Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikan koulutusohjelma

KÄNNYKÄN KERÄÄMÄT TIEDOT

Jori Kosonen

Työn tarkastaja(t): Tutkijaopettaja (TkT) Jouni Ikonen

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikan koulutusohjelma
Jori Kosonen

Kännykän keräämät tiedot
Kandidaatintyö 2021

41 sivua, 7 kuvaa, 2 taulukkoa
Työn tarkastajat: Tutkijaopettaja (TkT) Jouni Ikonen

Hakusanat: Android, Älypuhelin, tiedonkeräys, käyttäjätiedot
Keywords: Android, Smartphone, data acquisition, user data

Nykyään melkein jokainen ihminen omistaa kännykän, ja tyypillisesti kantaa sitä mukanaan liikkuessaan muiden ihmisten parissa. Kännykkään jää käyttäjästä aina jonkinlainen jälki käyttäjän toimista ja tapahtumista. Globaali COVID-19 virusepidemia on johtanut tilanteeseen, jossa on tarve selvittää viruksen leviämisketjuja. Tässä kanditaatintyössä tarkastellaan, millaista dataa puhelin oikein tallentaa käyttäjästä, hänen kohtaamistansa kontakteista ja vierailluista paikoista. Lisäksi tarkastellaan tiedonhankintamenetelmiä, ja jo olemassa olevia kaupallisia ja avoimen lähdekoodin työkaluja, jotka hakevat sekä koostavat käyttäjätietoja tarkasteltavaan muotoon. Lopuksi toteutettiin kirjallisuuskatsauksen pohjalta opitun tiedon perusteella yksinkertainen sovellus, joka demonstroi miten käyttäjätiedoista voidaan koostaa aikajana.

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Degree Programme in Software Engineering
Jori Kosonen

Information collected by the smartphone
Bachelor's Thesis 2021

41 pages, 7 figures, 2 tables
Examiner: Associate Professor Jouni Ikonen

Keywords: Android, Smartphone, data acquisition, user data

Nowadays almost everyone has a smartphone, and typically carries with them when interacting with other people. Smartphone always tracks the user and leaves some trace of this interaction when they use their device. Global COVID-19 pandemic has led to situation, where there is a need to track the spread of the virus. In this work we investigate, what data smartphone stores about their user, the people the person meets, and places the user visits. On top of that we investigate methods of acquiring this data, and already existing commercial and open-source tools for acquiring and presenting user data. Lastly, we developed a simple application to demonstrate how to present acquired user data as a timeline.

ALKUSANAT

Työ on tehty Lappeenrannassa Lappeenrannan–Lahden Teknillisessä yliopistossa. Haluisin kiittää ystäviä ja kaikkia niitä, jotka ovat tämän työn tekemisessä tarjonneet tukeaan ja varsinkin työn ohjaajaa Jouni Ikosta, joka kärsivällisesti auttoi pitkäksi venyneen työn valmiiksi saattamisessa.

SISÄLLYSLUETTELO

SYMBOLI- JA LYHENNELUETTELO

ACM	Association for Computing Machinery
ADB	Android debug bridge
API	Application programming interface
DELTA	Data Extraction and Logging Tool for Android
IEEE	Institute of Electrical and Electronics Engineers
JTAG	Joint Test Action Group
LiME	Linux Memory Extractor
OS	Operating System
PDF	Portable Document Format
RAM	Random Access Memory
SD	Secure Digital
TWRP	Team Win Recovery Project
URI	Uniform Resource Identifier
USB	Universal Serial Bus

1 JOHDANTO

Tässä luvussa esitellään kandidaatintyön aiheelle tausta, määritellään työn tavoitteet ja rajaukset, sekä lopuksi käydään läpi työn rakenne.

1.1 Tausta

Nykyään melkein jokainen ihminen omistaa kännykän, ja tyypillisesti kantaa sitä mukanaan liikkuessaan muiden ihmisten parissa. Se on mukana useimmissa vuorovaikutuksissa ja nykyisin sillä pystyy jopa maksamaan ostokset kaupassa, tarvitsematta ottaa lompakkoa esille. Näistä vuorovaikutuksista yleensä jää jonkinlainen jälki käyttäjän puhelimeen, kuten maksutapahtuma ostoksista, paikkamerkintä vierailluista paikoista tai esimerkiksi valokuva jostain nähtävyydestä tai arkisesta asiasta.

Globaali COVID-19 virusepidemia on johtanut tilanteeseen, jossa on tarve selvittää viruksen leviämisketjuja. Leviämisketjujen selvittäminen on yleensä suuri ja vaativa urakka, koska jokaisesta ketjusta pitää selvittää kaikki pienetkin yksityiskohdat uusien tartuntojen välttämiseksi. Nykyisistä tartuntaketjujen selvittämiskeinoista ei ole vielä kauheasti tutkimustietoa saatavilla, ja varmasti keinot vaihtelevat maitten välillä. Yksi varmasti käytössä oleva menetelmä tartuntakeinojen selvittämiseen on haastattelut, joissa tartunnan saaneilta kysytään missä he ovat liikkuneet, ja keiden kanssa he ovat olleet kanssakäymisissä. Ihminen on kuitenkin erehtyväinen, eikä välttämättä arkisista tapahtumista välttämättä jää muistiin selkeää muistijälkeä, joten muistia ei voida pitää luotettavana tietolähteenä ja leviämisketjuista saattaa jäädä olennaisia linkkejä puuttumaan. Tässä tilanteessa on selvästi tarvetta luotettavalle paikka- ja kanssakäymistietojen keräämismenetelmälle.

1.2 Tavoitteet ja rajaukset

Tässä kanditaatintyössä tarkastellaan, mihin ja missä muodossa Android tallentaa käyttäjätietoja, millaista dataa puhelin oikein tallentaa käyttäjästä, hänen kohtaamistansa kontakteista ja vierailluista paikoista. Lisäksi tarkastellaan loogisia ja fyysisiä tiedonhankintamenetelmiä, ja jo olemassa olevia kaupallisia ja avoimen lähdekoodin

työkaluja, jotka hakevat sekä koostavat käyttäjätietoja tarkasteltavaan muotoon. Työn tavoitteena on selvittää mitä tietoja puhelin tallentaa, mihin tieto tallentuu ja missä muodossa, sekä miten tähän työhön päästään käsiksi ja millä työkaluilla. Selvitetään myös, onko mahdollista toteuttaa yksinkertainen sovellus, joka on muodostettu läpikäydyistä avoimen lähdekoodin työkaluista, ja joka kasaisi yksityiskohtaisen aikajanan käyttäjän kohtaamista kontakteista ja paikkatiedoista. Aikajanan on tarkoitus toimia työkaluna muistuttamaan käyttäjää liikkumisistaan ja tapaamisistaan, ja näin ollen auttaa esimerkiksi viranomaisia viruksen leviämisketjujen selvittämisessä. Aihe on laaja, resursseja rajallisesti ja työkaluja on paljon, joten ei keskitytä tekemään kokonaista toimivaa sovellusta. Työkalujen testaus on rajattu Android käyttöjärjestelmään. Tutkimusmenetelmänä käytetään kirjallisuuskatsausta, jonka avulla kirjallisuudesta selvitetään vastauksia seuraaviin tutkimuskysymyksiin:

1. Millaista käyttäjän seurantaan liittyvää tietoa puhelin tallentaa?
2. Millä työkaluilla tähän tietoon päästään käsiksi?
3. Miten kerätystä tiedosta voidaan koostaa aikajana?

1.3 Työn rakenne

Luvussa 2 esitellään tutkimusmenetelmä ja käytetyt lähteet sekä hakusanat. Kolmannessa luvussa käsitellään kirjallisuuskatsauksen tulokset. Luvussa 4 kerrotaan tutkimuksen tulosten soveltamista käytäntöön aikajanan muodostavan sovelluksen muodossa. Viimeinen luku sisältää tutkimuksen ja toteutetun sovelluksen yhteenvedon.

2 TUTKIMUSMENETELMÄ

Tutkimusmenetelmänä käytetään kirjallisuuskatsausta. Kirjallisuuskatsaukseen tietoa kerätään verkkotietokannoista. Tietokannoiksi valittiin IEEE Xplore ja ACM Digital Library testihakujen tuloksien perusteella. Tietokannoista haettiin tietoa kyseisillä hakusanoilla:

- (“Data acquisition” OR “Data gathering”) AND “Android Operating System”
- (“Data acquisition tools” OR “Data gathering tools”) AND “Android Operating system”
- “Mobile Application” AND “Covid-19”

Ensimmäisen hakulausekkeen kohdalla hakutuloksia rajoitettiin lisäksi vuosien 2016 ja 2020 välille, koska hakutuloksia kyseisellä lausekkeella oli paljon, joten määrää haluttiin hieman rajoittaa. Lisäksi tutkimukseen etsittiin tietoa muista verkkolähteistä, kuten foorumeilta ja julkisista ohjelmavarastoista (repository). Näitä tietolähteitä käytettiin varsinkin käytännön osion suunnittelussa ja työkalujen etsimisessä, sekä kyseisten työkalujen toiminnan ymmärtämisessä.

