovaná analýze, návrhu a implementácii natívnej Android aplikácie bezpečného správcu súborov. Dôraz bol kladený najmä na dosiahnutie bezpečnosti výslednej aplikácie, uchovanie súkromia používateľa, dosiahnutie ľahkej použiteľnosti, rozšíriteľnosti a údržby pomocou otvoreného zdrojového kódu. Opísané sú taktiež existujúce riešenia a aktuálna kybernetická bezpečnosť a súkromie. Zdrojový kód je zverejnený na platforme Gitlab. Výsledná aplikácia je publikovaná na platforme Google Play a F-Droid. K aplikácii je vytvorený audit závislosti, audit súkromia, používateľské testovanie a navrhnuté odporúčané rozšírenia aplikácie.

Kľúčové slová v SJ

Android, aplikácia, bezpečnosť, súkromie, Kotlin, použiteľnosť

Abstrakt v AJ

The thesis is dedicated to the analysis, design, and implementation of a native Android application, in particular the secure file management. In this work, the focus is mainly on the resulting application to be secure, to preserve the private user, to be easy to use, and at the same time to be easily extensible and maintained using open source code. Existing solutions and current cybersecurity and privacy are also discussed. The source code is published on the Gitlab platform. The application is published on the Google Play and F-Droid platforms. A dependency audit, privacy audit, user testing, and suggested recommended application extensions are discussed in the results.

Kľúčové slová v AJ

Android, application, security, privacy, Kotlin, usability

Bibliografická citácia

OLENČIN, Michal. *Natívna Android aplikácia na bezpečnú správu súborov*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2021. 69s. Vedúci práce: Ing. Ján Perháč, PhD.

60805

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Natívna Android aplikácia na bezpečnú správu súborov

Native Android application for secure file management

Študent: **Bc. Michal Olenčin**
Školiteľ: **Ing. Ján Perháč, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce:
Pracovisko konzultanta:

Pokyny na vypracovanie diplomovej práce:

1. Analyzovať zámer a rozsah diplomovej práce v synergii s analýzou existujúcich riešení so zameraním na bezpečnosť správy súborov operačného systému Android.
2. Na základe vykonanej analýzy navrhnúť vhodnú funkcionalitu a zabezpečenie výslednej aplikácie.
3. Implementovať navrhnutú aplikáciu so zameraním na dodržanie pilierov použiteľnosti, súkromia a bezpečnosti.
4. Vyhodnotiť výslednú aplikáciu na základe vhodne zvolených metrík.
5. Vypracovať záverečnú prácu v prostredí typografického systému LaTeX, podľa pokynov vedúceho záverečnej práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 23.04.2021
Dátum zadania diplomovej práce: 30.10.2020

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -12-

.....
prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 13.5.2021 .....

Vlastnoručný podpis

Poďakovanie

Chcel by som sa poďakovať predovšetkým svojmu školiteľovi, Ing. Jánovi Perháčovi, PhD., za jeho cenné rady a odborné vedenie v rámci procesu tvorby mojej záverečnej práce. Rovnako by som sa rád poďakoval svojím rodičom, partnerke a priateľom za ich podporu a motiváciu počas celého môjho štúdia.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam zdrojových kódov

Motivácia

Mobilné zariadenia sa stali našimi každodennými pomocníkmi, ktoré nám zjednodušujú život, naše každodenné bytie. Okrem toho sa stali jedným z najväčších nositeľov nášho súkromia. V svojich zariadeniach máme uložené citlivé informácie ako sú napríklad fotky, videá, dokumenty, história prehliadania, samotné dáta v nainštalovaných aplikáciách a mnoho iného. Všetky tieto informácie môžu byť ukradnuté, zmazané alebo zneužité. Zraniteľnosť týchto informácií závisí od užívateľa i od toho, do akej miery si chráni svoje údaje.

Rovnako ako informačné technológie, tak aj mobilné zariadenia sú ľahko zneužiteľné. Neustále sa vyskytujú nové bezpečnostné incidenty, ktoré majú za následok únik informácií alebo iné neželané následky. Okrem bezpečnostných incidentov sú naše mobilné zariadenia terčom špehovania či už organizáciami, alebo samotnými štátmi. Väčšina ľudí si tieto hrozby neuvedomuje, nie je o nich informovaná, alebo im neprikladá veľkú pozornosť a prioritu, prípadne má chybný argument *„nemám čo skrývať“*[1].

Z týchto dôvodov je potrebné ľudí vzdelávať a informovať v oblasti kybernetickej bezpečnosti a ochrany súkromia. Okrem toho je potrebné vytvárať nástroje, ktoré pomôžu ľuďom chrániť ich údaje. Motiváciou práce je vytvoriť jeden z takýchto nástrojov, a to konkrétne mobilnú aplikáciu - bezpečný správca súborov.

Bezpečný správca súborov je postavený na troch pilieroch, a to konkrétne pilieroch použiteľnosti, bezpečnosti a súkromia. Pilier použiteľnosti umožní, aby aplikáciu vedel ovládať aj užívateľ s nižšou počítačovou gramotnosťou, a zároveň, aby sa aplikácia ovládala pohodlne. Pilier bezpečnosti zabezpečí ochranu užívateľa pred potencionálnymi útokmi rôzneho druhu a pilier súkromia zabezpečí, aby práca s dátami uchovala súkromie užívateľa.

✍ *V práci sú použité cudzie slová, ktorých význam je uvedený v slovníku.*

---

1 Rozbor riešenia

V rámci analýzy je najskôr zhrnutá aktuálna kybernetická bezpečnosť a súkromie. Následne sú zanalyzované jednotlivé funkcie aplikácie, ku ktorým bude v ďalších kapitolách navrhnutý dizajn a popísaná implementácia. Na konci kapitoly sú zanalyzované existujúce riešenia a zhrnutá samotná analýza.

1.1 Aktuálna kybernetická bezpečnosť a súkromie

Bezpečnosť a ochrana súkromia sú navzájom prepojené, a preto je potrebné ich zohľadňovať pri téme **kybernetickej bezpečnosti** alebo pri téme **súkromia**. Preto táto podkapitola zahrňuje obe témy zároveň.

1.1.1 Súkromie

Keď sa človek dostane do kontaktu s informačnými technológiami, respektíve s počítačom akéhokoľvek druhu, je vystavený chtiac-nechtiac odhaleniu svojho súkromia. Ako napríklad, ak človek vlastní mobilný telefón, neustále je možné **sledovať** jeho

- **polohu** pomocou WWAN, GPS, Wi-Fi alebo Bluetooth modulu v zariadení prostredníctvom triangulácie a podobne,
- **záujmy**, **správanie** a **osobné informácie** pomocou analytických nástrojov v aplikáciách, operačných systémov a webových stránok,
- **identitu** pomocou telefónneho čísla, IMEI čísla, MAC adresy, IP adresy a podobne,
- a mnoho ďalšieho.

Veľké organizácie ako napríklad Google, Facebook a Amazon zhromažďujú informácie a používajú ich na cielené reklamy alebo predaj. Bližšie informácie, ako veľké organizácie zbierajú a zneužívajú informácie užívateľoch a aké hrozby

to so sebou prináša, sú popísané v knihe *„The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power“* [2].

Informácie nie sú zbierané len organizáciami, ale taktiež samotnými vládami. Bližšie informácie ako vlády zbierajú informácie a špehujú ľudí je popísané v knihe *„Permanent Record“* [3], ktorú napísal známy whistleblower Edward Snowden. Autor vo svojom životopisnom diele popisuje, ako americké organizácie špehujú ľudí nielen v USA.

Väčšina ľudí nesprávne chápe pojem súkromie a majú chybný argument *„ja nemám čo skrývať“*, ako popisuje článok *„'I've Got Nothing to Hide' and Other Misunderstandings of Privacy“* [1], a tým pádom pravdepodobne nevedome odhaľujú svoje súkromie. V dnešnom svete moderných technológií je veľmi ťažké ochrániť svoje súkromie. Jednou z možných techník zachovania súkromia je mať vo svojom mobilnom telefóne nainštalovaného bezpečného správcu súborov, čomu je venovaná táto práca.

1.1.2 Kybernetická bezpečnosť

Ak chceme uchovať svoje súkromie, je potrebné dodržiavať základné aspekty kybernetickej bezpečnosti ako je dôvernosť, integrita, dostupnosť, utajenosť, autentickosť a overiteľnosť.

V článku *„Data Security on Mobile Devices: Current State of theArt, Open Problems, and Proposed Solutions“* [4] je rozoberaná aktuálna bezpečnosť mobilných operačných systémov iOS a Android. V článku sú diskutované možné techniky odcudzenia dát a aktuálne dostupné forenzné nástroje pre reverzné inžinierstvo. Taktiež je navrhnutých niekoľko odporúčaných vylepšení pre obe operačné systémy. K operačnému systému Android sa viažu tieto vylepšenia:

- Zašifrovať údaje používateľa pri zamknutí obrazovky.
- Implementovať end-to-end šifrovanie pre natívnu aplikáciu komunikácie.
- Implementovať end-to-end pre Google aplikácie.
- Zvýšiť bezpečnosť nízkoúrovňového programovacieho rozhrania firmvéru.
- Rozšíriť využitie bezpečného hardvéru.
- Zlepšiť spôsob aktualizácie.
- Využiť vzájomný prospech zo zdieľaného kódu.
- Využiť povahu operačného systému Android.

V závere článku je zhodnotená potreba spolupráce výskumníkov, inžinierov a politikov pre vytvorenie noriem a bezpečnostných protokolov s cieľom zlepšiť bezpečnosť a ochranu súkromia v mobilných zariadeniach.

Podľa správy hrozieb na mobilne zariadenia od firmy McAfee z roku 2019 [5] v druhej polovici roku 2019 bol detekovaný zvýšený **nárast škodlivého softvéru**, a to konkrétne:

- **falošných aplikácii**,
- aplikácii so škodlivým softvérom typu **cryptojacking**,
- aplikácii so škodlivým softvérom typu **backdoor**,
- a aplikácii so škodlivým softvérom typu **banking trojan**.

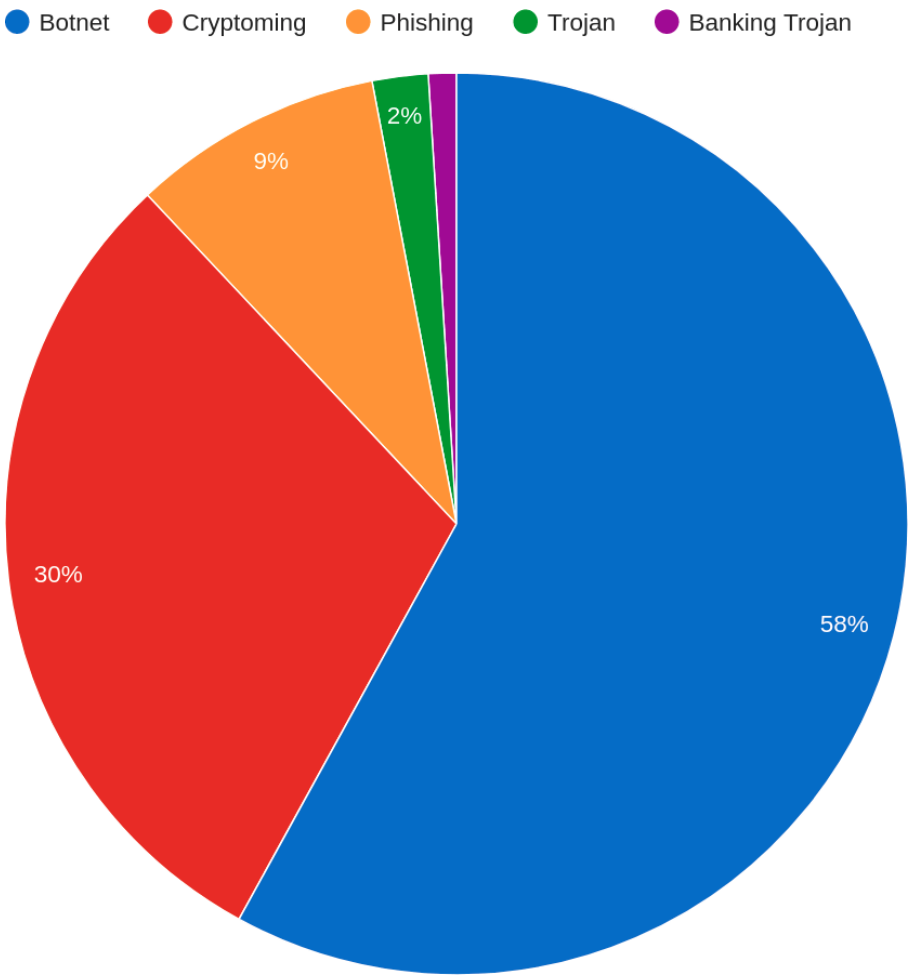
Motiváciou väčšiny útočníkov spomenutých v správe k vykonaniu kybernetického útoku bol práve zárobok.

Zhrnutie správy bezpečnosti od firmy Check Point z roku 2020 [6] je následovný:

- Jedným z najbežnejších zdrojov kompromitovania mobilných zariadení sú mobilné aplikácie.
- V prvej polovici roku 2019 došlo k **50% nárastu útokov na mobilné aplikácie bankovníctva** v porovnaní s rokom 2018.
- V roku 2019 viaceré typy škodlivého softvéru prešli do mobilného prostredia a zároveň bolo **nájdených viacej zraniteľností v mobilných zariadeniach, mobilných aplikáciách a mobilných operačných systémoch**.
- V celosvetovom meradle bolo **27% organizácii ovplyvnených** kybernetickými útokmi, v ktorých boli zahrnuté mobilné zariadenia.
- Medzi najpočetnejší škodlivý softvér zameraný na mobilné zariadenia patrí *„Hiddad“*, *„xHelper“*, *„Necro“*, *„AndroidBauts“* a *„Guerilla“*.

Podľa správy bezpečnosti od spoločnosti Cisco z roku 2019 [7] je možné pozorovať na obrázku 1.1 na strane 5 výskyt škodlivého softvéru v mobilných zariadeniach v časovom horizonte od júla 2018 do februára 2019.

Podľa zhrnutia aktuálnej kybernetickej bezpečnosti v článku *„Automated configuration of a Linux web server security“* [8] publikovaného na základe mojich výsledkov bakalárskej práce, sa v roku 2018 výskyt škodlivého softvéru typu



Obr. 1.1: Výskyt škodlivého softvéru v mobilných zariadeniach

- **cryptojacking** zvýšil o 8 500%,
- **útoku** na IoT zariadenia **zvýšil** o 600%,
- **ransomware zvýšil** o 46%,
- **útoku** na android zariadenia **zvyšil** o 54%.

Podľa spomenutých správ je možné jednoznačne dedukovať potrebnú zvýšenú pozornosť v oblasti kybernetickej bezpečnosti na mobilných zariadeniach. Vzhľadom na komplexnosť mobilných zariadení a všeobecne informačných technológií je veľmi ťažké udržať s kybernetickú bezpečnosť v mobilných zariadeniach, ktoré nám v mnohých aspektoch pomáhajú v každodennom živote.

1.2 Jadro aplikácie

Vzhľadom na to, že vývoj samotného správcu súborov je časovo náročný, rozhodol som sa moju prácu implementovať na základe existujúcej aplikácie a následne

do nej pridať jednotlivé analyzované funkcie. Spomedzi existujúcich riešení som vybral aplikáciu *„Simple File Manager“*[1]. Riešenie ako jediné spĺňalo kritériá

- otvoreného zdrojového kódu,
- licenciu umožňujúcu vytvoriť nové oddelené riešenie na základe existujúceho riešenia,
- malý počet závislostí,
- ľahko rozšíriteľný, malo komplexný a čitateľný zdrojový kód,
- riešenie s ohľadom na ochranu súkromia používateľa,
- užívateľský použiteľné rozhranie.

Riešenie bolo v čase písania práce postavené na programovacom jazyku **Kotlin** a podpore najnižšej verzie Android 5 s cieľovou verziou Android 10. Z dôvodu použitých knižníc, ktoré sú popísane v kapitole 2 má moje výsledné riešenie podporu najnižšej verzie Android 8.

Existujúce riešenie bolo v čase písania práce bolo súčasťou licencie *„GNU General Public License v3.0“* [9]. Pre splnenie licenčných podmienok je potrebné

- vyhlásenie o licencii a autorských právach,
- uviesť zmeny,
- zverejniť zdroj,
- použiť rovnakú licenciu.

Všetky licenčné podmienky boli vo výslednom riešení splnené. V kapitole 4 je popísaná publikácia výslednej aplikácie na platforme F-Droid a Google Play. Pred zverejnením som požiadal autora aplikácie *„Simple File Manager“* o povolenie publikovania aplikácie na spomenutých platformách. Autor súhlasil pod podmienkou, že aplikácia nebude obsahovať reklamy a bude uvedená pôvodná aplikácia. Vo výslednom riešení som dané podmienky splnil.

[1] `https://github.com/SimpleMobileTools/Simple-File-Manager`
dostupné online: 26. november 2020

1.3 Rozsah funkcií

Pred návrhom aplikácie bolo potrebné vytýčiť si funkcie, ktoré budú implementované. Všetky tieto funkcie je potrené zamerať na čo najväčšiu ochranu súkromia, bezpečia súborov užívateľa a najlepší užívateľský zážitok.

1.3.1 Skrývanie súborov

Ak sa súbory na zariadení nachádzajú na externom úložisku, všetky aplikácie, ktoré majú povolenie na čítanie alebo zápis externého úložiska, majú prístup k týmto súborom[2]. Externé úložisko je súborový systém, ktorý obsahuje dokumenty, fotky, videá, zvukové stopy, stiahnuté súbory a ostatné súbory pre všetky legitímne aplikácie. Pre čiastočné splnenie aspektu utajenosti v kybernetickej bezpečnosti je potrebné, aby správca súborov vedel uchovať súbory užívateľa, a to formou skrývania. V zmysle operačného systému je potrebné, aby súbory bolo možné uložiť do úložiska, ktoré bude dostupné iba pre danú aplikáciu.

1.3.2 Šifrovanie súborov

Od verzie Androidu 7 po verziu 9 je v základe podporované šifrovanie celej pamäte[3]. Od verzie Androidu 7 je v základe podporované šifrovanie na základe súborov. Od verzie Android 10 je šifrovanie na základe súborov povinné[4]. Aj napriek natívnej podpore šifrovania v Android zariadeniach je potrebné pre zvýšenú úroveň ochrany, aby správca súborov vedel súbory užívateľa šifrovať, a to užívateľsky použiteľne. Preto je potrené zvoliť silný šifrovací algoritmus, pre ktorý bude potrebné vynaložiť veľa úsilia, zdrojov a času na prelomenie algoritmu alebo nájdenie kľúča. Taktiež je potrebné, aby šifrovanie súborov bolo pre užívateľa jednoduché a zvládol ho aj užívateľ s nízkou počítačovou gramotnosťou. Šifrovanie zabezpečí úplné splnenie aspektu utajenosti v kybernetickej bezpečnosti.

1.3.3 Overenie prístupu

Keďže aplikácia bude uchovávať skryté súbory, ku ktorým bude mať prístup iba samotná aplikácia, je potrebné overiť prístup k aplikácii overením identity užíva-

[2] `https://developer.android.com/training/data-storage/app-specific`
dostupné online: 26. november 2020

[3] `https://source.android.com/security/encryption/full-disk`
dostupné online: 26. november 2020

[4] `https://source.android.com/security/encryption/file-based`
dostupné online: 26. november 2020

teľa. Overenie uchová skryté súbory v utajení pred nežiaducim prístupom.

Overenie je možné uskutočniť buď heslom, biometrickým overením ako je napríklad odtlačok prsta, alebo inou technikou overenia. Biometrické overenie je užívateľsky viacej prívetivé ako overenie heslom, avšak bezpečnosť biometrického overenia je na na oveľa nižšej úrovni ako pri overení heslom [10]. Preto je potrebné nechať užívateľa, aby si vybral nižšiu formu utajenia ako je biometrické overenie, alebo vyššiu formu utajenia ako je overenie heslom, prípadne ich kombinácia na základe jeho modelu hrozieb.

Medzi ďalšie formy overenia patrí napríklad časovo založený algoritmus jednorazového hesla alebo hardvérové autentifikačné zariadenie. Spomenuté ďalšie formy overenia sú však užívateľsky málo prívetivé.

1.3.4 Ostatné funkcie

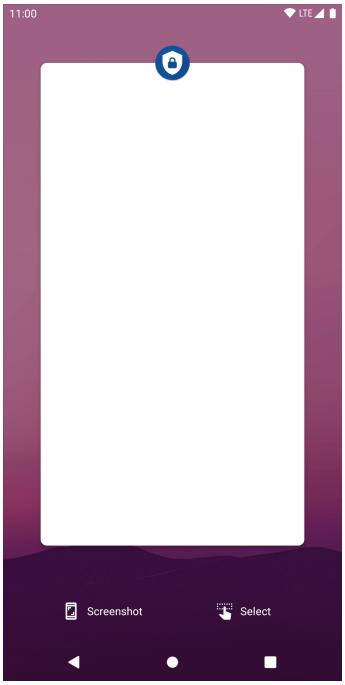
Spomenuté boli základné bezpečnostné prvky, ktoré by mala obsahovať aplikácia na zabezpečenie súborov. Ďalej sú popísané ostatné bezpečnostné prvky, respektíve funkcie, ktoré poskytnú dodatočnú ochranu súkromia a bezpečnosť aplikácie.

Integrita súborov

Na overenie aspektu integrity je vhodné, aby aplikácia vedela overiť integritu súborov formou kontrolného súčtu, prípadne viacerých kontrolných súčtov použitím rozdelených kryptografických transformačných funkcií. Kontrolný súčet umožní užívateľovi overiť, čí je zvolený súbor v nezmenenej forme, alebo či je súbor neporušený.

Zakázanie snímky obrazovky

Moderné mobilné operačné systémy ako je Android poskytujú možnosť vytvorenia snímky obrazovky zariadenia. Pre uchovanie utajenia informácií, ktoré sú zobrazované na obrazovke zariadenia, je potrebné systémovo zakázať zaznamenávať obraz aplikácie. Medzi obraz aplikácie patrí taktiež zakázanie náhľadu posledného stavu aplikácie v zozname nedávnych aplikácií. Zákaz zaznamenávania obrazu aplikácie má vplyv na aspekt utajenosti súborov užívateľa. Ukážku zákazu náhľadu aplikácie v zozname nedávnych aplikácií je možné vidieť na obrázku 1.2 na strane 9.



Obr. 1.2: Ukážka zákazu náhľadu aplikácie v zozname nedávnych aplikácií

Podpora kompresie a dekompresie šifrovaných a nešifrovaných Zip súborov

Medzi veľmi populárne formáty archívneho súboru patrí formát Zip. Tento formát poskytuje šifrovanú kompresiu súborov ako je popísané v špecifikácii formátu[5]. Pre vytváranie záloh, prípadne iný druh manipulácie s archivačnými súbormi, je vhodné umožniť vytvorenie šifrovanej alebo nešifrovanej kompresie Zip súboru rovnako ako jeho dekompresiu v navrhnutej aplikácii. Pridanie podpory kompresie a dekompresie šifrovaných Zip súborov má vplyv na aspekt utajenosti súborov užívateľa.

Určenie miesta uloženia mediálnych súborov

V rámci aplikácie by bolo vhodné umožniť užívateľovi špecifikovať miesto uloženia mediálnych súborov ako je fotka a video.

Mobilné zariadenie môže mať nastavené automatické zálohovanie mediálnych súborov po ich vytvorení, čo nemusí byť žiaduce správanie pri vytváraní mediálnych súborov ako je napríklad fotka občianskeho preukazu, úradné dokumenty, prípadne obyčajná fotka s tvárou užívateľa. Pri zvolení iného miesta uloženia ako je nastavená synchronizácia zálohovania sa zabráni nežiaducej synchronizácii mediálneho súboru na úložisko zálohy.

[5] `https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT`
dostupné online: 3. október 2020

Taktiež pri funkcii zvolenia miesta uloženia si bude môcť užívateľ uložiť vytvorený mediálny súbor priamo do skrytého úložiska, čím sa zachová aspekt utajenosti vytvoreného súboru.

Zákaz vytvárania náhľadu mediálnych súborov

Moderný správca súborov na zlepšenie používateľského zážitku vytvára náhľad mediálnych súborov na lepšie zorientovanie užívateľa. To však nemusí byť želané správanie aplikácie, ak sa jedná o tajný mediálny súbor. Tieto náhľady zvyknú moderné aplikácie ukladať do dočasnej pamäte na zlepšenie výkonu aplikácie a zároveň môže neželaná osoba pozerať na obrazovku užívateľa, čím môže byť porušený aspekt utajenosti. Preto je potrebné umožniť užívateľovi zakázať vytváranie náhľadov v nastaveniach aplikácie na uchovanie aspektu utajenosti.

Čistenie dočasnej pamäte

Ako už bolo spomenuté, moderné aplikácie zvyknú ukladať náhľady mediálnych súborov do dočasnej pamäte pri zlepšení výkonu aplikácie, čím môže byť porušený aspekt utajenosti. Preto by bolo vhodné zahrnúť v rámci aplikácie vyčistenie dočasnej pamäte náhľadov, aby sa zabránilo nežiaducemu potencionálnemu úniku dát z dočasnej pamäte.

Bezpečné mazanie súborov

Na zachovanie utajenosti súborov by bolo vhodné, aby súbory po ich zmazaní boli aj po hardvérovej stránke zmazané z pamäte mobilného telefónu bez možnosti obnovenia zmazaného súboru forenznou analýzou. Moderné mobilné zariadenia vyrábajú prevažne s UFS alebo eMMC typom pamäte. Bohužiaľ, v týchto pamätiach momentálne pravdepodobne neexistuje spôsob, ako na softvérovej úrovni bezpečne zmazať špecifický súbor z pamäte, preto túto funkciu momentálne nie je možné implementovať [11].

Prevencia voči tapjacking útokom

V bezpečnostnom audite Android aplikácie *„Mullvad“*[12] bola identifikovaná chýbajúca značka `filterTouchesWhenObscured`[6] v pohľadoch (*z anglicko slova „view“*), ktorá zabraňuje takzvanému tapjacking útoku. V riešení som sa rozhodol pridať túto značku z dôvodu zvýšenia úrovne bezpečnosti v aplikácii.

[6] `https://developer.android.com/reference/android/view/View`
dostupné online: 3. december 2020

Úvodná obrazovka a návod

Aby bola aplikácia ľahko použiteľná a užívateľský použiteľná, je potrebné pri prvom spustení aplikácie zobraziť úvodnú obrazovku, ktorá prevedie užívateľa počiatočným nastavením aplikácie a vysvetlí jednotlivé funkcie, ktoré aplikácia obsahuje. Návod priamo v aplikácii je taktiež vhodný pre užívateľskú použiteľnosť, aby bol užívateľ oboznámený s ovládaním aplikácie.

1.4 Existujúce riešenia

Spomedzi existujúcich riešení momentálne neexistuje žiadne, ktoré by zahŕňalo všetky funkcie spomenuté vyššie. Avšak momentálne existujú riešenia s otvoreným zdrojovým kódom, ktoré zahŕňajú niektoré spomenuté funkcie ako napríklad šifrovanie súborov.

1.4.1 OpenKeychain

OpenKeychain[7] je Android aplikácia, ktorá umožňuje šifrovať správy a súbory pomocou OpenPGP štandardu.

Medzi hlavné výhody aplikácie považujem:

- riešenie s otvoreným zdrojovým kódom,
- dostupnosť aplikácie na Google Play a F-Droid platformách,
- podporu OpenPGP štandardu,
- bezpečnostný audit od spoločnosti cure53[8], ktorý bol vykonaný v rámci aplikácie,
- integráciu s rôznymi aplikáciami a podporu siete Tor.

