ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

Σχολή Μηχανικών

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Πρόγραμμα Μεταπτυχιακών Σπουδών (ΠΜΣ) Κυβερνοασφάλεια (Cybersecurity)

Πρώτες ενέργειες κατά την αντιμετώπιση συμβάντος ασφαλείας σχετιζόμενο με συσκευές Android

Συγγραφείς

Δονάτος Δόσης & **Μιχαήλ Κωτσής**

ΑΜ: 2010 **ΑΜ: 2015**

Επιβλέπων:

Παναγιώτης Ριζομυλιώτης

Αθήνα, Φεβρουάριος 2022

UNIVERSITY OF WEST ATTICA

Faculty of Engineering

Department of Informatics and Computer Engineering

Cybersecurity

First steps in incident response related to Android Based Devices

Students:

Δονάτος Δόσης	**&**	**Μιχαήλ Κωτσής**
R.N.: 2010		**R.N.: 2015**

Supervisor:

Panagiotis Rizomiliotis

Athens, February 2022

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η μεταπτυχιακή διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/Α	ΟΝΟΜΑ- ΕΠΩΝΥΜΟ	ΥΠΟΓΡΑΦΗ
1	**ΠΑΝΑΓΙΩΤΗΣ ΡΙΖΟΜΥΛΙΩΤΗΣ**	PANAGIOTIS RIZOMYLIOTIS Aug 29 2023 11:05 AM PANAGIOTIS RIZOMYLIOTIS
2	**ΠΑΝΑΓΙΩΤΗΣ ΓΙΑΝΝΑΚΟΠΟΥΛΟΣ**	Panagiotis Giannakopoulos Digitally signed by Panagiotis Giannakopoulos Date: 2023.09.04 00:40:26 +03'00'
3	**ΣΤΕΦΑΝΟΣ ΓΚΡΙΤΖΑΛΗΣ**	STEFANOS GKRITZALIS 30/08/2023 11:56

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΩΝ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Οι κάτωθι υπογεγραμμένοι 1. **Δονάτος Δόσης** του Φωτίου με αριθμό μητρώου **2010** και 2. **Μιχαήλ Κωτσής** του Κωνσταντόινου, με αριθμό μητρώου **2015** φοιτητές του Προγράμματος Μεταπτυχιακών Σπουδών «Cybersecurity» («Κυβερνοασφλάλεια») του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαστε συγγραφείς αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνουμε ότι αυτή η εργασία έχει συγγραφεί από εμάς αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μας, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μας ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μας».

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι …………………… και έπειτα από αίτηση μου στη Βιβλιοθήκη και έγκριση του επιβλέποντα καθηγητή.

Οι Δηλούντες

1.

2.

Table of Contents

Abstract

Digital Forensics as a science is the process of acquiring, preserving, analyzing, and reporting of raw data residing in electronic or digital devices using scientific methods that are demonstrably reliable, verifiable and repeatable, such that they may be used in judicial and other format proceedings (SWGDE, 2014). One of the most particular circumstances that digital forensics must be applied is during the handling of a security incident where the operations of the first responders are extremely important and critical and a mistake is usually irreversible.

Mobile devices are in fact a computer in small form, equipped with high performance processor, huge storage, and enhanced functionality. They have become an essential part of the owner's live since they have drastically revolutionized the way that everyday activities (e.g., connecting with other people, e-banking) are achieved. As a result, a mobile device is now a huge repository that holds sensitive and personal information about its owner. A huge percentage of these personal and sensitive information is originated from social media and instant messaging applications like Viber, WhatsApp, and Telegram. In 2021, the number of mobile devices operating worldwide stood at almost 15 billion and their number is expected to reach 18.22 billion by 2025 (O' Dea S., 2021).

The huge proliferation of mobile devices and the social media and instant messaging applications have also shaped modern crime. Almost in every form of crime, bad actors make use of these technologies for planning and operating. This has, in turn, led to the development of a branch of digital forensics solely dedicated to mobile devices. Mobile forensics have unique characteristics and peculiarities since some of the core principles of other digital forensics do not apply like data preservation. Data inside a mobile device is continuously modified and the standard write protection techniques cannot be applied. For that reason, the actions of a first responder / examiner have even greater impact in the event of handling a mobile device as evidence.

The content of this Thesis is divided into four distinct parts. The first part introduces the concepts of digital criminology when dealing with a security incident, as well as basic concepts related to the forensic examination of a mobile phone device with Android operating system installed. The structure and basic elements of the SQLite database, on which most of the applications on such a device are based, are then analyzed. In addition, well-known and widespread forensic and non-forensic tools and software used to examine both Android devices and databases (SQLite) are presented. The second part presents suggested methods for obtaining administrative rights on two devices of different manufacturer (LG, Samsung) and different operating system version (Android 9 and Android 11), to create forensic copies (physical image) and examine them more effectively. In the third part, a description of a hypothetical scenario based on realistic incidents, the structure of the experiments and the followed methodology are given. Furthermore, in this part is explained the selection of the messaging applications Viber, WhatsApp and Telegram used for the experiments. The fourth part presents an analysis of the aforementioned messaging applications' operation, their examination and finally, the results of the experiments and a comparative analysis among them.

Key words: DFIR, First Responders, Mobile forensics, Android Applications, Viber, WhatsApp, Telegram.

Περίληψη

Η επιστήμη της ψηφιακής εγκληματολογίας (Digital Forensics) είναι η διαδικασία της συλλογής, διατήρησης, ανάλυσης και ερμηνείας δεδομένων που αποθηκεύονται σε ψηφιακή μορφή με επιστημονικό τρόπο, ώστε να μπορούν να αξιοποιηθούν σε νομικές διαδικασίες (SWGDE, 2014). Κύριος στόχος κατά την εξέταση μίας ψηφιακής συσκευής είναι η εξαγωγή και η ανάκτηση όλων των δεδομένων και πληροφοριών που βρίσκονται σε αυτή χωρίς όμως να επηρεαστεί η ίδια η συσκευή και τα δεδομένα που φέρει. Μία από τις πιο ιδιάζουσες εφαρμογές της ψηφιακής εγκληματολογίας είναι η αντιμετώπιση συμβάντων ασφαλείας (Incident Response), κατά την οποία οι ενέργειες των πρώτων ανταποκριτών είναι ιδιαίτερα σημαντικές και κάθε λάθος κίνηση συνήθως μη αναστρέψιμη.

Αναμφίβολα, στη σύγχρονη κοινωνία η πιο διαδεδομένη ψηφιακή συσκευή είναι το κινητό τηλέφωνο, το οποίο έχει μετατραπεί σε ψηφιακό συνεργάτη του ανθρώπου. Μία συσκευή κινητής τηλεφωνίας αποτελεί σήμερα ένα αποθετήριο με τεράστιο όγκο πληροφοριών, προερχόμενες σε μεγάλο ποσοστό από την αλληλεπίδραση του χρήστη μέσω εφαρμογών κοινωνικής δικτύωσης και ανταλλαγής μηνυμάτων, όπως για παράδειγμα το Viber, το WhatsApp και το Telegram. Ο τεράστιος όγκος πληροφοριών αυτών δικαιολογείται από το γεγονός ότι βρίσκεται σε συνεχή λειτουργία και διεπαφή με το χρήστη. Σήμερα, χρησιμοποιούνται σχεδόν δεκαπέντε (15) δισεκατομμύρια συσκευές κινητής τηλεφωνίας σε παγκόσμιο επίπεδο, ενώ εκτιμήσεις κάνουν λόγω για αύξηση του αριθμού πάνω από τα δεκαοκτώ (18) δισεκατομμύρια έως το 2025 (O' Dea S., 2021).

Η τεράστια διάδοση του κινητού τηλεφώνου και των εφαρμογών κοινωνικής δικτύωσης και ανταλλαγής μηνυμάτων έχουν διαμορφώσει και την σύγχρονη εγκληματικότητα. Σχεδόν σε όλα τα εγκλήματα η βασική μορφή επικοινωνίας μεταξύ των υπόπτων/δραστών πραγματοποιείται διαμέσου εφαρμογών ανταλλαγής μηνυμάτων που βρίσκονται εγκατεστημένες σε συσκευές κινητής τηλεφωνίας. Αυτή η ευρέα διάδοση των κινητών τηλεφώνων ως μέσο διευκόλυνσης ή ακόμη και διάπραξης εγκλημάτων οδήγησε στην ανάπτυξη ενός ξεχωριστού κλάδου ψηφιακής εγκληματολογίας, που εστιάζει αποκλειστικά σε αυτές (Mobile Forensics). Ο εν λόγω κλάδος έχει αρκετές ιδιαιτερότητες, καθώς βασικές αρχές αναιρούνται ή παρακάμπτονται, καθόσον τα δεδομένα της συσκευής συνεχώς μεταβάλλονται. Ως απόρροια αυτού, σε αντίθεση με τους γενικούς κανόνες της ψηφιακής εγκληματολογίας, τα δεδομένα είναι αδύνατο να παραμείνουν άθικτα σε κάθε πράξη. Έτσι, η κάθε επέμβαση από μεριάς των πρώτων ανταποκριτών και εξεταστών είναι ζωτικής σημασίας.