Taulukko 1. Tietokannat ja mukaan otetut tulokset

Tietokanta	Mukaan otetut / Tuloksia yhteensä
ACM Digital Library	6 / 68
IEEE Xplore	11 / 170
Yhteensä	17 / 238

Päätös tuloksien mukaan ottamisesta tutkimukseen tehtiin alustavasti otsikon ja tiivistelmien pohjalta. Näiden perusteella tutkimuksesta jätettiin pois aiheeseen kuulumattomat tulokset. Aiheeseen kuulumattomiksi tuloksiksi luokitellaan kaikki semmoiset tulokset, jotka eivät liity tutkimuksen aiheeseen (eivät sisällä aiheita liittyen tutkimuskysymyksiin) ja näin ollen ei hyödytä kirjallisuuskatsauksen tekemisessä. Mukaan valitut tutkimukset luettiin kokonaan

läpi, jonka perusteella päätettiin lopullisesti lopullinen mukaan otettujen tutkimuksien joukko.

3 KIRJALLISUUSKATSAUS

Tässä luvussa käydään läpi Android käyttöjärjestelmän muistirakennetta, mitä käyttäjän seurantaan liittyvää tietoa tämän käyttöjärjestelmän omaava puhelin tallentaa, ja sitä miten tähän tietoon pääsee käsiksi. Lisäksi käydään läpi mitä olemassa olevia sovelluksia Covid-19 pandemiaa varten on kehitetty.

3.1 Älypuhelimen käyttötarkoitukset

Jotta voidaan täysin ymmärtää mikä käyttäjätieto on tärkeää ja mistä sitä kannattaa älypuhelimesta etsiä, täytyy ymmärtää mitkä ovat älypuhelimen yleisimmät käyttötarkoitukset nykyaikana. Älypuhelimen käyttäjiä löytyy joka ikäluokasta, mutta varsinkin nuorista ja alle 30 vuotiaista amerikkalaisista aikuisista suurin osa omistaa älypuhelimen (Zhang & Costa, 2016). Toki kulttuurilla ja elinolosuhteilla on vaikutusta älypuhelimen omistamiseen ja käyttämiseen, mutta hieman yleistäen kännyköiden käyttö on yleistä melkein kaikkialla. Zhangin ja Costan tutkimuksen mukaan puhelin on mukana elämän joka osa-alueella, viestien lähettelystä viihdekäyttöön. Näistä älypuhelimista noin joka toinen on käyttöjärjestelmältään Android. Zhang ja Costa toteuttivat kyselyn nuorille amerikkalaisille korkeakouluopiskelijoille (pääasiassa vastaajat 18 v - 25 v). Kyselyn tuloksien mukaan nuoriso käyttää puhelinta usein ja moneen tarkoitukseen, ei vain peruskommunikointia varten.

Useimmin käytetyt sovellukset/toiminnot kyselyn mukaan:

1. Facebook
2. Instagram
3. Gmail
4. Chrome
5. Email

Älypuhelimen yleisimmät käyttötarkoitukset:

1. Peruskommunikointi
2. Email

3. Internetin selailu
4. Sosiaalinen media
5. Navigaatio/GPS

3.2 Android käyttöjärjestelmän muistirakenne

Älypuhelimet yleistyvät koko ajan, ja erilaisia sovelluksia tarjolla yhä enemmän. Monet näistä sovelluksista prosessoivat suurta määrää henkilökohtaista tietoa (esimerkiksi viestintäsovellukset). Prosessoinnin lisäksi on suuri mahdollisuus, että ne tallentavat tietoa paikallisesti muistiin. Tiedon keräämisessä (varsinkin rikollistutkinnassa, sekä tartuntaketjujen selvittämisessä) on tärkeä kerätä tietoa niin että tieto ei muutu sitä kerättäessä. Androidin muistirakenne on hyvin dokumentoitu jo valmiiksi, mutta tutkimusta varten täytyy ymmärtää yleisesti sen rakenne. Android on Linux pohjainen käyttöjärjestelmä, joka käyttää puurakennetta (single tree hierarchy) ja hyödyntää liitospisteitä (mount points), joilla otetaan massamuistin sisältö käyttöjärjestelmän käyttöön. (Boueiz, 2020). Android käyttöjärjestelmän muisti rakentuu seuraavasti (Drake et al., 2014):

- Alkulatausohjelma (Bootloader): Sisältää alkulatausohjelman. Ohjelma alustaa laitteiston ja käyttöjärjestelmän käyttöä varten. Hoitaa myös laitteen muiden tilojen käyttöä, esimerkiksi palautus tilan (recovery mode))
- Käynnistys: Sisältää käynnistys vedoksen (boot image). Vedos koostuu Linux kernelistä ja tiedostojärjestelmän RAM-levystä (root file system ram disk).
- Palautus: Sisältää miniminaalisen Android käyttöjärjestelmän, joka koostuu kernelistä ja RAM-levystä, jossa tulee mukana työkaluja laitteen ylläpitoon liittyen.
- Järjestelmä, eli ”/system” polku: Sisältää Androidin rungon, kirjastot, järjestelmän sovellukset ja ennalta asennetut sovellukset.
- Käyttäjätiedot (Userdata), eli ”/data” polku: Sisältää käyttäjän tietoja, kuten asennetut sovellukset, tiedostot, mediatiedostot ja niin edelleen.
- Sdcard, eli ”/sdcard” polku: Jos laitteessa on käytössä SD kortti, ”/sdcard” polku viittaa erilliseen sdcard osioon (partition), muuten polku sisältyy käyttäjätietojen osioon. Sisältää samanlaista dataa kuin käyttäjätieto osio.

Älypuhelin tallentaa käyttäjästä monenlaista tietoa käyttäjästä laitteen käytön helpottamiseksi. Varsinkin rikosteknistä tutkimusta tehdessä tietomäärä on valtava, joten on tärkeää erotella kerättävä tärkeä tieto muun tiedon seasta ja valita oikeat työkalut tämän tiedon keräämistä varten (Dian & Hudec, 2019). Dianin ja Hudecin mukaan Android käyttöjärjestelmä on luonteeltaan avoin, joten siitä on suhteellisen yksinkertaista kerätä hyödyllistä dataa.

Tietoa on monenlaisessa muodossa, ja monessa eri paikassa. Dian ja Hudec listaavat, että Android käyttöjärjestelmä tallentaa tietoa kuuteen kohteeseen:

- ”Shared preferences”, XML tiedostoja, jotka sisältävät dataa käyttäjistä ja laitteen asetuksista
- Sisäinen muisti
- Ulkoinen muisti
- SQLite tietokannat, sovelluksien käyttöön (selainten käyttö, paikkatiedot)
- Verkko, vaikka data ei välttämättä ole paikallisesti laitteella, voi siitä löytyä tietoa konfiguraatio tiedostoista
- Systeemin lokitiedostot

Tutkimuksen kannalta kiinnostavin muistiosio on pääasiallisesti käyttäjätiedot, eli laitteen sisäinen muisti. Tähän muistiosioon tallentuu kaikki käyttäjän seurantaan liittyvät tiedot, kuten valokuvat ja tiedot sovelluksista. Sovelluksien datan sijainti vaihtelee käyttöjärjestelmien ja eri sovelluksien mukaan, mutta yleisesti polku sovelluksen tallennuspaikkaan on *”/data/data/ <packageName>”.* Tämä sijainti on kuitenkin suojattu, joten tietoon käsiksi pääsemiseen tarvitaan suurimmat oikeudet.
Myös sdcard osiosta, eli ulkoisesta muistista voi sisältää hyödyllistä tietoa. Sieltä voidaan löytää esimerkiksi kuvia/videoita, ladattuja tiedostoja, sekä julkinen sovelluksen tallennuspaikka. Tämä sijainti on suojaamaton, eli jokaisella on oikeus tähän sisältöön. Viimeiseksi myös järjestelmän, kernelin ja sovelluksien lokitiedostot voivat sisältää tärkeää tietoa. (Scrivens ja Lin, 2017)

3.3 Käyttäjän seurantaan liittyvät tiedot

Älypuhelin tallentaa monenlaista tietoa, mutta kaikki tieto ei ole tarpeellista tutkimuksen kannalta. Tartuntaketjujen selvittämisen kannalta tärkeää tietoa ei ole eritelty kirjallisuudessa paljoa kirjallisuuskatsauksen tehdyn tutkimuksen perusteella, mutta tätä alaa vastaa läheisesti rikostekniikka, josta löytyy paljonkin tutkimustietoa. Rikosteknisestä näkökulmasta, sekä tartuntaketjujen selvittämisen kannalta, on tärkeä tietää mitä käyttäjä on tehnyt, mihin aikaan ja missä paikassa. Rikosteknisestä näkökulmasta kiinnostavaa tietoa käyttäjästä on puhelimessa saatavilla (Dogan & Akbal, 2017):

- Laitteen tiedot
- Sim kortin tiedot
- Kontaktit
- Sähköpostit/Viestit
- Tiedot kommunikoinnista ja Sosiaalisista verkostoista
- Käyttätilit
- Kalenteri
- Asennetut sovellukset ja sovelluksien data
- Systeemin lokit
- Käyttätiedot
- Multimedia (Kuvat, Videot, Audio)
- Paikkatiedot
- Poistetut tiedostot

3.4 Tiedonkeräys ja työkalut

Android käyttöjärjestelmästä voidaan kerätä tietoa kahdessa muodossa (Scrivens & Lin, 2017):

