UNIVERSITY OF TARTU
Institute of Computer Science
Cyber Security Curriculum

Vsevolod Djagilev

Android Chat Application Forensic Process Improvement & XRY Support

Master's Thesis (30 ECTS)

Supervisors: Toomas Lepik
Raimundas Matulevičius

Tartu 2017

Android suhtlemistarkvara digitaalse ekspertiisi protsessi paranemine & XRY abi

Annotatsioon: Tänapäeval seisab maailma silmitsi kiire mobiilseadmete arenguga ning see nõuab digimaailmas kohtuekspertiisi valdkonda. Eriti on see seotud mobiiltelefonide ja kaasaskantavate seadmetega, millel on erinevad platvormid ja viisid andmete salvestamiseks. See nõuab konkreetseid teadmisi, kuidas neid andmeid eraldada ja töödelda. Andmete eraldamine, analüüsimine ja esitamine inimesele loetaval kujul on kolm põhilist väljakutset, millega ekspertiisitöötajad puutuvad kokku igapäevaselt. Kõigil neist on kogum küsimusi ja takistusi. Teiseks ja kolmandaks on osad, mis on esitatud käesoleva väitekirjaga. Isegi kui valdkonnas on kogum äratuntavat (spetsialistide poolt) tarkvara, ei toeta need alati hilisemaid andmete formaate ja seetõttu ei saa pakkuda igal ajal inimestele loetavat varianti. Probleemide lahendamiseks on loodud ekspertide teenus, selle vestluserakenduse analüüs on tehtud nii käsitsi kui ka automaatselt.

Selle töö peamine tulemus lubab mitte vaid otsingut toestada, vaid kirjutada mooduleid ka Python'is, mis kitsendab otsingut ning iga moodul mõistab vajadusel esitatud failiformaati. Tulemused näitavad automaatotsingu ja -eraldamise häid ja halbu külgi ning võrdlevad analüüsitulemusi manuaalse lähenemisega (kui eksperdid analüüsivad faile käsitsi). Kommertsvahendit XRY toetab hulk vestlusrakendusi, mida võrreldakse peamise tulemuste tabeliga. Vähesel hulgal analüüsitakse avatud lähtekoodiga tarkvara (nende andmebaasi skeemi läbi erinevate versioonide), näitamaks et vestlusrakenduse andmete salvestamise vorm võib muutuda, mis vajaks kommertstarkvara uuendusi või käsitsi kogu andmete lugemist ja töötlemist.

Võtmesõnad: digitaalne kohtuekspertiis, android, suhtlemistarkvara, sqlite andmebaas, andmete analüüs

CERCS: P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Android Chat Application Forensic Process Improvement & XRY Support

Abstract: Nowadays world faces rapid mobile devices development and so requires forensic field in digital world. This is especially related to mobile phones & wearable devices, with various platforms and different ways of storing data. This requires certain knowledge on how to extract and process that data. Extracting, analyzing and presenting data in human readable way are three challenges, that each forensic specialist face in the working field. Each one of listed, have a set of issues and obstacles. Second and third are the parts, which are presented in this thesis. Even if there is a set of recognizable (by specialists) software in the field, it is not always support the latest data formats and therefore cannot provide human readable variant all the time. To solve a set of problems a forensic utility has been created, both manual & automated analysis of chat application data has been done.

Main result in this work allows not only to perform a search, but to write a modules in Python, which can make search narrower and each of modules can understand particular file format, if needed. Result shows, good and bad sides of automated way of searching and extracting results and compare analysis results with manual approach (as when forensic specialist do analyze files manually). A commercial tool - XRY, have a list of supported chat applications, which will be compared to the main results table. Few open source applications code will be analyzed (their database schema throughout different versions), to show, that chat application data storage format might change, which would require commercial software update or manually read and process all data.

Keywords: computer forensics, android, chat application, sqlite database, data analysis

CERCS: P170, Computer science, numerical analysis, systems, control

Abbreviations ja Definitions

OS	Operating System
YAML	Yet Another Markup Language (Yaml Ain't Markup Language)
JSON	JavaScript Object Notation
HTML	HyperText Markup Language
XRY	A tool (software + hardware) that can extract data from mobile devices
ADB	Android Debug Bridge
EULA	End User License Agreement
DESEFU	Digital Evidence Search Extract Forensic Utility
XML	eXtended Markup Language

Contents

List of Figures

List of Tables

1 Introduction

Mobile devices take essential place in modern society and this tendency continues to grow and spread rapidly. During only 5 years (from 2010 up to 2015), mobile phones reached great results [4] in replacing desktop computers. This changes gives a great flexibility for criminals cooperation and communication, comparing to desktop computers or other methods, including the fact that digital forensics on mobile devices can be more complicated and sometimes requires help of third party forensics companies [9]. Even if mobile forensics companies do provide a software (in this thesis, XRY [15] is being reviewed), which do not require deep forensics knowledge and this software have great user experience, they do not always support very latest versions of chat applications. Forensics software might not display chat messages in human readable way if chat application had major update. Update itself may deprecate previous message storage way, format or even path. Also if there is missing support for not very popular programs, then investigator should find by himself a file with evidence and make it more human readable. Knowing main information about data storage ways, paths and message databases structures helps investigator to save time.

Written thesis partly solves a problem of missing application support in commercial software (XRY [15]). By creating a solution (automated program) which helps investigator to find where messages are stored and export them, we would help specialists in different ways:

- Decrease evidence search time
- Narrow search by using different approaches and modules
- Export some specific formats in more human readable way
- Export whole search process as JSON format, which could be read and parsed by other software (specifically developed for this purposes)

This thesis will show both manual and automated methods of search & analysis upon different chat applications for Android platform. The reason for Android is simple, it's the most popular platform, where the probability to meet it in the case is much higher, rather than other platforms (Android has 86.2% of market share in second quarter of 2016 [7]). A table with supported chat applications, throughout different versions of XRY also has been done for comparison with main automated & manual result tables.

2 Preparations for Analysis

A list of chat applications which will be used for analysis, some of those applications do support calls (voice or video calls), some of them are tightly related to a social network (Facebook or dating applications).