Medzi hlavné nevýhody aplikácie považujem:

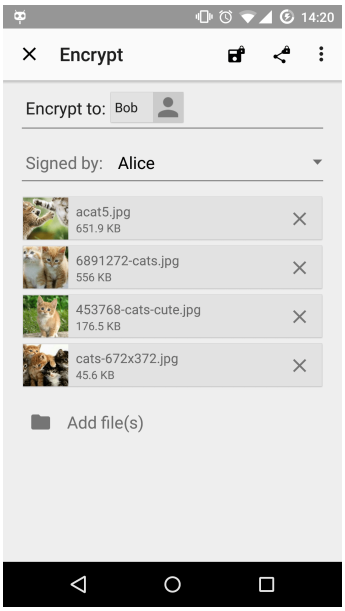
- málo užívateľsky použiteľné ovládanie aplikácie - štandard OpenPGP si vyžaduje pokročilé počítačové zručnosti,
- aktuálne málo aktívny vývoj a údržbu aplikácie - veľký počet nevyriešených úloh a malú frekvenciu nových záznamov v nástroji na správu verzií,

[7] `https://www.openkeychain.org/` dostupné online: 18. október 2020

[8] `https://github.com/open-keychain/open-keychain/wiki/cure53-Security-Audit-2015` dostupné online: 18. október 2020

- e-shop spoločnosti COTECH, ktorý aplikácia zahŕňa,
- softvér tretej strany, ktorý aplikácia propaguje.

Ukážku aplikácie OpenKeychain je možné vidieť na obrázku 1.3 na strane 12.



Obr. 1.3: Ukážka aplikácie OpenKeychain

1.4.2 DroidFS

DroidFS[9] je aplikácia, ktorá umožňuje šifrovať súbor v Android telefóne prostredníctvom súborového systému gocryptfs[10].

Medzi hlavné výhody aplikácie považujem:

- riešenie s otvoreným zdrojovým kódom,
- návrh aplikácie s cieľom čo najefektívnejšej ochrany bezpečia súborov,
- aktuálne aktívny vývoj aplikácie,
- možnosť povolenia menej bezpečných funkcií pre lepší komfort ovládania aplikácie, ako je napríklad povolenie ukladania transformácie hesla použitím odtlačku prsta a povolenie exportu súborov zo zašifrovaného úložiska.

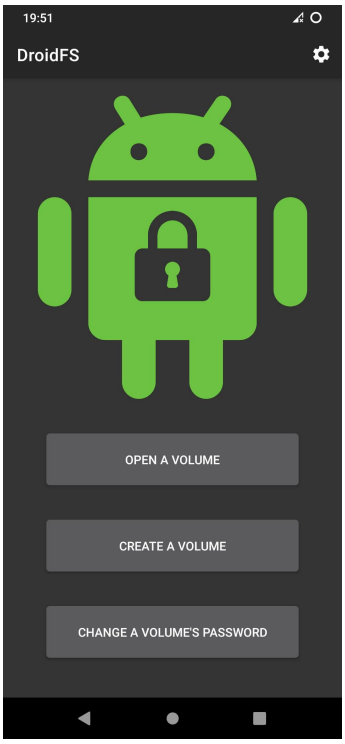
Hlavné hlavné nevýhody aplikácie považujem:

- užívateľsky málo použiteľný manažment šifrovaných úložisk,

[9] `https://github.com/hardcore-sushi/DroidFS` dostupné online: 18. október 2020
[10] `https://github.com/rfjakob/gocryptfs` dostupné online: 18. október 2020

- neprístupnosť aplikácie na platformách Google Play, alebo F-Droid,
- viaceré chyby, ktoré som odhalil pri vyskúšaní aplikácie, tieto chyby zhoršovali zážitok z aplikácie.

Ukážku aplikácie DroidFS je možné vidieť na obrázku 1.4 na strane 13.



Obr. 1.4: Ukážka aplikácie DroidFS

1.4.3 Cryptomator

Cryptomator[11] je aplikácia, ktorá umožňuje šifrovať súbory lokálne alebo na vzdialenom serveri prostredníctvom AES 256 bitovým kľúčom.

Medzi hlavné výhody aplikácie považujem:

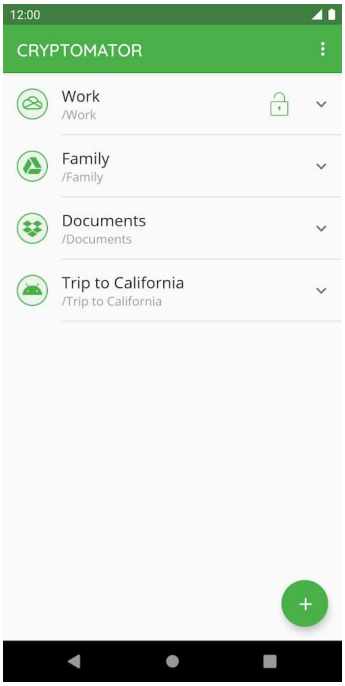
- dostupnosť aplikácie na Google Play platforme,
- kompatibilitu aplikácie s operačnými systémami Windows, MacOS, Linux a iOS,
- užívateľsky použiteľné šifrovanie na vzdialený server alebo lokálne,
- bezpečnostný audit od spoločnosti cure53[12],

[11] `https://cryptomator.org/` dostupné online: 18. október 2020

[12] `https://cryptomator.org/audits/2017-11-27%20crypto%20cure53.pdf` dostupné online: 18. október 2020

- bezpečnostný audit od bezpečnostného konzultanta Tim McLean[13].

Medzi hlavnú nevýhodu aplikácie považujem to, že aplikácia je platená pre Android zariadenia. Ukážku aplikácie Cryptomator je možné vidieť na obrázku 1.3 na strane 12.



Obr. 1.5: Ukážka aplikácie Cryptomator

1.5 Vyhodnotenie analýzy

Na základe analýzy aktuálneho súkromia je žiaduce, aby ľudia brali ohľad na svoje súkromie, pretože sú o nich zbierané množstvo informácii, ktoré môžu byť zneužité na marketing alebo iné neetické praktiky. Jedným z najväčších zdrojov informácií sú mobilné telefóny.

Aktuálna kybernetická bezpečnosť si taktiež vyžaduje zvýšenú pozornosť vzhľadom na zväčšujúci sa výskyt bezpečnostných incidentov. Oblasť mobilných zariadení nie je výnimka.

Spomedzi existujúcich riešení **momentálne neexistuje Android správca súborov, ktorý by zahŕňal všetky zanalyzované funkcie** (*kap. 1.3*). Avšak existujú riešenia s funkciou šifrovania súborov, prípadne riešenia s inými, podobnými

[13] `https://www.chosenplaintext.ca/publications/20161104-siv-mode-report.pdf` dostupné online: 18. október 2020

funkciami zameranými na súkromie a bezpečnosť. Spomedzi týchto riešení väčšina nemá užívateľsky prívetivé ovládanie. Tomuto sa v rámci návrhu a implementácie je potrebné vyvarovať. Pre dobrú dostupnosť aplikácie je žiaduce publikovať vytvorené riešenie pre platformu Google Play a/alebo F-Droid.

2 Návrh implementácie

Táto kapitola je venovaná návrhu jednotlivých funkcií, ku ktorým v ďalších kapitolách bude popísaná implementácia. Jednotlivé funkcie sú rozdelené do podkapitol. Návrh jednotlivých funkcií je zameraný na dodržanie pilierov použiteľnosti, súkromia a bezpečnosti.

2.1 Skrývanie súborov

Android súborový systém je rozdelený následovne[1]:

- **Interné úložisko** - na ukladanie perzistentných súborov a dočasných súborov aplikácie. Od verzie Android 10 je toto úložisko šifrované. Úložisko je prístupné len pre danú aplikáciu, ku ktorej je úložisko pridelené.
- **Externé úložisko** - na ukladanie perzistentných súborov a dočasných súborov aplikácie. Na rozdiel od interného úložiska je toto úložisko je prístupné pre ostatné aplikácie, ktoré majú pridelené právo čítať z externého úložiska. Externé úložisko obsahuje zdielané úložisko. **Zdielané úložisko** je časť externého úložiska, ktoré je určené pre súbory, ktoré majú byť prístupné pre iné aplikácie. Príkladom takýchto súborov sú mediálne súbory alebo dokumenty.

Z definície Android súborového systému vyplýva, že interné úložisko ideálne pre uchovanie skrytých súborov. Na ukladanie skrytých súborov je možné použiť štandardné API volanie na kopírovanie súboru, konkrétne `InputStream`[2]. Pri skrývaní bude cieľom súboru interné úložisko a pri odkrývaní súboru bude cieľom externé úložisko.

[1] `https://developer.android.com/training/data-storage/app-specific`
dostupné online: 26. november 2020

[2] `https://developer.android.com/reference/java/io/InputStream?hl=en`
dostupné online: 26. november 2020

Pri používaní interného úložiska je potrebné byť na pozore, keďže pri odinštalovaní aplikácie alebo zmazaní dát aplikácie je úložisko zmazané. Preto môže nastať situácia, kedy užívateľ odinštaluje aplikáciu alebo odstráni dáta aplikácie a príde nenávratne o skryté súbory. Z toho dôvodu je potrebné o tom varovať užívateľa a implementovať manažment úložiska aplikácie popísaný v kapitole 2.3.

Skrývanie súborov v jadre aplikácie

Aplikácia *„Simple File Manager“*, na ktorej je riešenie postavené, používa systém skrývania súborov pridaním znaku „.“ na začiatok súboru a zobrazuje iba súbory, ktorý daný znak neobsahujú. Skryté súbory zobrazí iba ak si to užívateľ špecificky vyžiada. Daný spôsob skrývania je známy v Unix operačných systémoch. Avšak daný spôsob neumožňuje prístup iba špecifickej aplikácii k zvoleným súborom, preto som sa rozhodol pôvodnú funkciu skrývania súborov odstrániť z jadra aplikácie a nahradiť ju spôsobom presúvania súborov do interného úložiska.

2.2 Šifrovanie súborov

Možností šifrovať súbory v operačnom systéme je mnoho. Existujú knižnice ako napríklad *„java–aes–crypto“*[3] alebo *„Conceal“*[4]. V týchto knižniciach však už nie je udržiavaný vývoj, nie sú ďalej vyvíjané a považujú sa za zastarané. Natívne Android podporuje šifrovanie dvomi spôsobmi:

- Pomocou natívnej knižnice `CipherInputStream`[5]/ `CipherOutputStream`[6], ktorá sa nachádza priamo v API operačného systému Android a tým pádom nie je potrebné pridávať dostatočné závislosti na knižnice.
- Pomocou knižnice `androidx.security.crypto`[7] , ktorá spadá pod sadu knižníc Jatpack[8], konkrétne Jatpack Security, alebo inak JetSec[9]. Knižnicu je potrebné v prípade používania pridať ako závislosť do aplikácie.

[3] `https://github.com/tozny/java-aes-crypto` dostupné online: 27. november 2020

[4] `https://github.com/facebookarchive/conceal` dostupné online: 27. november 2020

[5] `https://developer.android.com/reference/javax/crypto/CipherInputStream?hl=en` dostupné online: 27. november 2020

[6] `https://developer.android.com/reference/javax/crypto/CipherOutputStream?hl=en` dostupné online: 27. november 2020

[7] `https://developer.android.com/reference/androidx/security/crypto/package-summary` dostupné online: 27. november 2020

[8] `https://developer.android.com/jetpack` dostupné online: 27. november 2020

[9] `https://developer.android.com/topic/security/data.md` dostupné online: 27. november 2020

Jatpack Security je založený na knižnici „*Tink*“[10], ktorá má otvorený zdrojový kód a je vyvíjaná spoločnosťou Google.

Knižnica `androidx.security.crypto` je postavená na základe vzoru poskytujúcom bezpečné predvolené nastavenia s dodržaním vyváženia silného šifrovania, dobrého výkonu a maximálneho zabezpečenia. Táto knižnica používa Android KeyStore[11] a overené kryptografické algoritmy, ktoré sú často používané na vytváranie kryptografických protokolov. Knižnica používa AES256-GCM symetrické autentizované šifrovanie s asociovanými údajmi. AES256-GCM by malo byť aktuálne dostatočne bezpečné vzhľadom na použitie 256-bitového kľúča[13].

Vzhľadom na vyššie popísané som sa rozhodol v aplikácii šifrovať súbory prostredníctvom knižnice `androidx.security.crypto`, ktorá umožňuje šifrovať súbory pomocou kľúča uloženého v Android KeyStore. Tým pádom nie je potrebné zadávať heslo k zašifrovaniu a dešifrovaniu, čo je užívateľsky použiteľné. Kryptografické kľúče sú dostupné iba pre aplikáciu, ktorá dané kľúče vygenerovala. Keďže aplikácia bude mať prístup ku kryptografickým kľúčom, bude potrebné prístup k aplikácii autentifikovať, aby kryptografické kľúče nemohli byť použité neautorizovanou osobou.

Pri používaní Android KeyStore je potrebné byť na pozore, keďže kryptografické kľúče sú zmazané spolu s odinštalovaním aplikácie a odstránením dát aplikácie. Preto môže nastať situácia, kedy užívateľ odinštaluje aplikáciu, alebo odstráni dáta aplikácie, čím príde nenávratne o kľúče k zašifrovaným súborom, tým pádom nebude možné dané súbory dešifrovať. Z toho dôvodu je potrebné o tom varovať užívateľa a implementovať manažment úložiska aplikácie popísaný v kapitole 2.3.

2.3 Manažment dát v aplikácii

Android umožňuje pre aplikácie, ktoré v internom úložisku obsahujú citlivé dáta, definovať nasledujúce atribúty v manifeste aplikácie[12]:

- `android:hasFragileUserData` s hodnotou `true` zobrazí pri odinštalovaní aplikácie výzvu s možnosťou uchovania interného úložiska aplikácie a kryptografických kľúčov. Ak užívateľ odinštaluje aplikáciu a zároveň zvolí

[10] `https://github.com/google/tink` dostupné online: 27. november 2020

[11] `https://developer.android.com/training/articles/keystore` dostupné online: 27. november 2020

[12] `https://developer.android.com/guide/topics/manifest/application-element` dostupné online: 26. november 2020

možnosť uchovania dát aplikácie, po následnom spätnom budúcom nainštalovaní aplikácie budú dáta aplikácie obnovené. V rámci riešenia som sa rozhodol nastaviť atribútu hodnotu `true` pre možnosť prístupu k skrytým súborom a kryptografickým kľúčom po odinštalovaní a následnom nainštalovaní aplikácie.

- `android:manageSpaceActivity` s hodnotou úplného názvu aktivity umožní vlastnú správu dát aplikácie. V rámci riešenia som sa rozhodol implementovať vlastnú aktivitu pre správu dát aplikácie, nakoľko aplikácia bude obsahovať citlivé dáta vo forme skrytých súborov a kryptografických kľúčov.
- `android:allowClearUserData` s hodnotou `false` zakáže užívateľovi systémovo zmazať dáta aplikácie. V rámci riešenia som sa rozhodol nastaviť atribútu na hodnotu `false`, keďže si aplikácia bude manažovať správu dát aplikácie sama, vzhľadom na uchovávanie citlivých dát.
- `android:allowBackup` s hodnotou `false` zakáže natívne vytváranie záloh operačným systémom. Tieto zálohy sú odosielané na servery spoločnosti Google. V rámci riešenia som sa rozhodol nastaviť atribútu hodnotu `false`, keďže aplikácia bude obsahuje citlivé dáta a vytvorením zálohy na server sa môže porušiť aspekt utajenosti.

2.4 Overenie prístupu

Overenie prístupu je potrebné pre uchovanie utajenosti citlivých dát v aplikácii. Natívne operačný systém Android aktuálne nepodporuje samostatné overenie prístupu aplikácie, preto je potrebné vytvoriť vlastné riešenie. Overenie prístupu však vytvára bariéru v použiteľnosti aplikácie, preto je potrebné overenie prístupu navrhnúť tak, aby bolo užívateľsky použiteľné.

Na vyváženie bezpečnosti a použiteľnosti som sa rozhodol overiť prístup pri zapnutí aplikácie. Následne, ak bude overená identita užívateľa, bude aplikácia v odomknutom stave a bude môcť byť opätovne otvorená z pozadia operačného systému. Pozadie aplikácie je stav v životnom cykle aplikácie, kedy je aplikácia zastavená bez jej ukončenia[13]. Daný stav aplikácie nastane napríklad ak užívateľ otvorí inú aplikácie v popredí obrazovky. Zmeniť stav aplikácie do zamknutého stavu som sa rozhodol umožniť prostredníctvom ukončenia aplikácie, vypnutím obrazovky zariadenia, alebo zamknutia aplikácie cez notifikáciu pre čo

[13] `https://developer.android.com/guide/components/activities/activity-lifecycle` dostupné online: 3. december 2020

najlepší používateľský zážitok. V operačnom systéme však môžu nastať situácie, kedy môže nastať neželané uchovanie odomknutého stavu aplikácie. Medzi dané udalosti patrí nečakané zlyhanie aplikácie vnútornou alebo systémovou chybou alebo samotné vypnutie zariadenia. Z tohto dôvodu je potrebné, aby aplikácia pozorovala, či nenastala daná situácia a následne reagovala na danú situáciu. Taktiež môže nastať nečakaná situácia vypnutia zariadenia. Reagovať možno v takom prípade zamknutím aplikácie pri zapnutí zariadenia.

V konečnom dôsledku, aplikácia sa teda zamkne ak nastane jedena z nasledujúcich udalosti:

- alikáciu ukončí užívateľ,
- užívateľ zamkne aplikáciu cez notifikáciu,
- užívateľ zamkne obrazovku zariadenia,
- aplikácia zlyhá,
- zapnutie zariadenia.

Samotné overenie prístupu som sa rozhodol na základe analýzy vykonať prostredníctvom biometrickej autentifikácie a autentifikácie heslom.

Biometrické overenie prístupu

Android ponúka natívne biometrickú autentifikáciu prostredníctvom knižnice `androidx.biometric.Biometric`[14], ktorá je súčasťou knižníc Jatpack. Daná knižnica používa biometrickú autentifikáciu na základe nastavení v operačnom systéme.

Overenie prístupu heslom

Pri overení prístupu heslom je potrebné heslo ukladať v transformovanej forme, aby sa zabránilo jednoduchému získaniu hesla v čistej forme pri potencionálnom útoku. Aby sa útočníkovi sťažila šanca na prelomenie hesla útokom hrubou silou, je potrebné použiť silný transformačný algoritmus. V rámci riešenia som sa rozhodol použiť Argon2[14], ktorý v roku 2015 vyhral súťaž transformácií hesiel[15].

[14] `https://developer.android.com/reference/androidx/biometric/package-summary` dostupné online: 3. december 2020

[15] `https://www.password-hashing.net` dostupné online: 3. december 2020

Android natívne nepodporuje transformačný algoritmus Argon2. Avšak existuje knižnica tretej strany `Argon2Kt` [16], ktorá pridáva podporu Argon2 pre programovací jazyk Kotlin, v ktorom je riešenie navrhnuté.

Tento algoritmus umožňuje nastaviť parametre transformácie pre prispôsobenie prostrediu. Rozhodol som sa použiť následovne parametre v rámci riešenia:

- **Verzia:** Argon2id.
- **Počet iterácií:** 1.
- **Počet vlákien:** počet procesorov v aktuálnom zariadení.
- **Veľkosť pamäte:** 5% z pamäte RAM v aktuálnom zariadení.
- **Veľkosť transformácie:** 258 bitov.
- **Soľ (*anglicky „salt“*):** náhodne vygenerovaných 258 bitov knižnicou `java.security.SecureRandom` [17].

V prípade, že sa nepodarí získať veľkosť pamäte RAM v aktuálnom zariadení, rozhodol som sa použiť 65 MB ako predvolenú hodnotu parametra veľkosti pamäte. Pri daných parametroch trvá overenie hesla približne 1 sekundu v závislosti od zariadenia. Verzia Argon2id je navrhnutá ako kompromis rezistencie voči GPU a zároveň CPU lámaním hesla.

Výsledné vygenerované heslo som sa rozhodol ukladať v šifrovanej forme pomocou už spomínanej knižnice `androidx.security.crypto`, konkrétne pomocou `EncryptedSharedPreferences` [18], čím sa zvýši bezpečnosť uloženého hesla.

Aby bola aplikácia chránená pred útokom hrubou silou, slovníkovým útokom, prípadne iným druhom útoku zameraného na odhalenia hesla, je potrebné, aby užívateľ mal nastavené silné heslo. Silné heslo je možné u užívateľa vynútiť kontrolou sily hesla pri nastavení hesla. Kontrola sily hesla je síce používateľsky nevhodná, ale na druhú stranu chráni užívateľa voči spomínaným útokom. Silu hesla som sa rozhodol v aplikácii vynútiť nasledovnými podmienkami, na vyváženie použiteľnosti a sily hesla:

- heslo musí obsahovať aspoň 8 znakov,
- heslo musí obsahovať kombináciu znakov a číslic.

[16] `https://github.com/lambdapioneer/argon2kt`
dostupné online: 3. december 2020

[17] `https://developer.android.com/reference/java/security/SecureRandom`
dostupné online: 3. december 2020

[18] `https://developer.android.com/reference/androidx/security/crypto/EncryptedSharedPreferences` dostupné online: 3. december 2020

Overenie prístupu v jadre aplikácie

Aplikácia *„Simple File Manager“*, na ktorej je riešenie postavené, používa systém overenia prístupu prostredníctvom PIN, vzoru alebo odtlačku prsta. Overenie prístupu je vykonané pri zapnutí aplikácie, čím sa zmení na odomknutý stav. Na zmenu stavu na stav zamknutia je potrebné ukončiť aplikáciu, čo prináša bezpečnostnú hrozbu. Overenie prístupu prostredníctvom PIN a vzoru považujem za málo bezpečné, keďže obsahujú počet možných kombinácií, ktoré sú potenciálne náchylné na útok hrubou silou. Na overenie prístupu prostredníctvom odtlačku prsta je použitá knižnica tretej strany *„Reprint“*[19], ktorá je interne postavená na natívnej Android knižnici `android.hardware.fingerprint.FingerprintManager`[20]. Knižnica *„Reprint“* je v čase písania práce viac ako rok nevyvíjaná a zároveň natívna knižnica `android.hardware.fingerprint.FingerprintManager` sa považuje za zastaralú. Namiesto nej je odporúčané použiť knižnicu `androidx.biometric.Biometric`. Knižnica `androidx.biometric.Biometric` na rozdiel od knižnici `android.hardware.fingerprint.FingerprintManager` podporuje okrem overením odtlačku prsta aj iné biometrické overenia ako napríklad overenie tvárou. Na základe vyššie popísaného som sa rozhodol systém overenia prístupu prostredníctvom PIN, vzoru alebo odtlačku prsta odstrániť a nahradiť ho vlastným riešením popísaným v kapitole.

2.5 Ostatné funkcie

V tejto podkapitole je navrhnutý dizajn pre ostatné funkcie, ktoré majú ciel zvýšiť bezpečnosť a súkromie používateľa.

2.5.1 Integrita súborov

„National Institute of Standards and Technology“, alebo skrátene NIST, vydal publikáciu s detailným popisom odporúčaných kryptografických transformačných funkcií[15]. V tejto publikácii sú odporúčané transformačné funkcie SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 a SHA-512/256. V rámci riešenia som sa rozhodol použiť najpoužívanejšie z nich, a to konkrétne SHA-1, SHA-256 a SHA-512. Okrem týchto transformačných funkcií je taktiež populárna transfor-

[19] `https://github.com/ajalt/reprint` dostupné online: 3. december 2020

[20] `https://developer.android.com/reference/android/hardware/fingerprint/FingerprintManager` dostupné online: 3. december 2020

mačná funkcia MD5 so zraniteľnosťou kolízií[16]. Z dôvodu popularity MD5 som sa rozhodol v rámci riešenia použiť taktiež transformačnú funkciu MD5 pre vykonanie kontroly integrity súborov prostredníctvom kontrolného súčtu aj napriek spomenutej zraniteľnosti.

V operačnom systéme Android je natívne podporený výpočet kontrolného súčtu prostredníctvom `MessageDigest`[21]. Daná knižnica podporuje výpočet všetkých transformačných funkcií, ktoré som sa rozhodol použiť v riešení. Výpočet kontrolného súčtu pri veľkých súboroch môže byť časovo náročný, preto je potrebné výpočet vykonať na samostatnom vlákne. Pre užívateľsky použiteľné riešenie som sa rozhodol kontrolný súčet zobraziť v dialógu vlastností súboru. Ako už bolo spomenuté, výpočet kontrolného súčtu je časovo náročný, preto je potrebné kontrolný súčet vykonať až po potvrdení užívateľom pre ušetrenie batérie zariadenia, aby aplikácia nevykonávala náročné operácie zbytočne a nezaťažovala tak zariadenie.

2.5.2 Zakázanie snímky obrazovky

Operačný systém Android natívne podporuje zakázanie snímky obrazovky, a to pridaním značky `FLAG_SECURE`[22]. Danú značku je možné pridať do okna aplikácie priamo pri vytvorení aktivity.

2.5.3 Podpora kompresie a dekompresie šifrovaných Zip súborov

Riešenie, na ktorom je práca postavená, umožňuje vytvárať Zip súbory pomocou natívnej Android knižnice `Zip`[23]. Avšak táto knižnica nepodporuje vytváranie šifrovaných Zip súborov. Preto som sa rozhodol na vytvorenie šifrovaných Zip súborov použiť knižnicu tretej strany *„Zip4j“*[24]. Táto knižnica je podporovaná operačným systémom Android, keďže je implementovaná v jazyku Java. Vzhľadom na použitie tejto knižnice je potrebné pôvodné riešenie implementované v knižnici `Zip` odstrániť a implementovať prácu so Zip súbormi pomocou knižnice *„Zip4j“*. Kompresia a dekompresia Zip súborov je časovo náročná, preto je

[21] `https://developer.android.com/reference/java/security/MessageDigest` dostupné online: 3. december 2020

[22] `https://developer.android.com/reference/android/view/WindowManager.LayoutParams#FLAG_SECURE` dostupné online: 3. december 2020

[23] `https://developer.android.com/reference/java/util/zip/package-summary` dostupné online: 4. december 2020

[24] `https://github.com/srikanth-lingala/zip4j` dostupné online: 4. december 2020

potrebné funkcie implementovať tak, aby sa vykonala na pozadí. Na dlho vykonávané operácie Android natívne poskytuje službu (*z anglického slova „service“*)[25], pre ktorú je možné vytvoriť notifikáciu na monitorovanie aktuálneho stavu operácie a možnosť ukončenia danej operácie. Jadro aplikácie používa na vykonanie kompresie a dekompresie Zip súborov vlákno (*z anglického slova „thread“*)[26], ktoré nie je ideálne na túto operáciu vzhľadom na jeho vlastnosti. Pri použití služby operačný systém zaručí úspešné dokončenie operácie, aj keď aplikácia, ktorá danú službu vykonáva, bude ukončená alebo bude zastavená. V rámci riešenia som sa rozhodol použiť službu na kompresiu a dekompresiu Zip súborov.

2.5.4 Určenie miesta uloženia mediálnych súborov

Prostredníctvom natívnych Android API volaní je možné aplikáciu s implementovanou funkciou fotoaparátu požiadať o vytvorenie videa alebo obrázka pomocou akcie `ACTION_VIDEO_CAPTURE` alebo akcie `ACTION_IMAGE_CAPTURE`[27]. Pre vyvolanie akcie je potrebne v zámere (*z anglického slova „intent“*) zavolať metódu `startActivityForResult` nad aktivitou s extra hodnotou `EXTRA_OUTPUT`, ktorá obsahuje absolútnu cestu pre nový mediálny súbor.

2.5.5 Čistenie dočasnej pamäte

Aplikácia *„Simple File Manager“*, na ktorej je riešenie postavené, je závislá na knižnici tretej strany *„Glide“*[28]. Táto knižnica slúži ako rámec pre správu médií a ich načítanie. Táto knižnica taktiež spravuje dočasnú pamäť, čím je možné volaním API metódy vyvolať vyčistenie dočasnej pamäte aplikácie.

2.5.6 Úvodná obrazovka a návod

Implementovať úvodnú obrazovku som sa rozhodol prostredníctvom knižnice *„AppIntro“*[29], ktorá zjednoduší a urýchli vývoj danej funkcii. Danú knižnicu som

[25] `https://developer.android.com/reference/android/app/Service`
dostupné online: ť. december 2020

[26] `https://developer.android.com/reference/java/lang/Thread`
dostupné online: ť. december 2020

[27] `https://developer.android.com/reference/android/provider/MediaStore`
dostupné online: 4. december 2020

[28] `https://github.com/bumptech/glide` dostupné online: 4. december 2020

[29] `https://github.com/AppIntro/AppIntro` dostupné online: 8. január 2021

sa rozhodol zvoliť vzhľadom na jej aktívny vývoj, funkcie a užívateľský používateľné rozhranie. V rámci úvodnej obrazovky som sa rozhodol dať nastavenie povolenia prístupu k externému úložisku, pretože aplikácia bez daného povolenia je zbytočná a nepoužiteľná. Taktiež som sa rozhodol pridať do úvodnej obrazovky nastavenie overenia, ktoré je kritické pre uchovanie utajenosti skrytých a zašifrovaných súborov užívateľa. Okrem toho som sa rozhodol do úvodnej obrazovky vložiť konfiguráciu aplikácie, aby mal užívateľ nakonfigurovanú aplikáciu ihneď pri prvom spustení spolu s vysvetlením jednotlivých bezpečnostných funkcií.