Το περιεχόμενο της παρούσας εργασίας διαχωρίζεται σε τέσσερα διακριτά μέρη. Στο πρώτο μέρος εισάγονται οι έννοιες της ψηφιακής εγκληματολογίας κατά την αντιμετώπιση ενός περιστατικού ασφαλείας, καθώς επίσης αναφέρονται και βασικές έννοιες που σχετίζονται με την εγκληματολογική εξέταση μίας συσκευής κινητού τηλεφώνου με εγκατεστημένο λειτουργικό Android. Στη συνέχεια αναλύονται η δομή και τα βασικά στοιχεία της βάσης δεδομένων τύπου SQLite, στην οποία βασίζεται η πλειονότητα των εφαρμογών σε μία τέτοια συσκευή. Επιπλέον παρουσιάζονται γνωστά και ευρέως διαδεδομένα εγκληματολογικά και μη εργαλεία και λογισμικά που χρησιμοποιούνται για την εξέταση τόσο των Android συσκευών όσο και των βάσεων δεδομένων (SQLite). Στο δεύτερο μέρος παρουσιάζονται προτεινόμενοι μέθοδοι για απόκτηση διαχειριστικών δικαιωμάτων σε δύο συσκευές διαφορετικού κατασκευαστή (LG, Samsung) και διαφορετικής έκδοσης λειτουργικού συστήματος (Android 9 και Android 11), με απώτερο σκοπό την δημιουργία εγκληματολογικών αντιγράφων (physical image) και την αποτελεσματικότερη εξέτασή τους. Ακολούθως, στο τρίτο μέρος περιγράφονται ένα υποθετικό σενάριο βασισμένο σε ρεαλιστικά περιστατικά, πάνω στο οποίο θα βασιστούν τα πειράματα που θα πραγματοποιηθούν στην παρούσα εργασία και η μεθοδολογία που ακολουθήθηκε τόσο για την πειραματική διαδικασία, όσο και για την επιλογή των εφαρμογών ανταλλαγής μηνυμάτων Viber, WhatsApp και Telegram. Στο τέταρτο μέρος παρουσιάζεται η ανάλυση, η εξέταση και τα αποτελέσματα των πειραμάτων για την εκάστοτε προαναφερθείσα εφαρμογή, καθώς επίσης παρουσιάζεται και η συγκριτική τους ανάλυση.

Λέξεις κλειδιά: DFIR, First Responders, Mobile forensics, Android Applications, ενέργειες πρώτου προσαράξαντα σε περιστατικό ασφαλείας, εγκληματολογική εξέταση κινητών, εφαρμογές Android, Viber, WhatsApp, Telegram.

Acknowledgements

We would like to express our sincere thankfulness to our beloved wives Helen and Spyridoula for supporting us to our educational journey and give us all the needed time and space.

Of course, want to say a big thanks to our professor P. Rizomiliotis who put his trust on our idea and support us till the end of this Thesis.

Last but not least comes our Chief Commander of our digital forensics' lab, A. Vasilaras without his will for everyday progress and excel of our tasks nothing of this would have been done.

Disclaimer

This Thesis has been for educational purposes and there is no intention to inspire persons acting malicious or illegal.

The methods are presented made it much easier to make conclusions and is not advised to use if there are not specific and reasoned purposes.

Unlocking and rooting a device can easily set the device unstable or even nonfunctioning. Moreover, the daily usage of these devices is defined as insecure since all security fail safes have been disabled.

Tables List

Figures List

Chapter 1.

Introduction in DFIR for Android Devices

The acronym DFIR stands for Digital Forensics Incident Response and as its name states, is about forensics in digital evidence (hardware devices or software) during an incident response. In this chapter basic concepts about an incident response (IR) plan and team are presented. Moreover, there is an introduction to Android forensics basics, such as data acquisition methods, filesystems, partitions, connection methods (ADB) and known practices on how to prepare and acquire an android device during a security incident. Since the vast majority of android applications and operating systems utilizes the SQLite database format, a thorough forensic analysis of its structure and features is included. Finally, a brief reference of several forensic software and toolkits solutions for Android in general and SQLite in specific, is made.

1.1. Incident Response

An easy way to determine what the term "incident response" means exactly, is to think a how-to guide for a potential cyberattack in an organization. Is a plan in which are referred as many threats and scenario attacks as possible, with detailed plan on how to identify, restrain, prioritize, and eliminate each and every one of them (Cynet, 2021). Throughout the incident response plan, an organization make its employees be aware of security incidents in order to prevent severe attacks and be ready to spot as soon as possible an attacker and stop his acts. The purpose is to minimize the caused damage and the arisen loss.

1.1.1. An Incident Response Plan

In order to make an incident response plan an organization must determine what is the main reason of doing so. To be more specific if the main reason is to avoid data loss the procedures in the response plan should include a way to always keep secure a backup of the essential files. Of course, as everything in security, the accepted budget-to-invest is for sure something to bear in mind. Especially if the hole plan is undertaken from outside of organization experts. Below is presented a summary of common use strategy of making an incident response plan (Cynet, 2021).

1.1.1.1 Preparation

Firstly, there must be an assessment of risks and specification of critical incidents based on sensitive assets in order to build a team to handle all these incidents. This team is known also as "Computer Security Incident Response Team (CSIRT)". The preparation step should include:

- Making a strict and clear policy with rules, principles, and practices to help security processes. For example, there would be a displayed banner informing the users that every activity is being monitored and any unauthorized act or violation will be punished.

- A response strategy on how to prioritize and handle incidents according to its impact on organization. There is quite a difference between a login credentials hack of a simple user from an admin.
- A who-to-contact-with policy. In any instance that may occur there should be specific persons from organization' s IR team or/and law enforcement.
- Creating documentation on who, what, when, where, why and how any procedure in organization works. If something is against this documentation, there may be an intruder' s act.
- Making the IR Team a good one. By choosing members from different expertise (security, IT, human resources, public relations etc.), keep them up to date with training and access to professional tools, an organization make it more feasible to be protected from the known threats.

1.1.1.2 Identification

In this step the IR Team must detect any deviations from normal organization' s operations and determine whether represents a security incident or not, and how important is. An identification procedure may include:

- A monitoring set up for all sensitive infrastructure.
- Setting up a platform to analyze and report, if possible, any security incident which occurred from several sources, such as log files, alerts, error/fault messages, any other tools.
- Establish communication/support center for notifying IR Team.
- Documentation in detail of who, what, where, why and how incident responders must act.
- Making threat detection and prevention capabilities.

1.1.1.3 Containment

In order to limit the caused damage and prevent any spreading while keep untouched any evidence needed there are a few measures that an organization should take. These measures could be:

- A short-term limitation of the damage before gets worse, by isolating specific network spaces for example, preventing the propagation.
- Taking a forensic image of the affected hard disks before wipe and rewrite on them. Thus, there will be evidence for further investigation or accusing the intruder in court.
- A long-term limitation of the damage by addressing the root cause, fixing any back doors, and deleting any accounts or authentication mechanism used by attacker.

1.1.1.4 Eradication

In this step any identified malwares or malicious artifacts are removed, and the affected systems are restored back to normal. An eradication process may include:

- A complete wipe and re-image of all affected hard disks in order to find out if all malicious content has gone successfully.
- Making sure that the root cause is not able to damage/hack any system in the future.
- Making sure that any system is up to date, any software is upgraded and there is no unnecessary service running on them.
- Using anti-malware and anti-virus software for detecting any remaining malicious packages.

1.1.1.5 Recovery

In this step is ensured that all systems are cleaned and fully operational again. In recovery process:

- Based on IR Team findings and prompts will be decided the exact date and time of system services restoring.
- After restoring, tests and verification is needed to ensure that everything works as it should.
- Observe any abnormal behaviors by monitoring for a while after the incident.
- Applying anything could be helping to prevent the same attack.

1.1.1.6 Lessons Learned

After every security incident, the IR Team should gather all the important information acquired due to handling and the whole process, for extracting any lessons that could be used in future incident response activity. So, this step could include:

- Document a full picture of the incident until it' s end.
- Publish a detailed report answering once again who, what, where, why, and how.
- Notice what didn't go as good as it could or gone bad.
- Share and discuss learned lessons and findings with others working in the same field to immediately implement preventing measures.

1.1.1.7 IR Handling Report

A well-presented IR handling report has to include:

- The time and date when the incident was first detected.
- How it detected and by whom.
- The extend of the impact.

- What took to contain and eradicate it.
- Acts during recovery.
- The best practices used by IR Team. What went as it should.
- What requires improvement.

1.1.2. Incident Response Team

1.1.2.1 IR Team Model

An IR Team could take one of the following three models, based on NIST framework (Cynet, 2021):

1. Central, where the whole organization' s incidents are managed from a centralized body.
2. Distributed, where there are several teams coordinate each other if needed. Each team is responsible for a specific part of the IT infrastructure, physical location, or department.
3. Coordinated, where there is command or knowledge center consists of a central team, which monitors systems and sound the alarm for distributed teams as required, providing the necessary assistance.

There is no restriction about the general form of IR Teams, but based on their tasks they are usually divided in three main types:

- A team which is responsible for preventing, detecting, and responding to incident response cyber security events or incidents, known as Computer Security Incident Response Team (CSIRT).
- A team with similar task as the previous one but with prioritizing the development on threat intelligence and best practices based on security responses. This team focuses on partnerships with government, law enforcement, academia, and industry and it is known as Computer Emergency Response Team (CERT).
- A team which is responsible for monitoring and directing incident response and defending systems. Its tasks include controls configuration and overseeing general operations. This team is well known as Security Operations Center (SOC) and in general includes a CSIRT or CERT teams with a broader scope of cyber security.