Table 1: A List of Analyzed Chat Applications

Name	**Identifier**	**Version**	**Open source**	**Social**
Messenger	com.facebook.orca	92.0.0.13.70	No	Partly
Hangouts	com.google.hangouts.talk	13.0.134811930	No	No
ICQ	com.icq.mobile.client	6.10	No	Yes
Kik	kik.android	10.16.1.9927	No	No
Odnoklassniki	ru.ok.android	16.10.16	No	Yes
Signal	org.thoughtcrime.securesms	3.20.4	Yes	No
Skype	com.skype.raider	7.21.0.358	No	No
Snapchat	com.snapchat.android	9.42.0.0	No	Yes
Telegram	org.telegram.messenger	3.13.1	Yes	No
Tinder	com.tinder	6.1.2	No	Yes
Viber	com.viber.voip	6.3.1.63	No	No
Vkontakte	com.vkontakte.android	4.4.2	No	Yes
WhatsApp	com.whatsapp	2.16.310	No	No

2.1 Data Acquisition and Analysis

Data acquisition process has been done on Samsung X Cover 3 device, which already have TWRP custom recovery software. TWRP is used to install custom software (like rooting a device or install another firmware) [1]. , data acquisition process does not require any kind of commercial forensic software or tools and logical extraction can be done using ADB utility (version 1.0.32).
Before data acquisition, certain actions are done in chat applications.

Table 2: Actions Before Data Acquisition

Action	Data	Information
Send message	Hello there	-
Send message	You can find the object in stash	-
Send message	This is test message for thesis	-
Make a call	-	Call is done only if chat application supports it

All messages are sent from the phone, where data acquisition will happen. Same rule applies for calls.

2.1.1 Possible Solutions for Data Acquisition

Possibilities of data acquisition on android devices is not limited in provided list, but required for research of how application data is being stored on device, in our case - chat applications.

Data Backup

A method which allows to save a backup (reserved copy) of data and restore it later if needed. Unfortunately this approach cannot make a backup of all applications. During android application development it's possible to set an option where android OS will or will not allow to backup application data. The option android:allowBackup is set in application manifest file and is true by default[3]. During the research, backup were possible only for a few applications as Vkontakte, Kik, Viber, Odnoklassniki. This variant is not appropriate as data acquisition method because not all applications are possible to backup and backed up copy folder and file names are changed

Using Emulator

It's possible to use an emulator, this option gives full access to the filesystem. But this approach opens few issues: (1) EULA of some chat applications does not allow to run application in emulated environment. (2) AVD and Genymotion (emulators) do not have Google play services installed.

Custom Recovery

Recovery is a limited boot mode, independent of your normal Android operating system. In the stock recovery, you can install OTA update packages, wipe data, and wipe the cache partition. The feature set available depends on the specific

recovery installed[2]. Using a custom recovery gives a possibility to have a full access to device. In order to access recovery mode - 3 keys should be pressed simultaneously: Volume UP + Home key + Power key until TWRP logo appears. In this mode data acquisition itself can done using ADB pull command.

2.1.2 Manual Analysis

For data analysis a list of tools will be used.

- DB Browser for SQLite (version 3.9.1). For browsing SQLite format databases
- HxD - Hexeditor (version 1.7.7.0). Reading and analyzing files in hexademical view

Manual analysis includes evidence data search by corresponding names:

- Database names like `messages`, `calls`
- Database tables like `messages`, `calls`, `conversations`
- Analyze the content of database records (timestamps, type of messages)

2.1.3 Automated Analysis

For automated analysis, there is a application has been developed using Python programming language. This program name is DESEFU [6] (Digital Evidence Search Extract Forensic Utility). This program reads a configuration file, where is written a set of rules on how analyze a set of files (filter them out, which data to collect, which data to extract).

2.2 Goals of Analysis

Main goals of analysis are to find interesting data for investigator. Main evidence objects are

- Messages and their content
- Calls which has been done using messaging applications[1]
- Identify additional details about found data (example: When message was sent)

Both of those can be found in SQLite databases and certain tables in them.

[1] Calling on other phones is not supported by all chat applications

3 Manual Analysis

Manual analysis consists of main parts:

- Find a file where messages are stored
- Find a file where call logs are stored
- Place where file is located
- Describe record:
  - Which file format is used?
  - Is data encrypted
  - Does data **always** contain some other binary information? (Example: Messages are not in plain text, but do contain some other information, like on Figure 1)

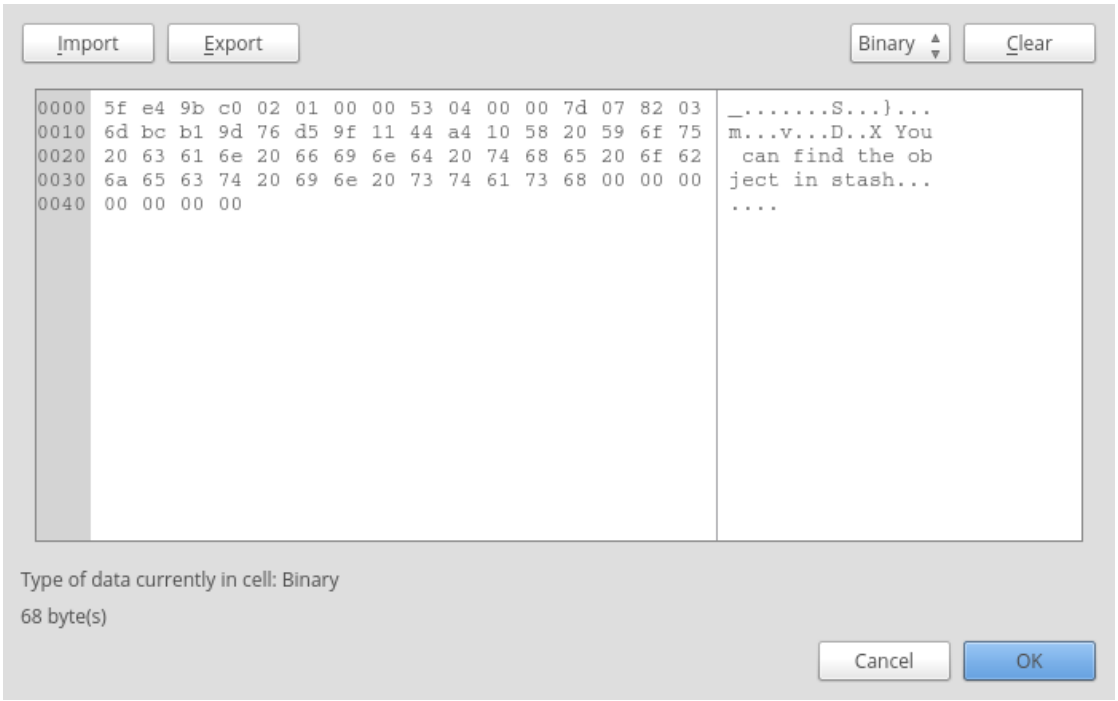


Figure 1: Telegram Message Content

On Android platforms, data related to applications is stored in `/data/data/{application_id}`. Application identifier can be found on Google Play page. Messenger application url is `https://play.google.com/store/apps/details?id=com.facebook.orca`. In this case identifier is `com.facebook.orca`. Once messages has been sent and calls has been done, all data has been exported from the phone. Using command:

```
$ adb pull /data/data/com.facebook.orca/ folder/
```

3.1 Results

All results[2] were in SQLite 3 format. A file database format which is stored on a disk drive[13].