Na implementáciu návodu som sa rozhodol použiť knižnicu *„TapTargetView“*[30], ktorá taktiež zjednoduší a urýchli vývoj danej funkcie. Danú knižnicu som sa rozhodol použiť hlavne ako návod na vysvetlenie práce so súbormi, konkrétne so skrývaním a šifrovaním súborov, vytváraním mediálnych súborov v zvolenom adresári.

2.6 Návrh používateľského rozhrania

Pre implementované funkcie som sa rozhodol pre zachovanie konzistencie používateľského rozhrania použiť Material Design[31]. Material Design je dizajnérsky jazyk od spoločnosti Google. Aktuálne väčšina Android aplikácií používa tento dizajnérsky jazyk, pretože je to primárny jazyk pre vývoj Android aplikácií. Na zjednodušenie vývoja Google ponúka knižnicu na prácu s týmto jazykom. Konkrétne to je je knižnica `com.google.android.material` [32], ktorá je použitá v aplikácii. Návrh dizajnu a jeho správania je navrhnutý na základe dizajnových prvkov popísaných v dokumentácii dizajnového jazyka[33].

2.7 Vyhodnotenie návrhu

Výsledkom kapitoly je podrobný návrh implementácie funkcií postavených nad jadrom aplikácie, konkrétne nad aplikáciou *„Simple File Manager“*. Jednotlivé funkcie sú navrhnuté s cieľom zabezpečiť čo najlepšiu užívateľskú používateľnosť so zameraním na zachovanie súkromia a bezpečia súborov používateľa. V rámci návrhu boli popísané funkcie skrývania súborov, kompresie a dekompresie šifrovaných a nešifrovaných Zip súborov, overenia prístupu, ktoré sú implemen-

[30] `https://github.com/KeepSafe/TapTargetView` dostupné online: 8. január 2021

[31] `https://www.material.io` dostupné online: 4. apríl 2021

[32] `https://material.io/develop/android/docs/getting-started` dostupné online: 4. apríl 2021

[33] `https://material.io/design` dostupné online: 4. apríl 2021

tované v jadre aplikácie, avšak s nedostatočným spôsobom ochrany súkromia a zabezpečenia. Preto budú tieto funkcie odstránené a implementované spôsobom popísaným v návrhu. Taktiež budú implementované aj ďalšie funkcie na zlepšenie súkromia a bezpečia súborov používateľa, a to konkrétne šifrovanie súborov, integrita súborov, manažment dát aplikácie, zakázanie snímky obrazovky, určenie miesta uloženia mediálnych súborov, čistenie dočasnej pamäte, prevencia voči tapjacking útokom, úvodná obrazovka a návod. Na zachovanie konzistencie je v návrhu aplikácie použitý Material Design.

3 Implementácia

Táto kapitola je venovaná implementácii jednotlivých funkcií. Jednotlivé funkcie sú rozdelené do podkapitol. Kapitola 3.1 je venovaná implementácii jadra aplikácie, z ktorej výsledné riešenie vychádza. Implementácia jednotlivých funkcií je zameraná na dodržanie pilierov použiteľnosti, súkromia a bezpečnosti.

3.1 Jadro aplikácie

Pri vytváraní jadra aplikácie je potrebné najskôr rozvetviť (*z anglického slova „fork“*) zdrojový kód. Po vytvorení kópie zdrojového kódu je následne potrebné zmeniť balík aplikácie. Rozhodol som sa použiť balík `com.securefilemanager.app`.

Riešenie *„Simple File Manager“* je závislé od knižnice `com.simplemobiletools:commons`[1], na ktorej je postavený celý balík aplikácií od autora riešenia *„Simple File Manager“*, ako napríklad *„Simple Gallery“*[2], alebo *„Simple Calendar“*[3]. Vzhľadom na to, že vo výslednej aplikácii je potrebné robiť komplexné zmeny, rozhodol som knižnicu `com.simplemobiletools:commons` integrovať priamo do aplikácie. Keďže daná knižnica obsahuje základ kódu pre všetky aplikácie autora riešenia *„Simple File Manager“*, v rámci integrácie knižnice do aplikácie je potrebné odstrániť nepotrebný kód na zmenšenie komplexnosti kódu a jeho čitateľnosti.

Na zjednodušenie kódu a čiastočné zamedzenie potencionálnych bezpečnostných chýb som sa rozhodol taktiež odstrániť nasledujúce funkcie z riešenia *„Simple File Manager“*:

[1] `https://github.com/SimpleMobileTools/Simple-Commons`
dostupné online: 26. november 2020

[2] `https://github.com/SimpleMobileTools/Simple-Gallery`
dostupné online: 26. november 2020

[3] `https://github.com/SimpleMobileTools/Simple-Calendar`
dostupné online: 26. november 2020

- Podporu OTG zariadení.
- Podporu root zariadení.
- Zmenu farieb v rozhraní.
- Pridanie predpony alebo prípony pri premenovaní súboru.
- Mriežkového zobrazenia súborov.
- Viacjazyčnú podporu.

Na jednoduchosť a čo najširšie pokrytie používateľov som sa rozhodol v aplikácii podporiť iba najrozšírenejší jazyk - angličtinu. V rámci implementácie som sa rozhodol taktiež zmeniť niektoré z funkcií:

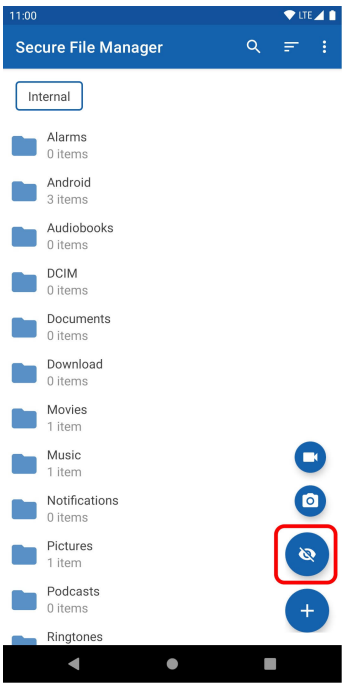
- Obrazovku *„o nás“* (*z anglického slova „about“*).
- Obrazovku *„často kladené otázky“*.
- Obrazovku licencii tretích strán.
- Obrazovku nastavení aplikácie.
- Podporu dekompresie a kompresie Zip súborov (*viac v podkapitole 2.5.3 a 3.6.3*).
- Podporu skrývania súborov (*viac v podkapitole 2.1 a 3.2*).
- Overenie prístupu pomocou PIN, vzoru a odtlačku (*viac v podkapitole 2.4 a 3.5*).

3.2 Skrývanie súborov

Ako už bolo spomenuté v analýze, na skrývanie súborov je možné použiť natívnu knižnicu `InputStream` s cieľovým adresárom interného úložiska. Na uchovanie štruktúry v internom úložisku som sa rozhodol skryté súbory ukladať v adresári `.hidden` v koreni interného úložiska aplikácie. Na získanie absolútnej cesty k internému úložisku je možné v aktivite zavolať metódu `this.context.filesDir.absolutePath`. Absolútna cesta tak môže v závislosti od zariadenia a jeho cesty k internému úložisku vyzerať nasledovne:

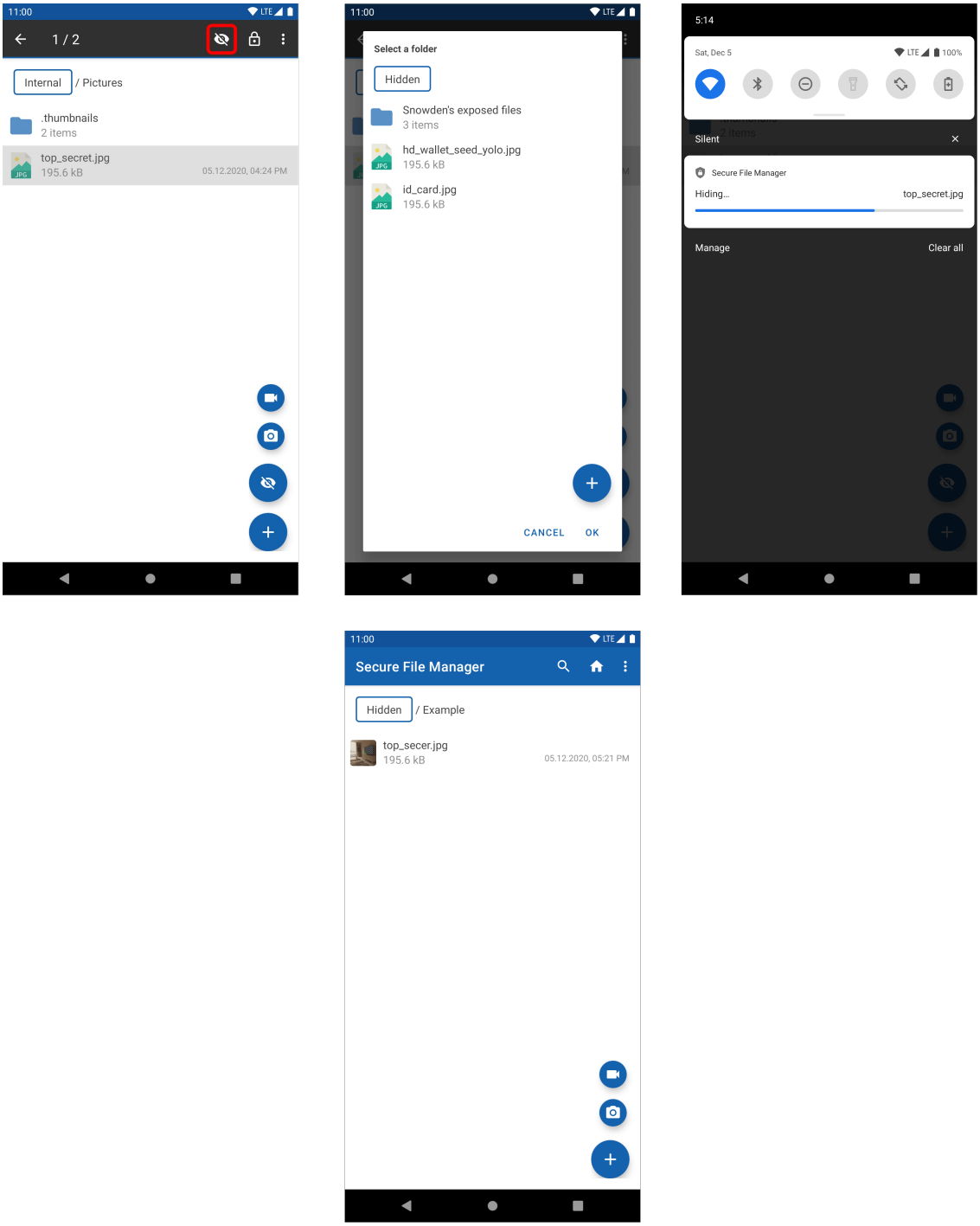
`/data/user/0/com.securefilemanager.app/files/.hidden`

Vzhľadom na to, že skrývanie je primárnou funkciou aplikácie, je potrebné užívateľské rozhranie navrhnúť tak, aby bolo skryté úložisko ľahko a rýchlo prístupné. Preto som sa rozhodol zobraziť plávajúce tlačidlo akcie skrývania súborov, ak sa užívateľ nenachádza v skrytom úložisku. Po kliknutí na plávajúce tlačidlo akcie sa otvorí užívateľovi obsah skrytého úložiska jedným kliknutím. Ukážka plávajúceho tlačidla akcie skrývania súborov je v obrázku 3.1 na strane 29. Taktiež som sa rozhodol spríjemniť zážitok používateľa pridaním možnosti skrývania súboru na prvé miesto v menu pri manipulačnom režime súboru/súborov.



Obr. 3.1: Ukážka plávajúceho tlačidla akcie skrývania súborov

Aby si mohol užívateľ upravovať adresárovú štruktúru v skrytom adresári, rozhodol som sa po zadaní akcie skrytia súboru otvoriť dialóg na voľbu cieľovej cesty, ktorý je dostupný v jadre aplikácie v triede s názvom `FilePickerDialog`. Po zvolení cieľovej cesty sa skrývanie vykoná formou kopírovania súboru pomocou asynchrónnej úlohy, ktorá je dostupná v jadre aplikácie v triede s názvom `CopyMoveTask`. V prípade, že sa vykonáva skrývanie adresára, jednotlivé súbory a adresáre v danom súbore sú rekurzívne skopírované do cieľovej cesty. V prípade úspešnej operácie sú skopírované súbory zo zdroja zmazané. V prípade skrývania veľkých súborov je o aktuálnom stave skrývania užívateľ informovaný pomocou notifikácie. Ukážku procesu skrývania súborov je možné vidieť na obrázku 3.2 na strane 30.



Obr. 3.2: Ukážka procesu akcie skrývania súboru

3.3 Šifrovanie súborov

Pri implementácii šifrovania súborov je možné taktiež použiť triedu `CopyMoveTask`, ktorú je potrené upraviť na podporu funkcie šifrovania súborov. Triedu je potrebné upraviť nasledovne:

1. V prípade nájdenia konfliktu, a teda rovnakého názvu súboru v aktuálnom pracovnom priečinku pri operácii šifrovania alebo dešifrovania, zobraziť správu s upozornením nájdenia konfliktu a nevykonaním operácie.

2. Pri operácii šifrovania použiť šifrovací výstupný prúd a pri operácii dešifrovania použiť dešifrovací vstupný prúd.

3. Po úspešnej operácii šifrovania alebo dešifrovania zmazať zdrojový súbor, ak je daná akcia povolená v nastaveniach aplikácie.

4. V prípade šifrovania veľkých súborov je o aktuálnom stave šifrovania potrebné informovať užívateľa pomocou notifikácie.

Na získanie šifrovacieho výstupného prúdu a dešifrovacieho vstupného prúdu je možné použiť fragment kódu 3.1 na strane 32. Metóda `getEncryptedFile` používa statickú metódu `EncryptedFile.Builder` [4] na získanie šifrovaného súboru a následne šifrovacieho výstupného prúdu alebo dešifrovacieho vstupného prúdu. Na získanie kľúča sa používa statická trieda `FileCrypto`, ktorá vráti kľúč k súboru na základe špecifikovaných parametrov v triede pomocou statickej triedy `MasterKey.Builder`. V rámci aplikácie som sa rozhodol použiť kľúče pre súbory s nasledovnými parametrami:

- **Šifrovací typ bloku:** GCM
- **Šifrovacia výplň:** bez výplne
- **Účel:** šifrovanie a dešifrovanie
- **Veľkosť:** 256 bitov
- Použiť bezpečný modul hardvéru (*Strongbox*), ak ho zariadenie podporuje.

Bližšie informácie generovania kľúča sú popísané v dokumentácii triedy `MasterKey.Builder` [5].

Na rozlíšenie zašifrovaných a nezašifrovaných súborov som sa rozhodol pri šifrovaní súboru nastaviť `.aes` súborový formát pre daný súbor a pri dešifrovaní tento formát odstrániť. Súbor tak po zašifrovaní môže mať nasledujúci názov:

`názov_súboru.jpg.aes`

[4] `https://developer.android.com/reference/androidx/security/crypto/EncryptedFile.Builder?hl=en` dostupné online: 28. december 2020

[5] `https://developer.android.com/reference/androidx/security/crypto/MasterKey.Builder?hl=en` dostupné online: 28. december 2020

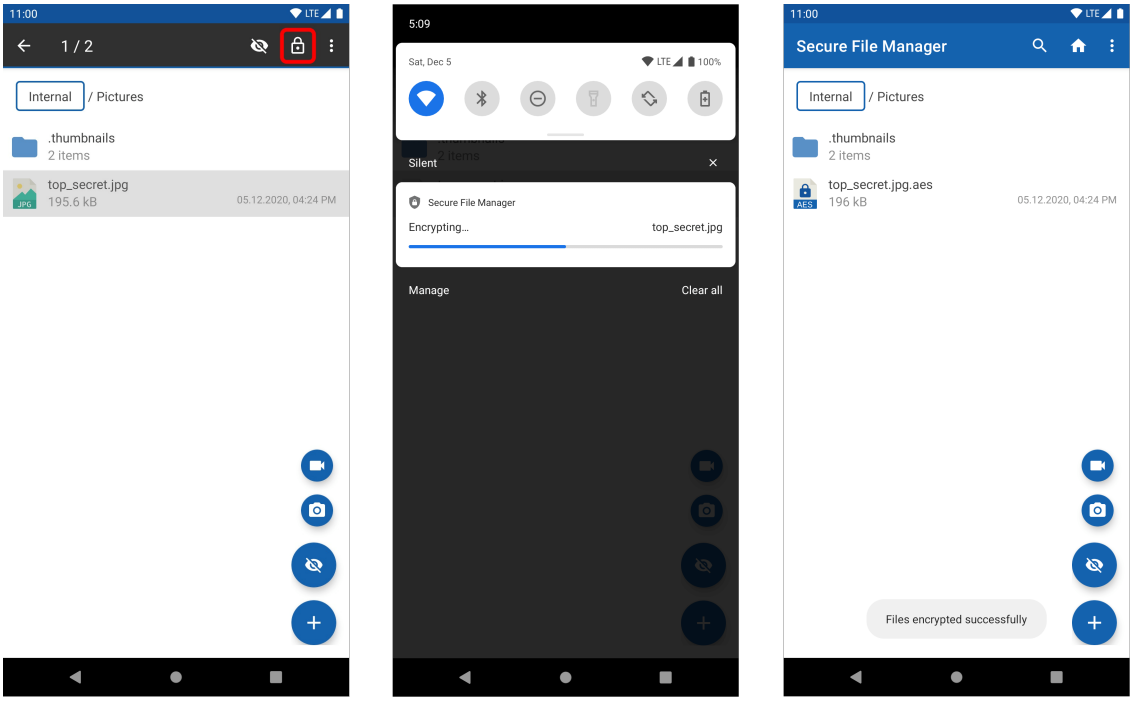
Pre užívateľskú používateľnosť som sa rozhodol zašifrované súbory rozlíšiť špecifickou ikonou náhľadu súboru. Ukážku procesu skrývania súborov je možné vidieť na obrázku 3.3 na strane 33.

```
fun Context.getEncryptedFile(file: File): EncryptedFile =
    EncryptedFile.Builder(
        this,
        file,
        FileCrypto.getKey(this),
        FileCrypto.ENCRYPTION_SCHEME
    ).build()

fun Context.getFileInputEncryptedStreamSync(path: String)
    : FileInputStream =
        getEncryptedFile(File(path)).openFileInput()

fun Context.getFileOutputEncryptedStreamSync(file: File)
    : FileOutputStream =
        getEncryptedFile(file).openFileOutput()
```

Zdrojový kód 3.1: Metódy získania šifrovacieho prúdu

3.4 Manažment dát v aplikácii

Na zamedzenie nechceného zmazania citlivých súborov ako sú kryptografické kľúče a skryté súbory je potrebné implementovať manažment dát aplikácie. Na implementáciu vlastnej správy internej pamäte je najskôr potrebné definovať v manifeste aplikácie atribút `android:manageSpaceActivity` na hodnotu `com.securefilemanager.app.activities.ManageStorageActivity`, čo predstavuje absolútnu cestu k triede aktivity, ktorá slúži na správu interného úložiska. Následne je možné implementovať danú aktivitu. Aktivitu som sa rozhodol implementovať tak, že aktivita vloží do svojho pohľadu fragment správy pamäte. Fragment správy pamäte následne používa abstraktnú triedu `SettingsAbstractFragment`, pomocou ktorej sú inicializované nastavenia čistenia internej pamäte a nastavenia čistenia dočasnej pamäte. Implementácia čistenia dočasnej pamäte je popísaná v kapitole 3.6.3. Na implementáciu čistenia internej pamäte je možné rozšíriť triedu `Activity` o metódu `deleteAppData`.

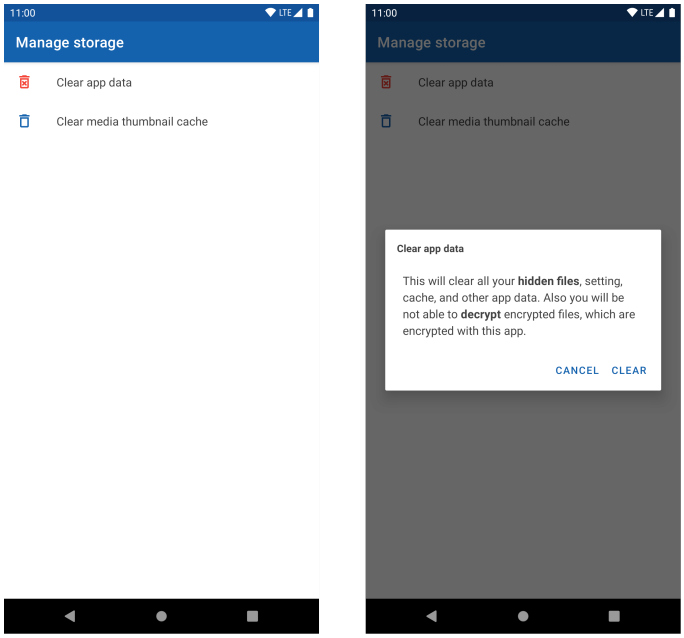


Obr. 3.3: Ukážka akcie šifrovania súboru v menu

Danú metódu je možné implementovať fragmentom kódu 3.2 na strane 33. Daný fragment kódu spusti príkaz vyčistenia všetkých dát, ktoré sú asociované s balíkom aplikácie a následne sa aplikácia vypne, pretože je v nekonzistentnom stave, kedy je zmazané interné úložisko aplikácie. Po opätovnom spustení je aplikácia v rovnakom stave, ako pri prvom zapnutí aplikácie. Keďže mazanie internej pamäte aplikácie je deštruktívna akcia, je potrebné informovať užívateľa o tom, že budú zmazané všetky dáta aplikácie, vrátane skrytých súborov, nastavení aplikácie, a zároveň nebude možné dešifrovať zašifrované súbory aplikáciou. Taktiež som sa rozhodol implementovať zmazanie interného úložiska tak, že sa zmaže až po piatom potvrdení akcie zmazania, aby sa predišlo vykonaniu nechcenej akcie. Ukážku manažovania dát aplikácie je možné vidieť na obrázku 3.4 na strane 34.

```
fun Activity.deleteAppData() {
    val packageName = this.applicationContext.packageName
    Runtime.getRuntime().exec("pm clear $packageName")
    this.quitApp()
}
```

Zdrojový kód 3.2: Metóda zmazania internej pamäte



Obr. 3.4: Ukážka aktivity manažovania dát aplikácie

3.5 Overenie prístupu

Vzhľadom na komplexnosť implementácie overenia prístupu je implementácia rozdelená na:

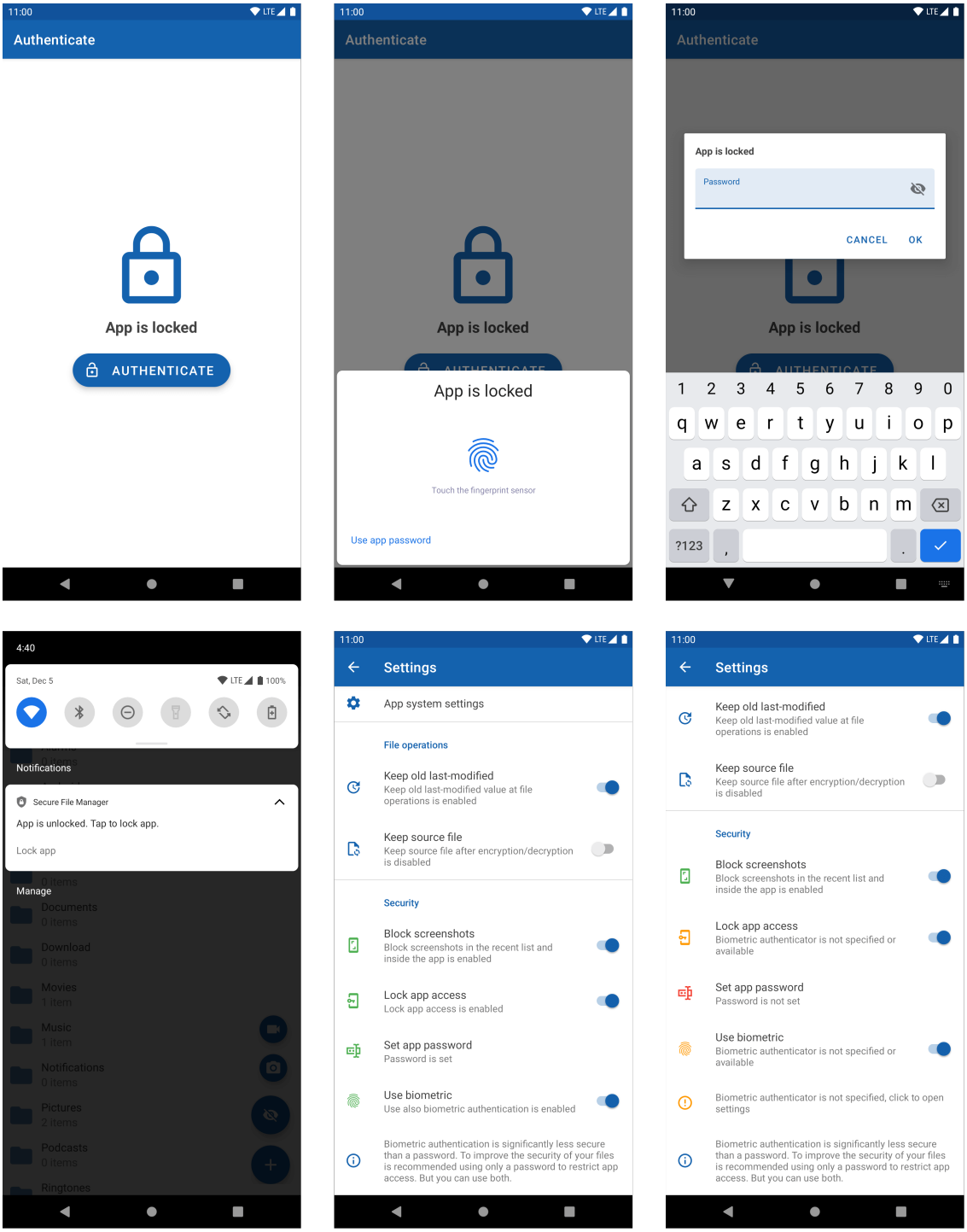
- implementáciu nastavenia overenia,
- implementáciu jadra overenia,
- implementáciu overenia hesla,
- implementáciu biometrického overenia.

Jednotlivý popis implementácií je rozdelený na podkapitoly. Ukážku overenia prístupu v aplikácii je možné vidieť na obrázku 3.5 na strane 35.

3.5.1 Implementácia nastavenia overenia

Pre užívateľskú používateľnosť som sa rozhodol implementovať nastavenia overenia prístupu skrývaním a odkrývaním jednotlivých nastavení na základe aktuálneho stavu. Diagram stavu nastavenia overenia prístupu je možné vidieť na strane 36.

V prípade, že nie je zapnutá funkcia overenia prístupu, nie je možné nastaviť heslo ani biometrické overenie. V prípade, že je zapnutá funkcia overenia prístupu, zobrazí sa možnosť nastavenia hesla a biometrického overenia spolu s upozornením, že je potrebné nastaviť aspoň jednu možnosť overenia. Po kliknutí na



Obr. 3.5: Ukážka overenia prístupu

nastavenie hesla je zobrazený dialóg nastavenia hesla. Ukážku dialógu je možné vidieť na obrázku 3.7 na strane 37.