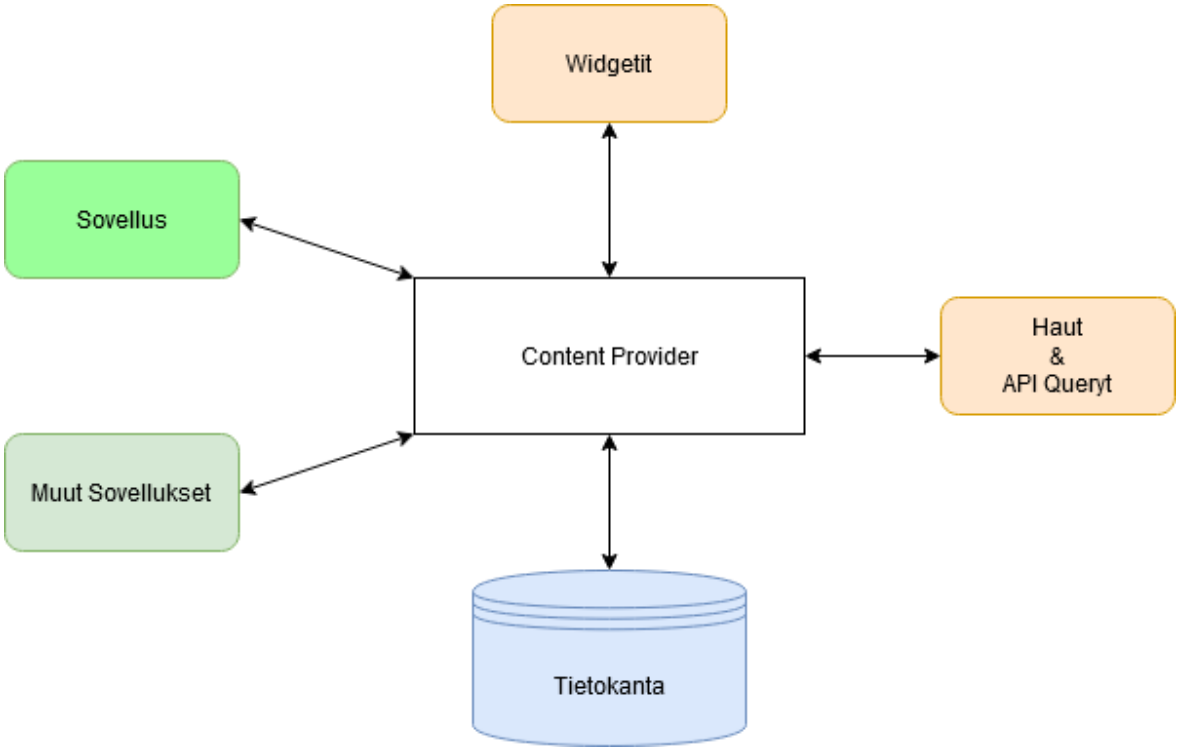
1. Loogisen tiedon keräys. Tieto kopioidaan ymmärrettävässä muodossa, eli otsikkotiedostot (file header) ovat paikoillaan ja tiedostojärjestelmä pysyy kunnossa. Hyötynä tässä metodissa on, että saadun datan kanssa on helppo työskennellä, sillä

se on jo ymmärrettävässä muodossa. Haittana on, että jotain oleellista tietoa voi jäädä keräämättä.

2. Fyysisen tiedon keräys. Bittitarkka kopio tai data dump tallennuslaitteesta tai osiosta. Kaikki data mukaan lukien poistetut tiedostot ja käyttämätön data kopioidaan. Hyötynä tässä on suurempi tietomäärä (potentiaalisesti enemmän hyödyllistä dataa), mutta dataa voi olla hankalaa uudelleenrakentaa ymmärrettävään muotoon.

3.4.1 Loogisen tiedon keräys

Loogisen tiedon keräyksessä tietoa haetaan laitteesta yleensä käyttäen jotain laitteen sovellusta, tai tietokonetta. Android käyttöjärjestelmässä tärkeä osa kaikessa laitteen sisäisessä tiedonkulussa on Content Provider. Content Provider vastuussa sovelluksien oikeuksista laitteen keskustietokantaan. Se on osa jokaista Android sovellusta ja sen avulla määritetään mihin kyseisen sovelluksen tietoon muut sovellukset voivat päästä käsiksi. Kuvassa 1 näkyy, miten kyseinen komponentti toimii laitteessa kaiken tietoliikenteen keskellä tiedonvälittäjänä. (Android Developers, 2020)



Kuva 1. Content Provider ja yhteydet muihin komponentteihin. (Android Developers 2020)

Loogisia tiedonhankintamenetelmiä on useita, jolla tietoa saadaan laitteesta ulos loogisessa muodossa. Menetelmiä ovat muun muassa (Scrivens & Lin, 2017):

- Rikostekniset ohjelmistot (Forensic software suite)
- ADB (Android Debug Bridge)
- Varmuuskopiointi sovellukset
- Laiteohjelmiston (Firmware) päivitys protokollat
- Mukautettu laitteen palautustila

Taulukossa 1 näkyvät kyseisten menetelmien hyödyt ja haitat.

Taulukko 1. Loogiset tiedonhankintamenetelmät (Scrivens and Lin, 2017)

Menetelmä	Hyödyt	Haitat
Rikostekniset ohjelmistot	Keräävät tiedot, sekä näyttävät ja tallentavat kerätyn tiedon järkevästi.	Kalliita. Uudet puhelimet ei välttämättä tuettuja. Puhelimia voidaan joutua muokkaamaan/roottaamaan ennen tiedon hankinnan aloittamista.
ADB	Voidaan käyttää tiedostojen hakemiseen laitteesta, komentojen suorittamiseen sekä lokitiedostoihin käsiksi pääsemiseen.	Laite täytyy olla rootattu, jotta käyttäjätietoihin voidaan päästä käsiksi. Myös ”USB-virheenkorjaus” täytyy olla laitteessa käytössä.
Varmuuskopiointi sovellukset	Helppo käyttöönotto. Tarvitaan vain ohjelma ja mahdollisesti microSD	Laite joudutaan roottaamaan, jotta kaikki tieto on haettavissa.

	kortti johon tiedot kopioidaan.	Muokkaavat käyttäjätietojen sisältämää osiota toimiakseen.
Laiteohjelmiston päivitys protokollat	Kehittynyt tapa päästä tietoihin käsiksi muuttamatta tietoa millään tavalla. Laitteen muistista saadaan myös fyysinen kuva (physical image), ilman että puhelinta tarvitsee purkaa fyysisesti.	Takaisinmallintaminen vie paljon aikaa, mutta tarvitsee suorittaa vain kerran tietylle laitemallille.
Mukautettu laitteen palautustila	Vain laitteen muistin palautusosiota muokataan, muu muisti pysyy koskemattomana. Suhteellisen yksinkertainen prosessi, jos palautusosio on hankittavissa. Voidaan tuottaa fyysinen ja looginen vedos. Joustava menetelmä.	Palautus vedoksen luonti, joka tukee useampaa laitetta, voi olla aikaa vievä urakka, mutta tarvitsee suorittaa vain kerran. Laitteen muistin palautusosiota täytyy muokata. Alkulatausohjelma voi olla lukittuna, ja tämän lukituksen avauksen yhteydessä on kuitenkin hyvin todennäköistä menettää käyttäjän tiedot (Wu et al., 2017).

Rikostekniset ohjelmistot

Monet rikostekniset ohjelmistot hyödyntävät Androidin Content Provider komponenttia, joka mahdollistaa sovelluksien välisen dataliikenteen. Suurin osa näistä ohjelmistoista voi hakea paljon enemmän tietoa hyödyntämällä myös tiedon fyysisiä kopiointi menetelmiä. (Scrivens & Lin, 2017)

ADB (Android debug bridge)

Komentorivi työkalu, joka mahdollistaa kommunikaation tietokoneen ja Android laitteen välillä. Ohjelma, joka mahdollistaa tämän kommunikoinnin koostuu kolmesta komponentista (Scrivens & Lin, 2017):

1. Asiakasohjelma (client), joka lähettää käskyt Android laitteeseen.
2. Daemon. Pyörii taustalla Android laitteessa, suorittaa sille lähetetyt käskyt.
3. Palvelin. Pyörii taustalla tietokoneessa, hoitaa yhteyden ohjelman ja daemonin välillä.

ADB:n käyttäminen tehokkaasti vaatii kuitenkin laitteen roottauksen. Roottaus on siis menetelmä, jolla saadaan täysi pääsy laitteen tietoihin hankkimalla Superuser oikeudet laitteeseen. Voidaan käyttää ilmaisia sovelluksia kuten Framaroot, Firmware.mobi ja niin edelleen. Toinen tapa rootata puhelin on avata alkulatausohjelman lukitus ja asentaa muokattu palautus vedos (esimerkiksi TWRP). Roottauksen yhteydessä on kuitenkin mahdollista menettää kaikki data. Roottaus on yleensä pakollinen toimenpide, jos laitteen flash muistista halutaan saada fyysinen bittitarkka (bit-by-bit) kopio, jos ei haluta käyttää laitetta muokkaavia fyysisen tiedonhankinnan menetelmiä kuten JTAG tai chip-off. Laitteen ollessa käyttökunnossa, yleensä roottaus on parempi toimenpide, mutta jos laitteen muisti on vioittunut JTAG tai chip-off ovat myös hyviä vaihtoehtoja. (Wu et al., 2017)

Wu:n tutkimuksessa verrattiin roottaamattomasta ja rootatusta puhelimesta saatuja tietoja. Roottaamattomasta puhelimesta löydettiin laitteen palautustoiminnolla 2553 kuvaa.
Rootatusta puhelimesta löydettiin yli 31000 kuvaa, tekstiviestejä, sähköposteja ja jopa vanhaa poistettua dataa seuraavalla menetelmällä:

1. Puhelin rootattiin Dr.Phone ohjelmistolla
2. Laitteen muistista etsittiin ”/data” osio
3. Syötettiin ”dd” komennot, joilla saatiin laitteen vedos.
4. Vedos analysoitiin Autopsy ohjelmistossa.