3.1.1 Messages

Table 3: Manual Data Analysis Results

Application	**Path**	**Table**	**E**	**M**
Messenger	databases/threads_db2	messages	N	N
Hangouts	databases/babel3.db	messages	N	N
ICQ	databases/agent-dao	MESSAGE_DATA	N	N
Kik	databases/[UUID].kikDatabase.db	messagesTable	N	N
Odnoklassniki	databases/odnklassniki.db	messages	N	Y
Signal	databases/messages.db	sms	Y	-
Skype	files/[ProfileName]/main.db	Messages	N	N
Snapchat	databases/tcspahn.db	Chat	N	N
Telegram	files/cache4.db	messages	N	Y
Tinder	databases/tinder.db	messages	N	N
Viber	databases/viber_messages	messages	N	N
Vkontakte	databases/vk.db	messages	N	N
WhatsApp	databases/msgstore.db	messages	N	N

Column E means encrypted. Where column M means mixed (information **always**[3] contain other binary data). Values Y means Yes and N means No.
Value in square brackets $[UUID]$ stands for Unified Unique IDentifier[8].

[2]Digital evidence containing messages and calls

[3]Sometimes messages can contain some XML or other data, in this case, meant only those messages where some other binary data always exists

The [$ProfileName$] field means Skype service account name.

Common approaches in finding correct database were:

- Database name gives a tip about it's content (For example: "viber_messages", application name, "main.db")
- Databases which contain messages might differ in size (Figure 3)
- Other articles in the internet might help to find exact database name (but it might be deprecated, if article is old)

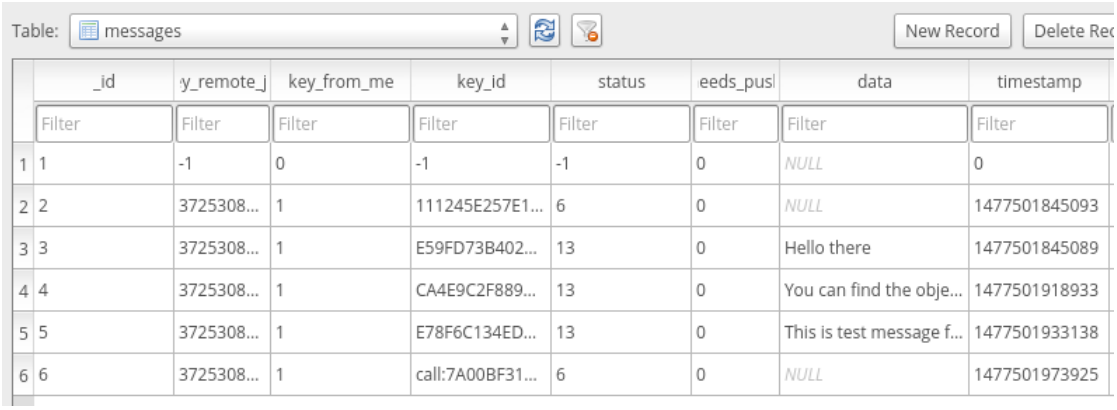
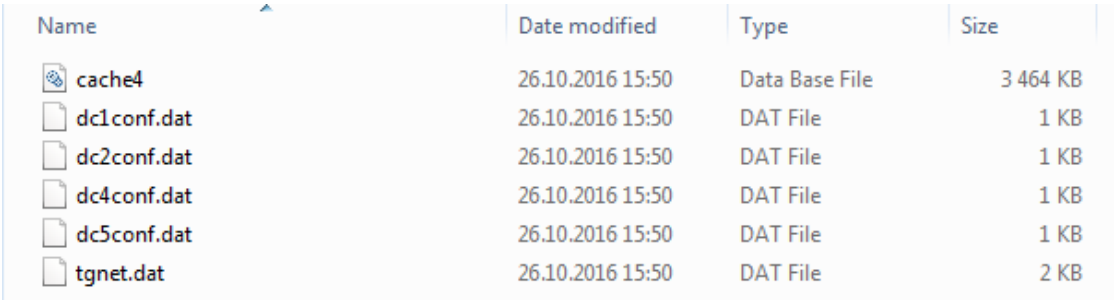


Table: messages

New Record | Delete Rec

	_id	y_remote_j	key_from_me	key_id	status	eeds_pusl	data	timestamp
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	-1	0	-1	-1	0	NULL	0
2	2	3725308...	1	111245E257E1...	6	0	NULL	1477501845093
3	3	3725308...	1	E59FD73B402...	13	0	Hello there	1477501845089
4	4	3725308...	1	CA4E9C2F889...	13	0	You can find the obje...	1477501918933
5	5	3725308...	1	E78F6C134ED...	13	0	This is test message f...	1477501933138
6	6	3725308...	1	call:7A00BF31...	6	0	NULL	1477501973925

Figure 2: WhatsApp Messages Table



Name	Date modified	Type	Size
cache4	26.10.2016 15:50	Data Base File	3 464 KB
dc1conf.dat	26.10.2016 15:50	DAT File	1 KB
dc2conf.dat	26.10.2016 15:50	DAT File	1 KB
dc4conf.dat	26.10.2016 15:50	DAT File	1 KB
dc5conf.dat	26.10.2016 15:50	DAT File	1 KB
tgnet.dat	26.10.2016 15:50	DAT File	2 KB

Figure 3: Telegram Databases List

3.1.2 Calls

Not all chat applications support calling function, in provided table, stored information of those applications where it was possible to make a call. Sometimes information about call can be found in different places at once. For example: information about outgoing call is stored in messages table and at the same time information about calls can be found in different database.

Table 4: Manual Data Analysis Results

Application	**Path**	**Table**	**E**	**M**
Messenger	databases/call_log.sqlite	person_summary	N	N
Hangouts	databases/babel3.db	messages	N	N
ICQ	databases/agent-dao	MESSAGE_DATA	N	N
Signal	databases/messages.db	sms	Y	-
Skype	files/[ProfileName]/main.db	Calls	N	N
Viber	databases/viber_data	calls	N	N
WhatsApp	databases/msgstore.db	messages	N	N

4 Approach to Automated Analysis

In order to make search more automated, there were developed a several programs, which might improve the way how digital evidence might be searched and extracted. The first program is *DESEFU*, which is abbreviation of Digital Evidence Search Extract Forensic Utility. It has open source code [6] and written in Python programming language. Second program is *DESEFUexport*, which reads result file of *DESEFU* program and converts it to HTML page. This operation makes result more readable and it's possible to print it in PDF format later.