V prípade, že je zapnuté biometrické overenie, zobrazí sa bezpečnostné upozornenie s odporúčaním uprednostnenia hesla pred biometrickým overením na zvýšenie bezpečnosti. V prípade, že je zapnuté biometrické overenie, ale zariadenie nemá nastavené biometrické overenie na úrovni operačného systému, ap-

Obr. 3.6: Diagram stavov nastavenia overenia prístupu

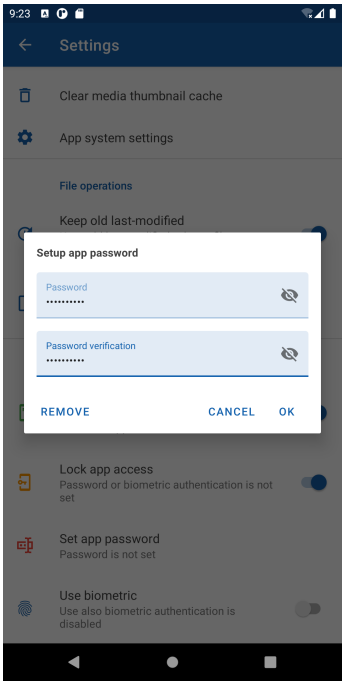
likácia o tom upozorní užívateľa a umožní mu jednoduché otvorenie nastavení operačného systému po kliknutí na upozornenie, na nastavenie biometrického overenia. Jednotlivé stavy overenia sú na zvýšenie užívateľskej použiteľnosti farebne odlíšené.

Aby bola aplikácia zabezpečená už pri prvom spustení aplikácie, rozhodol som sa nastavenie overenia vložiť taktiež v obrazovke overenia, ktorá je popísaná v kapitole 49

3.5.2 Implementácia jadra overenia

Pri implementácii jadra overenia prístupu je potrebné uchovávať stav zamknutia v zdieľaných predvoľbách aplikácie (*SharedPreferences*)[6] v privátnom režime. Privátny režim nebezpečí, aby ostatné aplikácie nevedeli pristupovať k predvoľbám

[6] `https://developer.android.com/reference/android/content/SharedPreferences` dostupné online: 6. január 2021



Obr. 3.7: Ukážka dialógu nastavenia hesla

aplikácie. Vo fragmente kódu 3.3 na strane 37 je možné vidieť ukážku uchovania stavu aplikácie, ktorý sa nachádza v triede `Config` aplikácie.

```
var wasAppProtectionHandled: Boolean
    get() = prefs.getBoolean(WAS_APP_PROTECTION_HANDLED, false)
    set(wasAppProtectionHandled) = prefs.edit()
        .putBoolean(WAS_APP_PROTECTION_HANDLED, wasAppProtectionHandled)
        .apply()
```

Zdrojový kód 3.3: Ukážka uchovania stavu zamknutia aplikácie

V prípade, že je aplikácia v zamknutom stave, pri jej otvorení sa zobrazí aktivita overenia. V danej obrazovke je užívateľ overený buď heslom alebo biometrickým overením podľa nastavení aplikácie. Implementácia biometrického overenia je popísaná na strane 40 a implementácia overenia heslom je popísaná na strane 41. V prípade, že dôjde k overeniu užívateľa, aplikácia zmení svoj stav na odomknutý.

Aby aplikácia pozorovala, kedy nastane jej otvorenie, je možné implementovať triedu, ktorá implementuje `LifecycleObserver` rozhranie a následne definovaním metódy s anotáciou `@OnLifecycleEvent(Lifecycle.Event.ON_STOP)` pozorovať stav vypnutia a s anotáciou

`@OnLifecycleEvent(Lifecycle.Event.ON_START)` pozorovať stav zapnutia[7]. Ukážku implementácie pozorovateľa v aplikácii je možné vidieť vo fragmente kódu 3.4 na strane 38.

```
class AuthenticationObserver(activity: Activity) : LifecycleObserver {
    private var mActivity: Activity = activity
    @OnLifecycleEvent(Lifecycle.Event.ON_STOP)
    fun onAppBackground() {
        this.mActivity.startStopUnlockAppService()
    }
    @OnLifecycleEvent(Lifecycle.Event.ON_START)
    fun onAppForeground() {
        this.mActivity.startStopUnlockAppService()
        this.mActivity.startAuthenticationActivity()
    }
}
```

Zdrojový kód 3.4: Trieda implementácie pozorovateľa stavu aplikácie

Následne je potrebné vo všetkých koreňových aktivitách registrovať pozorovateľa pri vytvorení koreňovej aktivity a odstránení pozorovateľa pri ukončení koreňovej aktivity. To je možné dosiahnuť príkladovým fragmentom kódu aktivity 3.5 na strane 39.

Pri otvorení aplikácie pozorovateľ vytvorí aktivitu overenia. Aby sa predišlo vytvoreniu viacerých inštancií aktivít, je potrebné aktivitu vytvoriť so značkou `FLAG_ACTIVITY_SINGLE_TOP`[8].

Vo fragmente kódu 3.4 je možné vidieť, že prí ukončení a otvorení aplikácie sa volá metóda `this.mActivity.startStopUnlockAppService()`. Táto metóda slúži na ukončenie/vytvorenie služby na správu notifikácie zamknutia obrazovky. V prípade, že je aplikácia v zamknutom stave, metóda ukončí notifikáciu zamknutia obrazovky, inak ju vytvorí. V prípade kliknutia na notifikáciu je aplikácia zamknutá a notifikácia je ukončená. Vytváranie notifikácií je popísané v dokumentácii operačného systému Android[9].

[7] `https://developer.android.com/topic/libraries/architecture/lifecycle` dostupné online: 6. január 2021

[8] `https://developer.android.com/reference/android/content/Intent#FLAG_ACTIVITY_SINGLE_TOP` dostupné online: 6. január 2021

[9] `https://developer.android.com/training/notify-user/build-notification`

```
class Acitivity : AppCompatActivity() {
    private lateinit var mAuthenticationObserver: AuthenticationObserver
    private var lockReceiver: LockReceiver = LockReceiver()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        this.mAuthenticationObserver = AuthenticationObserver(this)
        ProcessLifecycleOwner.get().lifecycle
            .addObserver(this.mAuthenticationObserver)
        this.registerReceiver(this.lockReceiver, LockReceiver.getIntent())
        Thread.setDefaultUncaughtExceptionHandler { _, _ ->
            this.appLock()
        }
        ...
    }
    override fun onDestroy() {
        ProcessLifecycleOwner.get().lifecycle
            .removeObserver(this.mAuthenticationObserver)
        this.unregisterReceiver(this.lockReceiver)
        ...
        super.onDestroy()
    }
}
```

Zdrojový kód 3.5: Príklad implementácie triedy zodpovednej za zamknutie aplikácie

Aby užívateľ nevedel obísť overenie prístupu tlačidlom späť v aktivite overenia, ktoré v základnom správaní ukončí aktivitu a vráti sa na predchádzajúcu aktivitu, je potrebné prepísať akciu tlačidla späť v aktivite overenia, aby sa aplikácia ukončila v prípade stlačenia tlačidla späť v aktivite overenia. Dané je možné implementovať fragmentom kódu 3.6 na strane 40.

Zamknutie aplikácie pri vypnutí obrazovky a zapnutí zariadenia je možné implementovať rovnakou metodikou, a to pomocou triedy `BroadcastReceiver` s prijímaním akcií `ACTION_LOCKED_BOOT_COMPLETED` a `ACTION_SCREEN_OFF`. Implementácia je popísaná v dokumentácii operačného systému Android[10]. Registráciu triedy je možné vidieť vo fragmente kódu 3.5 na

dostupné online: 6. január 2021

[10] `https://developer.android.com/guide/components/broadcasts`

dostupné online: 6. január 2021

```
override fun onBackPressed() {
    this.finishAffinity()
}
```

Zdrojový kód 3.6: Metóda akcie späť v aktivite overenia

strane 39 za predpokladu, že trieda `LockReceiver` implementuje `BroadcastReceiver`.

Uzamknúť aplikáciu v prípade jej zlyhania je možné prostredníctvom metódy `setDefaultUncaughtExceptionHandler` nad triedou `Thread` [11]. Ukážku implementácie metódy `setDefaultUncaughtExceptionHandler` je možné vidieť vo fragmente kódu 3.5 na strane 39.

Zamknutie aplikácie v prípade ukončenia aplikácie je možné detektovať v hlavnej aktivite pomocou preťaženia metódy `onDestroy` [12].

3.5.3 Implementácia biometrického overenia

Ako už bolo spomenuté v analýze riešenia, na implementáciu biometrického overenia je možné použiť knižnicu `androidx.biometric.Biometric`. Biometrické overenie je možné implementovať pomocou oficiálnej dokumentácie operačného systému Android[13].

V prípade, že je nastavené v aplikácii overenie heslom a zároveň biometrické overenie, je možné preťažiť v rozhraní `BiometricPrompt.AuthenticationCallback` metódu `onAuthenticationError` a následne vytvoriť podmienku, či bolo stlačené negatívne tlačidlo. Ak áno, je možné otvoriť dialóg overenia heslom, ktorý je popísaný v podkapitole implementácie overenia heslom. Text negatívneho tlačidla je možné nastaviť volaním metódy `setNegativeButtonText` nad triedou `BiometricPrompt.PromptInfo.Builder` pri vytváraní výzvy na biometrické overenie.

[11] `https://developer.android.com/reference/java/lang/Thread#setDefaultUncaughtExceptionHandler(java.lang.Thread.UncaughtExceptionHandler)` dostupné online: 6. január 2021

[12] `https://developer.android.com/reference/android/app/Activity#onDestroy()` dostupné online: 9. január 2021

[13] `https://developer.android.com/training/sign-in/biometric-auth` dostupné online: 6. január 2021

3.5.4 Implementácia overenia heslom

Pre nastavenie overenia heslom je možné implementovať dialóg nastavenia hesla. Pre implementáciu overenia heslom je možné aplikovať dialóg overenia hesla. Dialóg nastavenia hesla a dialóg overenia hesla je možné implementovať pomocou triedy `AlertDialog.Builder`, ktorá je dostupná natívne v operačnom systéme Android[14]. Na kontrolu sily hesla v dialógu nastavenia hesla na základe návrhu implementácie je možné použiť nasledujúci regulárny výraz:

```
^.*(?=.{8,})(?=.*[a-zA-Z])(?=.*\d).*&
```

Pred uložením hesla je potrebné heslo najskôr transformovať. Na základe návrhu implementácie je zvolený algoritmus Argon2, konkrétne knižnica `Argon2Kt`. Pre transformovanie hesla je možné zavolať medódu `hash` nad triedou `Argon2Kt` s parametrami popísanými v návrhu riešenia. Na overenie hesla s transformovanou podobou je možné použiť metódu `verify`. Na získanie soli je možné použiť triedu `SecureRandom` a jej metódu `nextBytes`. Získanie počtu procesorov v zariadení je možné zavolaním nasledujúcej metódy:

`Runtime.getRuntime().availableProcessors()` [15]

Na získanie veľkosti pamäte RAM je možné prečítať súbor `/proc/meminfo` v operačnom systéme Android pomocou triedy `RandomAccessFile` a jej metódy `readLine()` [16]. Po prečítaní súboru je potrebné dáta rozobrať a tým získať požadovanú hodnotu veľkosti pamäte RAM a z nej vyrátať 5%. V aplikácii je použitá metóda 3.7 na strane 42 na získanie veľkosti pamäte na transformáciu hesla.

Po transformovaní hesla na základe návrhu implementácie je potrebné heslo uložiť do šifrovanej podoby pomocou `EncryptedSharedPreferences`. Implementácia práce s danou knižnicou je popísaná v oficiálnej dokumentácii operačného systému Android[17]. Na získanie kľúča použitého na uloženie hesla sa využíva statická trieda `PrefCrypto`, ktorá vráti kľúč k heslu na základe špecifikovaných parametrov v triede pomocou statickej triedy `MasterKey.Builder`. V rámci aplikácie som sa rozhodol použiť kľúč pre heslo s nasledujúcimi parametrami:

- **Šifrovací mód bloku:** GCM

[14] `https://developer.android.com/reference/androidx/appcompat/app/AlertDialog.Builder?hl=en` dostupné online: 6. január 2021

[15] `https://developer.android.com/reference/java/lang/Runtime#availableProcessors()` dostupné online: 6. január 2021

[16] `https://developer.android.com/reference/kotlin/java/io/RandomAccessFile#readline` dostupné online: 6. január 2021

[17] `https://developer.android.com/topic/security/data#edit-shared-preferences` dostupné online: 6. január 2021

```
private fun getMemoryCost(): Int {
    val randomAccessFile = RandomAccessFile("/proc/meminfo", "r")
    val readLine: String = randomAccessFile.readLine()
    randomAccessFile.close()
    val p: Pattern = Pattern.compile("(\\d+)")
    val m: Matcher = p.matcher(readLine)
    if (m.find()) {
        val group: String? = m.group(1)
        if (group != null) {
            val memoryContKB: Int = (group.toInt() * 0.05).toInt()
            return memoryContKB.convertKBtoKiB()
        }
    }
    return defaultMemoryCost
}
```

Zdrojový kód 3.7: Metóda získania veľkosti pamäte na transformáciu hesla

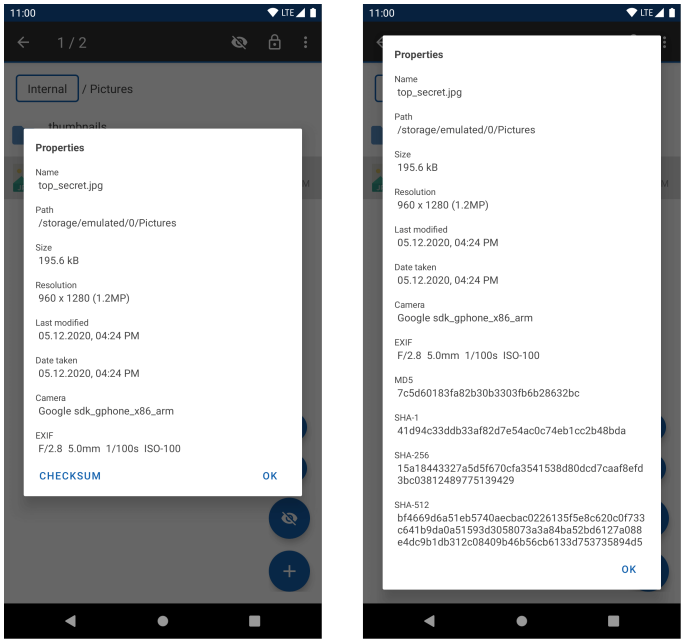
- **Šifrovacia výplň:** bez výplne
- **Účel:** šifrovanie a dešifrovanie
- **Veľkosť:** 256 bitov
- Použiť bezpečný modul hardvéru (*Strongbox*), ak ho zariadenie podporuje.

3.6 Ostatné funkcie

V nasledujúcich kapitolách je popísaná implementácia ostatných funkcií aplikácie na základe návrhu. Jednotlivé funkcie sú rozdelené do podkapitol s popisom riešenia implementácie.

3.6.1 Integrita súborov

Na základe návrhu je možné implementovať fragment kódu 3.8 na strane 44, ktorý obsahuje rozšírenie triedy `File` metódou `getDigest`, ktorá vypočíta kontrolný súčet súboru pomocou natívnej knižnice `java.security.MessageDigest`. Pri vstupe daná metóda dostane názov transformačnej funkcie a pri výstupe vráti



Obr. 3.8: Ukážka akcie kontrolného súčtu

reťazec obsahujúci kontrolný súčet v hexadecimálnej podobe. Na vykonanie operácie v samostatnom vlákne je možné použiť metódu `ensureBackgroundThread`, ktorá je dostupná v jadre aplikácie. Ukážku integrity súborov je možné vidieť na obrázku 3.8 na strane 43.

3.6.2 Zakázanie snímky obrazovky

Ako už bolo spomenuté v analýze, je potrebné pridať značku `FLAG_SECURE` do okna aplikácie na zakázanie snímky obrazovky. Najskôr je však potrebné pridať do nastavení aplikácie možnosť nastavenia zákazu snímky obrazovky. Rozhodol som sa v základom nastavení zapnúť zákaz snímok obrazovky pre zvýšenie bezpečnosti a súkromia užívateľa aplikácie. Ak je nastavený zákaz snímky obrazovky, je potrebné do všetkých aktivít aplikácie pridať značku `FLAG_SECURE`. Na dodržanie čistého kódu som sa rozhodol rozšíriť triedu `Activity` o metódu `addFlagsSecure`, ktorá pridá danú značku, ak je nastavený v aplikácii zákaz vytvorenia snímky obrazovky. Ukážku implementácie `addFlagsSecure` je možné vidieť v 3.9 na strane 44.

```
fun File.getDigest(algorithm: String): String =
    this.inputStream().use { fis ->
        val md = MessageDigest.getInstance(algorithm)
        val buffer = ByteArray(8192)
        generateSequence {
            when (val bytesRead = fis.read(buffer)) {
                -1 -> null
                else -> bytesRead
            }
        }.forEach { bytesRead -> md.update(buffer, 0, bytesRead) }
        md.digest().joinToString("") { "%02x".format(it) }
    }
```

Zdrojový kód 3.8: Metóda pre výpočet kontrolného súčtu

```
fun Activity.addFlagsSecure() {
    if (this.config.disableScreenshots) {
        this.window.setFlags(
            WindowManager.LayoutParams.FLAG_SECURE,
            WindowManager.LayoutParams.FLAG_SECURE
        )
    }
}
```

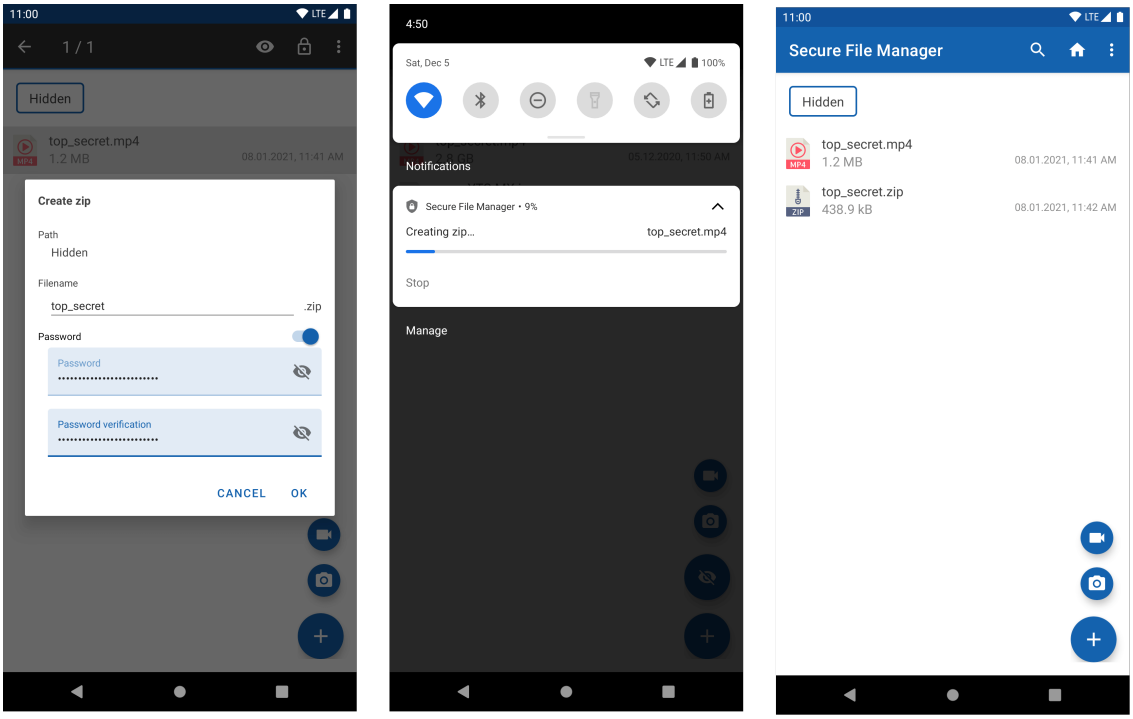
Zdrojový kód 3.9: Metóda na pridanie značky zákazu snímky obrazovky

3.6.3 Podpora kompresie a dekompresie šifrovaných Zip súborov

Na základe návrhu je potrebné najskôr odstrániť pôvodné riešenie podpory kompresie a dekompresie Zip súborov a nahradiť ho vlastným riešením postaveným na knižnici *„Zip4j“*. Dokumentácia ku knižnici je dostupná v jej zdrojovom kóde[18]. Popis implementácie služby popredia a notifikácie je dostupný v oficiálnej dokumentácii operačného systému Android[19]. Pri nahradení pôvodného riešenia je najskôr potrebné pridať do dialógu vytvorenia Zip súboru `CompressAsDialog`

[18] `https://github.com/srikanth-lingala/zip4j` dostupné online: 8. január 2021

[19] `https://developer.android.com/guide/components/foreground-services` dostupné online: 8. január 2021

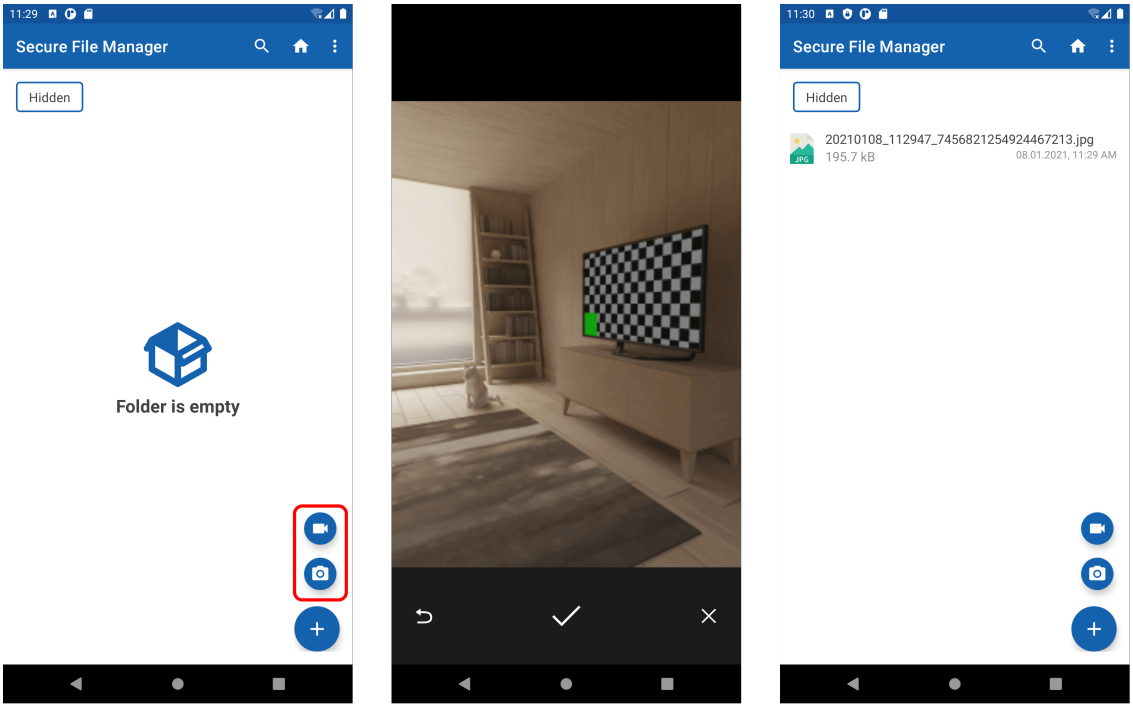


Obr. 3.9: Ukážka vytvárania Zip súboru

možnosť zadania hesla. Keďže heslo nie je potrebné pri vytvorení Zip súboru, užívateľ si môže zadať heslo zapnutím prepínacieho tlačidla a následne zadaním hesla. Po pridaní zadania hesla je možné odstrániť pôvodné riešenie a nahradiť ho novým riešením podľa dokumentácie ku knižnici, ktorá je použitá v novom riešení. Na obrázku 3.9 na strane 45 je možné vidieť ukážku implementovanej funkcie kompresie a dekompresie šifrovaných Zip súborov.

3.6.4 Určenie miesta uloženia mediálnych súborov

Ná základe návrhu implementácie je možné naprogramovať určenie miesta uloženia mediálnych súborov podľa obrázku 3.10 na strane 47. Pre užívateľsky použiteľné rozhranie som sa rozhodol pridať plávajúce tlačidlo akcie vytvorenia fotky a plávajúce tlačidlo akcie vytvorenia videa. Po kliknutí na dané plávajúce tlačidlá sa otvorí aplikácia fotoaparátu a po vytvorení mediálneho súboru sa súbor uloží do aktuálneho adresára, v ktorom sa užívateľ aktuálne v aplikácii nachádza. V prípade, že zariadenie obsahuje viacej aplikácií fotoaparátu, užívateľ si môže na úrovni systému Android zvoliť, ktorú aplikáciu použije na vytvorenie mediálneho obsahu. V prípade, že zariadenie neobsahuje aplikáciu fotoaparátu v zariadení, systém Android natívne o danej udalosti informuje užívateľa. Ukážka funkcie je na obrázku 3.10 na strane 46.



Obr. 3.10: Ukážka určenia miesta uloženia mediálnych súborov

3.6.5 Zákaz vytvárania náhľadu mediálnych súborov

Pri implementácii zákazu vytvárania náhľadu mediálnych súborov je potrebné pridať pri vytváraní náhľadov podmienku, aby sa dané náhľady vytvorili iba vtedy, ak je dané povolené. V prípade nastavenia zákazu vytvárania náhľadov je potrebné zmazať už vytvorené mediálne súbory. Keďže náhľady sú ukladané v dočasnej pamäti, stačí dočasnú pamäť zmazať. Možnosť nastavenia zobrazenia náhľadov som sa rozhodol pridať do nastavení aplikácie a úvodnej obrazovky. V ukážke zdrojového kódu 3.11 na strane 48 je možné vidieť ukážku aplikácie s povoleným a zakázaným vytváraním náhľadov mediálnych súborov.