1.1.2.2 Members of the IR Team

To achieve the effectiveness of an IR Team, must include a variety of professions and expertise. As it is true, there are some prerequisites to bear in mind while building an IR Team, such as:

- Availability. There is a need for 24/7 assistance and critical incident handling.
- If human resources are limited a virtual or on-call members may be required for dealing with problems when they occur, no matter the time or the place.

- A senior-level executive, like a chief information security officer (CISO) is a good asset in a team. This is the person who can communicate the importance of the incident to high level personnel and also find the required funding when needed.
- Technical diversity between the members to more easily identify and handle threats and find solutions.
- All persons should be moral and have a team spirit in order to create strong relationships which will lead to great collaboration and communication through the stressful situations that definitely will arise.

1.1.2.3 IR Team Preparing

No matter how good prepared the members of a team as individuals are, always should train together. Some wise principles to train an IR Team are: **i).** The supplementation of cyber security management tools to gain a more secure profile of what an incident is. For instance, the members of the team could monitor carefully to spot local network (intranet) traffic's misuse or abuse. Not only find anomalies or suspicious acts in accesses, but could also watch for abnormal consumptions, like when there is a dropdown in performance due to big demands for exporting data. In these situations, the team could note what a normal user/system behavior is look like and what is not. **ii).** The centralization of all members efforts when something occurs. In an organization there are so many different systems and subsystems with different software and logs files. So, in order to be always aware of what is happening there must be a central system of parsing, monitoring and alerting when needed. A way to achieve that is by implementing a system information and event management (SIEM). In addition, there are also security orchestration, automation, and response (SOAR) solutions, making easier to get alerted when something goes wrong and automate protective procedures, standardizing responding eco-systems. **iii).** Learn to act base investigating findings and not by gut feelings. All members must be aware of when you underestimate an attack and you think that you got everything write because of your experience is likely to miss something that will cause your organization' s revenue, brand fame or worse. So, is better to make an accurate conclusion during investigations, than based on faulty assumptions. Once an IR Team gets a clear view of the whole incident affection, could response effectively and efficiently remediate the damage.

1.1.2.4 IR Team Roles

As a good point to start is to divide IR Team into following roles:

- A team leader, who will be responsible for coordinating activities and reporting to upper-level management.
- A communications guy who has the responsibility for managing communications not only throughout the team and organization, but also throughout stakeholders, customers, and public authorities so that everyone is properly informed about any incidents.
- An investigation's leader, who will perform as a first investigation commander, giving guidance in analysis and evaluating the primary results.

- More than one member, if possible, who perform as analysts and researchers. They are about to provide threat intelligence and context for an incident. These members usually conduct the first responders' activities.
- Lastly, there must be a legal representant, who acts like a lawyer and make that every act is compliant with law enforcement, and standards of integrity for forensic evidence.

1.2. Android Forensics

To cut long story short, digital forensics is a science or a scientific procedure that like common forensics looking for evidence in a digital world. A recovered deleted message from a mobile phone, is like a fingerprint from the burglar on the banker's desk. Like in all the forensics, when something happened and there is a need for digital investigation, the "crime scene" must remain intact, in order to keep evidence unaltered. In this thesis we would try to clarify how the interfering with digital evidence, and to be more specific android-based devices, alter the integrity of data and the course of investigations (Gogolin, 2021). At the next chapters there will be focus on android forensics and the DFIR including such devices, but before that we must define the main terms that will be used.

1.2.1. Data acquisition

In digital forensics data acquisition, is the process of copying the data without tampering and make them available for presenting, examining, analyzing, and storing purposes. The data are located or stored to be more accurate, in storages like hard disks, memories (i.e., RAM), usb, cd-rom, on motherboard memory chips (mobile cell phones), etc.

1.2.2. A forensic image

In order to acquire the data from a storage space of a device a clone must be created, and thus not modifying them. A forensic image is just that, an unaltered copy of electronic data which are stored in a device. The proper way to take a forensic image is by using forensic tools or/and write-block hardware or software. There are multiple methods of data imaging, but not each one includes every single bit of data from the device they are extracted from. The choice of the method or type of imagining is up to device unique specifications, encryption, and security restrictions. The source storage capacity is also a factor to keep in mind, because if there is no need of keeping everything inside it, the best to do is to make a selective acquiring. Once the image is taken, the forensics examinations are good to go without the fear of tampering and alter the physical device's stored data. Thus, we also gain the integrity of acquired data.

1.2.2.1 Physical image (full file system)

A physical image is a full clone of a data storage source. With term "clone" it means a bit-by-bit copy including all partitions, unallocated and unused space of a storage. As always is written, a physical image includes all the source's zeros and ones (Vasilaras A., 2016). This type of imaging gives access throughout the examination process by forensic tools in the deleted and occasionally formatted files. A physical image could be a raw image file (i.e. .dd) or a compressed one (i.e. .E01) without all the zeros (no stored data) of the storage source. Not all the image' s formats are compatible with any forensic tool.

1.2.2.2 Logical image (partial logical or partial file system)

In contrast with physical, a logical image includes only the data that is visible to the file system. Is a sound forensic method to acquire data from a storage source, but there are missing much data fragments. For instance, if a mobile application has been deleted by the user, the examination couldn't be able to show that the app it was installed in first place (Tahiri S., 2016).

1.2.2.3 Selective or Targeted image

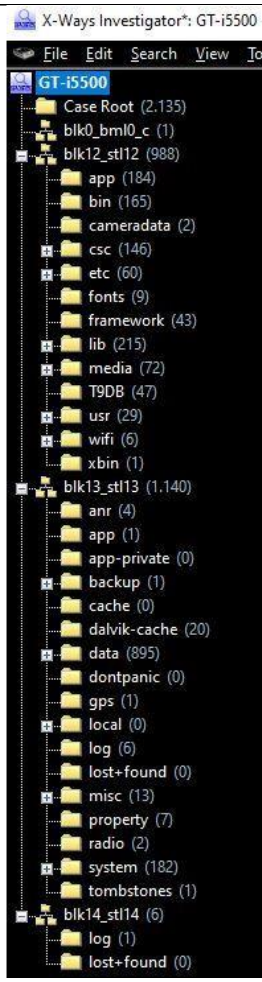
In addition to above mentioned methods, there is one more costless in terms of resources and time and quicker most of the times. This method requires to know exactly what you want to extract from the storage source for examining and analyzing. Needless to say, that this method is like logical and so no unused or unallocated space or file fragments is feasible to be restored (Precise Law, 2017).

1.2.3. Partition

Partitions are logical divisions of a storage memory (i.e., hard disk) which is being recognized by the operating system as a separate unit (Rahmanpour, 2019). When partitioning is applied, there is a more convenience way to store and organize files and also works as a protection for sensitive data. In an android device for example, there are partitions that user data are stored and partitions that user can tamper with, in which sensitive data for the operating device are included. Hence, no matter how user interact, the device can always be formatted and restored in order to operate without problems.

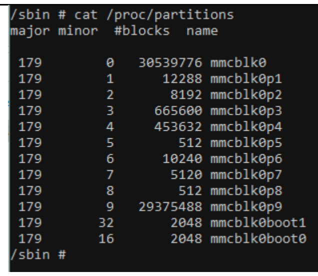
1.2.4. Android File System

State of the art android devices tend to have larger internal storage than their ancestors did, and their operating systems use their internal storage as a hard drive (i.e., sda) and not as an SD card (i.e., mmcblk0). The table below depicts the structure of the partition table in different android versions.



Android 2.2.2. Froyo (Samsung GT-i5500)

(created by: X-WAYS Investigator)

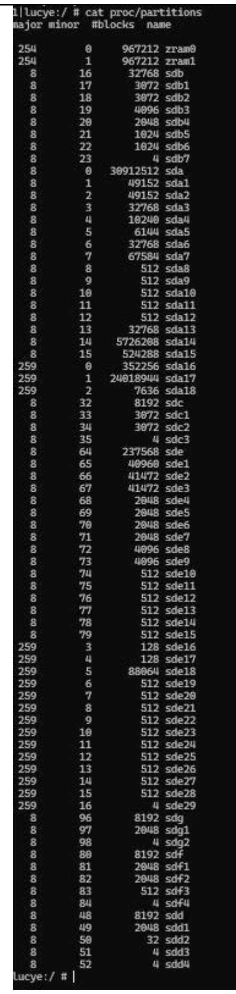


```
/sbin # cat /proc/partitions
major minor  #blocks  name

 179        0   30539776 mmcblk0
 179        1      12288 mmcblk0p1
 179        2       8192 mmcblk0p2
 179        3     665600 mmcblk0p3
 179        4     453632 mmcblk0p4
 179        5        512 mmcblk0p5
 179        6      10240 mmcblk0p6
 179        7       5120 mmcblk0p7
 179        8        512 mmcblk0p8
 179        9   29375488 mmcblk0p9
 179       32       2048 mmcblk0boot1
 179       16       2048 mmcblk0boot0
/sbin #
```

Android 5.1.1 Lollipop (Nexus 7)

(created by: adb shell command line view)