4.1 DESEFU

This program was developed to make easier search for digital evidence. It uses modular approach and allows to narrow search. Program itself is written on Python programming language and requires version at least 3.4. A list of library dependencies:

- colorama (To make colored output in console, Windows OS support included)
- termcolor (Create colored messages in more easier way)
- ruamel.yaml (Parse YAML file format)
- jsonpickle (Improved serialization and unserialization of a JSON file)

The reasons of choosing python programming language are:

- Language is easier to learn (rather that C++). So other forensic specialists can contribute and develop modules as well.
- It's easier to do modular approach of the application
- It works across all main platforms (Linux, Windows, MacOS)

DESEFU main workflow is:

1. Create a configuration (described in YAML format) [21]
2. Extract evidence folder to your system
3. Execute program with configuration and evidence folder paths as a parameters
4. Analyse result file (JSON format) or better export it as HTML using DESEFU export program

Command, which executes program:

```
./desefu.py /path/to/config.yml /path/to/evidence_folder
```

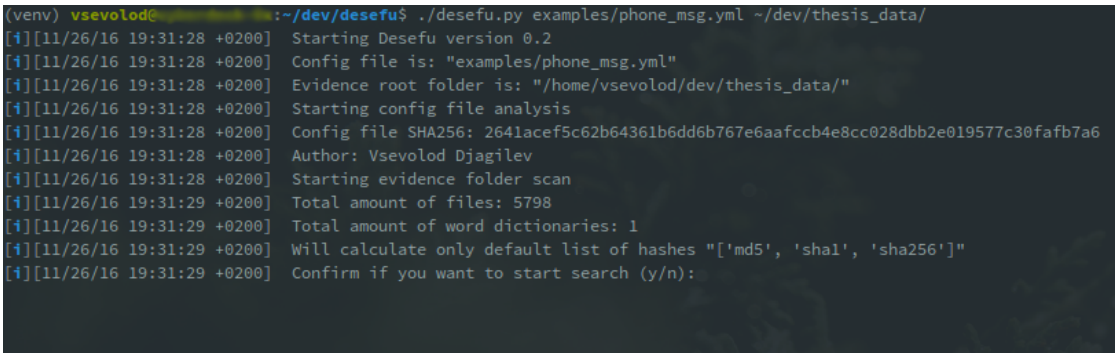


Figure 4: Execute desefu.py

Result file name is having this format: `result_26102016_193328.json`. The numbers represent current date and time, a file name provided in example were created at 26 of October, 2016, 19:33:28.

Steps which are done by the program are:

1. Parse provided arguments (Make sure configuration path & path to evidence folder do exist)
2. Load a configuration file data
3. Analyze configuration file
   (a) Parse YAML file data
   (b) Make sure all required fields exist in configuration file
   (c) Collect evidence files from evidence folder
   (d) Gather data about every module chain
   (e) Get and check all modules in every chain
4. Ask user about starting search & extract procedure
5. Begin search & extract procedure
   (a) Pass list of files to module chain
   (b) Execute N module of chain
      i. Filter not needed files [4]
      ii. Collect data
      iii. Extract data
6. Write results to the file

[4]Each module can have it's own strategy on handling evidence file. Some of them just filter files, some of them only extract, some of them can have combined variant

4.1.1 DESEFU Configuration File

Search & data extraction rules are described in each configuration file by investigator. Configuration file is written in YAML format, example which is used for this thesis is in Appendix 7. Most important and required parts of configuration file are:

- Author
- Module chain
- A list of modules (in each module chain)
- (Optional) module chain in module

4.1.2 Module Chain

Module chain is a list of modules which are executed one after another. A root module chain can have it's name (For example "sqlite_dbs" in Appendix 7). Every module can have it's module chain as well (it can be added via option: `sub`), so when module has been executed, module chain execution phase is started.

```
search:
  ModuleChainIdentifier:
    -
      mod: file.Extension
      args: ['jpg', 'jpeg']
    -
      mod: file.FileHeader
      args: [ [...] ]
```

In provided code example, root module chain identifier is `ModuleChainIdentifier`. There can be a several identifiers and each one of them will have the same file list which was collected before.

4.1.3 Module

Each module is an individual file located in `desefu/modules` folder, which is written in Python programming language. This file represents a class with few functions:

```
class ExampleModule(AbstractModule):
  def check(self):
```

```
  pass
def check_arguments(self):
  pass
def is_filter_files(self) -> bool:
  return True
def is_collect_data(self) -> bool:
  return False
def is_extract_data(self) -> bool:
  return False
def description(self) -> str:
  return "Module description"
def do_filter_files(self):
  pass
def do_collect_data(self):
  pass
def do_extract_data(self):
  pass
def execute(self):
  # Executing filter, collect and extract functions
  pass
```

There is two methods which have to exist in each module:

- `check` - Used for checking if module is ready to be executed (can be used for those modules which use external software)
- `check_modules` - Used to check inserted arguments (it's happening during configuration file analysis)

Each module has it's own 3 main properties

- File filtering (for example module `file.Extension` can filter files by extension, next module which receives a file list already have only those files which are filtered by some extension)
- Data collection (`search.Dictionary` module not only to filter files that contain certain words from dictionary files, but collect a list of words which were found in each file)
- Data extraction. Is required not just to scan raw files, but gather information in more intelligent way (Example: `file.type.SqliteDatabase` module which works with SQLite databases and extracts data using SQL queries.)

In DESEFU all modules are located in folder `modules`. For example module `SqliteDatabase` is located in `modules/file/type/SqliteDatabase.py` file. In configuration file, this module is written as `mod: file.type.SqliteDatabase`. Each module has it's own rules how to write arguments and extract options, those are checked in `check_arguments` function.