3.6.6 Čistenie dočasnej pamäte

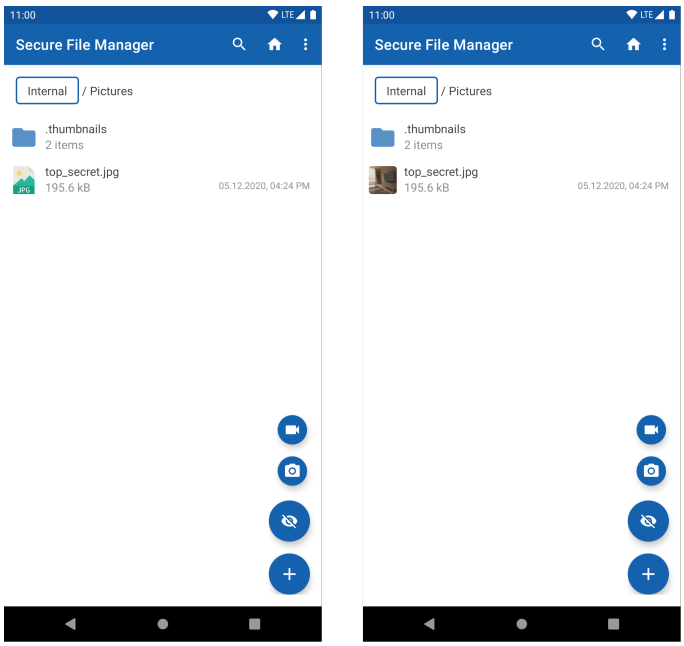
Knižnica tretej strany *„Glide“* pomocou volania metódy `clearDiskCache` [20] umožňuje vymazať dočasnú pamäť. Keďže operácia vyčistenia dočasnej pamäte môže byť časovo náročná, je potrebné danú operáciu vykonať v samostatnom vlákne. Programovací jazyk Kotlin umožňuje vykonanie operácie v samostatnom vlákne pomocou statickej metódy `GlobalScope.launch` [21]. Čistenie dočasnej pamäte som

[20] `https://bumptech.github.io/glide/doc/caching.html#disk-cache`
dostupné online: 28. december 2020

[21] `https://kotlinlang.org/docs/reference/coroutines/basics.html#coroutine-basics`
dostupné online: 28. december 2020

```
private fun takeMedia(action: String, extension: String, requestCode: Int) {
    Intent(action).also { takePictureIntent ->
        takePictureIntent.resolveActivity(context?.packageManager!!)?.also {
            createMediaFile(currentPath, extension).also {
                currentMediaFile = it
                val mediaURI: Uri = this.requireContext().getUriForFile(it)
                takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, mediaURI)
                startActivityForResult(takePictureIntent, requestCode)
            }
        }
    }
}

private fun takeVideo() {
    takeMedia(MediaStore.ACTION_VIDEO_CAPTURE, ".mp4", VIDEO_REQUEST_CODE)
}

private fun takePicture() {
    takeMedia(MediaStore.ACTION_IMAGE_CAPTURE, ".jpg", IMAGE_REQUEST_CODE)
}
```

Zdrojový kód 3.10: Fragment kódu na určenie miesta uloženia mediálnych súborov



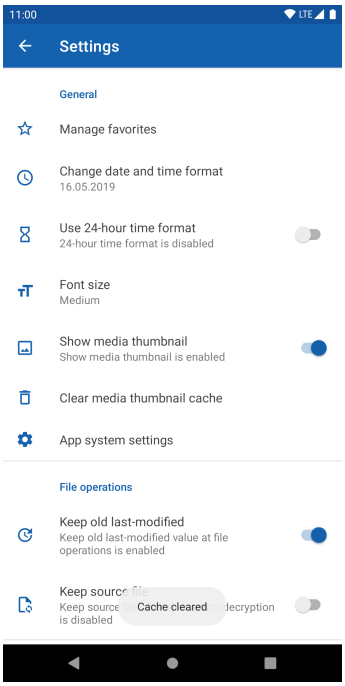
Obr. 3.11: Ukážka zobrazenia náhľadu mediálnych súborov

sa rozhodol pre užívateľskú použiteľnosť umiestniť do nastavení aplikácie, ako je možné vidieť na obrázku 3.12 na strane 48. Taktiež som sa rozhodol informo-

vať užívateľa o úspešnom zmazaní dočasnej pamäte pomocou správy. 3.11 na strane 48 je výsledný fragment kódu zodpovedný za mazanie dočasnej pamäte.

```
private fun clearGlideCache() {
    val activity = requireActivity()
    GlobalScope.launch {
        Glide.get(activity).clearDiskCache()
    }
    activity.toast(R.string.media_thumbnail_clear_cleared)
}
```

Zdrojový kód 3.11: Metóda mazania dočasnej pamäte



Obr. 3.12: Ukážka čistenia dočasnej pamäte

3.6.7 Prevencia voči tapjacking útokom

Pri implementácii prevencie voči tapjacking útokom je potrebné pridať značku `filterTouchesWhenObscured` do všetkých koreňových pohľadov. V 3.12 na strane 49 je možné vidieť ukážku definovania pohľadu so značkou `filterTouchesWhenObscured`.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/frame_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:filterTouchesWhenObscured="true"
    tools:context=".activities.ManageStorageActivity" />
```

Zdrojový kód 3.12: Ukážka pohľadu s preveciou voči tapjacking útokom

3.6.8 Úvodná obrazovka

Úvodnú obrazovku je možné implementovať pomocou dokumentácie ku knižnici *„AppIntro“*[22]. Detektovať prvé spustenie aplikácie je možné pomocou zdieľaných predvolieb aplikácie[23]. Keďže v úvodnej obrazovke sú fragmenty z obrazovky nastavení, rozhodol som sa kompletne prepísať správu nastavení v aplikácii, a to do abstraktnej triedy `SettingsAbstractFragment`, následne v jednotlivých fragmentoch pristupovať k metódam triedy `SettingsAbstractFragment` pomocou enkapsulácie. Keďže v jednotlivých krokoch úvodnej obrazovky je použité vlastné zobrazenie, rozhodol som sa pre každý krok v úvodnej obrazovke vytvoriť fragment pre definovanie zobrazenia a jeho správania. Aby sa užívateľ vedel časom vrátiť k úvodnej obrazovke, rozhodol som sa úvodnú obrazovku zobraziť okrem prvého spustenia aj pomocou kliknutia z menu aplikácie. Ak užívateľ otvorí úvodnú obrazovku kliknutím na menu, pre užívateľskú používateľnosť som sa rozhodol pridať možnosť ukončenia zobrazenia úvodnej obrazovky kliknutím tlačidla späť. Ak je úvodná obrazovka otvorená pri úvodnom spustení, užívateľ je nútený prejsť úvodnou obrazovkou. Ukážku úvodnej obrazovky je možné vidieť na obrázku 3.13 na strane 50.

3.6.9 Návod

Návod v aplikácii pri prvom spustení je možné implementovať pomocou dokumentácie ku knižnici *„TapTargetView“*[24].

Aby sa užívateľovi vedel nanovo zobraziť návod, rozhodol som sa úvodnú ob-

[22] `https://github.com/AppIntro/AppIntro` dostupné online: 8. január 2021

[23] `https://developer.android.com/reference/android/content/SharedPreferences` dostupné online: 6. január 2021

[24] `https://github.com/KeepSafe/TapTargetView` dostupné online: 8. január 2021

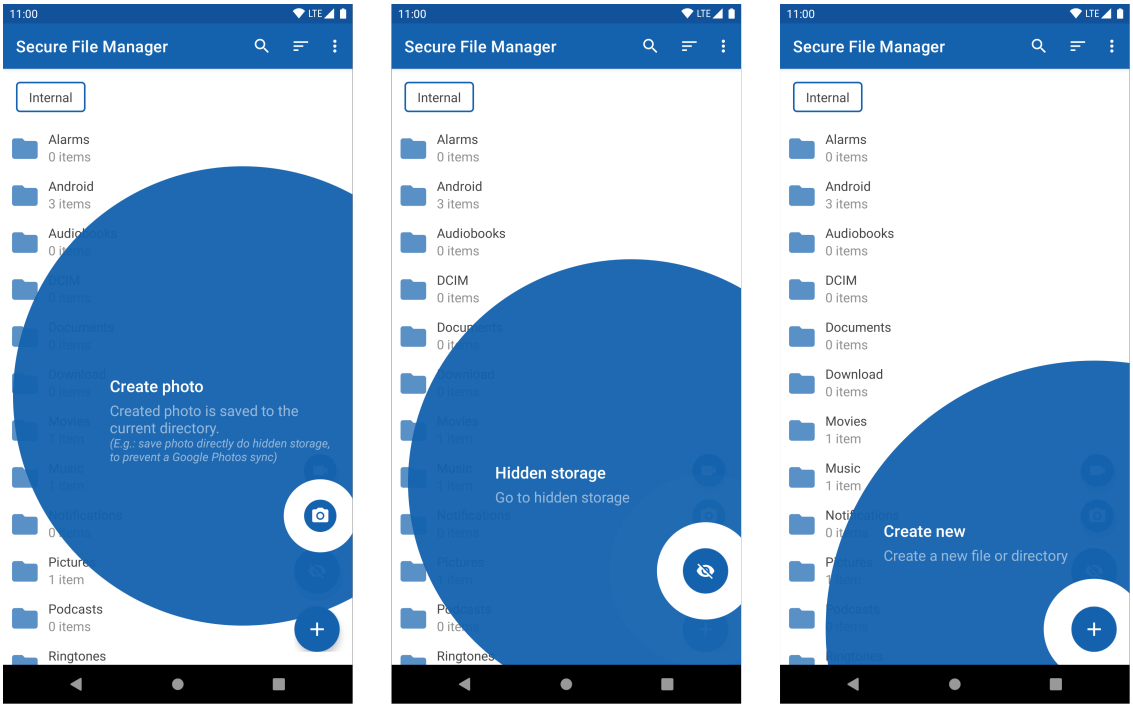
Obr. 3.13: Ukážka úvodnej obrazovky

razovku zobraziť okrem prvého spustenia aj pomocou kliknutia z menu aplikácie. Ak užívateľ otvorí návod kliknutím na menu, pre užívateľskú používateľnosť som sa rozhodol pridať možnosť ukončenia návodu kliknutím mimo aktuálneho cieleného pohľadu návodu. Ak sa užívateľovi zobrazí návod pri prvom otvorení aplikácie, užívateľ je nútený prejsť celým návodom. Ukážku návodu je možné vidieť na obrázku 3.14 na strane 51 a obrázku 3.15 na strane 52.

Obr. 3.14: Ukážka návodu 1. časť

3.7 Vyhodnotenie implementácie

Výsledkom kapitoly je podrobný popis samotnej implementácie funkcií postavených nad jadrom aplikácie, konkrétne nad aplikáciou *„Simple File Manager“*. Jednotlivé funkcie sú navrhnuté pre čo najlepšiu užívateľskú použiteľnosť so zameraním na zachovanie súkromia a bezpečia súborov používateľa.



Obr. 3.15: Ukážka návodu 2. časť

4 Vyhodnotenie

V kapitole je realizovaná analýza testovania aplikácie prostredníctvom platformy BrowserStack, publikácia aplikácie na platforme Google Play a F-Droid, audit súkromia a audit závislosti. Súčasťou kapitoly je vyhodnotenie použiteľnosti, popísané používateľské testovanie a jeho výsledky. V závere sú uvedené možné rozšírenia a zhrnuté výsledky vyhodnotenia.

4.1 Testovanie aplikácie prostredníctvom platformy BrowserStack

Platforma BrowserStack[1] je mobilná a zároveň webová testovacia platforma, ktorá umožňuje testovať aplikácie pomocou rôznych reálnych zariadení a ich verzií operačných systémov. Platforma BrowserStack poskytuje službu *„App Live“*, pomocou ktorej je možné testovať mobilné aplikácie na operačnom systéme Android a iOS. Táto služba je spoplatnená. Ak sa však testuje projekt s otvoreným zdrojovým kódom, daná službu je zadarmo. Pomocou služby *„App Live“* sa mi podarilo aplikáciu vyladiť na rôznych Android zariadeniach s rôznou verziou operačného systému Android. Pomocou platformy je jednoduchšie v niektorých prípadoch replikovať a vyladiť dané chyby. Na zabezpečenie zjednodušeného testovania je možné do nástroja Gradle nainštalovať BrowserStack rozšírenie, pomocou ktorého je možné nahrávať priamo balík aplikácie do platformy BrowserStack. Týmto rozšírením sa tak zjednoduší proces testovania aplikácie na platforme. Vo fragmente kódu 4.1 na strane 54 je možné vidieť konfiguráciu, ktorá je použitá pri nastavení rozšírenia BrowserStack v nástroji Gradle.

Prostredníctvom danej konfigurácie sú uchované vývojárske kľúče v bezpečí v súbore `secrets.properties`. Je však dôležité, aby sa tento súbor nachádzal medzi ignorovanými súbormi v nástroji na správu verzií. V prípade nástroja Git, v ktorom je práca tvorená, je ignorovanie súborov možné nastaviť v súbore

[1] `https://www.browserstack.com/` dostupné online: 11. marec 2021

```
buildscript {
    ext {
        ...
        browserstack_version = '3.0.2'
    }

    repositories {
        ...
        maven {
            url "https://plugins.gradle.org/m2/"
        }
    }

    dependencies {
        ...
        classpath "gradle.plugin.com.browserstack.gradle:browserstack-gradle-plugin:${browserstack_version}"
    }
}

static def getSecret(name){
    Properties props = new Properties()
    try {
        props.load(new FileInputStream(new File('secrets.properties')))
    } catch(ignored) {
        return ""
    }
    return props[name]
}

browserStackConfig {
    username = getSecret('BROWSERSTACK_USERNAME')
    accessKey = getSecret('BROWSERSTACK_ACCESSKEY')
}
```

Zdrojový kód 4.1: Ukážka konfigurácie rozšírenia BrowserStack v nástroji Gradle

`.gitignore`. Pre nahrávanie balíka aplikácie do platformy BrowserStack stačí spustiť následujúci príkaz v koreňovom adresári projektu aplikácie:

```
gradle clean uploadDebugToBrowserstackAppLive
```

4.2 Publikovanie aplikácie na platforme Google Play

Aplikáciu som sa rozhodol sprístupniť širokej verejnosti, a to publikovaním aplikácie na platforme Google Play[2], ktorá slúži ako natívny repozitár aplikácií pre väčšinu mobilných zariadení postavených na operačnom systéme Android. Na publikovanie je potrebné mať vytvorený Google účet so zaplateným vývojárskym poplatkom vo výške $25. Po zaplatení poplatku je vývojárovi sprístupnená Google play konzola (*z anglického slova "Google Play Console"*)[3], pomocou ktorej je možné publikovať aplikácie. Pre publikovanie aplikácie je potrebné vytvoriť podpísaný balík aplikácie, následne vyplniť údaje o aplikácii, ako napríklad názov,

[2] `https://play.google.com/store/apps/details?id=com.securefilemanager.app` dostupné online: 18. február 2021

[3] `https://play.google.com/console/about/` dostupné online: 18. február 2021

popis a obrázky aplikácie spoločne s nahraním vytvoreného podpísaného balíka. Po zaslaní žiadosti je aplikácia preverená a publikovaná. Podrobný návod publikovania aplikácie na platforme Google Play je popísaný v dokumentácii operačného systému Android[4].

Pri vytváraní aktualizácií je možné postupovať rovnako ako pri publikovaní aplikácie. Pri zautomatizovaní aktualizácií je možné použiť službu Fastlane[5], ktorá urýchli a zjednoduší vydávanie aktualizácií pomocou kontinuálnej integrácii a nasadenia. Popis integrácie služby Fastlane je popísaný v dokumentácii[6].

V rámci štatistík platformy Google play, kde je aplikácia publikovaná, mála aplikácia v čase písania 214 inštalácií, z toho 55 aktívnych zariadení. Aplikácia dostala hodnotenie 4.583/5 od 12 používateľov, z toho 5 hodnotenia boli hodnotenia s recenziou.

4.3 Publikovanie aplikácie na platforme F-Droid

Na sprístupnenie aplikácie širšej verejnosti som sa rozhodol aplikáciu publikovať taktiež na platforme F-Droid. Táto platforma je alternatíva k platforme Google Play, ktorá obsahuje iba aplikácie s otvoreným zdrojovým kódom. Na publikovanie aplikácie je potrebné v repozitári obsahujúcom dáta pre vytvorenie balíka aplikácie[7] vytvoriť žiadosť o zlúčenie s pridaním súboru, ktorý obsahuje dané dáta. Príklad takéhoto súboru je možné vidieť v ukážke zdrojového kódu 4.2 na strane 57. Po vytvorení žiadosti sa prispievatelia projektu F-Droid vyjadria k vašej žiadosti o zlúčenie. V prípade, že aplikácia spĺňa všetky podmienky pre začlenenie aplikácie do platformy, je žiadosť o zlúčenie akceptovaná. Podmienky pre začlenenie aplikácie do platformy F-Droid sú nasledujúce:

- Aplikácia musí byť zadarmo a s otvoreným zdrojovým kódom.
- Aplikácia musí mať jedinečný identifikátor.
- Ochranné známky a iné zákonné náležitosti musia byť dodržané.
- Aplikácia nesmie obsahovať binárne závislosti.
- Aplikácia nesmie obsahovať proprietárne Google služby.
- Aplikácia nesmie obsahovať proprietárne analytické a sledovacie nástroje.

[4] `https://developer.android.com/studio/publish` dostupné online: 18. február 2021
[5] `https://fastlane.tools/` dostupné online: 18. február 2021
[6] `https://docs.fastlane.tools/` dostupné online: 18. február 2021
[7] `https://gitlab.com/fdroid/fdroiddata` dostupné online: 21. február 2021

- Aplikácia nesmie obsahovať proprietárne knižnice reklám.
- Aplikácia nesmie obsahovať nástroje na vytvorenie balíka aplikácie, ktoré nie sú zadarmo.
- Aplikácia nesmie dodatočne sťahovať spustiteľné binárne súbory.
- Ak aplikácia obsahuje *„Anti-Features“*[8], musí byť daná informácia ohlásená.

V prípade mojej žiadosti o pridelenie aplikácie do platformy[9] prispievateľ, ktorý bol pridelený k mojej žiadosti, žiadal o objasnenie použitých povolení v aplikácii a následne priloženie daného objasnenia do popisu aplikácie. Prispievateľ mal taktiež odporúčanie pre spôsob pomenovania verzií v aplikácii. Po objasnení povolení, upravení popisu aplikácie a začlenení odporúčaní k spôsobu pomenovania verzií bola žiadosť o zlúčenie akceptovaná a aplikácia publikovaná na platforme F-Droid. Na zjednodušenie integrácie aplikácií v platforme používa F-Droid nástroj Fastlane na definovanie popisu, obrázkov a ostatných informácií o aplikácii. Podrobný popis ako postupovať v prípade publikácie aplikácie do platformy F-Droid, je popísané v dokumentácii projektu[10]. Platforma neposkytuje žiadne štatistiky ako napríklad počet inštalácií, na rešpektovanie ochrany súkromia používateľov.

4.4 Audit súkromia

Služba Exodus poskytuje audit súkromia pre Android aplikácie. Daný nástroj detekuje použité analytické nástroje a požadované povolenia v aplikácii. Ak aplikácia rešpektuje súkromie používateľa, tak neobsahuje žiaden analytický nástroj a obsahuje iba nevyhnutné povolenia pre chod aplikácie. V rámci vyhodnotenia som vytvoril audit súkromia s nasledujúcim výsledkom[11]:

- Audit nedetegoval žiadne analytické nástroje.
- Audit detegoval nasledujúce povolenia:

[8] `https://f-droid.org/docs/Anti-Features/` dostupné online: 21. február 2021

[9] `https://gitlab.com/fdroid/fdroiddata/-/merge_requests/8155` dostupné online: 21. február 2021

[10] `https://www.f-droid.org/en/docs/Submitting_to_F-Droid_Quick_Start_Guide/` dostupné online: 21. február 2021

[11] `https://reports.exodus-privacy.eu.org/en/reports/161335/` dostupné online: 18. február 2021

```
Categories:
  - Security
  - System
License: GPL-3.0-or-later
SourceCode: https://github.com/Secure-File-Manager/Secure-File-Manager
IssueTracker: https://github.com/Secure-File-Manager/Secure-File-Manager/issues
Changelog: https://github.com/Secure-File-Manager/Secure-File-Manager/blob/HEAD/CHANGELOG.md

RepoType: git
Repo: https://github.com/Secure-File-Manager/Secure-File-Manager

Builds:
  - versionName: 0.1.2-beta2
    versionCode: 4
    commit: v0.1.2-beta2
    subdir: app
    gradle:
      - yes

AutoUpdateMode: Version v%v
UpdateCheckMode: Tags
CurrentVersion: 0.1.2-beta2
CurrentVersionCode: 4
```

Zdrojový kód 4.2: Ukážka F-Droid súboru s obsahom dát aplikácie

- **FOREGROUND_SERVICE** - Povolenie slúži na používanie služieb na pozadí. Dané povolenie je potrebné pre vytvorenie notifikácií na zamknutia obrazovky a vytvorenie alebo rozbalenie Zip súboru. Viac v kapitole 3.5 a 3.6.3.
- **READ_EXTERNAL_STORAGE** - Povolenie slúži na povolenie prístupu na čítanie externého úložiska. Dané povolenie je potrebné pre chod aplikácie, bez daného povolenia by bola aplikácia nepoužiteľná.
- **RECEIVE_BOOT_COMPLETED** - Povolenie slúži na prijímanie signálu o úspešnom zapnutí zariadenia. Dané povolenie je potrebné na zamknutie aplikácie pri zapnutí zariadenia. Viac v kapitole 3.5.
- **REQUEST_INSTALL_PACKAGES** - Povolenie slúži na inštaláciu aplikácií. Dané povolenie je potrebné pre nainštalovanie aplikácie v prípade, že užívateľ klikne na súbor balíka aplikácie tretej strany v externom úložisku.
- **USE_BIOMETRIC** - Dané povolenie informuje o používaní biometrického hardvéru. Biometrický hardvér sa používa pri biometrickom overení aplikácie, ak je dané povolené. Viac v kapitole 3.5.
- **USE_FINGERPRINT** - Dané povolenie informuje o používaní snímača

odtlačku prsta. Snímač odlačok prsta sa používa pri biometrickom overení aplikácie, ak je dané povolené. Viac v kapitole 3.5.

- **WRITE_EXTERNAL_STORAGE** - Povolenie slúži na povolenie prístupu k zápisu externého úložiska. Dané povolenie je potrebné pre chod aplikácie, bez daného povolenia by bola aplikácia nepoužiteľná.

4.5 Audit závislosti

Nezisková organizácia OWASP vytovrila nástroj *„OWASP Dependency-Check"*[12], ktorý umožňuje vytvoriť audit závislostí s verejne známymi zraniteľnosťami v závislostiach aplikácie. Nástroj je možné použiť ako rozšírenie v nástroji Gradle, ktorý natívne používajú aplikácie Android. Tento nástroj v základnom nastavení hľadá zraniteľnosti vo všetkých závislostiach, vrátane anotačných procesoroch a testovacích knižníc. V rámci auditu je však postačujúci audit knižníc, ktoré sa nachádzajú iba vo výslednej aplikácii. Preto je možné nakonfigurovať Gradle zásuvný modul podľa fragmentu kódu 4.3 na strane 58.