Varmuuskopiointi sovellukset

Hyödyntää myös Androidin Content Provider palvelua. Sovelluksella päästään siis käsiksi tietoihin, joita laitteen sovellukset on ohjelmoitu jakamaan. Muokkaamattomasta laitteesta näillä sovelluksilla voidaan saada esimerkiksi seuraavan laista hyödyllistä tietoa: tekstiviestit, soittotiedot, kontaktit, kalenteri, selainten tiedot jne. (Scrivens & Lin, 2017)

Laiteohjelmiston päivitys protokollat

Tekninen tietotaitoa vaativa tiedonkeruu menetelmä. Lähestymistavan toteutus riippuu myös tarkasteltavasta laitteen valmistajasta/mallista, koska laitteet voi erota toisistaan ja näin ollen menetelmän eri vaiheet ovat laitekohtaisia. Menetelmässä hyväksikäytetään laitteen laiteohjelmiston päivitystilaa. Tämä tila on ainoa tapa päästä käsiksi suoraan laitteen flash muistiin. Takaisinmallintamalla (reverse engineering) alkulatausohjelma ja laiteohjelmiston päivitysohjelma, voidaan selvittää komennot ja keinot tietoon käsiksi pääsemiseksi. (Scrivens & Lin, 2017)

Mukautettu laitteen palautustila (custom recovery image)

Menetelmä, jolla laitteen muistista saadaan sekä looginen, että fyysinen kuva muokkaamalla laitteen palautusosiota. Näin ollen käyttäjätietojen sisältämä muistiosio pysyy koskemattomana. Käytännössä menetelmä toimii seuraavasti (Scrivens & Lin, 2017):

1. Luo/hanki mukautettu palautus vedos (recovery image) laitteesta
2. Asenna (flash) vedos laitteeseen
3. Uudelleenkäynnistä laite palautustilaan
4. Käytä hyväksi ”adb shell” komentoa tietojen hakemiseksi muistista. Muistista voidaan muodostaa fyysinen vedos apuohjelmia käyttäen (esim. nanddump, dd), tai looginen vedos (hakemalla tiedostoja suoraan)

3.4.2 Fyysisen tiedon keräys

Fyysisen tiedon keräyksessä laitteen tietoa kerätään yleensä kiertämällä laitteen suojauksia, tai muokkaamalla laitetta fyysisesti ja näin mahdollistamalla pääsyn suoraan raakaan bittitarkkaan dataan. Fyysisen tiedon keräyksen menetelmiä:

- Chip-off
- JTAG (Joint Test Action Group)
- Qualcomm tilojen hyväksi käyttäminen
- RAM:in (Random Access Memory) hyväksi käyttäminen

Chip-off

NAND flash sirut poistetaan piirilevystä, ja näiden sirujen liittymäkohtiin liitetään työkalut, joiden avulla saadaan laitteesta fyysisesti tietoon käsiksi. (Hoog, 2011)

JTAG (Joint Test Action Group)

Kommunikaatio protokolla, jolla tarjotaan pääsy prosessoreiden debuggaus/emulaatio ominaisuuksiin. JTAG yhdistetään suoraan fyysisesti prosessoriin juottamalla, ja kun yhteys on luotu, tiettyjä komentoja käyttämällä voidaan muodostaa täydellinen fyysinen vedos flash muistista. (Hoog, 2011)

Qualcomm tilojen hyväksi käyttäminen

Ensimmäinen tapa on käyttää hyväksi Qualcomm 9006 tilaa. Android pakotetaan Qualcomm 9006 tilaan tahallisesti tekemällä vahinkoa laitteen käynnistys osioon. Tämän jälkeen datan voi lukea tietokoneella käyttäen apuna jotain muistin kuvantamiseen tarkoitettua työkalua. Toinen tapa on käyttää Qualcomm 9008 tilaa. Tässä tavassa puhelin käynnistetään fastboot tilaan. Fastboot tila mahdollistaa tietokoneen kommunikoinnin laitteen alkulatausohjelman kanssa USB:n (Universal Serial Bus) kautta. Fastbootin avulla voidaan muokata tai poistaa muistin osioita komentoriviltä tai ladata laitteeseen muokattu vedos. Laitteen ollessa fastboot tilassa, asetetaan se Qualcomm 9008 tilaan käyttäen laitteelle sopivaa flash työkalua

ja muokattua TWRP (Team Win Recovery Project ohjelmisto) palautus vedosta. Menetelmä eroaa laitteen mallista riippuen. (Wu et al., 2017)

RAM:in hyväksi käyttäminen

Nisioti et al., 2017, tutkimuksen mukaan, esimerkiksi viestintäsovelluksista voidaan saada dataa Random Access Memorystä (RAM). Kyseisessä tutkimuksessa käytettiin LiME:ä (Linux Memory Extractor) muistivedoksen lukemiseen ja ADB:tä muistivedoksen siirtämiseen tutkijan tietokoneelle. Muistivedosta tutkittiin hex editorilla, jolla etsittiin viestintäsovelluksien viestejä, ja niiden sisältämää tietoa. Löydettyjen viestien perusteella tunnistettiin toistuvia kuvioita ja luotiin säännöllisiä lausekkeita, joiden avulla vedoksesta voitiin hakea kaikki mahdollinen data viesteihin liittyen. Data täytyi muuntaa myös hexadesimaalimuodosta luettavaan muotoon. Käytännössä tämä tarkoitti esimerkiksi siis aikaleiman muuntamisen hexadesimaalimuodosta päivämääräksi. Tämän jälkeen luotiin Python skriptejä hakemaan kaikki viestidata WhatsApp, Facebook ja Viber sovelluksista. Tämän jälkeen löydettyjen viestien määrää vertailtiin neljässä eri tilanteessa (Käynnistyksen jälkeen, akun poiston jälkeen, päivän käytön jälkeen ja päivän käytön sekä uudelleenkäynnistyksen jälkeen). Lopputuloksena huomattiin, että tulokset näissä tilanteissa eivät vaihtele huomattavasti.

3.4.3 Tiedonkeräyksen työkalut

Puhelimiin luodut tiedonhankintatyökalut on suunniteltu pääosin rikosteknisen tutkimuksen käyttöön. Nämä keräävät puhelimesta kaiken mahdollisen saatavilla olevan tiedon, kategorisoi tiedot omiin luokkiinsa ja koostaa raportin tutkijoiden käyttöön rikoksien selvittämisessä. Tällaisia työkaluja ovat esimerkiksi (Dogan & Akbal, 2017):

- Cellebrite
- Paraben’s Device Seizure
- XRY
- EnCase Neutrino
- Oxygen Forensic

- MOBILedit
- Faraday
- Tarantula

Nämä työkalut ovat hyvinkin kehittyneitä, sillä ne ovat tehty rikostekniseen käyttöön. Data puhelimesta täytyy saada vahingoittamatta dataa, jotta sitä voidaan käyttää hyväksi oikeudessa. Työkaluja varten laitteita ei yleensä tarvitse myöskään rootata, tai ainakaan se ei vaikuta tiedon hankinnan lopputulokseen (Hassan & Pantaleon, 2017). Jos roottaus vaaditaan, se on yleensä mukana ohjelmassa. Työkaluissa on paljon hyödyllisiä ominaisuuksia ja esimerkiksi Oxygen Forensics Suite voi muodostaa graafin sosiaalisista verkostoista ja hakea paikkatiedot sekä aikajanan käyttäjän luomista tapahtumista puhelimen käytön aikana. Nämä sovellukset ovat kuitenkin kaupallisia, joten sovelluksien lähdekoodi ei ole julkisesti saatavilla. Myös avoimen lähdekoodin työkaluja on kuitenkin saatavilla, esimerkiksi ANDROPHSY, Delta ja Magnet.

Työkaluja vertailevassa tutkimuksessa (Lwin et al., 2020), vertailtiin viittä eri työkalua: Autopsy, ADB Backup, Magnet, Belkasoft ja DD. Kyseisen tutkimuksen lopputuloksien mukaan loogisen tiedon keräyksessä Magnet oli vahvin, fyysisen tiedon keräyksessä DD ja Belkasoft saavuttivat yhtä hyvät lukemat, ja tiedon analysoinnissa Belkasoft oli Autopsy:ä parempi kerätyssä tietomäärässä, sekä nopeudessa. Seuraavaksi käydään läpi muita työkaluja ja niiden ominaisuuksia. Kappaleen lopussa näkyy taulukkoon 2 kerättynä kaikki tutkimuksessa vastaan tulleet työkalut ja karkeasti niiden ominaisuudet.