4.1.4 Configuration File Visualization

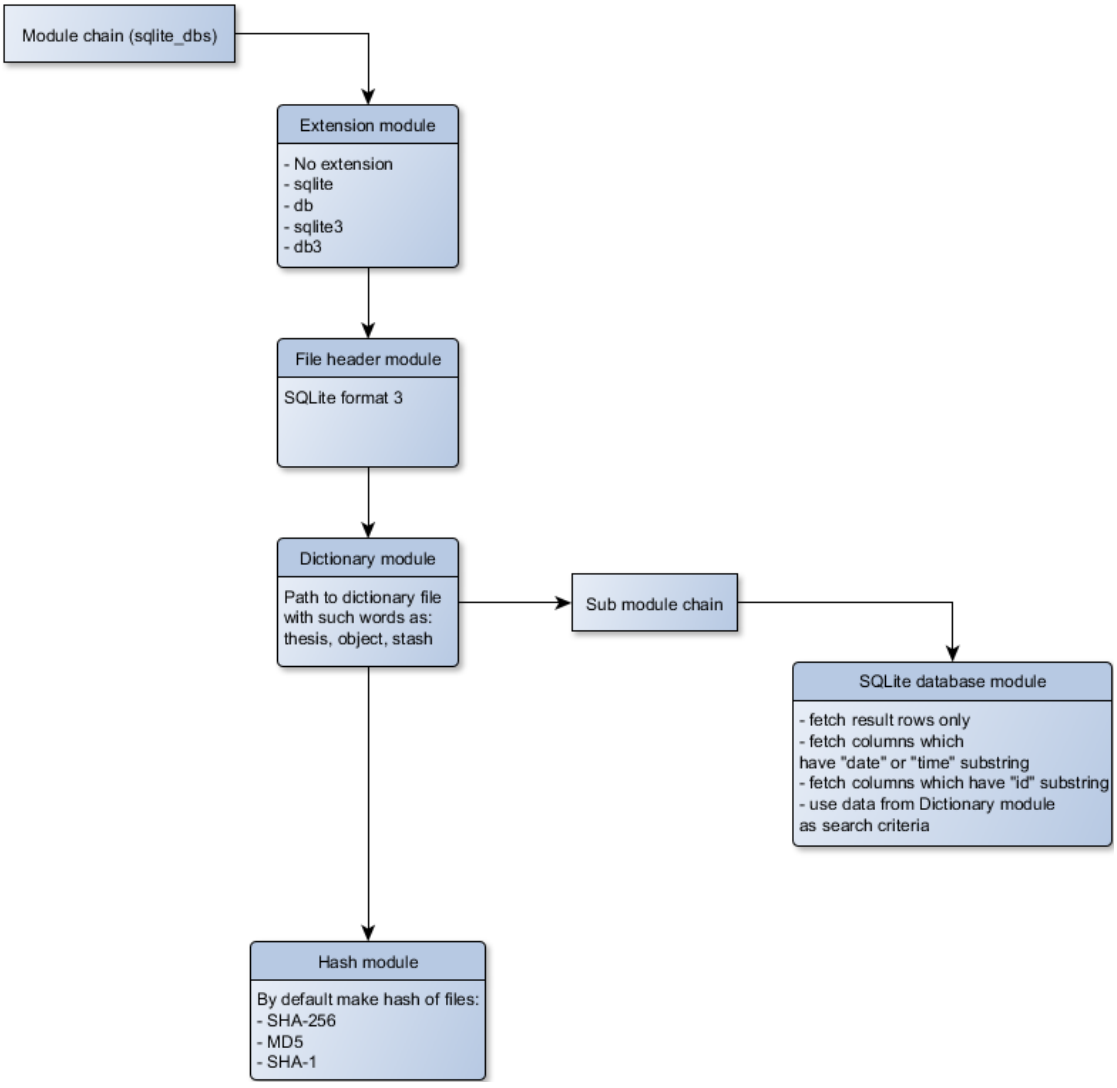


Figure 5: Visualized Configuration File

4.1.5 DESEFU Result File

DESEFU program result - is a serialized JSON file. Program result file contains information which were collected by each module in each module chain. An example of top level keys in result file:

```
{
  "author": "Vsevolod Djagilev",
  "evidence_folder": "/path/to/evidence/folder",
  "files_count": 5168,
  "meta": {
    "description": "..."
  },
  "config": {
    "file": "examples/phone_msg.yml",
    "sha256": "2641acef5c62b64361b6dd6..."
  },
  "result": [
    {
      "module_chain_id": "ModuleChainIdentifier",
      "modules": []
    }
  ]
}
```

In provided result file, some of the fields are copied from configuration file (like `author` and module chain identifiers, module names and their opptions). As seen on example `result` contains a list of module chains, which then contain a list of modules.

Result File: Module JSON Format

Example of module JSON format:

```
{
  "modules": [
    {
      "title": "Module title (which later can be shown in extracted
      ↪  HTML file)",
      "mod": "file.Extension",
      "files_count": 123,
      "data": {}
```

```
    },
    {
      "title": "Filter by file header",
      "mod": "file.FileHeader",
      "files_count": 20,
      "data": {}
    },
    {
      "title": "Filter by dictionary and collect data",
      "mod": "search.Dictionary",
      "files_count": 2,
      "files": ["file1.dat", "file2.db"]
    }
  ]
}
```

Each module record has `files_count` value - this shows how many files were filtered after module execution. A key `data` is an object which stores collected data. A key `files` is always stored in the last module record (to obtain less storage and do not repeat information, because only latest result is matters in this case). A key `extract_data` contains key-value pairs containing extracted data information.

Result File: Collected Data

Collected data stored in result file can have two approaches to store values:

- A simple array of strings for each file
- An array of tuples (consists of a number of values separated by commas [10]) for each file

Example of module result file which collects file hashes:

```
{
    "[path]/com.viber.voip/databases/viber_messages": [
    {
      "py/tuple": [
        "md5",
        "5196bbd265e6d469821231fc168b0ea7"
      ]
    },
```

```
    {
      "py/tuple": [
        "sha1",
        "132aa238b62d8e7ceadfbf56b9e1428d9684b208"
      ]
    }
  ]
}
```

A key `py/tuple` is automatically created by **jsonpickle** library. To read and write type of values equally in other programs (such as DESEFU export).

Result File: Extracted Data

Extracted data storage way is oriented on table view. For each file there is a keys (that represent table title) and this table has tuple data structure type [10]. Tuple zero index value is a list of columns for the table view, a first index value is a list of lists which contain values.

```
{
  "[path]/com.viber.voip/databases/viber_messages": {
    "messages": {
      "py/tuple": [
        [
          "_id",
          "message_global_id",
          "extra_upload_id",
          "extra_download_id",
          "group_id",
          "conversation_id",
          "participant_id",
          "date",
          "date_real",
          "body"
        ],
        [
          [
            154,
            0,
            0,
            null,
```

```
            0,
            2,
            3,
            1476391325043,
            0,
            "You can find the object in stash"
          ],
          [
            155,
            0,
            0,
            null,
            0,
            2,
            3,
            1476391339776,
            0,
            "This is test message for thesis"
          ]
        ]
      ]
    }
  }
}
```