```
dependencyCheck {
    scanConfigurations = configurations.findAll {
        !it.name.startsWithAny('androidTest', 'test', 'debug') &&
            it.name.contains("DependenciesMetadata") && (
            it.name.startsWithAny("api", "implementation", "runtimeOnly") ||
                it.name.contains("Api") ||
                it.name.contains("Implementation") ||
                it.name.contains("RuntimeOnly")
        )
    }.collect {
        it.name
    }
}
```

Zdrojový kód 4.3: Ukážka konfigurácie OWASP Dependency-Check v nástroji Gradle

V rámci auditu nástroj našiel nasledujúce zraniteľnosti v knižnici Kotlin[13]:

- CVE-2020-15824[14]

[12] `https://owasp.org/www-project-dependency-check/` dostupné online: 18. február 2021
[13] `https://kotlinlang.org/api/latest/jvm/stdlib/` dostupné online: 18. február 2021
[14] `https://nvd.nist.gov/vuln/detail/CVE-2020-15824` dostupné online: 18. február 2021

- CVE-2020-29582[15]

Dané zraniteľnosti sú však opravené v knižnici vo verzii 1.4.0, pričom aplikácia používa verziu 1.4.30, a tým pádom sa jedná o falošne pozitívne zraniteľnosti. Po nakonfigurovaní potlačenia daných zraniteľností nástroj nenašiel žiadne ďalšie zraniteľnosti.

4.6 Používateľské testovanie

V rámci používateľského testovania som sa rozhodol použiť chodbové testovanie a testovanie prostredníctvom dotazníka. Jednotlivé výsledky a spôsob testovania sú popísané v jednotlivých kapitolách.

4.6.1 Chodbové testovanie

Chodbové testovanie je druh používateľského testovania, kde sú náhodní ľudia požiadaní o použitie aplikácie alebo služby. V rámci používateľského testovania som vykonal chodbové testovanie na štyroch subjektoch, ktorí mali priemerné technologické znalosti. V rámci testovania som subjekty požiadal o použitie výslednej aplikácie a pýtal sa ich na dojmy z aplikácie i dôvody vykonaných interakcií v aplikácii. Zároveň som kládol otázky, ako vnímali jednotlivé prvky rozhrania pre zistenie, či boli správne pochopené. Taktiež som im zadal nasledujúce úlohy, ktoré mali subjekty vykonať v aplikácii:

- Otvoriť skryté úložisko.
- Zašifrovať ľubovoľný súbor.
- Skryť ľubovoľný súbor.
- Vytvoriť fotku v skrytom úložisku.
- Vytvoriť zašifrovaný Zip súbor.
- Zamknúť aplikáciu.

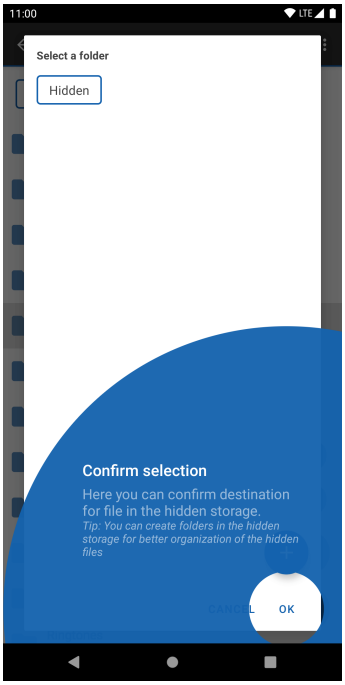
Analýza výsledkov ukázala, že každý z testovaných subjektov zvládol vykonanie úloh úspešne, bez väčších komplikácií a v relatívne krátkom čase. Na základe dojmov subjektov možno konštatovať, že sa im aplikácia ovládala ľahko, bez väčších komplikácii a aplikáciu považovali za užívateľsky použiteľnú. V rámci

[15] `https://nvd.nist.gov/vuln/detail/CVE-2020-29582` dostupné online: 18. február 2021

testovania som však zistil, že testované subjekty nevedeli rozlíšiť rozdiel medzi zašifrovaním súboru a schovaním súboru. Spätnými otázkami som dospel k poznatku, že príčinou toho bola neznalosť technického pojmu *„šifrovanie“*. V rámci vyriešenia tohto problému by bolo vhodné doplniť do aplikácie vysvetlivku, ktorá by vysvetlila technické pojmy a úkony, ktoré v aplikácii je možné vykonať. Vysvetlivku technických pojmov je možné v budúcnosti implementovať v rámci odporúčaných rozšírení.

V rámci chodbového testovania som dostal od subjektov nasledujúce odporúčania, ktoré som následne implementoval do výslednej aplikácie.

- Úprava niektorých anglických slov.
  - Príklad: compress > create, decompress > extract, passwordless > without password
- Zobraziť návod pri prvom skrytí súboru. Ukážku je možné vidieť na obrázku 4.1 na strane 60.
- Ak sa užívateľ nenachádza v skrytom úložisku, zobraziť plávajúce tlačidlo akcie skrývania súborov, ktoré po stlačení zobrazí skryté úložisko. Viac v kapitole 3.2.



Obr. 4.1: Ukážka návodu potvrdenia cieľového adresára skrytia súboru

4.6.2 Testovanie prostredníctvom dotazníka

Pri používateľskom testovaní prostredníctvom dotazníka subjekt najskôr aplikáciu vyskúšal a následne svoje dojmy z aplikácie vyjadril prostredníctvom dotazníka. Zhrnutie dotazníka je možné vidieť v tabuľke 4.1 na strane 61. Dotazník vyplnilo 16 subjektov.

Tabuľka 4.1: Výsledok dotazníku

Tvrdenie / otázka	**Priemerná odpoveď**
Moje počítačové zručnosti	7.26
Môj celkový dojem z aplikácie	8.00
Aplikácia sa mi ovládala ľahko	7.40
Aplikáciu plánujem každodenne používať	5.00
Aplikáciu považujem za užitočnú	8.13

Dotazník som navrhol tak, že subjekt hodnotil mieru súhlasu s tvrdením. Svoj súhlas hodnotil na škále od 1 do 10, kde odpoveď 1 znamenala *„vôbec nesúhlasím“* a odpoveď 10 znamenala *„plne súhlasím“* s tvrdením. Pri hodnotení *„Moje počítačové zručnosti“* subjekt hodnotil svoje počítačové zručnosti, kde odpoveď 1 znamenala *„Pri používaní počítača potrebujem pomoc“* a odpoveď 10 znamenala *„Som odborník v IT“*. Pri možnosti *„Môj celkový dojem z aplikácie“* subjekt hodnotil svoj celkový dojem z aplikácie, kde odpoveď 1 znamenala *„hrozný“* a odpoveď 10 znamenala *„vynikajúci“*.

4.7 Odporúčané rozšírenia aplikácie

Aplikáciu môže ktokoľvek ďalej rozširovať, pretože aplikácia má otvorený zdrojový kód. Jednotlivé odporúčané rozšírenia som rozdelil do podkapitol.

Odstránenie metadát

Súbory okrem vlastného obsahu obsahujú taktiež dáta, ktoré uvádzajú bližšie informácie o danom súbore. V prípade fotky to môže byť názov zariadenia, ktoré vytvorilo danú fotku, alebo GPS polohu, kde bola daná fotografia vytvorená. Dané informácie však nemusia byť žiadané, pretože môžu obsahovať citlivé informácie, čím sa narúša aspekt súkromia. Preto by bolo vhodné do aplikácie doplniť nástroj, ktorý by vedel odstrániť tieto informácie. Podobné nástroje s otvoreným

zdrojovým kódom existujú, napríklad Scrambled Exif[16], alebo ImagePipe[17]. Tieto riešenia však podporujú iba obrázky. Momentálne neexistuje riešenie s otvoreným zdrojovým kódom pre Android zariadenia, ktoré by podporovalo mazanie metadát aj iných súborov ako obrázkov. V operačnom systéme Android je natívne podporené taktiež iba mazanie metadát v obrázkoch[18].

Bezpečné zdieľanie súborov

Väčšina užívateľov zdieľa súbory so známymi. Na zdieľanie je potrebné použiť bezpečný nástroj, aby v rámci zdieľania neboli zdieľané súbory odcudzené neprivilegovanou osobou. Nato je ideálne použiť P2P sieť, ktorá je decentralizovaná a nie je potrebný centralizovaný server, ktorý prináša rôzne bezpečnostné riziká. V rámci zdieľania je potrebné, aby súbor bol zašifrovaný pre odolnosť voči man-in-the-middle útokom. Momentálne neexistuje riešenie pre operačný systém Android, ktoré by podporovalo bezpečné zdieľanie súborov pomocou P2P siete. Avšak existujú riešenia pre vstavané počítače, ako napríklad riešenie Croc[19], ktoré používa password-authenticated key agreement a end-to-end šifrovanie.

Otváranie súborov priamo v aplikácii

Momentálne sú súbory otvárané pomocou aplikácií nainštalovaných v zariadení. Ak napríklad používateľ otvorí obrázok, obrázok je otvorený aplikáciou, ktorá podporuje obrázky. Otváraním súborov aplikáciami tretích strán môže nastať odcudzenie súboru v prípade, že je aplikácia podporujúca daný súbor kompromitovaná. Zároveň má daná aplikácia dočasne prístup k súboru, čím môže dojsť k porušeniu aspektu súkromia. Na eliminovanie popísaného je vhodné do aplikácie implementovať podporu často používaných súborov, ako sú napríklad obrázky, videá a dokumenty. Takto užívateľom budú súbory otvárané priamo v aplikácii, čím sa zamedzí bezpečnostným hrozbám dočasným zdieľaním súboru v inej aplikácií.

Šifrovanie súborov heslom

Aktuálne je možné šifrovať súbory kľúčom uloženým v Android Keystore. Zašifrované súbory nie je možné dešifrovať mimo zariadenia, pretože kľúč z Android

[16] `https://gitlab.com/juanitobananas/scrambled-exif` dostupné online: 19. február 2021

[17] `https://codeberg.org/Starfish/Imagepipe` dostupné online: 19. február 2021

[18] `https://developer.android.com/reference/androidx/exifinterface/media/ExifInterface?hl=en` dostupné online: 19. február 2021

[19] `https://github.com/schollz/croc` dostupné online: 19. február 2021

Keystore nie je možné exportovať. Preto by bolo vhodné pridať možnosť šifrovania súboru pomocou hesla, čím by bolo možné zašifrované súbory dešifrovať aj mimo zariadenia.

Zoznam zašifrovaných súborov

Momentálne nie je v aplikácii ukladané, aké súbory boli zašifrované. To môže byť nevyhovujúce v prípade, ak by používateľ chcel odinštalovať aplikáciu a nepamätal by si cestu ku všetkým súborom, ktoré v minulosti zašifroval aplikáciou. Z tohto dôvodu by bolo vhodné ukladať zoznam zašifrovaných súborov do databázy aplikácie a vhodným spôsobom danú databázu zobrazovať.

Vytvorenie zálohy skrytých súborov

Momentálne v prípade odinštalovania aplikácie je potrebné manuálne odkryť skryté súbory, aby po odinštalovaní aplikácie používateľ o skryté súbory neprišiel. Odkrytím súborov sú dané súbory odhalené v externom úložisku, čím vniká bezpečnostné riziko. Z tohto dôvodu by bolo vhodné pridať do aplikácie vytvorenie zálohy skrytých súborov. Napríklad by aplikácia vytvorila zašifrovaný Zip súbor v externom úložisku. Takto by bolo možné taktiež zašifrovaný Zip súbor so skrytými súbormi premiestniť mimo zariadenia, čím by vznikla záloha skrytých súborov.

Zobrazenie aktuálne dostupného miesta

Momentálne veľkosť skrytého úložiska je obmedzená veľkosťou internej pamäte. Aby mal používateľ prehľad o dostupnom voľnom mieste pre skryté úložisko, prípadne celkovo zariadenie, bolo by vhodné do aplikácie pridať zobrazenie aktuálne dostupného miesta.

Upráva obrázkov pre odolnosť voči nástrojom rozpoznávania tváre

V nedávnej dobe výskumníci vytvorili nástroj[20], ktorý poskytuje 95% a viac ochranu voči systémom rozpoznávania tváre [17]. Tieto systémy sú skutočnou hrozbou pre osobné súkromie. Ochranou voči sledovaniu tohto charakteru môže byť pridanie podpory nástroja úpravy obrázkov pre odolnosť voči systémom rozpoznávania tváre do aplikácie.

[20] `https://github.com/Shawn-Shan/fawkes` dostupné online: 21. február 2021

4.8 Zhrnutie vyhodnotenia

V rámci vyhodnotenia bola aplikácia vyladená pomocou platformy BrowserStack na rôznych zariadeniach. Daným vyladením by mala aplikácia obsahovať menej chýb spojených s kompatibilitou. Aplikácia je publikovaná na platforme Google Play a F-Droid, čím je dostupná širokej verejnosti. V aplikácii bol vykonaný audit súkromia. Na základe výsledku auditu aplikácia z pohľadu ochrany súkromia nezískava od používateľa žiadne údaje a ani osobné informácie. Pri audite závislosti aplikácie boli nájdené 2 bezpečnostné zraniteľnosti, ktoré však boli falošne pozitívne. Pri používateľskom testovaní bolo vykonané chodbové testovanie a testovanie prostredníctvom dotazníka. Výsledok používateľského testovania ukázal, že sa aplikácia ľahko používa a subjekty ju považovali za užitočnú. No užívatelia so slabšou počítačovou gramotnosťou mali problém s pochopením termínov ako *„šifrovanie“*, tým pádom u nich vznikali problémy s používaním aplikácie. Na záver sú rozobraté možné rozšírenia aplikácie, ktoré by bolo vhodné v budúcnosti implementovať na zabezpečenie lepšej použiteľnosti a bezpečnosť aplikácie.

5 Záver

Výsledkom práce je aplikácia zameraná na bezpečnú správu súborov navrhnutá na dodržanie pilierov použiteľnosti, súkromia a bezpečnosti. Práca opisuje aktuálnu kybernetickú bezpečnosť a existujúce riešenie. Obsahuje návrh a implementáciu jednotlivých jednotlivých funkcií zameraných na bezpečnosť a použiteľnosť. V závere práce je výsledná aplikácia vyhodnotená.

Pri analýze aktuálneho súkromia bolo zistené, že aktuálne je zbierané veľké množstvo informácií, ktoré sú zneužívané na marketing a neetické praktiky. Pri analýze aktuálnej kybernetickej bezpečnosti bol zistený zväčšujúci sa výskyt bezpečnostných incidentov, pričom mobilné zariadenia nie sú výnimkou. Pri analýze existujúcich riešení bolo zistené, že momentálne neexistuje Android správca súborov, ktorý by zahŕňal všetky analyzované funkcie. Taktiež bolo zistené, že väčšina existujúcich riešení nemá užívateľský použiteľné rozhranie.

V rámci návrhu implementácie bol vyhotovený návrh implementácie funkcie skrývania súborov, šifrovania súborov, manažmentu dát aplikácie, overenia prístupu aplikácie, integrity súborov, zákazu snímky obrazovky, podpory šifrovaných Zip súborov, určenie miesta uloženia mediálnych súborov, čistenie dočasnej pamäte, úvodnej obrazovky aplikácie a návodu aplikácie. Tento návrh bol vyhotovený pre dosiahnutie čo najväčšieho bezpečia a použiteľnosti aplikácie. V rámci návrhu je taktiež popísaný spôsob implementácie jadra aplikácie správcu súborov.

Pri kapitole implementácie je popísaný podrobný postup implementácie jednotlivých funkcií a výslednej aplikácie. Táto implementácia je založená nevyhotovení návrhu implementácie. Aplikácia je implementovaná s použitím dizajnérskeho jazyka Material Design.

Pri vyhodnotení aplikácie bola aplikácia otestovaná na rôznych zariadeniach prostredníctvom platformy BrowserStack pre minimalizovanie chýb spojených s kompatibilitou. V záujme dosiahnutia čo najväčšej dostupnosti aplikácie širokej verejnosti bola aplikácia publikovaná na platforme Google Play a F-Droid. Na základe výsledku závislostí je evidentné, že aplikácia je bezpečná v zmysle pou-

žitých závislostí. Audit našiel 2 bezpečnostné zraniteľnosti, ktoré však boli falošne pozitívne. Pri audite súkromia bolo zistené, že aplikácia berie ohľad na ochranu súkromia používateľa. Vzhľadom na výsledok používateľského testovania je aplikácia ľahko použiteľná. V závere hodnotenia boli uvedené možné rozšírenia aplikácie.

V rámci štatistík platformy Google play, kde je aplikácia publikovaná, mala aplikácia v čase písania 214 inštalácií, z toho 55 aktívnych zariadení. Aplikácia dostala hodnotenie 4.583/5 od 12 používateľov, z toho 5 hodnotenia boli hodnotenia s recenziou. V rámci štatistík platformy Github, kde je publikovaný zdrojový kód aplikácie, má aplikácia 42 hviezd a 4 rozvetvenia kódu (*forks*), 1 žiadosť o spojenie kódu (*merge request*), 20 tiketov (*issues*), z čoho je 13 otvorených a 7 uzavretých tiketov. Z otvorených tiketov je 11 tiketov so žiadosťou o implementáciu novej funkcie aplikácie. Bohužiaľ, platforma F-Droid, na ktorej je aplikácia taktiež publikovaná, neposkytuje žiadne štatistiky pre rešpektovanie ochrany súkromia používateľov.

Medzi hlavné výhody aplikácie považujem:

- riešenie s otvoreným zdrojovým kódom,
- aktuálne unikátne riešenie,
- riešenie zadarmo s ochranou súkromia používateľa,
- riešenie bez zbytočných povolení,
- používateľsky ľahko ovládateľné rozhranie.

Medzi hlavné nevýhody aplikácie považujem:

- aplikácia je nová, môže obsahovať chyby a bezpečnostné zraniteľnosti,
- nad aplikáciou nebol vykonaný bezpečnostný audit,
- zašifrované súbory nie je možné dešifrovať mimo zariadenia.

Literatúra

1. SOLOVE, Daniel J. 'I've Got Nothing to Hide' and Other Misunderstandings of Privacy. *GWU Law School Public Law Research Paper*. 2007, roč. 44, č. 289. Dostupné tiež z: `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=998565`.
2. ZUBOFF, Shoshana. *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power*. First Trade Paperback Edition. PublicAffairs, 2019. ISBN 9781610395694.
3. SNOWDEN, Edward. *Permanent Record*. Metropolitan Books, 2019. ISBN 9781250237231.
4. ZINKUS, M.; JOIS, T. M.; GREEN, M. Data Security on Mobile Devices: Current State of theArt, Open Problems, and Proposed Solutions. In: 2021. Dostupné tiež z: `https://securephones.io/main.pdf`.
5. SAMANI, R.; DAVIS, G. *McAfee Mobile Threat Report*. 2019. Dostupné tiež z: `https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2019.pdf`. Technická správa.
6. POINT, Check. *Cyber Security Report 2020*. 2020. Dostupné tiež z: `https://www.ntsc.org/assets/pdfs/cyber-security-report-2020.pdf`. Technická správa.
7. CISCO. *Cisco Cybersecurity Report: 2019 Threat Report*. 2019. Dostupné tiež z: `https://engage2demand.cisco.com/LP=14878?CCID=cc000160&DTID=odicdc000016&OID=ebksc015215`. Technická správa.
8. OLENČIN, M.; PERHÁČ, J. Automated configuration of a Linux web server security. In: *2019 IEEE 15th International Scientific Conference on Informatics*. 2019, s. 000491–000496.
9. *GNU General Public License*. 2007. Verzia 3. Dostupné tiež z: `http://www.gnu.org/licenses/gpl.html`.

10. MATYAS, Vashek; RIHA, Zdenek. Security of Biometric Authentication Systems. *2010 International Conference on Computer Information Systems and Industrial Management Applications, CISIM 2010*. 2010, roč. 3. Dostupné z DOI: 10.1109/CISIM.2010.5643698.

11. WEI, Michael; GRUPP, Laura M.; SPADA, Frederick E.; SWANSON, Steven. Reliably Erasing Data from Flash-Based Solid State Drives. In: *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*. San Jose, California: USENIX Association, 2011, s. 8. FAST'11. ISBN 9781931971829.

12. HEIDERICH, M.; LARSSON, J.; PERAGLIE, R.; FÄSSLER, F.; MORITZ, S.; KEAN, C. *Pentest-Report Mullvad Apps, Clients & API 05.2020*. 2020. Dostupné tiež z: https://cure53.de/pentest-report_mullvad_2020_v2.pdf.

13. AUMASSON, Jean-Philippe. *Serious Cryptography: A Practical Introduction to Modern Encryption*. No Starch Press, 2017. ISBN 9781593278267.

14. BIRYUKOV, A.; DINU, D.; KHOVRATOVICH, D. Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications. In: *2016 IEEE European Symposium on Security and Privacy (EuroS P)*. 2016. Dostupné z DOI: 10.1109/EuroSP.2016.31.

15. DANG, Quynh H. Recommendation for Applications Using Approved Hash Algorithms. *NIST Pubs*. 2012. Dostupné tiež z: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=911479.

16. XIE, Tao; LIU, Fanbao; FENG, Dengguo. *Fast Collision Attack on MD5* [Cryptology ePrint Archive, Report 2013/170]. 2013. https://eprint.iacr.org/2013/170.

17. SHAN, Shawn; WENGER, Emily; ZHANG, Jiayun; LI, Huiying; ZHENG, Haitao; ZHAO, Ben Y. Fawkes: Protecting Personal Privacy against Unauthorized Deep Learning Models. In: *Proc. of USENIX Security*. 2020.

18. RAINS, Tim. *Cybersecurity Threats, Malware Trends, and Strategies: Mitigate exploits, malware, phishing, and other social engineering attacks*. Packt Publishing, 2020. ISBN 9781800206014.

19. COLETTA, Alberto; VAN DER VEEN, Victor; MAGGI, Federico. DroydSeuss: A Mobile Banking Trojan Tracker - Short Paper. In: *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2016. Lecture Notes in Computer Science (LNCS).

20. ESKANDARI, S.; LEOUTSARAKOS, A.; MURSC, T.; CLARK, J. A First Look at Browser-Based Cryptojacking. In: *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. 2018, s. 58–66.

21. TANG, C.; CHEN, S.; FAN, L.; XU, L.; LIU, Y.; TANG, Z.; DOU, L. A Large-Scale Empirical Study on Industrial Fake Apps. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 2019, s. 183–192.

22. NIST. *Secure Hash Standard (SHS)*. 2015. Dostupné tiež z: `http://dx.doi.org/10.6028/NIST.FIPS.180-4`.

23. BELLARE, Mihir; POINTCHEVAL, David; ROGAWAY, Phillip. Authenticated Key Exchange Secure against Dictionary Attacks. In: PRENEEL, Bart (ed.). *Advances in Cryptology — EUROCRYPT 2000*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. ISBN 978-3-540-45539-4.

24. HUMAYUN, M.; JHANJHI, N.; ALSAYAT, A.; PONNUSAMY, V. Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*. 2020. ISSN 1110-8665. Dostupné z DOI: `https://doi.org/10.1016/j.eij.2020.05.003`.

25. *Oxford English Dictionary*. 3rd edition. Oxford University Press, 2010. ISBN 9780199571123.

Zoznam skratiek

AES Advanced Encryption Standard.

API Application Programming Interface.

CPU Central Processing Unit.

eMMC embedded Multi-Media Controller.

GCM Galois/Counter Mod.