ANDHROPSY

Avoimen lähdekoodin vaihtoehto rikostekniseen tutkimukseen, joka kattaa kaikki tutkimuksen osa-alueet tutkimuksen tapauksen hallinnasta, puhelimen roottauksesta, datan keräyksestä aina sen esitykseen asti. Vaihtoehto kalliille kaupallisille ohjelmille kuten Oxygen. Sisältää loogisen ja fyysisen datan keräyksen. Kerää myös kernelin, muistin, käyttäjän aktiviteettien, tilien ja synkronisoinnin ja paikkatietojen lokitiedostot laitteesta tekstitiedoston muodossa hyödyntäen Linuxin kernel komentoja (logcat, demsg, dumpsys). (Akarawita et al., 2015)

AFLogical

Avoimen lähdekoodin rikosteknilliseen työhön suunniteltu työkalu, joka käyttää hyväkseen Androidin Content Provider komponenttia. Sopii nopeaan, muttei niin perusteelliseen tiedonhankintaan. Työkalulla voidaan laitteesta kerätä tietoja esimerkiksi puheluista, kontakteista, tekstiviesteistä. Haettu data tallennetaan SD (Secure Digital) kortille, josta data voidaan siirtää tietokoneelle analysoitavaksi. (Sathe & Dongre, 2018)

DELTA (Data Extraction and Logging Tool for Android)

DELTA on tehty tarkkailemaan Android käyttöjärjestelmän omaavaa puhelinta kokonaisvaltaisesti ja pitemmälläkin aikavälillä. Voidaan tarkkailla puhelimesta kaikkea näytöntoiminnoista sijaintitietoihin sekä sensoreihin asti. Työkalun koodi on vapaasti saatavilla ja muokattavissa. Lisäksi työkalussa on lisäosia tukeva toiminto, joten tiedonkeräystä voi muokata omiin tarpeisiin sopivaksi. Sisältää Android ohjelman, ja työpöytäohjelman datan tarkastelemiseen. (Spolaor et al., 2018)

Menthal

Andone et al., 2016, kehittämä työkalu eroaa muista työkaluista sillä se ei ole rikosteknistä työtä varten suunniteltu, vaan sen ominaisuutena on tapahtumapohjainen ja reaaliaikainen tiedonkeräys. Se kerää laitteen käytön ajalta tietyn aikaa käyttäjädataa ja analysoi datan. Data analysoidaan osittain laitteessa ja kun dataa on kerätty tarpeeksi, data lähetetään serverille lopulliseen analyysiin. Sovellus kerää esimerkiksi tietoja:

- Sovelluksien käytöstä
- Näytön sammutuksista/käynnistyksistä
- Laitteen sammutuksista
- Tekstiviesteistä, puheluista
- GPS koordinaateista
- Käyttäjän mielialasta kyselyiden muodossa

Oxygen Forensic Suite

Suosittu rikosteknisessä tutkimuksessa käytetty kaupallinen työkalu Euroopassa ja myös Suomessa. Mahdollistaa nopean tiedon keruun paikan päällä. Toimii kaapelin, infrapunan tai Bluetoothin välityksellä muuttamatta laitteen dataa. Sisältää datan keräyksen, tulkinnan ja analyysin tietokoneessa sekä puhelimessa. Myös mahdollisuus yhdistää ”Lifeblog and geotagging” ominaisuuteen Symbian OS:ssä (Operating System). Ohjelmisto käyttää apunaan ”special agent” sovellusta, joka suorittaa data-analyysin. Tämä sovellus mahdollistaa sekä fyysisen, että loogisen tiedon keräyksen laitteesta. (Yates, 2010)

Paraben’s Device Seizure

Matalat järjestelmävaatimukset omaava kaupallinen työkalu, eli pyörii missä vain tietokoneessa. Työkalulla on myös mahdollista suorittaa analyysia sitä tukemattomissa laitteissa, jos ne ovat tuettujen valmistajien laitteita. Työkalussa on vähemmän ominaisuuksia kuin Oxygenissä, mutta eroavana ominaisuutena sillä voidaan käydä läpi myös laitteen muistivedos. (Yates, 2010)

Taulukko 2. Erilaisten tiedonhankitatyökalujen vertailu.

Työkalu	**Avoin lähdekoodi**	**Looginen tiedon hankinta**	**Fyysinen tiedon hankinta**	**Graafinen käyttöliittymä**	**Analysointi ominaisuus**
ADB Backup	Kyllä	Kyllä	Ei	Ei	Ei
AFLogical	Kyllä	Kyllä	Ei	Kyllä	Kyllä
Autopsy	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
ANDHROPSY	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Belkasoft	Ei	Kyllä	Kyllä	Kyllä	Kyllä
DD	Kyllä	Ei	Kyllä	Ei	Ei
Delta	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Magnet	Kyllä	Kyllä	Kyllä	Kyllä	Ei
Menthal	Ei	Kyllä	Ei	Kyllä	Kyllä

3.5 Älypuhelinsovellukset ja Covid-19

COVID-19 pandemia on riehunut maailmalla noin kahden vuoden ajan, joten se on vielä suhteellisen uusi ilmiö. Tämän takia saatavilla olevat sovellukset ovat vielä hyvin rajoittuneita, mutta aktiivisesti vielä kehityksessä, joten uusia sovelluksia luodaan ja vanhoja

kehitetään koko ajan. Näitä sovelluksia vertailevan tutkimuksen (Islam et al., 2020) mukaan, on olemassa 25 sovellusta, jotka on luotu COVID-19 pandemian tarkoituksiin. Tarkoitukset on kategorisoitu yhdeksään eri kategoriaan:

1. Etätuki – Konsultaatio ja taudin oireiden arviointi
2. Potilaiden tarkkailu
3. Tämänhetkinen tilanne – Paikallinen/maailmanlaajuinen tilanne ja statistiikat
4. Taudin ennaltaehkäisy – Ajankohtaiset uutiset ja ohjeistukset
5. Taudin hallinta – Ohjeet eristäytymiseen ja taudista raportoiminen
6. Tuki – Kommunikointi viranomaisten kanssa
7. Hoitopalvelut
8. Tiedon levittäminen
9. Mielenterveyden parantaminen

Suurin osa näistä sovelluksista on kuitenkin kehitetty kehitysmaan kontekstin pohjalta, eikä välttämättä palvele muiden maiden tarkoituksia. Jokaisessa maassa tilanne on kuitenkin erilainen, joten näitä sovelluksia tarvitaan myös. Sovellukset yleensä myös sisältävät useita näistä tarkoituksia palvelevista ominaisuuksista, muttei yksikään sisällä kaikkia niistä. (Islam et al., 2020)

Sovelluksia on siis kattavasti jokaiseen tarkoitukseen, ja sekä globaalisti että lokaalisti saatavilla. Tutkimuksessa ei kuitenkaan löytynyt yhtään käyttäjätiedoista käyttäjän sijainnit ja kontaktit aikajanalle koostavaa sovellusta. Lähimpänä tätä oli rikosteknisiin tarkoituksiin kehitetyt sovellukset, mutta pandemiaa varten tämmöistä yleiseen tarkoitukseen käytettävää työkalua ei vielä ole olemassa. Kaikki taudin hallintaan ja ennaltaehkäisyyn tarkoitetut sovellukset ”vain” seuraavat tilannetta ja ilmoittavat mahdollisesta tartunnasta, eivätkä esimerkiksi tarjoa ominaisuuksia käytyjen paikkojen tai kontaktien tarkastelemiseen.

Seurantasovellukset ovat auttaneet osaltaan merkittävästi Covid-19 taudin hallinnassa, mutta varsinkin yksityisyyden suojan kannalta näissä sovelluksissa on vielä kehittämisen varaa. Käyttäjän yksityisyydensuojan kannalta olisi hyvä suosia hajautettuja järjestelmiä (Azad et al., 2020).

Hajautetuissa sovelluksissa kontaktit määrittää käyttäjän laite itse (Nguyen et al., 2020). Hajautetussa jokaiselle puhelimelle määritetään osittain satunnainen tunniste nimeltään Ephemeral ID (EID). Näitä tunnisteita ei tallenneta keskitetysti mihinkään, vaan kun sovelluksen käyttäjä ilmoittaa tartunnasta sovellukselle, kyseisen käyttäjän ID lähetetään ilmoituksen mukana sovellukseen. Tämä ID lähetetään muille sovelluksen käyttäjille ja tartunnan saaneen käyttäjän kanssa kontaktissa olleet käyttäjät saavat sovellukseensa varoituksen kontaktista. Esimerkkinä tällaisesta lähestymistavasta on Suomen Koronavilkku ja Saksan Corona-Warn-App. (Desai, 2020)

Keskitetyssä järjestelmässä puhelimen sisältämä tieto lähetetään ulkoisen järjestelmän säilytykseen ja tulkittavaksi, jonka järjestelmän tekemän analyysin perusteella muita käyttäjiä varoitetaan kontaktista (Desai, 2020). Keskitettyjä järjestelmiä on käytössä esimerkiksi Singaporessa, Australiassa, Ranskassa, ja Iso-Britanniassa (Nguyen et al., 2020). Nguyenin tutkimuksessa esimerkkinä mainittiin Singaporen TraceTogether sovellus.

Xiong et al., 2020 ovat kehittäneet myös hieman poikkeavan lähestymistavan kontaktien jäljitykseen: REACT (REAl-time Contact Tracing). Sovellus, jonka avulla voidaan jäljittää kontakteja sekä käyttäjä voi itse reaaliajassa seurata riskejä tartunnan saamiseen vierailtujen sijaintien perusteella. Sovellus seuraa käyttäjän sijaintitietoja ja oireita. Käyttäjän itse valitsemat tiedot kerätään vapaaehtoisesti valituin väliajoin palvelimelle. Oireita kysytään päivittäin kyselyiden välityksellä. Lähikontaktit tunnistetaan Bluetoothin avulla ja sijainti GPS:n avulla (jos käyttäjä on tämän sallinut). Käyttäjän henkilöllisyys turvataan käyttämällä Ephemeral ID:tä.

4 SOVELLUKSEN SUUNNITTELU JA TOTEUTUS

Tässä luvussa käydään läpi sovelluksen suunnittelu ja toteutus vaiheet. Lopuksi tarkastellaan toteutettua sovellusta ja mahdollisia ideoita sovelluksen jatkokehitykseen.