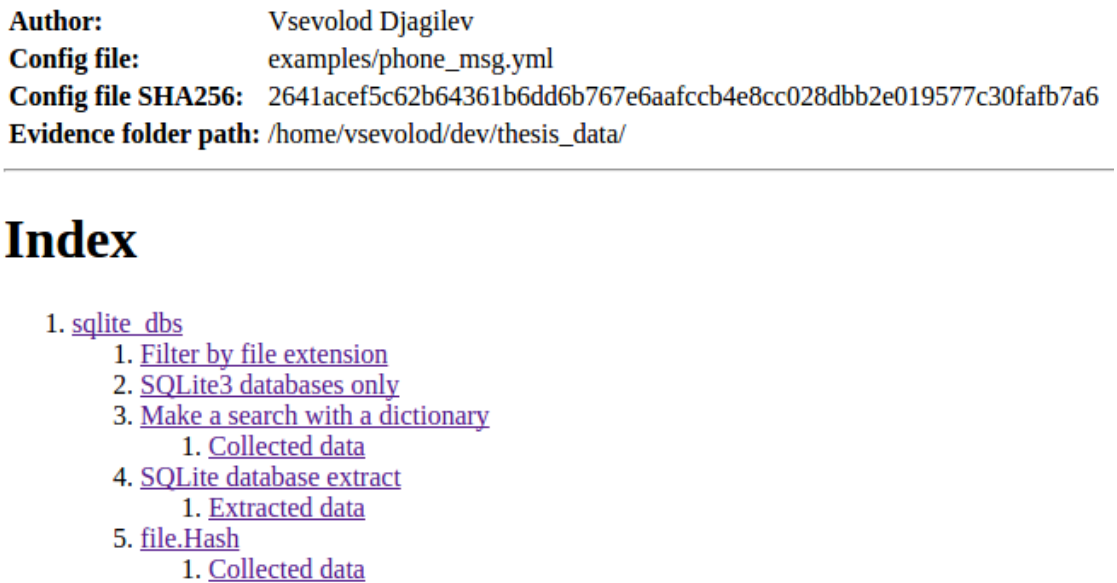
The module `file.type.SqliteDatabase` provides such `data_extract` option values. The table in this particular case mean not SQLite 3 database table, but table view.

4.2 DESEFU Export

DESEFU export is a program written in Python programming language, which can convert DESEFU result file to HTML page file. The same as DESEFU, this project is open source [5]. To execute a program, this command is needed:

```
./desefu_export.py /path/to/result_123.json
```

This command will create a single html page file.



Author: Vsevolod Djagilev
Config file: examples/phone_msg.yml
Config file SHA256: 2641acef5c62b64361b6dd6b767e6aafccb4e8cc028dbb2e019577c30fafb7a6
Evidence folder path: /home/vsevolod/dev/thesis_data/

Index

1. sqlite_dbs
    1. Filter by file extension
    2. SQLite3 databases only
    3. Make a search with a dictionary
        1. Collected data
    4. SQLite database extract
        1. Extracted data
    5. file.Hash
        1. Collected data

Result

sqlite_dbs

Figure 6: DESEFU Export HTML Page

HTML page file basically represents everything what is listed in JSON result file. A collected of data (which has two main view variants) or

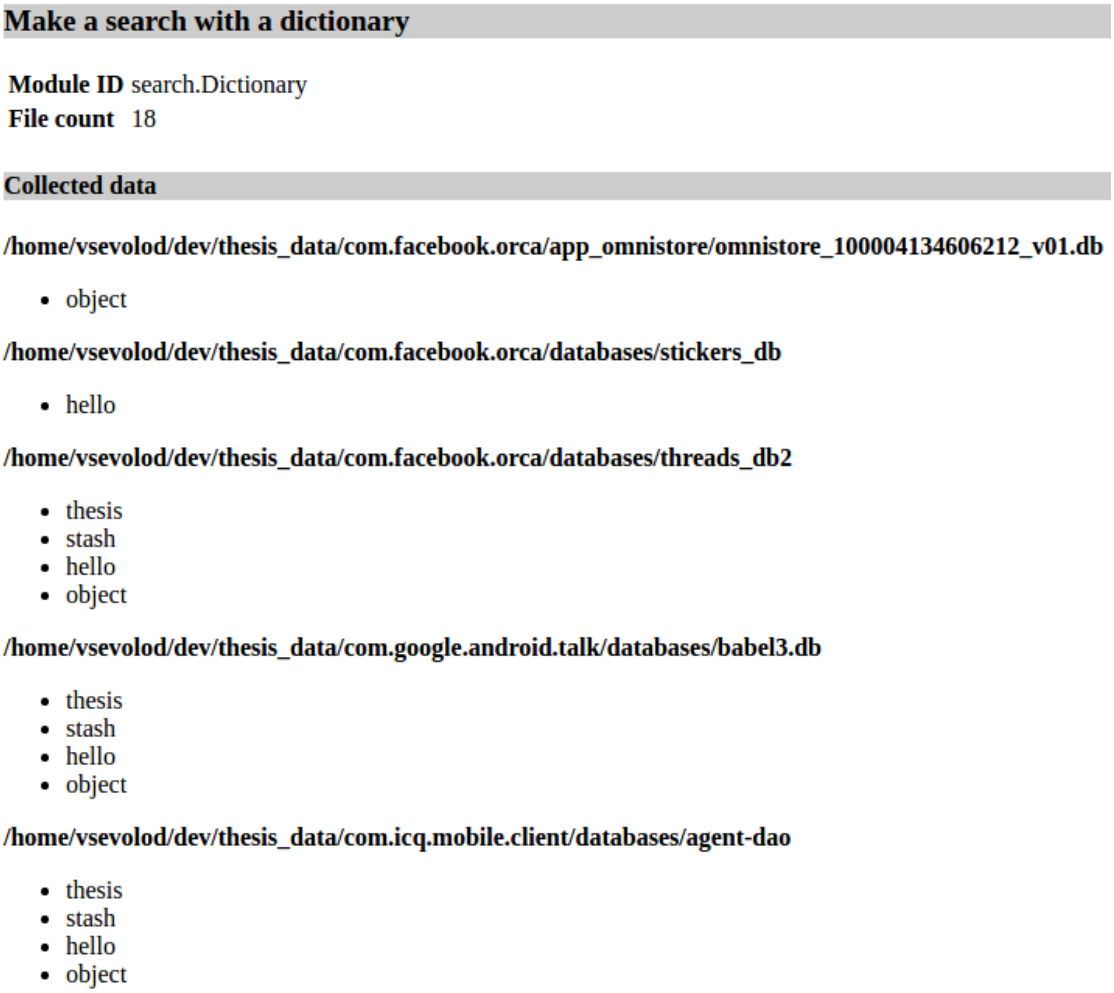


Figure 7: Dictionary Module. Collected Data View

file.Hash

Module ID file.Hash
File count 18

Collected data

/home/vsevolod/dev/thesis_data/com.facebook.orca/app_omnistore/omnistore_100

md5	5bdb654e5706efa9cb2af6853a81e97e
sha1	cb6d5dc5b5891f4147683072ce77e4d55fbd9985
sha256	a60f18c40d33e7fbe4549320e7f7e446be87ee055cd92a8531bdc133a5f2a2c5