GPS Global Positioning System.

GPU Graphics Processing Unit.

IMEI International Mobile Equipment Identity.

IP Internet Protocol.

MAC Media Access Control.

NIST National Institute of Standards and Technology.

OTG On-The-Go.

OWASP Open Web Application Security Project.

P2P Peer To Peer.

PIN Personal Identification Number.

RAM Random-Access Memory.

SHA Secure Hash Algorithms.

UFS Universal Flash Storage.

USA United States of America.

WWAN Wireless Wide Area Network.

Slovník

Android KeyStore je v operačnom systéme Android úložisko na ukladanie kryptografických kľúčov v kontajneri, ktoré je možné ťažšie extrahovať zo zariadenia. Kryptografické kľúče sú ukladané v bezpečnom module hardvéru (*Strongbox*) v zariadeniach, ktoré majú bezpečný modul hardvéru a podporujú verziu Android 9 a novšie.

backdoor (*v preklade z anglického jazyka „zadné vrátka“*) je škodlivý softvér, ktorý umožňuje útočníkovi obísť bežnú autentifikáciu [18].

banking trojan (*v preklade z anglického jazyka „bankový trójsky kôň“*) je typ trójského koňa, ktorý je vyvinutý na účel ukradnutia informácií z bankového softvéru [19].

cryptojacking je škodlivý softvér, ktorý na pozadí zariadenia ťaží kryptomeny [20].

end-to-end šifrovanie je šifrovaný typ komunikácie, v ktorom môžu správy čítať iba komunikujúci používatelia.

enkapsulácia (*z anglického slova „encapsulation“*) je v objektovo orientovanom programovaní označenie prístupu tried, metód a členských premenných.

falošné aplikácie (*z anglického slova „fake apps“*) je škodlivý softvér, ktorý sa tvári ako legitímna aplikácia [21].

kryptografická transformačná funkcia (*z anglického slova „cryptographic hash function“*) je matematický jednosmerný kryptografický algoritmus, ktorý zmapuje dáta do fixnej dĺžky na overenie autenticity alebo integrity správy [22].

lámanie hesla (*z anglického slova „cracking password“*) je proces obnovy hesla z transformovanej formy, väčšinou skúšaním metódou útoku hrubou silou.

man-in-the-middle je typ kybernetického útoku, kde útočník tajne odpočúva, prípadne aj mení komunikáciu medzi dvoma stranami, ktoré sa domnievajú, že komunikujú priamo.

password-authenticated key agreement je metóda pre dve alebo viac strán na vytvorenie kryptografických kľúčov na základe znalosti hesla jednej alebo viacerých strán [23].

ransomware je škodlivý softvér, ktorý zašifruje dáta obete privátnym kľúčom útočníka, čo znemožní prístup obete k jej dátam. Útočník žiada výkupné od obete za privátny kľúč, prevažne prostredníctvom kryptomeny [24].

slovníkový útok (*z anglického slova „dictionary attack“*) je typ útoku hrubou silou na prelomenie hesla pomocou vopred pripraveného slovníka obsahujúceho heslá získané z predošlých únikov, alebo zoznamu slov v určitom jazyku.

tapjacking útok je typ útoku, kde škodlivá aplikácia sleduje ako užívateľ interaguje so zariadenia pomocou úderov, čím môžu byť odcudzené citlivé informácie, ako napríklad heslo.

trojan (*v preklade z anglického jazyka „trójsky kôň“, slangovo „trojan“*) je škodlivý softvér, ktorý vykonáva deštruktívnu činnosť [18].

whistleblower je v ponímaní tejto práce zamestnanec, ktorý ohlási nelegálnu, nemorálnu, neetickú činnosť, alebo iný druh negatívnej činnosti zamestnávateľa, a to najme vládneho orgánu alebo orgánu činného v trestnom stíhaní [25].

útok hrubou silou (*z anglického slova „brute-force“*) je typ útoku, kedy sa útočník snaží prelomiť zabezpečenie, napríklad zabezpečenie heslom, skúšaním rôznych kombinácií.

Zoznam príloh

Príloha A CD médium – záverečná práca v elektronickej podobe

Príloha B Používateľská príručka

Príloha C Systémová príručka

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Používateľská príručka

Príloha B

2021 **Bc. Michal Olenčin**

Obsah

Zoznam obrázkov

1 Funkcia aplikácie

Aplikácia slúži ako správca súborov so zameraním na bezpečnosť. Aplikácia je zadarmo, s otvoreným zdrojovým kódom, bez reklám, bez zbytočných povolení s ohľadom na súkromie používateľa. Medzi hlavné funkcie aplikácie patrí:

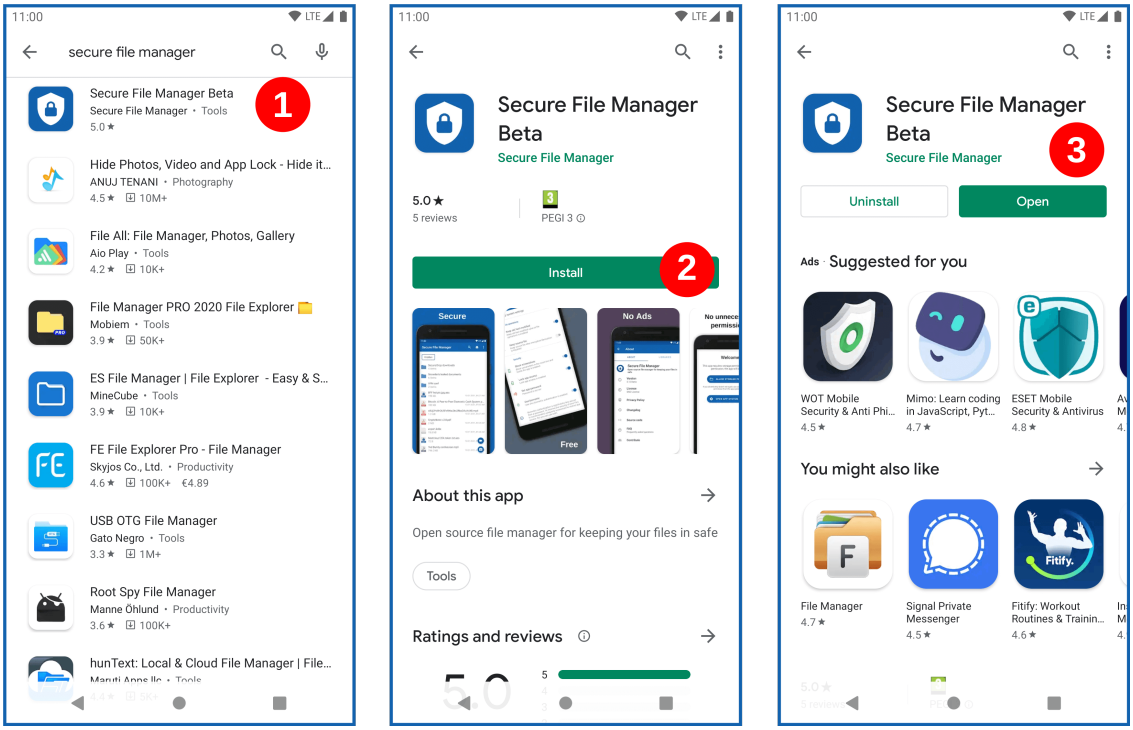
- prehľadávanie súborov,
- presúvanie súborov,
- kopírovanie súborov,
- premenovanie súborov,
- zobrazenie informácii o súbore,
- skrývanie súborov
- šifrovanie súborov,
- vytvorenie alebo rozbalenie šifrovaného a nešifrovaného Zip súboru,
- overenie prístupu,
- možnosť vypnutia snímky obrazovky,
- možnosť vypnutia vytvárania náhľadov,
- čistenie dočasnej pamäte náhľadov,
- určenie miesta uloženia mediálneho súboru,
- vytvorenie kontrolného súčtu súboru.

2 Inštalácia aplikácie

V kapitole je popísaná inštalácia aplikácie prostredníctvom platformy Google Play a F-Droid. Taktiež sú v kapitole popísané požiadavky na aplikáciu.

2.1 Inštalácia prostredníctvom platformy Google Play

Na obrázku 2.1 na strane 2 sú zobrazené postupnosti krokov, ako postupovať pri inštalácii aplikácie prostredníctvom platformy Google Play.



Obr. 2.1: Inštalácia prostredníctvom platformy Google Play

- **Bod 1**: Vyhľadajte v aplikácii Google Play aplikáciu *„Secure File Manager“*.
- **Bod 2**: Po kliknutí na aplikáciu stlačte tlačidlo *„Inštalovať“*.
- **Bod 3**: Inštalácia chvíľu trvá, počkajte prosím. Po nainštalovaní aplikácie je možné aplikáciu používať.

2.2 Inštalácia prostredníctvom platformy F-Droid

Na obrázku 2.2 na strane 4 sú zobrazené postupnosti krokov, ako postupovať pri inštalácii aplikácie prostredníctvom platformy F-Droid.

- **Bod 1**: Vyhľadajte v aplikácii F-Droid aplikáciu *„Secure File Manager“*.
- **Bod 2**: Po kliknutí na aplikáciu stlačte tlačidlo *„Inštalovať“*.
- **Bod 3**: Počkajte, kým sa aplikácia stiahne a následne stlačte tlačidlo *„Inštalovať“* v okne, ktoré sa vám automaticky otvorí po stiahnutí aplikácie.
- **Bod 4**: Inštalácia chvíľu trvá, počkajte prosím. Po nainštalovaní aplikácie je možné aplikáciu používať.

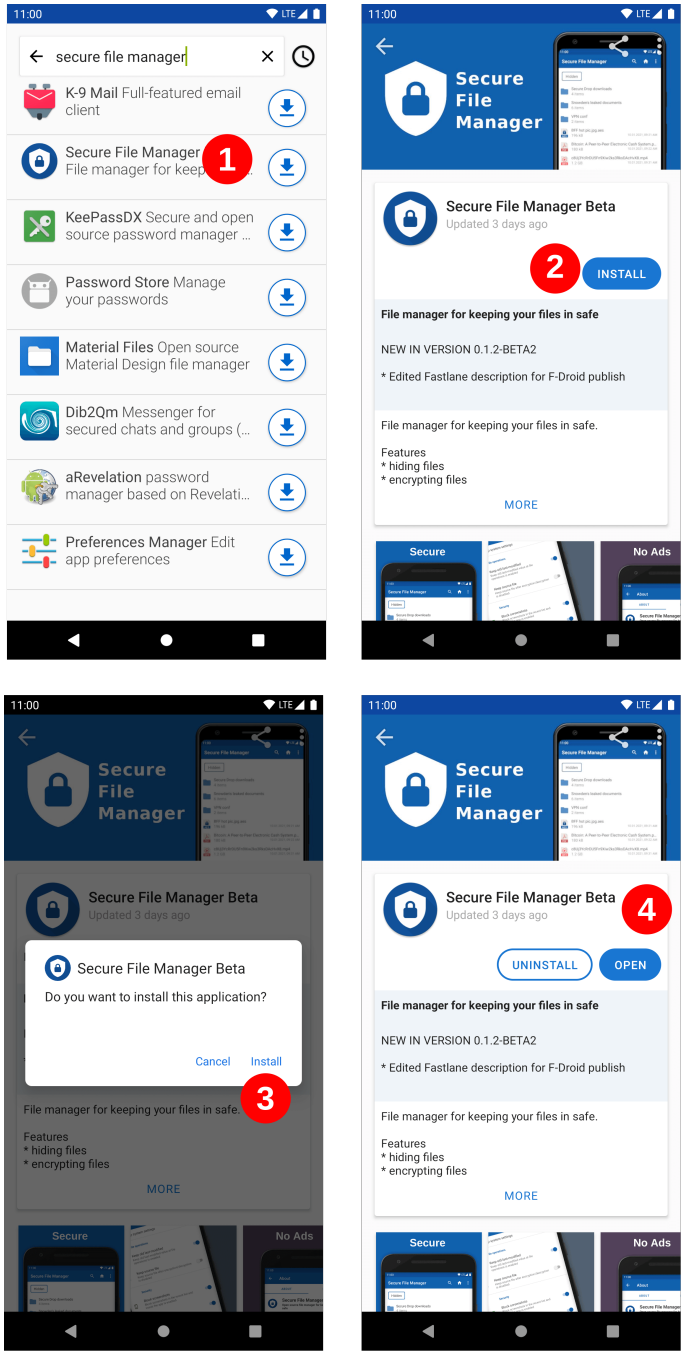
2.3 Požiadavky na zariadenie

Na spustenie aplikácie je potrebné, aby zariadenie spĺňalo nasledujúce požiadavky:

- operačný systém Android vo verzii 8.0 a novšie,
- voľnú pamäť minimálne 5MB.

Na zariadenie sa nevzťahujú technické parametre ako minimálna veľkosť pamäte RAM, alebo výkon CPU. Odporúča sa, aby zariadenie obsahovalo:

- biometricky hardvér ako napríklad čítačka odtlačkov prstov,
- kamera,
- bezpečný modul hardvéru.



Obr. 2.2: Inštalácia prostredníctvom platformy F-Droid

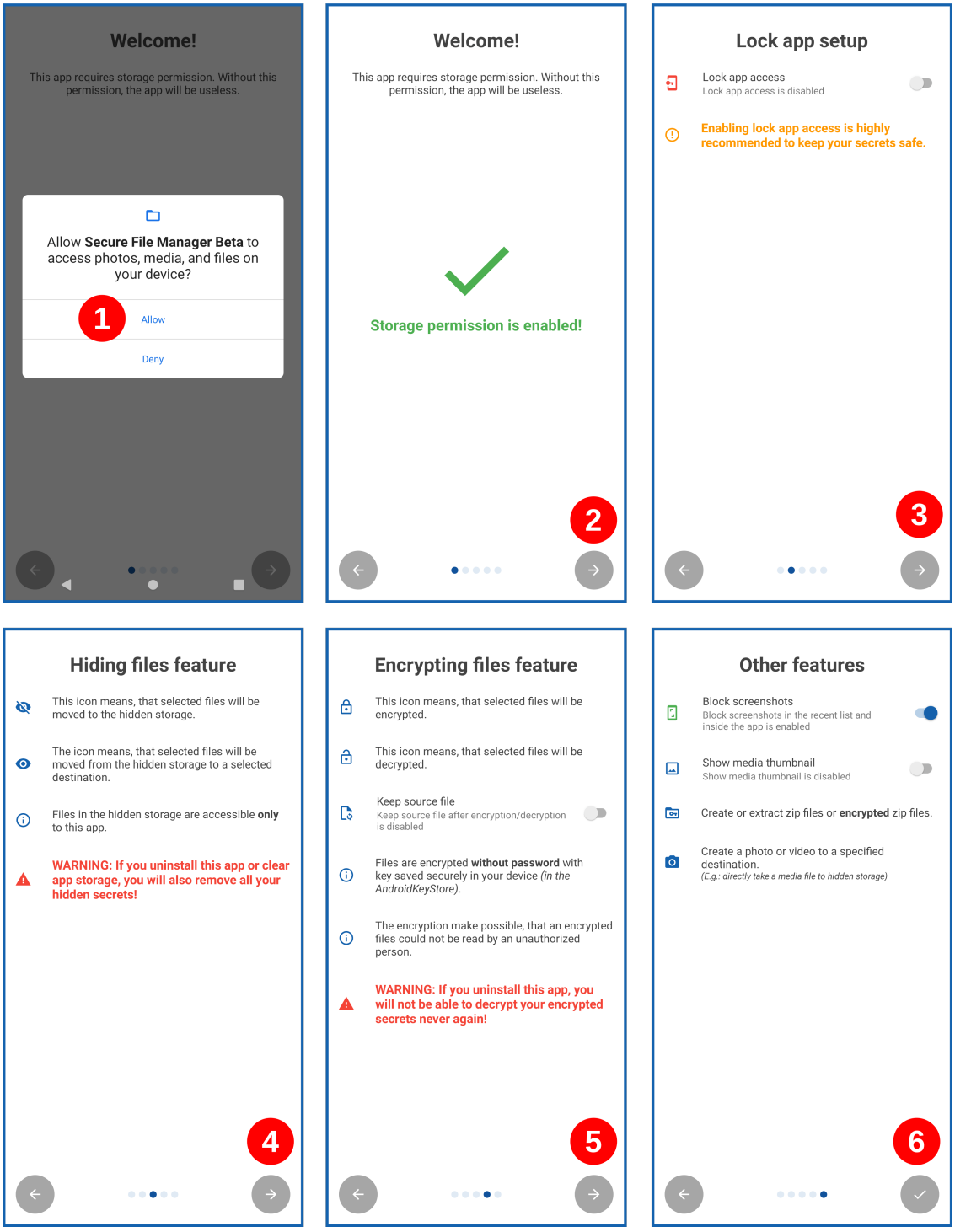
3 Použitie aplikácie

Kapitola je rozdelená na jednotlivé podkapitoly. V prvej podkapitole je popísané prvé spustenie a následne sú podkapitoly rozdelené podľa použitia jednotlivých funkcií. Jednotlivé postupy a texty sa môžu mierne líšiť na základe verzie operačného systému Android a predvoleného jazyka operačného systému.

3.1 Prvé spustenie aplikácie

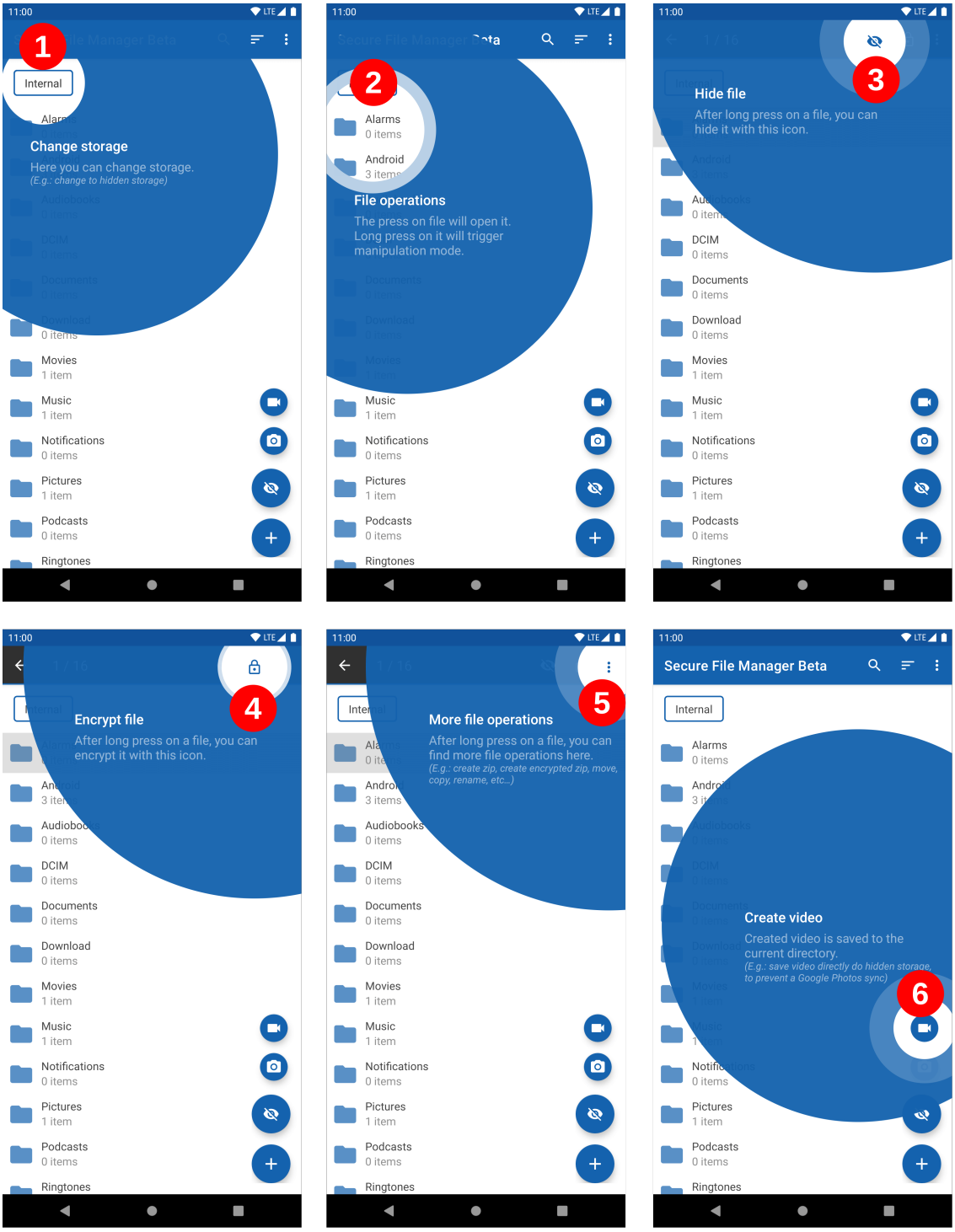
V prípade prvého spustenia aplikácie je užívateľ privítaný uvítacou obrazovkou aplikácie. Na obrázku 3.1 na strane 6 je možné vidieť postupnosť krokov, ako postupovať v rámci uvítacej obrazovke.

- **Bod 1**: Pre fungovanie aplikácie je potrebné potvrdiť povolenie prístupu k externému úložisku. Kliknite na tlačidlo *„povoliť“*.
- **Bod 2**: Kliknite na tlačidlo so šípkou ďalej.
- **Bod 3**: V prípade, že si želáte nastaviť overenie prístupu, môžete si ho nastaviť v tomto pohľade. Odporúča sa nastaviť overenie prístupu. Následne kliknite na tlačidlo so šípkou ďalej.
- **Bod 4**: Prečítajte si informácie zobrazené na obrazovke a kliknite na tlačidlo so šípkou ďalej.
- **Bod 5**: Prečítajte si informácie zobrazené na obrazovke. Na obrazovke si môžete nastaviť zachovanie pôvodného súboru po operácii šifrovania. Následne kliknite na tlačidlo so šípkou ďalej.
- **Bod 6**: Prečítajte si informácie zobrazené na obrazovke. Na obrazovke si môžete nastaviť snímok obrazovky a vytváranie náhľadov. Následne kliknite na tlačidlo so šípkou ďalej.

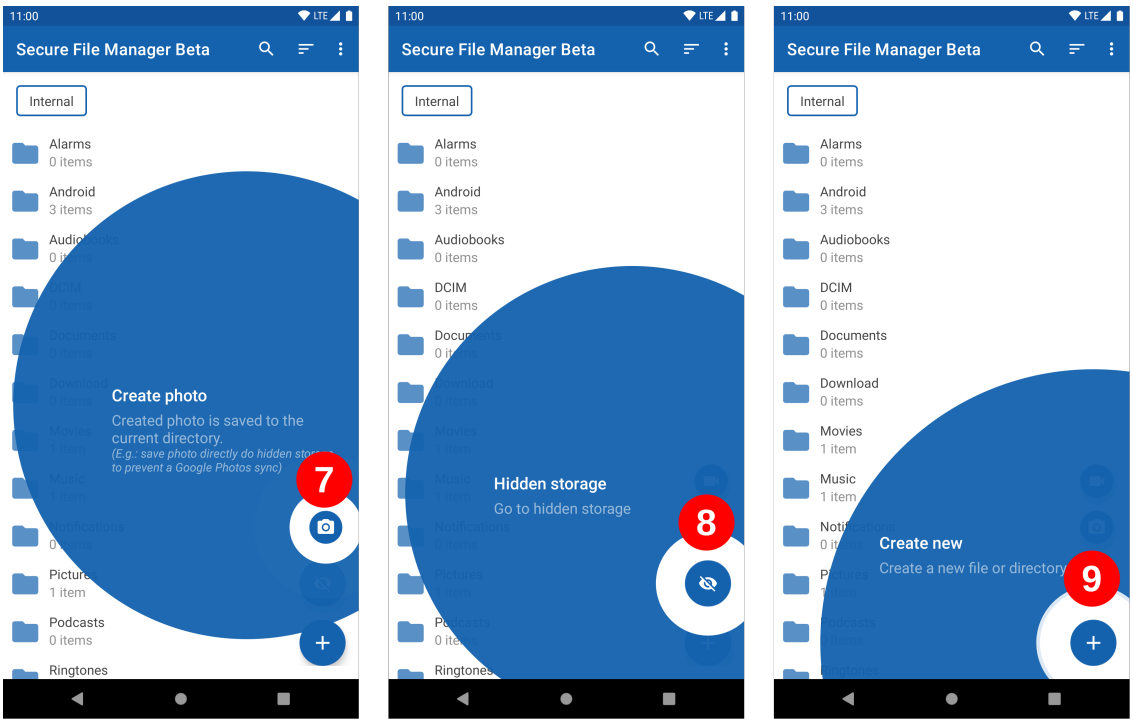


Obr. 3.1: Uvítacia obrazovka

Po zobrazení uvítacej obrazovky sa objaví návod. Na obrázku 3.2 na strane 7 a 3.3 na strane 8 sú zobrazené postupnosti krokov, ako postupovať v rámci návodu.



Obr. 3.2: Návod časť 1

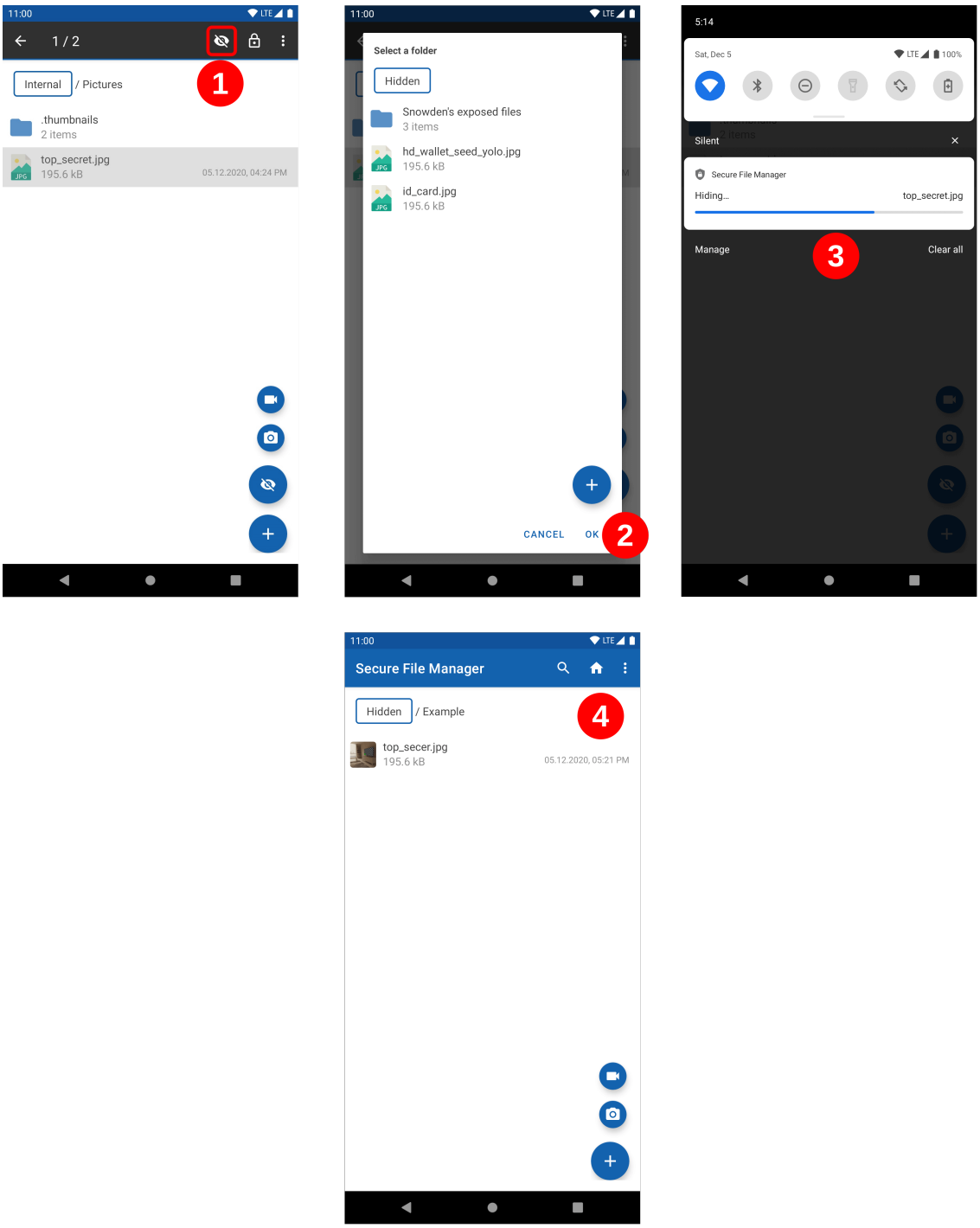


Obr. 3.3: Návod časť 2

3.2 Skrývanie súborov

Na obrázku 3.4 na strane 9 sú zobrazené postupnosti krokov, ako postupovať pri skrytí súborov.

- **Bod 1**: Zvoľte súbory, ktoré si želáte skryť a následne kliknite na ikonu skrytia súborov.
- **Bod 2**: Zvoľte priečinok v skrytom úložisku, do ktorého si želáte uložiť skryté súbory a následne kliknite na tlačidlo „*OK*“.
- **Bod 3**: Aktuálny stav skrývania súborov môžete sledovať cez notifikáciu.
- **Bod 4**: Po dokončení skrývania súborov sú skryté súbory dostupné v skrytom úložisku vo zvolenom priečinku.

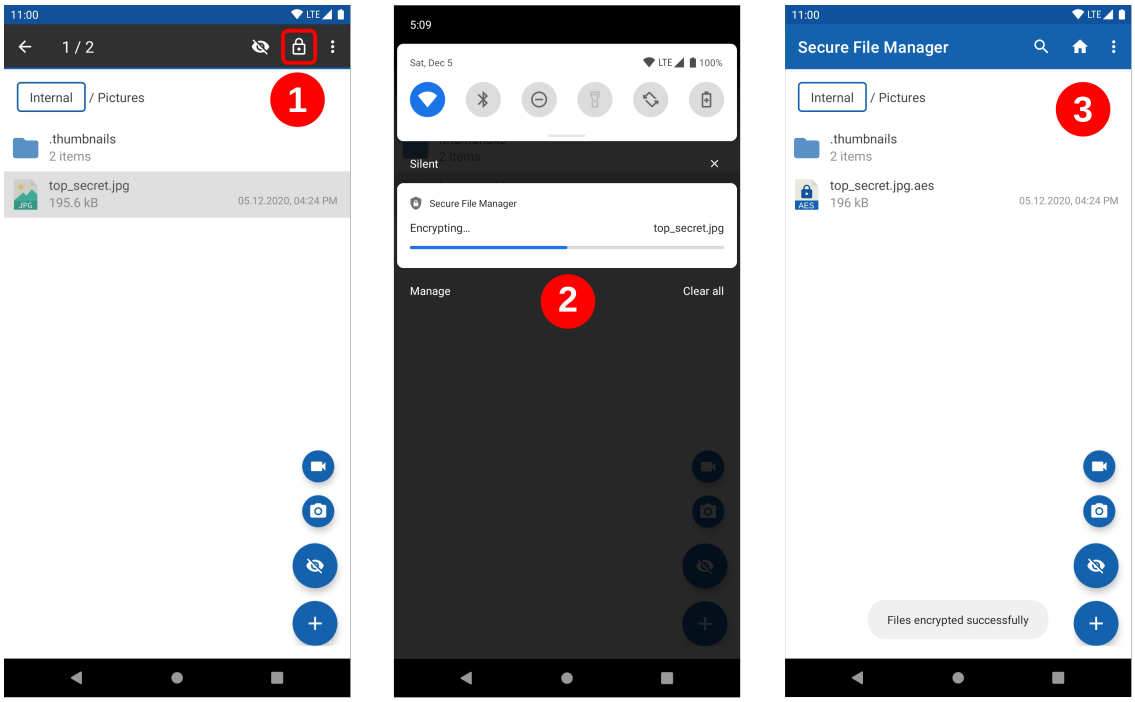


Obr. 3.4: Skrývanie súborov

3.3 Šifrovanie súborov

Na obrázku 3.5 na strane 10 sú zobrazené postupnosti krokov, ako postupovať pri šifrovaní súborov.

- **Bod 1**: Zvoľte súbory, ktoré si želáte zašifrovať/dešifrovať a následne kliknite na ikonu šifrovania/dešifrovania.
- **Bod 2**: Aktuálny stav šifrovania/dešifrovania môžete sledovať cez notifikáciu.
- **Bod 3**: Po dokončení šifrovania/dešifrovania sú zašifrované/dešifrované súbory dostupné.



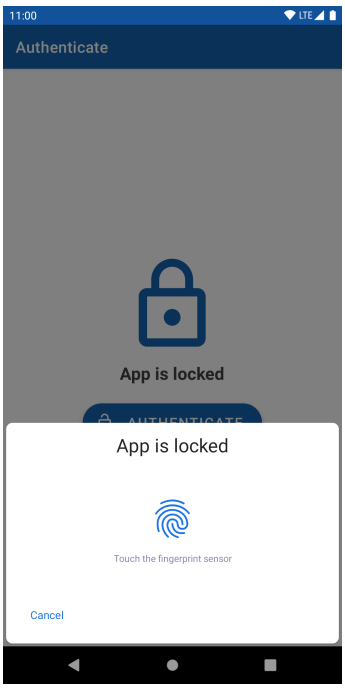
Obr. 3.5: Šifrovanie súborov

3.4 Overenie prístupu

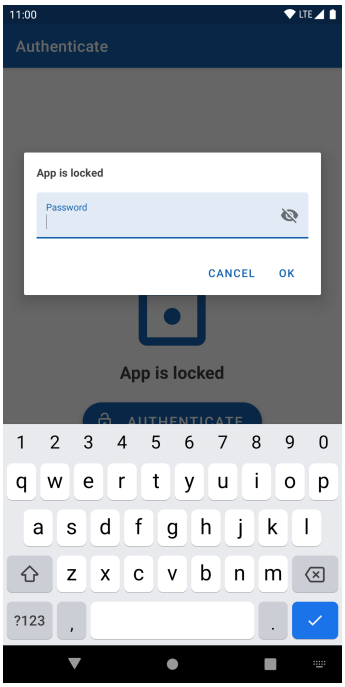
Overenie prístupu záleží od nastavenia aplikácie. V prípade, že je overenie prístupu nastavené cez biometrické overenie, je zobrazená obrazovka ako na obrázku 3.6 na strane 12. Pre sprístupnenie aplikácie je potrebné v tomto prípade sa biometricky overiť na základe nastavení biometrického overenia v operačnom systéme. V prípade, že je overenie prístupu nastavené na heslo, je zobrazená obrazovka ako na obrázku 3.7 na strane 12. Na sprístupnenie aplikácie je potrebné v tomto prípade sa overiť zadaním hesla. V prípade, že je overenie prístupu nastavené na heslo a zároveň na biometrické overenie, je zobrazená obrazovka ako na obrázku 3.8 na strane 13. Na sprístupnenie aplikácie je potrebné v tomto prípade sa biometricky overiť na základe nastavení biometrického overenia v operačnom systéme. Ak sa chce užívateľ overiť heslom, je potrebné stlačiť tlačidlo *„Použiť heslo“*. Po kliknutí na tlačidlo sa zobrazí obrazovka ako je na obrázku 3.7 na strane 12.

Po otvorení je aplikácia v odomknutom stave. Na zmenu stavu na zamknutý stav je potrebné vykonať jeden z nasledujúcich krokov:

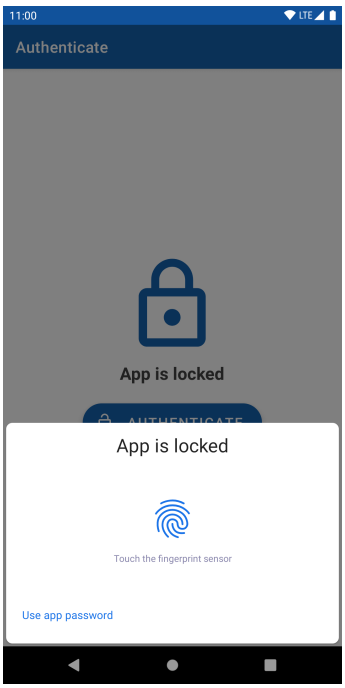
- ukončiť aplikáciu,
- zamknúť obrazovku,
- Zamknúť aplikáciu cez notifikáciu, ako je zobrazené na obrázku 3.9 na strane 13.



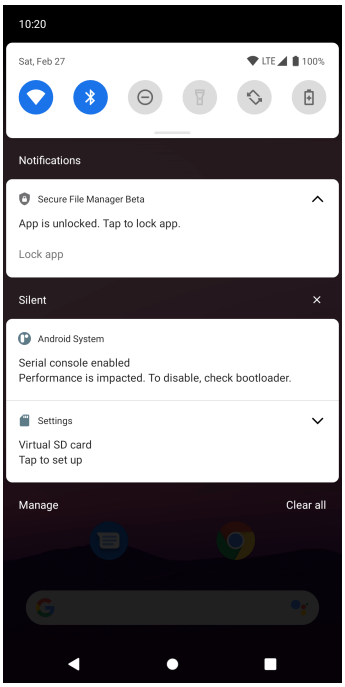
Obr. 3.6: Biometrické overenie prístupu



Obr. 3.7: Overenie prístupu heslom



Obr. 3.8: Kombinované overenie

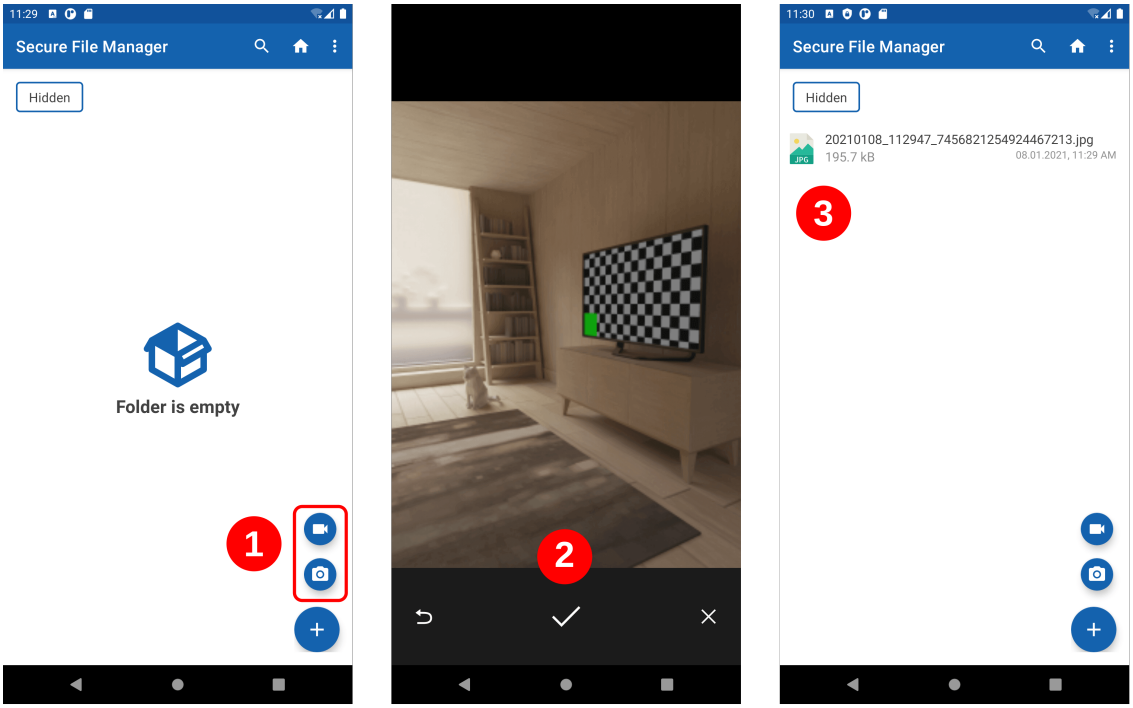


Obr. 3.9: Notifikácia zamknutia aplikácie

3.5 Určenie miesta uloženia mediálneho súboru

Na obrázku 3.10 na strane 14 sú zobrazené postupnosti krokov, ako postupovať pri určení miesta uloženia mediálneho súboru.

- **Bod 1**: Otvorte priečinok, do ktorého si želáte vytvoriť mediálny súbor. V prípade, že si želáte vytvoriť video, kliknite na ikonu videa. V prípade, že si želáte vytvoriť fotku, kliknite na ikonu fotoaparátu.
- **Bod 2**: Otvorí sa vám aplikácia, ktorá podporuje vytvorenie mediálneho súboru. V prípade, že vaše zariadenie obsahuje viacero aplikácií podporujúcich vytvorenie mediálneho súboru, je potrebné zvoliť aplikáciu, pomocou ktorej si želáte vytvoriť mediálny súbor. Po vytvorení mediálneho súboru potvrďte uloženie mediálneho súboru.
- **Bod 3**: Mediálny súbor je uložený vo zvolenom priečinku.

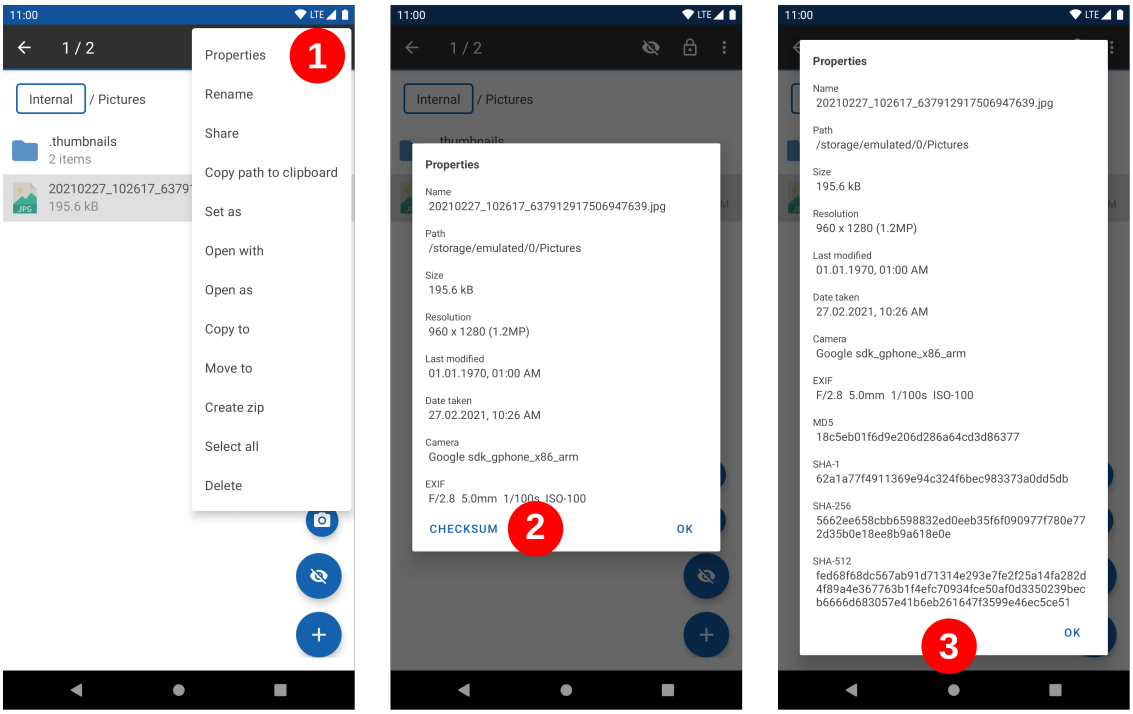


Obr. 3.10: Určenie miesta uloženia mediálneho súboru

3.6 Zobrazenie kontrolného súčtu

Na obrázku 3.11 na strane 15 sú zobrazené postupnosti krokov, ako postupovať pri zobrazení kontrolného súčtu.

- **Bod 1**: Zvoľte súbor, ku ktorému si želáte vytvoriť kontrolný súčet. Následne cez menu zvoľte *„Vlastnosti“*.
- **Bod 2**: Po zvolení sa zobrazia vlastnosti súboru. Kliknite na *„Kontrolný súčet“*.
- **Bod 3**: Po kliknutí sa zobrazí kontrolný súčet súboru.

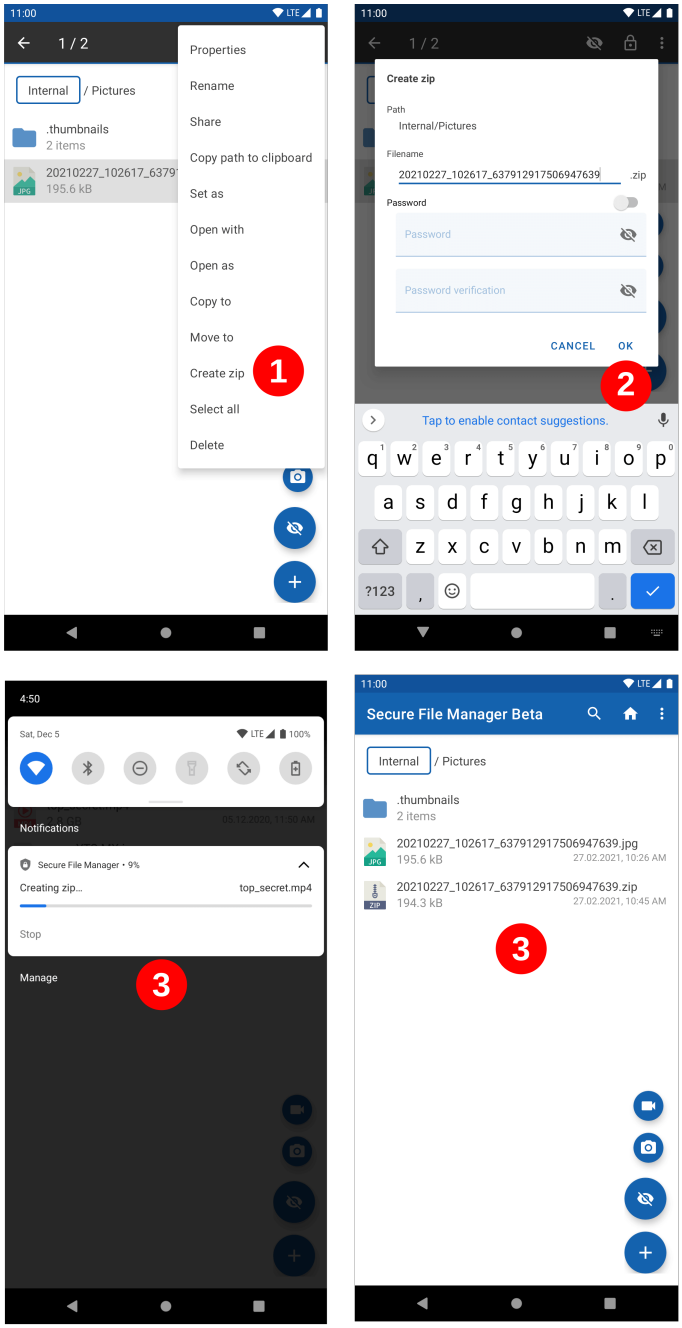


Obr. 3.11: Zobrazenie kontrolného súčtu

3.7 Vytvorenie Zip súboru

Na obrázku 3.12 na strane 17 sú zobrazené postupnosti krokov, ako postupovať pri vytvorení Zip súboru.

- **Bod 1**: Zvoľte súbory, ktoré si želáte začleniť do Zip súboru. Následne cez menu zvoľte *„Vytvoriť Zip“*.
- **Bod 2**: V nasledujúcej obrazovke si môžete definovať názov súboru, prípadne heslo k Zip súboru. Na vytvorenie Zip súboru potvrďte krok stlačením *„OK“*.
- **Bod 3**: Aktuálny stav vytvárania Zip súboru môžete sledovať cez notifikáciu.
- **Bod 4**: Po vytvorení Zip súboru, je súbor dostupný v aktuálnom priečinku.



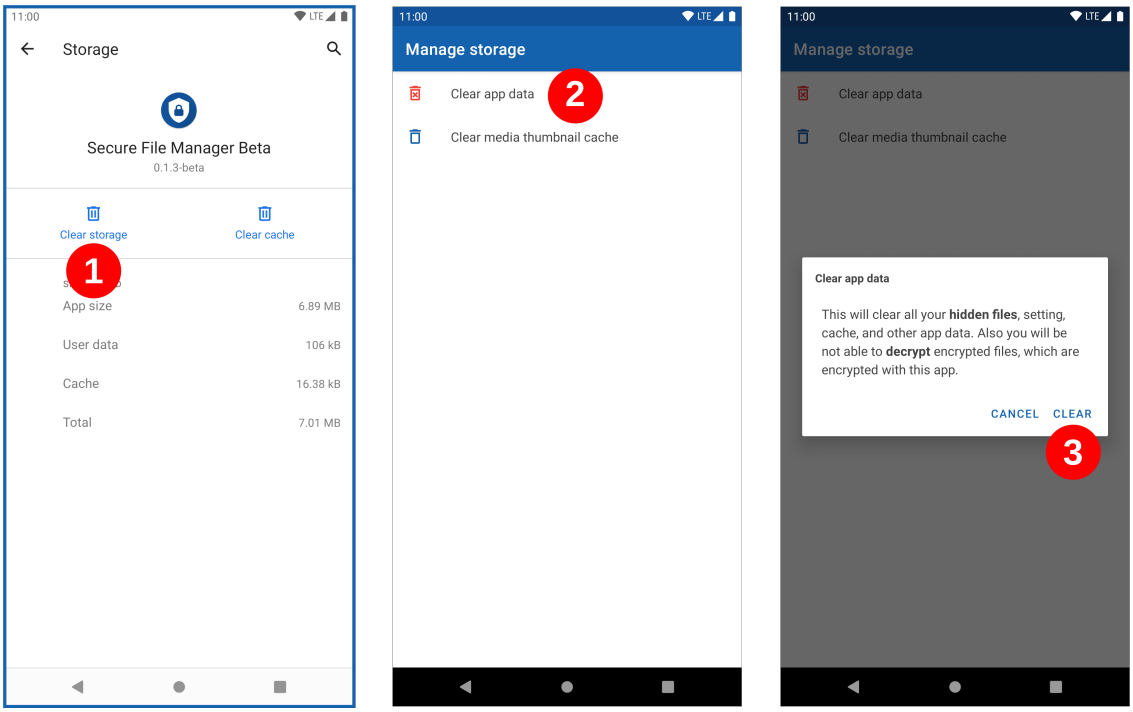
Obr. 3.12: Vytvorenie Zip súboru

3.8 Zmazanie dát aplikácie

Na obrázku 3.13 na strane 18 sú zobrazené postupnosti krokov, ako postupovať pri zmazaní dát aplikácie.

- **Bod 1**: V nastaveniach úložiska aplikácie zvoľte *„Vyčistiť úložisko“*.
- **Bod 2**: Následne kliknite na *„Vyčistiť dáta aplikácie“*.

- **Bod 3**: Prečítajte si zobrazené upozornenie. Ak si naozaj želáte zmazať dáta aplikácie, kliknite 5-krát na tlačidlo *„Vyčistiť“*.



Obr. 3.13: Zmazanie dát aplikácie

4 Obmedzenia aplikácie

Aplikácia uchováva skryté súbory v úložisku aplikácie. Aplikácia taktiež uchováva kryptografické kľúče k zašifrovaným súborom v Android Keystore. V prípade odinštalovania aplikácie sú odstránené všetky údaje v úložisku aplikácie vrátane skrytých súborov a kryptografických kľúčov. Z tohto dôvodu je potrebné pred odinštalovaním aplikácie dešifrovať všetky zašifrované súbory aplikáciou a odkryť všetky skryté súbory, aby sa predišlo neželanému zmazaniu citlivých súborov a kryptografických kľúčov.

Aplikácia nemá podporu OTG zariadení, z tohto dôvodu nie je možné pracovať v aplikácii s OTG zariadeniami.

Aplikácia nereaguje na interakcie užívateľa v prípade, že užívateľ má nainštalovanú aplikáciu tretej strany podporujúcu nočné svetlo. V prípade, že je nočné svetlo zapnuté, aplikácia nemusí reagovať na interakcie užívateľa z dôvodu prevencie voči tapjacking útokom. Ak si želáte používať nočné svetlo, použite prosím natívnu funkciu nočného svetla, ak to vaše zariadenie podporuje.

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Systémová príručka

Príloha C

2021 **Bc. Michal Olenčin**

Obsah

Zoznam zdrojových kódov

1 Funkcia aplikácie

Aplikácia slúži ako správca súborov so zameraním na bezpečnosť. Aplikácia je zadarmo, s otvoreným zdrojovým kódom, bez reklám, bez zbytočných povolení s ohľadom na súkromie používateľa. Medzi hlavné funkcie aplikácie patrí:

- prehľadávanie súborov,
- presúvanie súborov,
- kopírovanie súborov,
- premenovanie súborov,
- zobrazenie informácii o súbore,
- skrývanie súborov
- šifrovanie súborov,
- vytvorenie alebo rozbalenie šifrovaného a nešifrovaného Zip súboru,
- overenie prístupu,
- možnosť vypnutia snímky obrazovky,
- možnosť vypnutia vytvárania náhľadov,
- čistenie dočasnej pamäte náhľadov,
- určenie miesta uloženia mediálneho súboru,
- vytvorenie kontrolného súčtu súboru.

2 Popis aplikácie

Aplikácie je postavená na jadre aplikácie *„Simple File Manager“*[1]. Jadro aplikácie používa knižnicu *„Simple Commons“* na pomocné funkcie a tým je jadro silno previazané s touto knižnicou. Aplikácia je implementovaná tak, že všetky závislosti na knižnici *„Simple Commons“* boli presunuté priamo do aplikácie. Kód aplikácie je zverejnený na platforme Github[2].

2.1 Popis štruktúry súborov

Štruktúra zdrojových súborov aplikácie je nasledujúca:

- `/app` - priečinok obsahujúci súbory aplikácie
- `/app/src` - priečinok obsahujúci zdrojové súbory aplikácie
- `/app/src/debug` - priečinok obsahujúci zdrojové súbory aplikácie pre ladenie
- `/app/src/main` - priečinok obsahujúci zdrojové súbory aplikácie pre produkčnú verziu
- `/assets` - priečinok obsahujúci doplnkové súbory
- `/fastlane` - priečinok obsahujúci súbory pre nástroj Fastlane
- `/gradle` - priečinok obsahujúci súbory pre nástroj Gradle

[1] `https://github.com/SimpleMobileTools/Simple-File-Manager`
dostupné online: 28. február 2021

[2] `https://github.com/Secure-File-Manager/Secure-File-Manager`
dostupné online: 28. február 2021

2.2 Popis štruktúry balíka aplikácie

Aplikácia sa nachádza v balíku s názvom `com.securefilemanager.app`. Tento balík má nasledujúcu štruktúru:

- `activities` - balík obsahuje triedy, ktoré implementujú aktivity
- `adapters` - balík obsahuje triedy, ktoré implementujú návrhový vzor Adapter
- `asynctasks` - balík obsahuje triedy, ktoré dedia z triedy AsyncTask
- `dialog` - balík obsahuje triedy, ktoré obsahujú triedy dialógových okien
- `extensions` - balík obsahujúci metódy, ktoré rozširujú definície špecifických tried
- `fragments` - balík obsahuje triedy, ktoré implementujú fragmenty
- `fragments.intro` - balík obsahuje triedy, ktoré implementujú fragmenty úvodnej obrazovky
- `fragments.settings` - balík obsahuje triedy, ktoré implementujú fragmenty nastavení
- `helpers` - balík obsahuje pomocné triedy
- `helpers.crypto` - balík obsahuje pomocné kryptografické triedy
- `interfaces` - balík obsahuje rozhrania
- `models` - balík obsahuje triedy modelov
- `observers` - balík obsahuje triedy, ktoré implementujú návrhový vzor Observer
- `receivers` - balík obsahuje triedy, ktoré dedia triedu BroadcastReceiver
- `services` - balík obsahuje triedy, ktoré dedia triedu Service
- `views` - balík obsahuje triedy, ktoré implementujú triedy rozhrania

2.3 Popis funkcií

V kapitole sú popísané jednotlivé hlavné funkcie aplikácie.

2.3.1 Skrývanie súborov

Skrývanie súborov v aplikácii funguje na základe presúvania súborov z externého úložiska na interné úložisko aplikácie. Na presun súborov je použitá natívna knižnica `InputStream` . Podrobný spôsob práce s touto knižnicou je popísaný v dokumentácii knižnice[3]. Pri skrývaní súborov je súbor uložený do adresára `.hidden` v koreni interného úložiska aplikácie. V prípade, že užívateľ vytvoril adresár v skrytom úložisku, adresár sa nachádza v adresári `.hidden` . Pre operáciu presunu súboru sa používa trieda `CopyMoveTask` .

2.3.2 Šifrovanie súborov

Šifrovanie súborov je implementované pomocou knižnice `androidx.security.crypto` . Podrobný spôsob práce s touto knižnicou je popísaný v dokumentácii knižnice[4]. Daná knižnica používa na šifrovanie súborov Android Keystore[5]. Tým je dosiahnutie šifrovanie pomocou kryptografických kľúčov bez potreby zadania hesla pri šifrovaní súboru. Na šifrovanie súborov sa používajú kľúče definované v triede `FileCrypto` . Táto trieda sa nachádza v balíku `com.securefilemanager.app.helpers.crypto` , ktorý slúži na prácu s kryptografickými operáciami. Na získanie šifrovaného vstupného a výstupného prúdu aplikácia používa metódy vo fragmente kódu 2.1 na strane 6. Pre operáciu šifrovania súboru sa používa trieda `CopyMoveTask` .

2.3.3 Manažment dát aplikácie

Pre nastavenie manažmentu dát aplikácie je potrebné v manifeste aplikácie zakázať mazať užívateľské dáta aplikácie atribútom `android:allowClearUserData` a nastaviť cestu k aktivite manažmentu dát atribútom

[3] `https://developer.android.com/reference/java/io/InputStream`
dostupné online: 13. marec 2021

[4] `https://developer.android.com/reference/androidx/security/crypto/package-summary`
dostupné online: 13. marec 2021

[5] `https://developer.android.com/training/articles/keystore`
dostupné online: 13. marec 2021