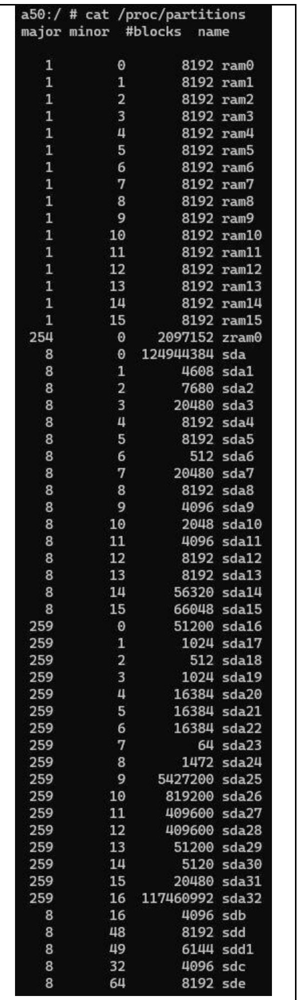
```
1|lucye:/ # cat proc/partitions
major minor  #blocks  name

 254        0     967212 zram0
 254        1     967212 zram1
   8       16      32768 sdb
   8       17       3072 sdb1
   8       18       3072 sdb2
   8       19       4096 sdb3
   8       20       2048 sdb4
   8       21       1024 sdb5
   8       22       1024 sdb6
   8       23          4 sdb7
   8        0   30912512 sda
   8        1      49152 sda1
   8        2      49152 sda2
   8        3      32768 sda3
   8        4      10240 sda4
   8        5       6144 sda5
   8        6      32768 sda6
   8        7      67584 sda7
   8        8        512 sda8
   8        9        512 sda9
   8       10        512 sda10
   8       11        512 sda11
   8       12        512 sda12
   8       13      32768 sda13
   8       14    5726208 sda14
   8       15     524288 sda15
 259        0     352256 sda16
 259        1   24018944 sda17
 259        2       7636 sda18
   8       32       8192 sdc
   8       33       3072 sdc1
   8       34       3072 sdc2
   8       35          4 sdc3
   8       64     237568 sde
   8       65      40960 sde1
   8       66      41472 sde2
   8       67      41472 sde3
   8       68       2048 sde4
   8       69       2048 sde5
   8       70       2048 sde6
   8       71       2048 sde7
   8       72       4096 sde8
   8       73       4096 sde9
   8       74        512 sde10
   8       75        512 sde11
   8       76        512 sde12
   8       77        512 sde13
   8       78        512 sde14
   8       79        512 sde15
 259        3        128 sde16
 259        4        128 sde17
 259        5      88064 sde18
 259        6        512 sde19
 259        7        512 sde20
 259        8        512 sde21
 259        9        512 sde22
 259       10        512 sde23
 259       11        512 sde24
 259       12        512 sde25
 259       13        512 sde26
 259       14        512 sde27
 259       15        512 sde28
 259       16          4 sde29
   8       96       8192 sdg
   8       97       2048 sdg1
   8       98          4 sdg2
   8       80       8192 sdf
   8       81       2048 sdf1
   8       82       2048 sdf2
   8       83        512 sdf3
   8       84          4 sdf4
   8       48       8192 sdd
   8       49       2048 sdd1
   8       50         32 sdd2
   8       51          4 sdd3
   8       52          4 sdd4
lucye:/ # |
```

Android 9 Pie

(LG G6 [H870])

(created by: adb shell command line view)



```
a50:/ # cat /proc/partitions
major minor  #blocks  name

   1        0       8192 ram0
   1        1       8192 ram1
   1        2       8192 ram2
   1        3       8192 ram3
   1        4       8192 ram4
   1        5       8192 ram5
   1        6       8192 ram6
   1        7       8192 ram7
   1        8       8192 ram8
   1        9       8192 ram9
   1       10       8192 ram10
   1       11       8192 ram11
   1       12       8192 ram12
   1       13       8192 ram13
   1       14       8192 ram14
   1       15       8192 ram15
 254        0    2097152 zram0
   8        0  124944384 sda
   8        1       4608 sda1
   8        2       7680 sda2
   8        3      20480 sda3
   8        4       8192 sda4
   8        5       8192 sda5
   8        6        512 sda6
   8        7      20480 sda7
   8        8       8192 sda8
   8        9       4096 sda9
   8       10       2048 sda10
   8       11       4096 sda11
   8       12       8192 sda12
   8       13       8192 sda13
   8       14      56320 sda14
   8       15      66048 sda15
 259        0      51200 sda16
 259        1       1024 sda17
 259        2        512 sda18
 259        3       1024 sda19
 259        4      16384 sda20
 259        5      16384 sda21
 259        6      16384 sda22
 259        7         64 sda23
 259        8       1472 sda24
 259        9    5427200 sda25
 259       10     819200 sda26
 259       11     409600 sda27
 259       12     409600 sda28
 259       13      51200 sda29
 259       14       5120 sda30
 259       15      20480 sda31
 259       16  117460992 sda32
   8       16       4096 sdb
   8       48       8192 sdd
   8       49       6144 sdd1
   8       32       4096 sdc
   8       64       8192 sde
```

Android 11

(Samsung A50)

(created by: adb shell command line view)

Table 1. Examples of the evolution of Androids' partition tables.

The fundamental principle that lies in every device is that the largest partition contains the user data. However, several changes in the partition structure of the OS are observed, depending not only on the variations of the android versions, but also on the proprietary vendor's firmware. From forensic view, this is the partition which contains the vast majority of valuable evidence data. Examiners should be aware that only super user (su) has access to all of the filesystem (**Figure 2**).

```
a50:/ $ cd /data/data
a50:/data/data $ ls
ls: .: Permission denied
1|a50:/data/data $ |
```

Figure 1. Denial of access on Samsung A50 device's filesystem as simple user.

1.2.5. Android Debug Bridge

Creating a forensic image of an android device is impossible without setting up a connection bridge first. The way to achieve that is by a command-line tool known as ADB or Android Debug Bridge. This communication method is based on Linux kernel and works like command-line shell. In the official android development site, they refer to ADB capabilities as «a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device» (SalvationDATA, 2020).

1.2.6. Android Boot Loader

Bootloader known also as boot manager or bootstrap loader is a vendor-proprietary image responsible for boot the kernel on an android device. It monitors the device state and is responsible for initializing the T.E.E (Trusted Execution Environment). The main property is to verify the boot and recovery partitions' integrity, before continuing to the execution of kernel. When something is out of order displays the corresponding boot state warnings. When is unlocked it makes it possible to install a custom firmware or/and giving full access privileges. Hence, modifications and more advanced access on the system are allowed (Bair J., 2017). In general bootloader on android devices is locked and is responsible for what programs are loading and starts to run when the device is turned on.

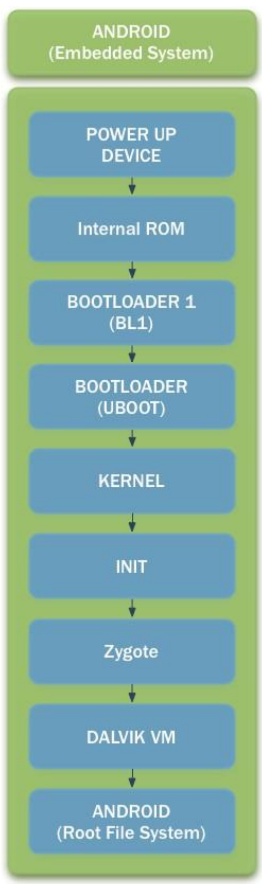


Figure 2. Flow of Boot Sequence on Android Devices (González J., 2012)

1.2.7. Known IR Practices on acquiring Android devices

Every time an android device is examined for a security incident the extraction of data is crucial. For this reason, the acquisition of a device's physical image is important, in order to have extensive access to its stored data. But as it is already mentioned not every incident complies with examiner's or incident responder's convenience. As Mahalik et al. (2021) points out, before proceeding in data acquisition there are several steps to walk through in order to prepare the android device for the extraction. The power status, of most android devices, is a crucial factor relating to the procedures that have to be followed.

1.2.7.1 Device is powered off

If the device is powered off an attempt to get a physical acquisition is a sound choice, but if the extraction fails then the device must be turned on and continue as shown in the next two occasions.

1.2.7.2 Device is powered on and unlocked

The steps that are followed on this case are:

- Network connections must be disabled such as Wi-Fi, Hotspots, Bluetooth or mobile data and the device must be set on airplane mode while removing any sim cards.
- Preparation of the device for data extraction by removing any passcodes and enabling the "USB Debugging" and "stay awake" option from the settings and disabling any screen lock features.
- Choosing the acquiring method. First of all, a physical acquisition must be attempted and if this method is not supported, then other extractions (logical, file system, backup) have to be attempted.
- Imaging extra media and sim cards. The examination of any sim cards or/and external memory cards included in source device have also to be added to the evidence collection procedure.

1.2.7.3 Device is on and locked

There are at least two kind of lock protections on android devices. The first one is before android system is fully loaded which is a passphrase with digits and/or letters and the second one is screen lock (pattern, pin code, password, biometrics). If a physical image acquisition can be achieved without tampering device settings, then is highly recommended to proceed with the extraction without any further delay. Network isolation step is still the first thing to do when you want to examine a device. Some forensic tools have their custom bootloaders and will try to use them for gaining access to the device's data, by bypassing any lock, but if this fails then there are not many options left. The searches must be focused on how to get the passcode from the owner or other sources (use known pin codes like birthdays or examine associated media or hard disks). After having completed the above procedure and the device is now unlocked, the described in previous paragraph steps remain the same.