/home/vsevolod/dev/thesis_data/com.facebook.orca/databases/stickers_db

md5	773417d180b58334fa9a8daa68ae4ab3
sha1	debb44f85ae5fd90f15fcc8fb082fa540b60d1f2
sha256	bdc53a646d1bbd6a5b2d6d8441849c573da04eebab22e322a3603bc5a90244f6

/home/vsevolod/dev/thesis_data/com.facebook.orca/databases/threads_db2

md5	9b7972092dc290883589f3a3e0c8005f
sha1	c60b27408163cb57cdb97b799d0bda1f782ed7f2
sha256	dafd4f1622abb9edcb49e547daa4f1bca56fcf7a1761b8c6b0a0540b01d86c51

/home/vsevolod/dev/thesis_data/com.google.android.talk/databases/babel3.db

md5	9c1dfb1f991bd7711671ae90b6ef9069
sha1	a850c0592ca7677ca90dae088dad9a57e0a1fc74
sha256	0184e7cf6219423e2959902802768d93e79ba3751592a79538475e5920cdeaea

/home/vsevolod/dev/thesis_data/com.icq.mobile.client/databases/agent-dao

Figure 8: Hash Module. Collected Data View (Another Variant)

Unlike single list with data, approach with highlighted row-columns gives more readable format.

/home/vsevolod/dev/thesis_data/com.vkontakte.android/databases/vk.db

messages

mid	random_id	time	text
13395	-1701208395	1477256259	Hello there
13396	-1701208396	1477256301	You can find the object in stash
13397	-1701208393	1477256324	This is test message for thesis

/home/vsevolod/dev/thesis_data/com.whatsapp/databases/msgstore.db

messages

_id	key_remote_jid	key_id
3	37253[redacted]s.whatsapp.net	E59FD73B402211B079A74251F13F85
4	37253[redacted]s.whatsapp.net	CA4E9C2F8898F3FDD959D037186783
5	37253[redacted]s.whatsapp.net	E78F6C134ED93D148CC82D3C96E093

messages_fts

Figure 9: Table Data Display of Vkontakte & WhatsApp Applications

4.3 Search & Extract Result (For Chat Messages)

Table 5: Automated Data Analysis Results

Application	**Found**
Messenger	Yes
Hangouts	Yes
ICQ	Yes
Kik	Yes
Odnoklassniki	Yes
Signal	**No**
Skype	Yes
Snapchat	Yes
Telegram	Yes
Tinder	Yes
Viber	Yes
Vkontakte	Yes
WhatsApp	Yes

There is no results for calls, main reason is: This kind of search can be done manually. Database tables contain only technical information (like call duration, contact identifier, date and time). Main drawback is that some of the files which contained keywords from dictionary files were included to the result:

- Facebook
  - /app_omnistore/omnistore_100004134606212_v01.db
  - databases/stickers_db
- Skype
  - [$skype\_profile$]/media_messaging/emo_cache_v2/asyncdb/cache_db.db
- Snapchat (tcspahn.db file)
  - StickerSynonymTable
  - StickerTagTable

5 Comparing Automated & Manual Analysis

5.1 Manual Analysis

Good Sides

- Concentration on details
- Table names & Columns have understandable naming, which corresponds to it's purpose. Mostly it is easy to search for evidence.
- Not all applications (commercial forensic software) support some unknown, brand new or those applications which had major update with significant changes. Analyzing data manually is only one solution
- Finding evidence in not obvious places (encrypted database storage)

Bad Sides

- Analyzing big amount of data (and understand how it is stored) is challenging and taking more time.
- All information needs to be converted into more understandable way (to be presented in court). This requires post processing of evidence data.

5.2 Automated Analysis Approach

Good Sides

- Extremely fast in finding new data (comparing to manual analysis)
- More flexible
- Provides more presentable way of showing digital evidence

Bad Sides

- Not needed data might be included (in this thesis 'hello' and 'object' keywords were met quite often in such elements as stickers or smiley's)
- As seen in DESEFU program result. Signal messaging application data were not found using this approach (messages were encrypted)
- It requires writing additional code for non standard situation
- Missing details (type of message, sender, events type)

As a recommendation, best approach would be a combination of both.

6 XRY Support

Table 6: Chat Applications Support in XRY

Application	**XRY 7.2**	**XRY 7.1**	**XRY 7.0**	**XRY 6.16**
Messenger	Yes	Yes	Yes	Yes
Hangouts	No	No	No	Yes
ICQ	Yes	No	No	No
Kik	No	Yes	Yes	Yes
Odnoklassniki	Yes	No	No	No
Signal	No	No	No	No
Skype	Yes	Yes	Yes	Yes
Snapchat	Yes	Yes	Yes	Yes
Telegram	Yes	No	No	No
Tinder	Yes	No	No	No
Viber	Yes	No	Yes	Yes
Vkontakte	No	No	No	Yes
WhatsApp	Yes	Yes	Yes	Yes

XRY, pronounced "ex-arr-why", is a forensic system specifically designed for analyzing mobile digital devices written by Micro Systemation. The software is designed to run on a Windows computer and will retrieve information from mobile devices for immediate display of the results or files can be saved for later analysis. At the time of writing support levels included smartphones, gps units and mobile tablets such as the iPad. [16]

All supported applications in XRY 6.16 [17], XRY 7.0 [18], XRY 7.1 [19] and XRY 7.2 [20], were found in release notes document. Some of those versions are not available on the internet. Some application which was supported in version **6.16**, but not supported in future versions, potentially might work in future releases as well.

6.1 Application Continuous Support Issues

Commercial software should always keep an eye on latest chat application updates. It does not necessary means to check and test all possible changes. For example if we do follow semantic versioning [11], it does not always mean that minor updates and patches for applications will affect the way data is stored, but nothing guarantees the opposite.
Telegram & Signal messaging applications had some database schema changes even through out minor version upgrade. As seen in the code [12] for Signal messaging application, every new version has some new indexes or columns added to database schema. Same changes exist for Telegram chat application [14], where database columns are changed throughout the versions. In this case we talk about open source applications, where source code is available and could be seen and understood, with proprietary applications, it make things more harder, since it's impossible to track those changes without making blackbox testing every new version and compare results with previous versions. Theoretically it can be improved using automated approach, for example:

1. Execute application in virtual environment
2. Perform some actions
3. Dump the data
4. Find differences with previous versions

But this specific case has main drawback. EULA does not always allow to execute applications in emulated environment. Possible solution would be: To create digital evidence description format, a unified format which can describe what certain sequence of bytes (low level) or data (higher level) meaning and provide more detailed description about each object in the file.