4.1 Suunnittelu

Sovelluksen tarkoituksena on koostaa käyttäjän älypuhelimen käyttäjätiedoista aikajanan, joka sisältää tietoja vierailluista paikoista ja tavatuista henkilöistä. Aikajanaa voidaan käyttää apuna tartuntaketjujen selvittämisessä ja käyttäjän oman muistin virkistämisessä. Sovelluksen täytyy olla siis yksinkertainen käyttää, sekä mahdollisimman nopea ottaa käyttöön haastattelutilanteessa. Ideaali tilanne olisi siis, että älypuhelimen voisi yhdistää tietokoneeseen ja sovellus keräisi älypuhelimesta tarvittavat tiedot aikajanan muodostamiseen.

Loogisen tiedon hankintamenetelmät ovat sovelluksen kannalta mielenkiintoisempia, koska fyysiset menetelmät vaativat tarkkaa tietotaitoa aiheesta, sekä resursseja ja aikaa. Loogisilla menetelmillä voidaan saada tuloksia paljon nopeammin, vaikkakin mahdollisesti saatavissa olevan tiedon määrä on pienempi. Tiedon määrä on kuitenkin varmasti riittävä tämän työn tarkoituksiin. Haettu tieto on myös helpommin suoraan käsiteltävissä, eikä siihen tarvitse käyttää aikaa. Content Provider komponentin avulla todennäköisesti saadaan helpoimmin eniten tietoa älypuhelimesta ulos, sillä Androidin tarjoamien API:en (Application programming interface) käyttäminen on hyvin dokumentoitu.

Alustavasti lähdettiin tekemään tietokoneelle web-pohjaista sovellusta, jossa siis käyttöliittymä toteutettaisiin JavaScriptiä ja HTML:ää käyttäen. Tiedonkäsittelyssä apuna voitaisiin käyttää apuna Python tai C ohjelmointikieliä. Molemmista kielistä on hieman aikaisempaa kokemusta, joten kielen valinta riippuu siitä, kummalla on helpompi yhdistää saada yhteys älypuhelimen ja tietokoneen välille. Tiedon hankinta laitteesta toteutetaan ADB:tä ja Content Provideria hyväksikäyttäen.

Sovelluksen kannalta kiinnostavia käyttäjätietoja, joista mahdollisesti saadaan irti tietoa kontakteista tai vierailluista paikoista:

- Valokuvien metadata
- Kalenterin tapahtumat
- Instagram/Facebook julkaisut, ja julkaisuihin merkityt ihmiset ja paikat
- Googlen tallentamat sijaintitiedot
- Kontaktit ja soittohistoria

Käyttäjätiedot on numeroitu kiinnostavuuden ja potentiaalin mukaan. Valokuvista voi saada hyvinkin tarkkaa tietoa käyttäjän liikkeistä ja tavatuista ihmisistä. Tämä kuitenkin vaatii sen, että käyttäjä on valokuvauksellinen ja on hyväksynyt kameran asetuksista sijaintitietojen liittämisen valokuviin, sillä se on oletuksena pois päältä. Kalenterin tapahtumat on myös mahdollisesti paljon tietoa sisältävä paikka, sillä kalenteritapahtumiin voi liittää paikan lisäksi myös tapahtumaan osallistuvat ihmiset ja paljon muuta. Instagram ja Facebook julkaisut yleensä sisältävät myös paikkatietoja ja niihin voi liittää tietoa niihin liittyvistä ihmisistä, mutta käytännössä näihin julkaisuihin käsiksi pääseminen ei välttämättä ole realistista. Google tallentaa osaltaan sijaintitietoja käyttäjästä, jos käyttäjä on näin valinnut. Googlella on kuitenkin jo itsellään oma aikajana, josta näitä tietoja voi kätevästi tarkastella, joten nämä sijaintitiedot ei ole sovelluksen kannalta niin tärkeitä. Kuitenkin jos nämä Googlen sijaintitiedot on kätevästi mahdollista sisällyttää sovellukseen, näin tehdään. Viimeisenä osiona, josta voisi saada jotain hyödyllistä tietoa irti on kontaktit ja soittohistoria. Soittohistoria ei kuitenkaan sisällä muuta tietoa kuin soittajan tiedot ja kellonajan, joten käyttäjän täytyisi muistaa puhelun syy ja tapahtuiko puhelun jälkeen tapaaminen puheluun liittyen.

Itse sovelluksessa ei käyttäjän näkökulmasta tarvitse olla paljoa toimintoja. Kaikki tärkeät toiminnot tapahtuvat taustalla käyttäjältä piilossa. Käytännössä sovelluksessa ei tarvitse olla kuin kaksi näkymää:

1. Alkutietojen syöttäminen (Miten pitkältä ajalta tietoja haetaan, Instagram tunnus ja niin edelleen).

2. Aikajanan tarkasteleminen. Tässä näkymässä on hyvä myös olla mahdollisuus tehdä muutoksia ja lisätä tietoa aikajanalle. Aikajana pitää voida myös tallentaa tai voida jakaa esimerkiksi sähköpostilla tutkijalle.

4.2 Toteutus

Heti alkuun huomattiin, että lähestymistapaa täytyy muuttaa hieman, sillä tietokoneella älypuhelimesta tiedon kerääminen osoittautui yllättävän hankalaksi nykyisten tietoturvamenetelmien johdosta. Hylättiin ajatus web-pohjaisesta sovelluksesta, ja alettiin rakentamaan natiivia Android sovellusta, jonka ohjelmointikielenä on Kotlin. Android sovelluksen etuna on, että ne voivat suoraan käyttää hyväksi Content Provider komponenttia, kunhan vain sovelluksen käynnistyessä käyttäjältä kysytään lupa tietojen käyttöön. Sovellus voidaan myös nopeasti asentaa käyttäjän laitteeseen, joko USB:llä tietokonetta käyttäen, tai laittamalla sovelluksen nettiin, josta sen asennuspaketin voi ladata laitteeseen. Nyt siis Android sovellus hoitaa kaiken tiedon hakemisesta, tiedon käsittelyyn ja aikajanan muodostamiseen. Päätettiin myös keskittyä hankkimaan tietoa pääosin valokuvista ja kalenteritapahtumista, sillä vain näidenkin ominaisuuksien saaminen toimivaksi oli yllättävän ongelmallista.

4.2.1 Tiedon ja paikkatietojen kerääminen

Sekä valokuvien, että kalenteritapahtumien hakeminen onnistuu Androidin Content Resolver luokan kautta. Content Resolver käytännössä hakee URI:n (Uniform Resource Identifier) tietylle Content Providerille, josta Content Provider hakee lopullisen datan. Tämä luokka tarjoaa sovelluksille pääsyn laitteen sisältöön (Android Developers, 2021). Content Resolverin avulla valokuvien hakeminen tapahtuu MediaStore.Images.Media luokan projektiota käyttäen, kuten kuva 2 sen havainnollistaa. Kalenteritapahtumien hakeminen on

käytännössä identtinen tapahtuma, mutta se tapahtuu CalendarContract.Events luokan projektion avulla. Kalenteritapahtumien hakeminen havainnollistetaan kuvassa 3.

```
val imageProjection = arrayOf(
    MediaStore.Images.Media.DISPLAY_NAME,
    MediaStore.Images.Media.SIZE,
    MediaStore.Images.Media.DATE_TAKEN,
    MediaStore.Images.Media._ID
)
val imageSortOrder = "${MediaStore.Images.Media.DATE_TAKEN} DESC"
val cursor = context.contentResolver.query(
    MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
    imageProjection,
    selection: null,
    selectionArgs: null,
    imageSortOrder
)
```

Kuva 2. Valokuvien hakeminen

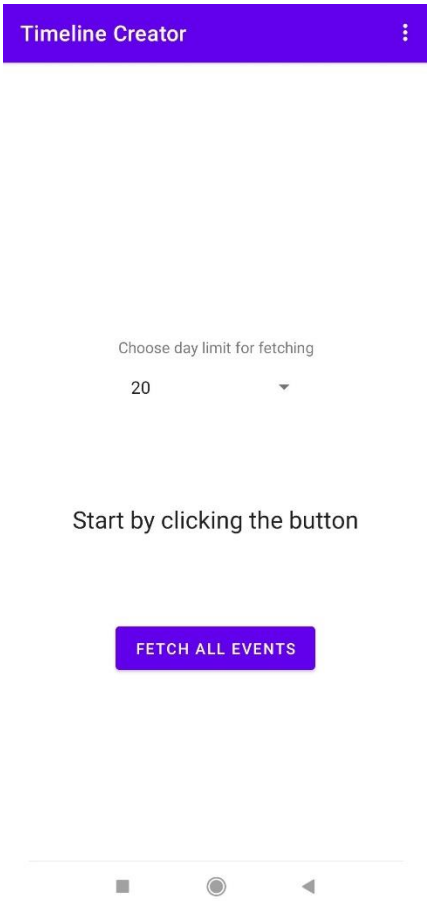
```
val eventProjection: Array<String> = arrayOf(
    "_id",                      // 0
    CalendarContract.Events.TITLE,
    CalendarContract.Events.EVENT_LOCATION,
    CalendarContract.Events.DTSTART,
    CalendarContract.Events.DTEND,
    CalendarContract.Events.DESCRIPTION
)

val uri: Uri = CalendarContract.Events.CONTENT_URI
try {
    val cur: Cursor? = context.contentResolver.query(
        uri,
        eventProjection,
        selection: CalendarContract.Events.DTSTART + "< ?",
        arrayOf(System.currentTimeMillis().toString()),
        sortOrder: "${CalendarContract.Events.DTSTART} DESC"
    )
```

Kuva 3. Kalenteritapahtumien hakeminen

Paikkatietojen hakeminen valokuvista onnistuu käyttäen ExifInterface luokkaa. Kyseinen luokka mahdollistaa Exif tagien lukemisen valokuvista. ExifInterface luokalla kuvista haetaan vain tarpeellinen tieto, eli pituus- ja leveyspiiri, sekä aikaleima. Kuvasta saatu pituus- ja leveyspiiri muunnetaan osoitteeksi Geocoder luokan avulla. Kalenteritapahtumissa kaikki tieto on valmiina käytettävissä heti hakemisen jälkeen.