1.3. Structured Query Language (SQL) and SQLite Data

SQLite is lightweight, self-contained database type that fulfills the ACID (Atomicity-Consistency-Isolation-Durability) properties for reliable transactions. The database can be easily integrated into any application without the necessity of any server software (Pawlaszczyk & Hummert, 2021). The efficiency of an android application's operation depends on both the android and SQL programming skills of the developer.

1.3.1. SQLite Database Structure

1.3.1.1 Header

The filename extension of an SQLite database is irrelevant as it can be found as db2, db3, SQLite or none. The proper way to examine the properties of a database is by inspecting the header of a file. The header of an SQLite database is comprised of it's first one hundred (100) bytes divided into several fields. (SQLite, 2021)

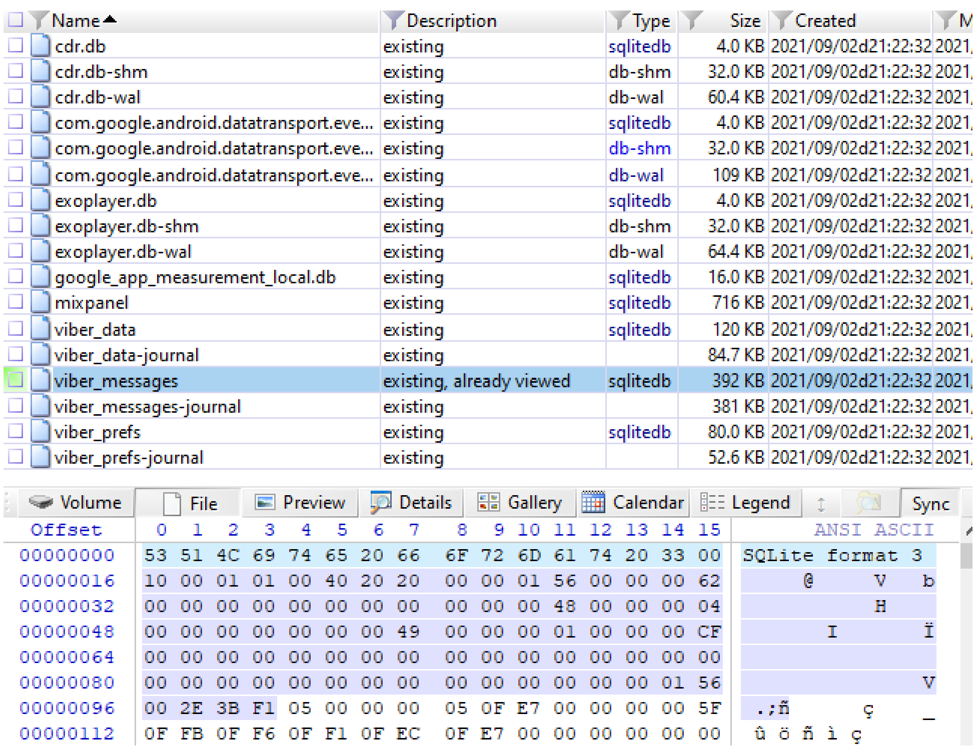
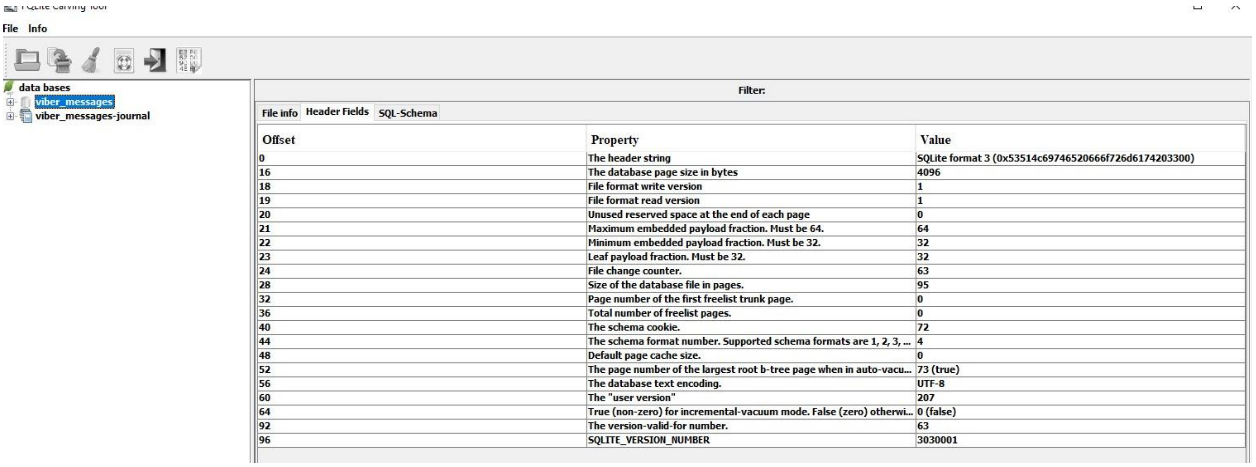


Figure 3. Viber-messages SQLite database header parsed via X-Ways forensics Software in hex mode.

Nowadays, several forensic software products provide a way to interpreter those one hundred (100) header bytes by parsing them.



Offset	Property	Value
0	The header string	SQLite format 3 (0x53514c69746520666f726d6174203300)
16	The database page size in bytes	4096
18	File format write version	1
19	File format read version	1
20	Unused reserved space at the end of each page	0
21	Maximum embedded payload fraction. Must be 64.	64
22	Minimum embedded payload fraction. Must be 32.	32
23	Leaf payload fraction. Must be 32.	32
24	File change counter.	63
28	Size of the database file in pages.	95
32	Page number of the first freelist trunk page.	0
36	Total number of freelist pages.	0
40	The schema cookie.	72
44	The schema format number. Supported schema formats are 1, 2, 3, ...	4
48	Default page cache size.	0
52	The page number of the largest root b-tree page when in auto-vacu...	73 (true)
56	The database text encoding.	UTF-8
60	The "user version"	207
64	True (non-zero) for incremental-vacuum mode. False (zero) otherwi...	0 (false)
92	The version-valid-for number.	63
96	SQLITE_VERSION_NUMBER	3030001

Figure 4. Viber-messages SQLite database header parsed via FQLite.

In terms of forensics, some interesting points in the header, are the number of freelist pages (bytes 36 to 40), the auto vacuum feature (bytes 52 to 56) and the incremental vacuuming (bytes 64 to 68). The above information can determine whether deleted records can be recovered. (Nemetz et al., 2018)

1.3.1.2 SQLite Pragma Statements

As it is defined by the official site of SQLite (2021), PRAGMA statements are SQL extensions for SQLite and are used for controlling various environmental variables and state flags within the SQLite database. There is a vital difference between PRAGMA statements and SQLite commands although they are using the same interface. There are specific PRAGMA statements that can be removed or added in an updated future release of an SQLite database, with no guarantee of backwards compatibility. Below is depicted some of the “Pragmas” that can be viewed or edited by a database tool.

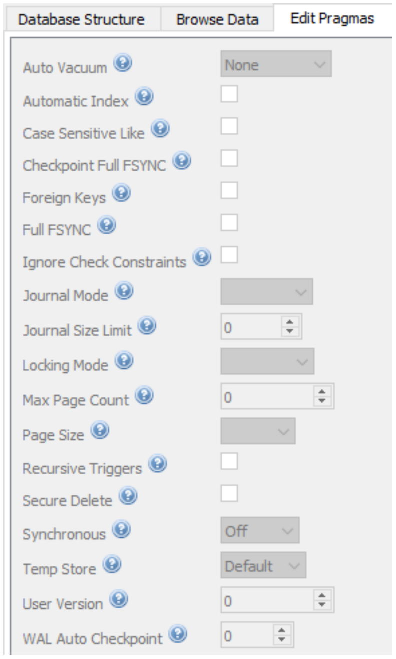


Figure 5. SQLite Pragma Statements from DB Browser for SQLite interface.

The most interesting parameters, as shown at the above picture, for database forensics, from DFIR spectre are the following:

- **Auto Vacuum**: this parameter indicates how database freelist pages are being handled when a data deletion commit occurs. When it is enabled (registered value "Full" or "1"), the freelist pages are moved to the end of the database file and are removed after the end of every transaction commitment. On the other hand, if it has a value of "None" or "0" that means the auto-vacuum PRAGMA statement is disabled and no matter what insertions or deletions are committed, the database's file size is retained, as unused data are added to freelist pages and reused for subsequent inserts.

- **Journal Mode**: this parameter defines how the journal file interacts with the "main" database every time a transaction is committed. If value of "DELETE" is selected (default value), the rollback journal is invalidated at the completion of each transaction. The other used method for journaling mode is "WAL" that uses the write-ahead log instead of the rollback journal.

- **Journal Size Limit**: this parameter may be used for size limitation of rollback-journal and WAL files. By default, the size parameter takes the value of "-1", for unlimited journal size, but as it will be proved during in the experiments held in this thesis the android application's database have a size limitation of their rollback journal files.

- **Secure Delete**: this is a boolean parameter and holds values "on" or "off". When the value is 'on' SQLite overwrites deleted content with zeros thus wiping the deleted data. "Secure_delete" is quite a CPU and disk intensive process with great impact in battery life. For this reason, the vast majority of applications on android devices has the "secure_delete" option disabled.