7 Conclusion

New challenges and changing world giving birth to new excellent methods and tools in forensic field and not only digital one. In the information age and general IT rapid development it's a shame not to use all possible potential of IT. Information about evidence data, that was found for each chat application, might come useful for other forensic specialists and law enforcement agencies, but beyond that, newer approaches on data search & extract are shown.

Created forensic utility would definitely ease the job for digital forensic specialists, where manual file analysis cannot be skipped. Automated solution showed that program capable of doing it's job in search and extract activities, even with some of the limitations, like encrypted data, but these cases are mostly special and modules for those can be developed as well.

Extraction program, which could transfer non-readable format into HTML page, also managed to complete it's task in providing more understandable data representation, even if it's done in a very primitive way.

At this point, there is a lot of room for new modules, additions & fixes. Making more test cases and their feedback would give new ideas and improvements to the program in general. A program which has been created for exporting also can be improved, for example, by adding customized view (themes).

Unfortunately chat applications which are not open source, require additional effort in understanding, how data is stored and particular meaning for each byte sequence. Most often databases, which were analyzed, do have appropriate table & column names, but not the values. Possible solution would be digital evidence description program, this description files can be managed by chat application (and not only them) vendors and given to law enforcement agencies, without revealing the source code.

References

[1] *About TWRP.* (Visited: 16.10.2016). URL: https://twrp.me/about/.

[2] *All About Recovery Images. License: Attribution-ShareAlike 3.0 Unported.* (Visited: 17.10.2016). URL: https://wiki.cyanogenmod.org/w/All_About_Recovery_Images.

[3] *Android developers. Application Manifest file description.* (Visited: 16.10.2016). URL: https://developer.android.com/guide/topics/manifest/application-element.html.

[4] Danyl Bosomworth. *Mobile Marketing Statistics compilation.* (Visited: 23.10.2016). URL: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/.

[5] *DESEFU Export program source code repository.* (Visited: 24.11.2016). URL: https://github.com/vdjagilev/desefu-export/.

[6] *Digital Evidence Search Extract Forensic Utility code repository.* (Visited: 17.10.2016). URL: https://github.com/vdjagilev/desefu/.

[7] *Gartner Says Five of Top 10 Worldwide Mobile Phone Vendors Increased Sales in Second Quarter of 2016).* (Visited: 23.10.2016). URL: http://www.gartner.com/newsroom/id/3415117.

[8] *IETF. RFC 4122.* (Visited: 06.11.2016). URL: https://www.ietf.org/rfc/rfc4122.txt.

[9] Jose Pagliery Laurie Segall and Jackie Wattles. *FBI says it has cracked terrorist's iPhone without Apple's help.* (Visited: 23.10.2016). URL: http://money.cnn.com/2016/03/28/news/companies/fbi-apple-iphone-case-cracked/.

[10] *Python documentation. Data structures. Tuples and Sequences.* (Visited: 28.11.2016). URL: https://docs.python.org/3.5/tutorial/datastructures.html#tuples-and-sequences.

[11] *Semantic Versioning.* (Visited: 29.11.2016). URL: http://semver.org/.

[12] *Signal Messaging Application source code part.* (Visited: 29.11.2016). URL: \url{https://github.com/WhisperSystems/Signal-Android/blob/master/src/org/thoughtcrime/securesms/database/DatabaseFactory.java#L704}.

[13] *SQLite. Database File Format.* (Visited: 31.10.2016). URL: https://www.sqlite.org/fileformat2.html.

[14] *Telegram application source code part.* (Visited: 29.11.2016). URL: https://github.com/DrKLO/Telegram/commit/e313885ac540c31ea02c85122907a5845fc576d2#diff-bbdbb10b03771eb1e46a8c1d0424adf4R155.

[15] *XRY - Extract - MSAB page.* (Visited: 11.12.2016). URL: https://www.msab.com/products/xry/.

[16] *XRY. Forensics Wiki. Licensed as (CC BY-SA 2.5).* (Visited: 29.11.2016). URL: http://www.forensicswiki.org/wiki/.XRY.

[17] *XRY Release notes v6.16.* (Visited: 29.11.2016). URL: http://aimtech.ru/upload/XRY_6.16_release_notes_EN.pdf.

[18] *XRY Release notes v7.0.* (Visited: 29.11.2016). URL: https://www.msab.com/download/release_notes/en/english_xry_release_notes/XRY_7.0_release_notes_EN.pdf.

[19] *XRY Release notes v7.1.* (Visited: 29.11.2016). URL: https://www.msab.com/download/release_notes/en/english_xry_release_notes/XRY_7.1_release_notes_EN.pdf.

[20] *XRY Release notes v7.2.* (Visited: 11.12.2016). URL: https://www.msab.com/download/release_notes/en/english_xry_release_notes/XRY_7.2_release_notes_EN.pdf.

[21] *Yet Another Markup Language.* (Visited: 25.11.2016). URL: http://yaml.org/spec/history/2001-05-26.html.

Appendices

A Appendix 7 - DESEFU Configuration File

```
author: Vsevolod Djagilev
meta:
  description: |
    This config file is specifically needed to search for message databases on
    phone. It traverses all possible files (mainly it's SQLite databases)
    and extracts the data

search:
  sqlite_dbs:
    -
      title: "Filter by file extension" # Titles can be anything, used by desefu-extract
      mod: file.Extension
      args: ['', 'sqlite', 'sqlite3', 'db', 'db3']
    -
      title: "SQLite3 databases only"
      mod: file.FileHeader
      args:
        types:
          - [53, 51, 4C, 69, 74, 65, 20, 66, 6F, 72, 6D, 61, 74, 20, 33] # SQLite format 3
    -
      title: "Make a search with a dictionary"
      mod: search.Dictionary
      args:
        dictionary:
          - './tests/modules/search/dictionaries/dictionary1.txt'
          #- './tests/modules/search/dictionaries/dictionary2.txt'
        encoding:
          - 'utf-8'
          - 'ascii'
          - 'latin-1'
          - 'cp866'
      sub:
```

```
33        -
34          title: "SQLite database extract"
35          mod: file.type.SqliteDatabase
36          extract:
37            columns: # If no "result" value is provided, all
               ↪ variants are included
38              - result # Columns where evidence were found (from
                 ↪ parent Dictionary module)
39              - timestamps # Possible timestamp values
40              - id # Fields which might be identifiers - contain
                 ↪ "id" keyword
41            where: ~mod.3.data # "~" is a reference to module number
               ↪ 3
42            order:
43              timestamps: DESC
44            result:
45              - data # Fetch data which were
46    -
47      hmod: file.Hash
```

Non-exclusive licence to reproduce thesis and make thesis public

I, Vsevolod Djagilev (date of birth: 29th of October 1991),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Type Inference for a Fourth Order Logic Formulae

supervised by Toomas Lepik and Raimundas Matulevičius

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tallinn, 06.01.2017