```
fun Context.getEncryptedFile(file: File): EncryptedFile =
    EncryptedFile.Builder(
        this,
        file,
        FileCrypto.getKey(this),
        FileCrypto.ENCRYPTION_SCHEME
    ).build()

fun Context.getFileInputEncryptedStreamSync(path: String)
    : FileInputStream =
        getEncryptedFile(File(path)).openFileInput()


fun Context.getFileOutputEncryptedStreamSync(file: File)
    : FileOutputStream =
        getEncryptedFile(file).openFileOutput()
```

Zdrojový kód 2.1: Fragment kódu pre získanie šifrovaného prúdu

`android:manageSpaceActivit`. Podrobný popis atribútov je v dokumentácii operačného systému Android[6]. Aktivita manažmentu dát `ManageStorageActivity` vkladá do svojho pohľadu fragment konfigurácie dát aplikácie `SettingsManageStorageFragment`. Na vymazanie dát aplikácie je použitý fragment kódu 2.2 na strane 7, ktorý nad správcom balíkov spustí vyčistenie dát aplikácie.

2.3.4 Overenie prístupu

Pre overenie prístupu sa používa biometrické overenie a/alebo overenie heslom. Pre biometrické overenie je použitá knižnica `androidx.biometric`. Podrobný spôsob práce s touto knižnicou je popísaný v dokumentácii knižnice[7]. Pri overení hesla je heslo transformované pomocou transformačnej funkcie Argon2. Na prácu s touto transformačnou knižnicou je použitá knižnica `Argon2Kt`. Podrobný spô-

[6] `https://developer.android.com/guide/topics/manifest/application-element` dostupné online: 13. marec 2021

[7] `https://developer.android.com/jetpack/androidx/releases/biometric` dostupné online: 13. marec 2021

```
fun Activity.deleteAppData() {
    val packageName = this.applicationContext.packageName
    Runtime.getRuntime().exec("pm clear $packageName")
    this.quitApp()
}

fun Activity.quitApp(canLock: Boolean = true) {
    if (canLock) {
        this.appLock()
    }
    this.finishAffinity()
}
```

Zdrojový kód 2.2: Fragment kódu vymazania dát aplikácie

sob práce s touto knižnicou je popísaný v dokumentácii knižnice[8]. Na overenie prístupu aplikácia používa aktivitu `AuthenticationActivity`. Stav overenia je ukladaný do predvolieb aplikácie. Heslo je ukladané po transformácii do šifrovaných predvolieb aplikácie. Pri práci s šifrovanými predvoľbami je použitá knižnica
`androidx.security.crypto`. Na zmenu stavu na zamknutý stav je potrebné, aby nastala jedna z možností:

- **Zamknutie pomocou notifikácie**. Pre dané je v aplikácii služba na pozadí `UnlockAppService`.
- **Zamknutie pomocou zamknutím a zapnutím zariadenia**. Pre dané je v aplikácii trieda `LockReceiver`, ktorá pri odochytení akcie `ACTION_LOCKED_BOOT_COMPLETED` a `ACTION_SCREEN_OFF` zmení stav aplikácie na stav zamknutý.
- **Zamknutie v prípade ukončenia aplikácie**. V prípade ukončenia aplikácie je pomocou metódy `onDestroy` zmenený stav na zamknutý.
- **Zamknutie v prípade chyby v aplikácii**. Pre dané je v aplikácii pri vytvorení aktivity nastavené odchytávanie chýb pomocou triedy `Thread`, konkrétne pomocou metódy `setDefaultUncaughtExceptionHandler`

[8] `https://github.com/lambdapioneer/argon2kt` dostupné online: 13. marec 2021

2.3.5 Integrita súborov

Na výpočet kontrolného súčtu je použitá natívna knižnica MessageDigest. Podrobný spôsob práce s touto knižnicou je popísaný v dokumentácii knižnice[9]. Na výpočet kontrolného súčtu je v aplikácii možné použiť fragment kódu 2.3 na strane 8, ktorý podporuje transformačné funkcie MD5, SHA-1, SHA-256 a SHA-512.

```
fun File.getDigest(algorithm: String): String =
    this.inputStream().use { fis ->
        val md = MessageDigest.getInstance(algorithm)
        val buffer = ByteArray(8192)
        generateSequence {
            when (val bytesRead = fis.read(buffer)) {
                -1 -> null
                else -> bytesRead
            }
        }.forEach { bytesRead -> md.update(buffer, 0, bytesRead) }
        md.digest().joinToString("") { "%02x".format(it) }
    }

fun File.md5(): String = this.getDigest(MD5)
fun File.sha1(): String = this.getDigest(SHA1)
fun File.sha256(): String = this.getDigest(SHA256)
fun File.sha512(): String = this.getDigest(SHA512)
```

Zdrojový kód 2.3: Fragment kódu výpočtu kontrolného súčtu

2.3.6 Zakázanie snímky obrazovky

Pre zakázanie snímky obrazovky je použitá značka `FLAG_SECURE`[10]. Zákazanie snímky je možné konfigurovať v nastaveniach. Na základe konfigurácie je pridaná do pohľadu značka `FLAG_SECURE`. Danú značku je potrebné pridať pri vy-

[9] `https://developer.android.com/reference/java/security/MessageDigest?hl=en`
dostupné online: 13. marec 2021

[10] `https://developer.android.com/reference/android/view/WindowManager.LayoutParams`
dostupné online: 13. marec 2021

tvorení aktivity. Nato slúži funkcia pridania tejto značky, ktorú je možné vidieť na fragmente kódu 2.4 na strane 9.

```
fun Activity.addFlagsSecure() {
    if (this.config.disableScreenshots) {
        this.window.setFlags(
            WindowManager.LayoutParams.FLAG_SECURE,
            WindowManager.LayoutParams.FLAG_SECURE
        )
    }
}
```

Zdrojový kód 2.4: Fragment kódu pridania značky FLAG_SECURE

2.3.7 Kompresia a dekompresia Zip súborov

Pri práci so Zip súbormi je použitá knižnica *„Zip4j“*. Podrobný spôsob práce s touto knižnicou je popísaný v dokumentácii knižnice [11].

V aplikácii je daná knižnica používaná v službe `ZipManagerService`. Táto služba sa využíva pri práci so Zip súbormi, konkrétne pri kompresii a dekompresii súborov. Služba je implementovaná ako služba na pozadí pri použití notifikácie na oznámenie aktuálneho stavu.

2.3.8 Úvodná obrazovka

Úvodná obrazovka je implementovaná pomocou knižnice *„AppIntro“*. Podrobný spôsob práce s touto knižnicou je popísaný v dokumentácii knižnice[12].

Aplikácia využíva túto knižnicu v rámci aktivity *„IntroActivity“*. Jednotlivé pohľady sú implementované ako fragmenty z dôvodu použitia vlastného dizajnu. Implementácia jednotlivých fragmentov pre úvodnú obrazovku sa nachádza v balíku `com.securefilemanager.app.fragments.intro`. Tieto fragmenty do seba zapúzdrujú fragmenty nastavení využitím polymorfizmu pre prehľadnosť z konzistenciu kódu.

[11] `https://github.com/srikanth-lingala/zip4j` dostupné online: 8. január 2021
[12] `https://github.com/AppIntro/AppIntro` dostupné online: 8. január 2021

2.3.9 Návod

Návod je implementovaný pomocou knižnice *„TapTargetView“*. Podrobný spôsob práce s touto knižnicou je popísaný v dokumentácii knižnice[13].

Pre prácu s knižnicou je v aplikácii vytvorená pomocná trieda `TapTargetTutorial`, ktorá slúži na vytváranie jednotlivých cieľov v rámci návodu a združuje ich do zoznamu. Vo fragmente kódu 2.5 na strane 10 je možné vidieť prácu s touto pomocnou triedou a knižnicou.

```
val tapTarget = TapTargetTutorial(this)
TapTargetSequence(activity)
                    .targets(tapTarget.getTutorialTapTargets(cancellable))
                    .listener(tapTarget.getTutorialListener())
                    .start()
```

Zdrojový kód 2.5: Ukážka použitia pomocnej triedy pre vytvorenie návodu

[13] `https://github.com/KeepSafe/TapTargetView` dostupné online: 8. január 2021

3 Naviazanosť na knižnice

V tejto kapitole sú popísané knižnice, na ktoré je aplikácia naviazaná. Kapitoly sú rozdelené na naviazanosť na natívne knižnice a na knižnice tretích strán.

3.1 Naviazanosť na natívne knižnice

Aplikácia je naviazaná na natívne knižnice Kotlin a Android.

Kotlin

Aplikácia je napísaná v programovacom jazyku Kotlin. Z tohto dôvodu aplikácia používa nasledujúce natívne knižnice:

- `org.jetbrains.kotlin:kotlin-stdlib-jdk8` - štandardná Kotlin knižnica pre JVM 8
- `org.jetbrains.kotlinx:kotlinx-coroutines-core` - jadro knižnice Kotlin na podporu asynchrónnych operácii
- `org.jetbrains.kotlinx:kotlinx-coroutines-android` - knižnica Kotlin na podporu asynchrónnych operácii v operačnom systéme Android

Android

Aplikácia je vytvorená pre operačný systém Android. Z tohto dôvodu aplikácia používa nasledujúce natívne Android KTX Jetpack knižnice:

- `androidx.activity:activity-ktx:` - knižnica pre prácu s aktivitami
- `androidx.appcompat:appcompat` - knižnica pre spätnú kompatibilitu
- `androidx.biometric:biometric` - knižnica pre autentifikáciu pomocou biometrie

- `androidx.cardview:cardview` - knižnica pre prácu s kartovým Material dizajnom
- `androidx.constraintlayout:constraintlayout` - knižnica pre prácu s ConstraintLayout
- `androidx.core:core-ktx` - jadro KTX knižnice
- `androidx.documentfile:documentfile` - knižnica pre prácu s DocumentFile
- `androidx.exifinterface:exifinterface` - knižnica pre čítanie a zápis EXIF tagov
- `androidx.fragment:fragment-ktx` - knižnica pre prácu s fragmentami
- `androidx.lifecycle:lifecycle-process` - knižnica pre podporu procesov v životnom cykle
- `androidx.localbroadcastmanager:localbroadcastmanager` - knižnica pre podporu zberne udalostí
- `androidx.preference:preference-ktx` - knižnica pre vytvorenie obrazovky preferencií
- `androidx.recyclerview:recyclerview` - knižnica pre podporu RecyclerView
- `androidx.security:security-crypto` - knižnica pre podporu kryptografických operácií
- `androidx.swiperefreshlayout:swiperefreshlayout` - knižnica pre podporu SwiperefreshLayout

3.2 Naviazanosť na knižnice tretích strán

Aplikácia používa nasledujúce knižnice tretích strán:

- `com.google.android.material:material` - knižnica podporujúca Material komponenty pre Material dezajn
- `com.github.AppIntro:AppIntro` - knižnica pre vytvorenie úvodnej obrazovky

- `com.lambdapioneer.argon2kt:argon2kt` - knižnica pre podporu algoritmu Argon2
- `com.getkeepsafe.taptargetview:taptargetview` - knižnica pre vytvorenie návodu
- `com.github.srikanth-lingala:zip4j` - knižnica pre kompresiu a dekompresiu Zip súborov
- `com.mikepenz:aboutlibraries-core` - jadro knižnice pre zobrazenie *„O nás“*
- `com.mikepenz:aboutlibraries` - knižnica pre zobrazenie *„O nás“*
- `com.github.bumptech.glide:glide` - knižnica pre prácu s mediálnymi súbormi