4.2.2 Käyttöliittymä ja ominaisuudet

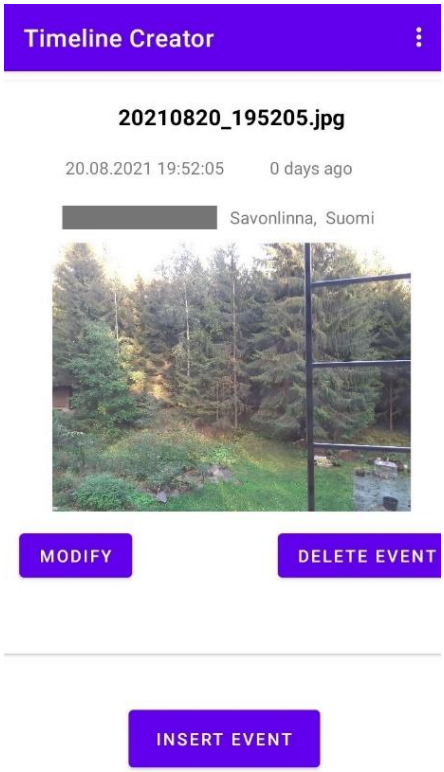
Sovelluksen kieli on englanti henkilökohtaisen preferenssin takia, mutta se on helposti vaihdettavissa Suomeksi, jos sovellus siirtyy jatkokehitykseen. Käyttöliittymään toteutettiin kaksi yksinkertaista näkymää: Aloitusnäkymä ja toinen näkymä aikajanan tarkastelemiseen. Aloitusnäkymä on hyvin yksinkertainen, sillä sovellus ei tarvitse käyttäjältä syötteenä kuin numeron, joka määrittää kuinka pitkältä ajalta tapahtumia haetaan laitteesta. Tapahtumaketjujen selvittämisessä ei tarvitse kaivaa koko käytön ajalta sijaintitietoja, joten nopeutetaan prosessia ja rajoitetaan aikajanalle päätyvän tiedon määrää asettamalla takaraja. Valittuaan pudotusvalikosta arvojen 10 ja 100 väliltä tarpeeseen sopivan rajan, painamalla ”Fetch all events” nappia sovellus aloittaa tietojenkeräämisprosessin ja ilmoittaa siitä käyttäjälle.



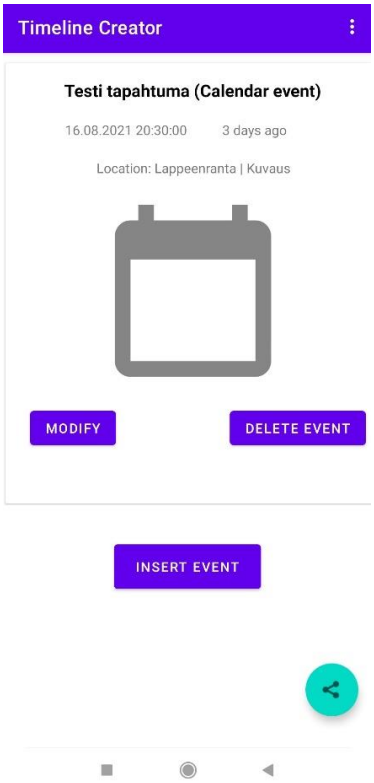
Kuva 4. Sovelluksen aloitusnäkymä

Tietojen hakeminen ja formatoiminen aikajanalle esitettävään muotoon riippuu toki valokuvien ja kalenteritapahtumien määrästä, mutta kahdella eri Android puhelimella ja tietomäärillä testattuna aikaa meni noin 5-10 sekuntia. Kun kaikki data on käsitelty, näkyviin ilmestyy kuvan 4 näköinen aikajana. Aikajanaa voi kosketuksella rullata ylös ja alas sovelluksessa, ja se on järjestetty uusimmasta tapahtumasta vanhimpaan. Aikajanalla näkyviä tapahtumia on siis kahdenlaisia: valokuvatapahtuma ja kalenteritapahtuma. Valokuvatapahtumassa näkyy itse valokuva, valokuvan nimi, tarkka aikaleima, kuinka pitkään aikaa sitten kuva on otettu, ja tarkka sijainti. Kuvasta 5 tarkka sijainti on peitetty, mutta ominaisuuden havainnollistamiseksi kaupunki on jätetty kuvakaappaukseen. Kalenteritapahtumassa ei tietenkään valokuvaa ole mukana, vaan sen paikalla on kalenteri ikoni. Muuten tapahtumassa on samat tiedot, mutta lisäksi kalenteritapahtumassa voi olla merkittynä henkilöitä ja lisätietoja, joten nämä tiedot tulisivat sijainnin perään erotettuna ”|” merkillä, kuten kuva 5 havainnollistaa. Jos tapahtumaan halutaan lisätä jotain olennaista

tietoa tai sovellus ei jostain syystä hakenut oikeita tietoja, voidaan tapahtumaa muokata napista ”Modify”. Tapahtuma voidaan myös poistaa napista ”Delete event”, jos koetaan että se ei ole tarpeellinen. Lisäksi ”Insert event” napista voidaan lisätä tapahtuma, jos aikajanaa tutkiessa käyttäjän mieleen tulee jokin tapahtuma, joka olisi tärkeä sisällyttää aikajanaan.

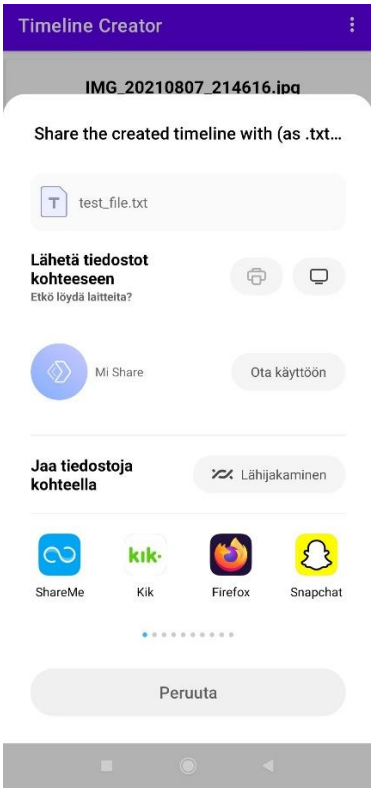


Kuva 5. Aikajananäkymä, valokuva ja sen tiedot

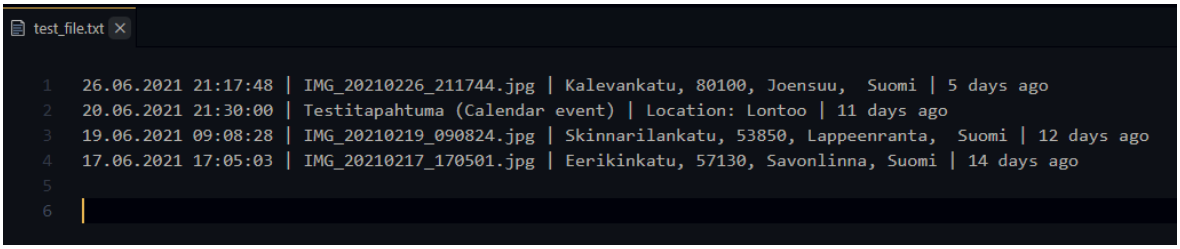


Kuva 6. Aikajananäkymän kalenteritapahtuma ja sen tiedot

Kun aikajana on käyttäjän mielestä valmis, voidaan aikajana jakaa tutkijalle esimerkiksi sähköpostilla, tai esimerkiksi muuntaa PDF (Portable Document Format) tiedostoksi samalla toiminnolla. Jakaminen aloitetaan painamalla pyöreää jakamisikonillista nappia näkymän oikeasta ala laidasta, joka näkyy kuvassa 5. Mahdolliset valinnat riippuvat laitteesta ja Android versiosta, mutta kuvassa 6 näkyy Xiaomi 10T Lite laitteen näkymä ja osa vaihtoehdoista. Tartuntaketjuja tutkivan tutkijan näkökulmasta on tärkeää vain sijainnit ja tavatut kontaktit, joten jaettu aikajana sisältää vain nämä olennaiset tiedot. Kuvassa 7 näkyy jaettu tekstitiedosto ja sen sisältämät tiedot. Jokainen tapahtuma on järjestyksessä omalla rivillään, ja tapahtuman tiedot on eroteltu ”|” merkillä luettavuuden helpottamiseksi.