1.3.1.3 B-Tree Pages

The B-tree pages store the active data from the tables and indexes divided in four main areas: the page header, the cell pointer array, the unallocated space, and the cell content area. The latter is comprised by the active records, the free blocks, and the fragmented bytes. In some cases, it is possible to recover data from the unallocated space and the freeblocks, but it is nearly impossible from the fragmented bytes, as their size is too small (less than 4 bytes).

1.3.1.4 Freelist Pages

Freelist pages are the ones, which are not currently used by a database. Their records were deleted, and they are flagged for re-use at a later time. By default, SQLite does not wipe their contents and so there is a possibility to recover deleted records. This is not possible, if the pragma statement of "secure_delete" is enabled meaning that SQLite overwrites deleted content with zeros (SQLite, 2021). Other page types are the overflow, the pointer map and the lockbyte pages but since, they have not any forensic interest will not be further examined.

1.3.1.5 Atomic Commit in SQLite

As it is already mentioned, SQLite is ACID compliant. For this reason, its atomic commit feature is very important. Atomic commit is achieved when all database changes occur within a single transaction (SQLite, 2021). This is implemented through the rollback journal (legacy) or, since SQLite 3.7, with the Write Ahead Log (WAL). These two methods utilize very different approaches for the atomic commit and rollback features of a database. The method is used can be determined by bytes 18 and 19 of the file header. If the value, in the header, is set to “01 01” the rollback journal is used and if the value is set to “02 02” then the WAL method is used.

1.3.1.6 Rollback Journal

SQL .journal file holds a copy of the page prior to committing a transaction and if the transaction is committed, it is invalidated, but if the transaction fails, the page is copied back to the main database. The reasons why there may be a failed transaction vary. Some examples are file overwriting, file locking issues or failure in synchronization with the storage media. There are five Rollback Journal modes: Delete, Truncate, Persist, Memory and Off. The Delete mode is the default one. On the contrary, Memory and Off modes are rarely encountered. The selected mode cannot be determined by the file header but can be found into PRAGMA statements (SQLite, 2021). The Rollback Journal could have an invalid state, which is potentially a previous version of the state of the main database or a valid/hot one which indicates that the last transaction failed leaving the main database unmodified. There are cases, especially after a fresh installation of an application in mobile applications in which the journal/wal file and the main database is not present in the device’s folder structure.

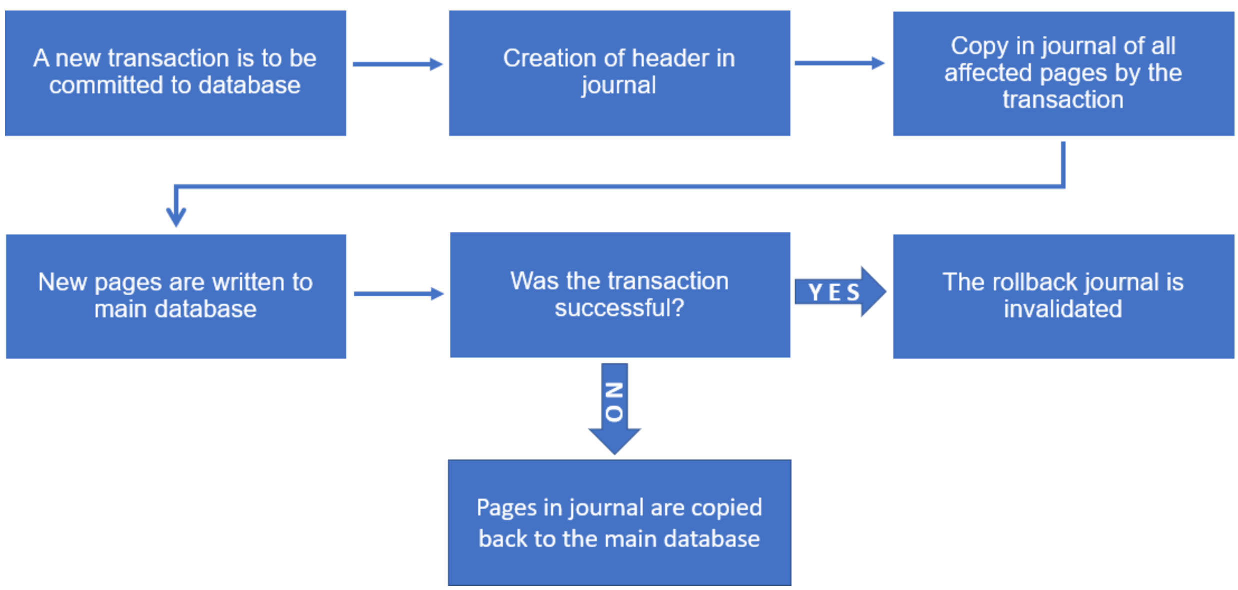


Figure 6. SQL journal transaction process flow.

1.3.1.7 Write Ahead Log (WAL) and Shared Memory (shm)

The WAL process, which was implemented after SQLite 3.7, takes a different approach by leaving the main database intact and committing the changes to a separate file. The pages of the main database are left unmodified until a WAL checkpoint is reached. Usually this happens after one thousand (1.000) commits, when the application forces a commit back to the main database. The examination of the WAL file provides information about the most recent state of the database. For instance, recently deleted data cannot be found in the main database but might be located in the database’s WAL file.

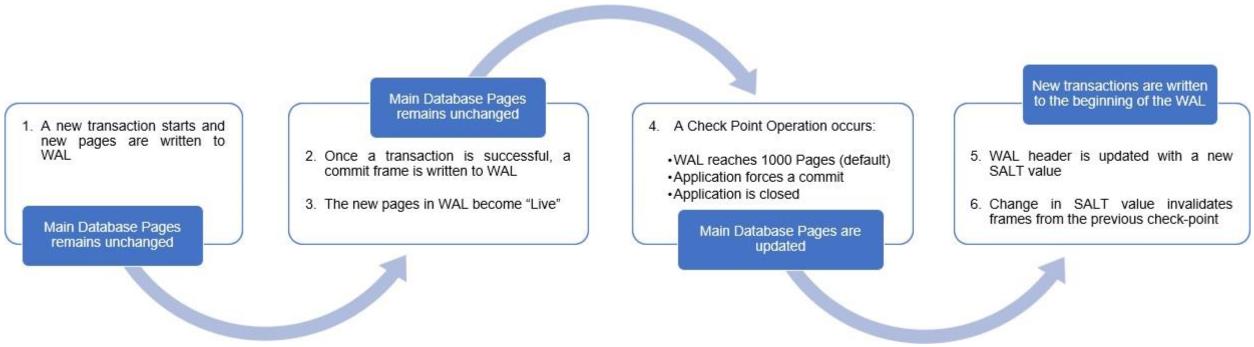


Figure 7. SQL -wal transaction process flow.

The Shared Memory file contains an index of the most current version of each page since the last checkpoint. Thus, an application can decide wherefrom the most recent data exists in order to pull them off.

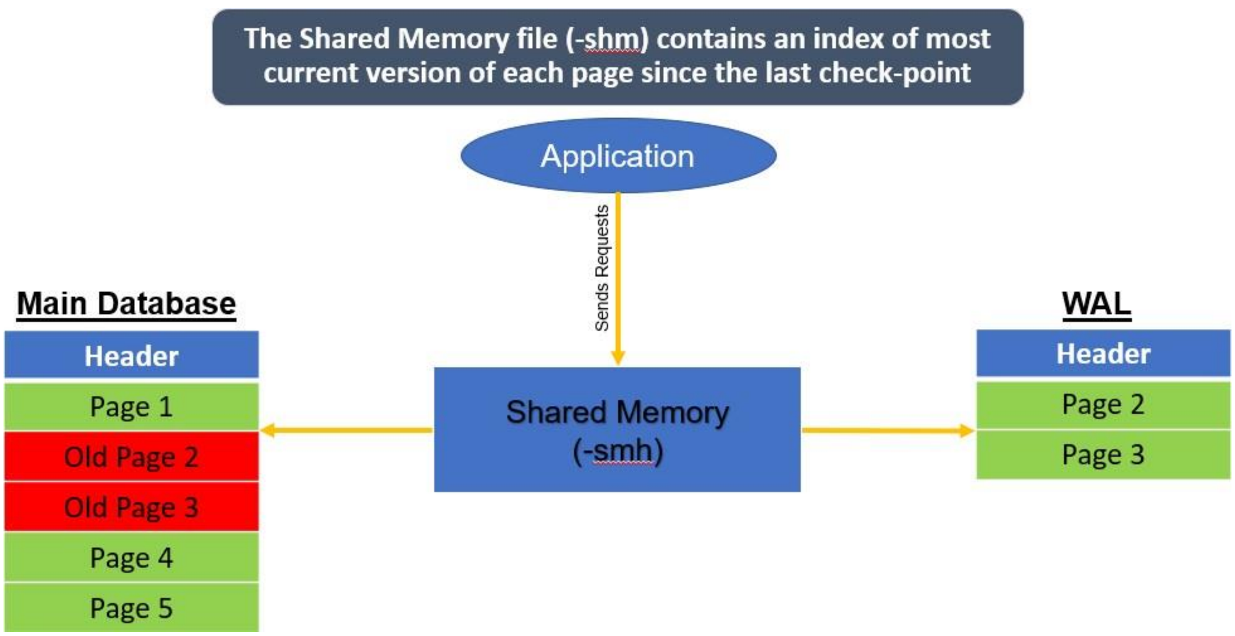


Figure 8. SQL -shm transaction process flow.

1.4. Forensics tools solutions

1.4.1. Android Mobile Devices Acquisition and Examination

There are a lot of information about forensic tools supporting android forensics. In fact there are plenty of sources that describe them and compare their capabilities and features (Easttom, C., 2021, Easttom, C., 2021, Raji et al., 2018, Vasilaras A., 2016). In the section below, the major capabilities of commercial and non-commercial well-known software and tools used for android forensic purposes are outlined. Some of the mentioned tools are used both for extracting and analyzing data, while others are used only for examinations. Most of the tools' user interfaces will be included in the next chapter.

- **Autopsy**. A free of cost software suite developed by Brian Carrier and his team. There is an Android Analyzer module for android devices which works with physical images and file system dumps. In newer versions (4.18 and above) the aLEAPP parser has been implemented in the Android Analyzer module. It supports images created from other tools such as Cellebrite and MSAB.
- **aLEAPP**. A noncommercial github tool written in Python by a law enforcement officer Alexis Brignoni. This tool is a log events and protobuf parser and works also with physical or file system dumps files (.zip, .tar) and creates a report including all valuable data.
- **Andriller**. A freeware utility including a collection of tools for android based devices, which can perform both forensic analysis and acquisition.
- **Cellebrite UFED**. A commercial tool that includes UFED touch or UFED 4PC and UFED physical Analyzer. The first two are utilized for extraction purposes providing a user-friendly interface. The last one is utilized for analyzing the device's extraction.
- **Magnet Forensics**. Magnet provides both an acquisition (Magnet Acquire) and an examination and analysis tool (Magnet Axiom). Also, it provides a very convenient report for viewing the evidence findings. It also supports various type of images including the ones created from Cellebrite.
- **Oxygen forensics**. Oxygen Extractor supports a variety of different chipsets as well as the abovementioned tools. The analyzing tool (Oxygen Detective) is capable of building connections and correlations among multiple acquired evidence devices as well as visualizing them in a user-friendly manner. It also supports images created from other tools such as Cellebrite and MSAB.
- **Belkasoft**. Belkasoft X is another useful tool combining extractions from android devices and supports their examination and analysis.
- **MSAB**. XRY digital data is able to extract data from mobile devices (both physical and logical) and XAMN is used to analyze and examine them.
- **MOBILedit**. Through MOBILedit Forensic Express tool, android physical and/or logical extractions as well as analysis of the extracted data are feasible. It supports Cellebrite and Oxygen exported images.
- **Access Data**. Since Forensic ToolKit (FTK) v7.4.104 there is an addon for mobile data analysis and chat application parsing tool It also provides a QView viewer for more convenient visualization of examined data. It supports Cellebrite and MSAB exports.
- **Hancom**. MD-Series tools provide several different solutions for acquiring and examining android devices. Among other extraction methods, JTag (via MD-box) is included.
- **Paraben E3 Forensic Platform**. A platform which supports all kind of acquisitions and forensic examinations.
- **X-Ways**. Since the android filesystem is similar/same to UNIX/LINUX, X-ways can provide an accurate and fast hex interpreter parser and file system viewer.

Most of the referred tools have the capabilities of bypassing, removing or brute forcing the pin code, pattern, or password security measures of android devices. In addition of the previous mentioned features, the tool should include a decryption method of the device's data. More and more applications implement cypher methods in order to encrypt their databases but not all of the tools are capable of decrypting and parsing every single one of them. Bear in mind that some versions of the abovementioned commercial tools are only available for law enforcement agencies and not for enterprises. Specifically, the law enforcement versions include extra features and artifacts for both extraction and analysis. Be aware that there is not a tool that supports every single android device or chipset. For this reason, the examiners should wisely select the tools that will help them in their work.

1.4.2. SQLite Databases Examination, Recovery and Analysis

The forensic analysis of an SQLite Database's live records is relatively easy compared to deleted ones. The SQLite Carving for deleted records is a difficult and tedious task for various reasons. Firstly, the number of forensically interesting applications, which use SQLite databases, and the speed these are being updated is too fast for any tool to keep up with. Secondly, the pragma statements (e.g., secure_delete, autovacuum etc.), which are implemented by the application's developers and the way unallocated space and free blocks (containing random fragments of data), are located in these pages, makes the carving of any useful information is challenging to say at least. Some options for recovering and analyzing data from SQLite Databases are:

- **aLEAPP** (Free). The integrated python scripts which are implemented in the tool can create a useful report of android artifacts, including records from applications' databases. In addition, the SQL queries inside the scripts could be used separately for browsing SQLite databases in other software such as DB Browser for SQLite. One disadvantage is that it cannot retrieve deleted records.
- **DB Browser for SQLite** (Free). One of the most known and useful tools is DB Browser for SQLite. It does not have the automated ability to parse journal or deleted records. It is not considered as forensic software as the user can modify the examined database's data.
- **SQlite Expert** (Free and Propietary). SQlite Expert can visualize and modify BLOB fields and present multiple databases in one tree. It is also not considered as forensic software as the user can modify examined database's data.
- **Oxygen forensic SQlite Viewer** (Propietary). Part of Oxygen forensic Suite which can recover deleted records and parse journal and wal files.
- **Magnet Axiom Forensics** (Propietary). Magnet Axiom suite integrates an SQLite Viewer which has the automated ability to parse and combine the main database with its corresponding WAL/Journal file.
- **UFED Physical Analyzer** (Proprietary). Ufed includes the most popular mobile application's database schemas and via App Genie and SQlite Wizard features, can potentially recover/parse deleted records.
- **Autopsy Suite** (Free). Autopsy, is the best known free forensic suite and with the usage of python plugins can parse SQLite Databases and their deleted records.
- **Paraben E3** (Free and Proprietary). Paraben E3 free edition can parse SQLite Databases but not Journal / Wal files.
- **Belkasoft X** (Proprietary). Belkasoft X can parse records from SQLite Database and their corresponding Journal / wal files.
- **X-ways Forensics** (Proprietary). X-ways, by creating virtual files, can analyze SQLite databases.
- **acquire Forensics** (Shareware). Acquire SQLite Forensic Explorer has the ability to parse and recover records from an SQLite database (including records residing in journal/wal file) even if they are corrupted or deleted.
- **FQlite** (Free). FQLite is a tool to find and restore deleted records in SQLite databases. It therefore examines the database for entries marked as deleted. Those entries can be recovered and displayed.

- **Andriller** (Free). Has a variety of tools for parsing and decrypting databases but it cannot retrieve deleted records.
- **KS DB Merge Tools for SQLite** (Free & Proprietary). A tool for comparing the same database in different states based on the schema, the tables, and the data inside them. The tool uses a GUI version of the "sqldiff" executable.
- **Forensic Toolkit for SQLite**. This is a toolkit which includes three separate tools: "The Forensic Browser", "Forensic Recovery" and "SQLite Forensic Explorer" for SQLite. Each of these tools has different usage. For example, "Forensic Browser" has the ability to reconstruct a database including the records from its wal/journal file.

Some other known command line tools which could be used are undark (Nemetz et al., 2018) and bring2lite (Meng & Baier, 2019).

Chapter 2.

Preparation and Examination of an Android Device

In this chapter, the preparation required for gaining admin privileges (root) and examining of two different Android OS devices is presented. An LG G6 (H870) with Android 9 and a Samsung A50 (A505FN/DS) device with Android 11 were utilized and granted root access by unlocking bootloader, installing custom recovery menu and software. The important folders and files of the filesystem are referenced. The commands (CLI) which were used to navigate in the filesystem and extract the desired data are also analyzed. In addition, user interfaces of several software and tools for extracting and examining data are presented.

2.1. Getting Root access

In order to gain a root access, the devices' bootloaders have to be unlocked and a custom recovery image known as Team Win Recovery Project (TWRP) has to be installed. This has as a result, third-party firmware can be installed, and full system backups can be created. The unlocking of the device's bootloader requires the enabling of USB-Debugging and OEM unlocking options. The above-mentioned settings can be found under Developer Options. After that, the device can be put in bootloader mode via Command Prompt or Terminal command with the command:

```
adb reboot bootloader
```

After this step the bootloader unlocking procedure varies depending on the device-vendor's requirements. For instance, LG requires to send the unique device id and IMEI through their developer portal in order to receive back a binary unlock file (unlock.bin), whereas Xiaomi uses the Mi Unlock application for this purpose (Xiaomi MIUI, 2021). In any case the userdata partition is wiped when the bootloader unlocks. After having unlocked the devices' bootloader, the "Magisk" solution (Magisk Manager, 2021) was chosen for getting root access. As the official developer, John Wu, refers «Magisk is a systemless rooting», which means that the core code of the mobile's operating system is not altered.

2.1.1. Getting root access on LG G6

The examined LG G6 device had the following specifications:

- Android Kernel version is 3.18.120.
- Android 9 with security patch dated in May of 2019 (Latest available).
- The official rom firmware which was installed is the latest available (v30b-EUR-XX).