Kuva 7. Aikajanan jakaminen



Kuva 8. Jaettu aikajana

4.3 Mahdollinen jatkokehitys

Kaiken kaikkiaan sovellus on vielä melko yksinkertainen, joten siihen on mahdollista kehittää monella tapaa. Tällä hetkellä sovellus kerää tietoja vain kalenteritapahtumista ja valokuvista, joten esimerkiksi Instagram julkaisujen hakeminen aikajanalle voisi tuoda sovellukselle lisäarvoa. Käytettävissä olevan ajan ja resurssien puitteissa kuitenkin tämä jätettiin työn ulkopuolelle, koska Instagram julkaisujen hakemisesta Android sovellukseen ei ole aikaisempaa kokemusta. Sovellukselle voisi kehittää myös työpöytäversion, joka

toimisi sovelluksen kanssa yhdessä. Muodostettua aikajanaa olisi varmasti helpompi tarkastella ja muokata tietokoneella, sekä aikajana olisi näin jo valmiina tutkijalla ilman jakamista. Lisäksi näkymät ovat hyvin yksinkertaisia ja ilman ohjeistusta, joten jos sovellukseen lisättäisiin esimerkiksi mahdollisuus hakea Instagram tilin julkaisuja, täytyisi sovellukseen lisätä ohjeistuksia. Myös Suomen kieli olisi tärkeä lisätä sovellukseen jatkokehityksen vaiheessa.

5 YHTEENVETO

Android älypuhelin tallentaa käyttäjästään tietoa monessa eri muodossa, ja moneen eri paikkaan. Tärkein paikka, josta käyttäjätietoa voidaan hankkia, on muistin ”/data” osio. Tästä osiosta voidaan dataa kaivaa tietoa niin loogisessa kuin fyysisessä muodossa, monella eri menetelmällä. Sijainti- ja käyttäjätietojen kannalta kiinnostavimpia tietoja löytyy valokuvista ja kalenteritapahtumista. Mikään tiedonhankintamenetelmä ei ole täydellinen, ja varsinkin fyysisissä menetelmissä on vaara vahingoittaa laitetta, joten tiedonkeruu menetelmä kannattaa valita huolella. Yleensä kannattaakin valita useita menetelmiä, jotta laitteesta saadaan varmasti kaikki tieto irti. Tiedon keräämiseksi on jo valmiiksi olemassa useita kaupallisia ja avoimen lähdekoodin työkaluja, jotka keräävät ja esittävät kerättyä tietoa, kuten esimerkiksi Andhropsy tai Oxygen Forensic.

Tutkimustiedon pohjalta voitiin myös kehittää sovellus, joka muodosti kerätystä tiedosta aikajanan. Sovellus kerää tietoa Content Provider komponenttia hyväksikäyttäen loogisessa muodossa. Sovellus on hyvin yksinkertainen, mutta onnistuu demonstroimaan, miten käyttäjädatasta voidaan muodostaa aikajana. Sovellusta voidaan jatkossa jatkokehittää sisällyttämään käyttäjätietojen lähteitä.

LÄHTEET

Akarawita, I.U., Perera, A.B., Atukorale, A., 2015. ANDROPHSY - forensic framework for Android, in: 2015 Fifteenth International Conference on Advances in ICT for Emerging Regions (ICTer). Presented at the 2015 Fifteenth International Conference on Advances in ICT for Emerging Regions (ICTer), pp. 250–258. https://doi.org/10.1109/ICTER.2015.7377696

Andone, I., Błaszkiewicz, K., Eibes, M., Trendafilov, B., Montag, C., Markowetz, A., 2016. Menthal: a framework for mobile data collection and analysis, in: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct. Presented at the UbiComp '16: The 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, Heidelberg Germany, pp. 624–629. https://doi.org/10.1145/2968219.2971591

Azad, M.A., Arshad, J., Akmal, S.M.A., Riaz, F., Abdullah, S., Imran, M., Ahmad, F., 2020. A First Look at Privacy Analysis of COVID-19 Contact Tracing Mobile Applications. IEEE Internet of Things Journal 1–1. https://doi.org/10.1109/JIOT.2020.3024180

Boueiz, M., 2020. Importance of rooting in an Android data acquisition, in: 2020 8th International Symposium on Digital Forensics and Security (ISDFS). Presented at the 2020 8th International Symposium on Digital Forensics and Security (ISDFS), pp. 1–4. https://doi.org/10.1109/ISDFS49300.2020.9116445

Content provider basics [WWW Document], 2020. . Android Developers. URL https://developer.android.com/guide/topics/providers/content-provider-basics (accessed 2.20.21).

ContentResolver | Android Developers [WWW Document], 2021. URL https://developer.android.com/reference/kotlin/android/content/ContentResolver (accessed 8.20.21).

Desai, B.C., 2020. Pandemic and big tech, in: Proceedings of the 24th Symposium on International Database Engineering & Applications. Presented at the IDEAS 2020: 24th International Database Engineering & Applications Symposium, ACM, Seoul Republic of Korea, pp. 1–10. https://doi.org/10.1145/3410566.3410585

Dian, F., Hudec, J., 2019. Efficient Sensitive Data Gathering with Forensic Analysis of Android Operating System, in: 2019 17th International Conference on Emerging ELearning Technologies and Applications (ICETA). Presented at the 2019 17th International Conference on Emerging eLearning Technologies and Applications (ICETA), pp. 149–155. https://doi.org/10.1109/ICETA48886.2019.9040136

Dogan, S., Akbal, E., 2017. Analysis of mobile phones in digital forensics, in: 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Presented at the 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1241–1244. https://doi.org/10.23919/MIPRO.2017.7973613

Drake, J.J., Lanier, Z., Mulliner, C., Fora, P.O., Ridley, S.A., Wicherski, G., 2014. Android hacker's handbook. Wiley Publishing.

Hassan, M., Pantaleon, L., 2017. An investigation into the impact of rooting android device on user data integrity, in: 2017 Seventh International Conference on Emerging Security Technologies (EST). Presented at the 2017 Seventh International Conference on Emerging Security Technologies (EST), pp. 32–37. https://doi.org/10.1109/EST.2017.8090395

Hoog, A., 2011. Android forensics: investigation, analysis and mobile security for google android. Elsevier.

Islam, M.N., Islam, I., Munim, K.M., Islam, A.K.M.N., 2020. A Review on the Mobile Applications Developed for COVID-19: An Exploratory Analysis. IEEE Access 8, 145601–145610. https://doi.org/10.1109/ACCESS.2020.3015102

Lwin, H.H., Aung, W.P., Lin, K.K., 2020. Comparative Analysis of Android Mobile Forensics Tools, in: 2020 IEEE Conference on Computer Applications(ICCA). Presented at the 2020

IEEE Conference on Computer Applications(ICCA), pp. 1–6. https://doi.org/10.1109/ICCA49400.2020.9022838

Nguyen, T.D., Miettinen, M., Sadeghi, A.-R., 2020. Long Live Randomization: On Privacy-preserving Contact Tracing in Pandemic, in: Proceedings of the 7th ACM Workshop on Moving Target Defense. Presented at the CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, ACM, Virtual Event USA, pp. 1–9. https://doi.org/10.1145/3411496.3421229

Nisioti, A., Mylonas, A., Katos, V., Yoo, P.D., Chryssanthou, A., 2017. You can run but you cannot hide from memory: Extracting IM evidence of Android apps, in: 2017 IEEE Symposium on Computers and Communications (ISCC). Presented at the 2017 IEEE Symposium on Computers and Communications (ISCC), pp. 457–464. https://doi.org/10.1109/ISCC.2017.8024571

Sathe, S.C., Dongre, N.M., 2018. Data acquisition techniques in mobile forensics, in: 2018 2nd International Conference on Inventive Systems and Control (ICISC). Presented at the 2018 2nd International Conference on Inventive Systems and Control (ICISC), pp. 280–286. https://doi.org/10.1109/ICISC.2018.8399079

Scrivens, N., Lin, X., 2017. Android digital forensics: data, extraction and analysis, in: Proceedings of the ACM Turing 50th Celebration Conference - China on - ACM TUR-C '17. Presented at the the ACM Turing 50th Celebration Conference - China, ACM Press, Shanghai, China, pp. 1–10. https://doi.org/10.1145/3063955.3063981

Spolaor, R., Santo, E.D., Conti, M., 2018. DELTA: Data Extraction and Logging Tool for Android. IEEE Transactions on Mobile Computing 17, 1289–1302. https://doi.org/10.1109/TMC.2017.2762692

Wu, S., Xiong, X., Zhang, Y., Tang, Y., Jin, B., 2017. A general forensics acquisition for Android smartphones with qualcomm processor, in: 2017 IEEE 17th International Conference on Communication Technology (ICCT). Presented at the 2017 IEEE 17th International

Conference on Communication Technology (ICCT), pp. 1984–1988. https://doi.org/10.1109/ICCT.2017.8359976

Xiong, L., Shahabi, C., Da, Y., Ahuja, R., Hertzberg, V., Waller, L., Jiang, X., Franklin, A., 2020. REACT: real-time contact tracing and risk monitoring using privacy-enhanced mobile tracking. SIGSPATIAL Special 12, 3–14. https://doi.org/10.1145/3431843.3431845

Yates, M., 2010. Practical investigations of digital forensics tools for mobile devices, in: 2010 Information Security Curriculum Development Conference on - InfoSecCD '10. Presented at the 2010 Information Security Curriculum Development Conference, ACM Press, Kennesaw, Georgia, p. 156. https://doi.org/10.1145/1940941.1940972

Zhang, S., Costa, S., 2016. A Survey Study of Young Generation's Mobile Phone Usage and Security Concerns, in: 2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). Presented at the 2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp. 321–324. https://doi.org/10.1109/PDCAT.2016.075