The "3.2.3-Nebula-Alpha_20180813 TWRP" from XDA-developers forum (zefie, 2017) was used as a recovery image, and the Magisk version 23.0 was installed for granting root access privileges to the system. For further convenience, an extra Magisk module was installed, named “Busybox for android NDK 1.33.1”, which offers additional Unix commands (i.e., netcat).

2.1.2. Getting root access on Samsung A50

The examined Samsung device had the following specifications:

- Android Kernel version is 4.14.113.
- Android 11 with security patch dated in July of 2021.
- The official rom firmware which was installed has the built number “a505fxxs9cuf1”.

The “3.5.0_9_0 (UNOFFICIAL recovery)”[1] from XDA-Developers forum (redymedan, 2021) was used as a recovery image and the unofficial Magisk version 22.1 was installed in order to get administrating (root) permissions. For further convenience, the Magisk module, “Busybox for android NDK 1.33.1” was also installed. In addition, on the custom recovery image (twrp), the “multidisabler-samsung 3.1” from Macdonald, I. (2020) addon was installed, in order to disable the auto-reflash of stock recovery, which is implemented from Samsung as a countermeasure, against the unauthorized installation of third-party firmware.

2.1.3. Wiping the device

By enabling the parameter “Use rm -rf instead of formatting” in the Settings menu of TWRP, a force deletion of every file/folder in the userdata directory was succeeded.

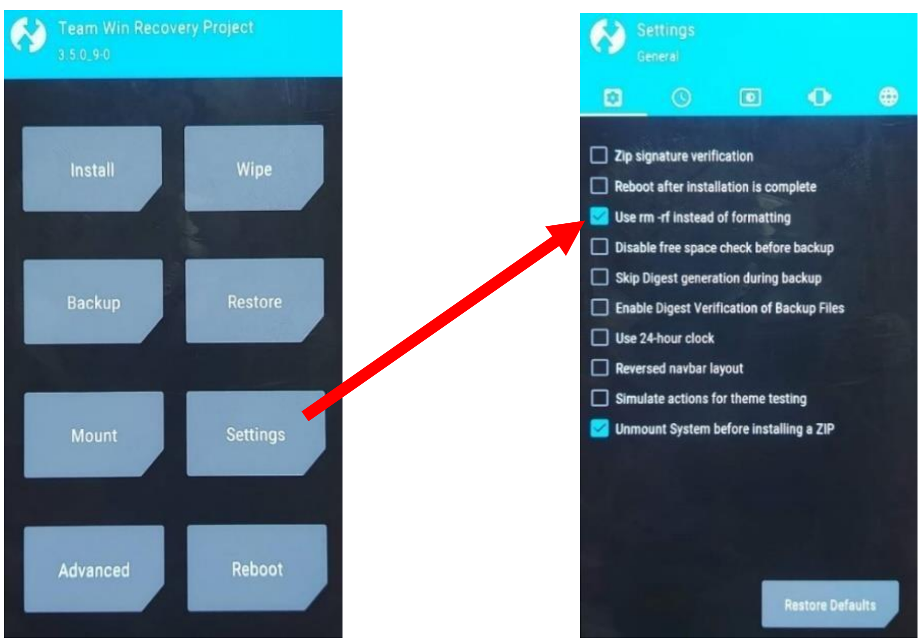


Figure 9. TWRP settings parameter “Use rm -rf instead of formatting”.

[1] This is an unofficial TWRP. At the time there is not an official TWRP supporting android 11 for Samsung A50.

2.2. Extracting the data manually

For the acquisition of the data with ADB commands (Tamma et el., 2020), both Windows and Linux environment systems were utilized. In the Windows environment (Microsoft Windows 10) the SDK platform tools v. 31.03 were utilized. Furthermore, the “ncat” binary was used instead of the netcat executable that is detected as a malicious one from the majority of antiviruses. Additionally, all the executables were added to the environmental path of the operating system for convenience. In the Linux environment (ubuntu 20.04) the same version of the SDK platform tools was installed by copying them to the usr/bin folder instead of installing the obsolete ones from the repository of the Ubuntu operating system. The following commands were used:

- Directing commands to the only attached USB device (mobile phone) (“-d”) or (“-s”)[2]

$ adb -d shell

- Getting root access (super user)

$ su

- Previewing partition table[3]

cat /proc/partitions

```
lucye:/ # cat /proc/partitions
major minor  #blocks  name

 254        0     967212 zram0
 254        1     967212 zram1
   8        0   30912512 sda
   8        1      49152 sda1
   8        2      49152 sda2
   8        3      32768 sda3
   8        4      10240 sda4
   8        5       6144 sda5
   8        6      32768 sda6
   8        7      67584 sda7
   8        8        512 sda8
   8        9        512 sda9
   8       10        512 sda10
   8       11        512 sda11
   8       12        512 sda12
   8       13      32768 sda13
   8       14    5726208 sda14
   8       15     524288 sda15
 259        0     352256 sda16
 259        1   24018944 sda17
 259        2       7636 sda18
```

Figure 10. Partition table view of LG G6 device via ADB command.

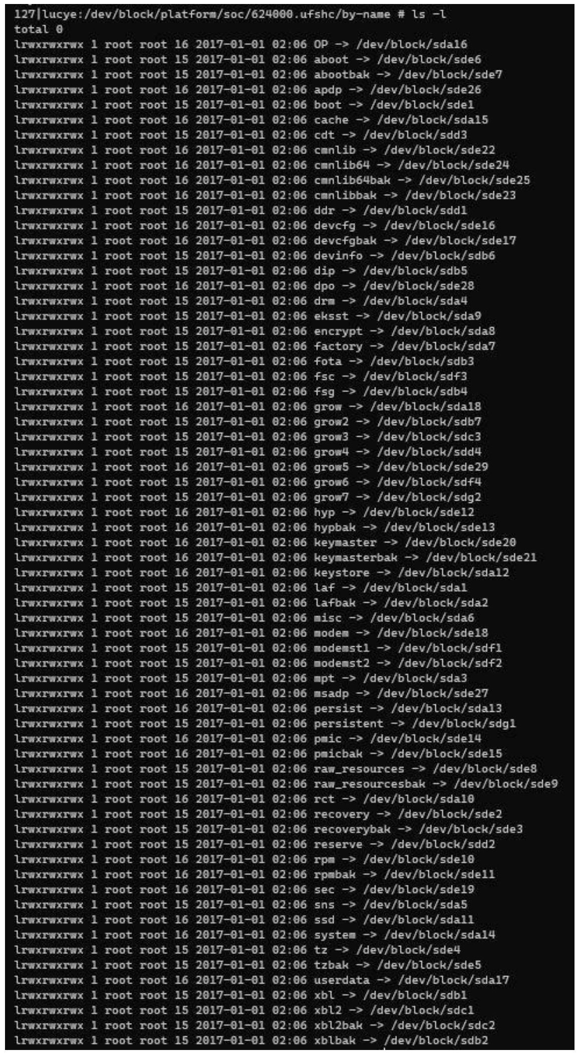
- Previewing the symlinks of the partitions table by navigating to the path «dev/block/platform/soc/624000.ufshc/by-name» and listing all the files.

cd / dev/block/platform/soc/624000.ufshc/by-name

[2] if multible devices were connected.

[3] Notice: the largest partitions usually is the whole block.

ls -l



```
127|lucye:/dev/block/platform/soc/624000.ufshc/by-name # ls -l
total 0
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 OP -> /dev/block/sda16
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 aboot -> /dev/block/sde6
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 abootbak -> /dev/block/sde7
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 apdp -> /dev/block/sde26
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 boot -> /dev/block/sde1
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 cache -> /dev/block/sda15
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 cdt -> /dev/block/sdd3
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 cmnlib -> /dev/block/sde22
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 cmnlib64 -> /dev/block/sde24
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 cmnlib64bak -> /dev/block/sde25
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 cmnlibbak -> /dev/block/sde23
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 ddr -> /dev/block/sdd1
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 devcfg -> /dev/block/sde16
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 devcfgbak -> /dev/block/sde17
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 devinfo -> /dev/block/sdb6
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 dip -> /dev/block/sdb5
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 dpo -> /dev/block/sde28
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 drm -> /dev/block/sda4
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 eksst -> /dev/block/sda9
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 encrypt -> /dev/block/sda8
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 factory -> /dev/block/sda7
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 fota -> /dev/block/sdb3
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 fsc -> /dev/block/sdf3
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 fsg -> /dev/block/sdb4
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 grow -> /dev/block/sda18
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 grow2 -> /dev/block/sdb7
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 grow3 -> /dev/block/sdc3
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 grow4 -> /dev/block/sdd4
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 grow5 -> /dev/block/sde29
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 grow6 -> /dev/block/sdf4
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 grow7 -> /dev/block/sdg2
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 hyp -> /dev/block/sde12
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 hypbak -> /dev/block/sde13
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 keymaster -> /dev/block/sde20
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 keymasterbak -> /dev/block/sde21
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 keystore -> /dev/block/sda12
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 laf -> /dev/block/sda1
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 lafbak -> /dev/block/sda2
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 misc -> /dev/block/sda6
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 modem -> /dev/block/sde18
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 modemst1 -> /dev/block/sdf1
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 modemst2 -> /dev/block/sdf2
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 mpt -> /dev/block/sda3
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 msadp -> /dev/block/sde27
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 persist -> /dev/block/sda13
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 persistent -> /dev/block/sdg1
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 pmic -> /dev/block/sde14
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 pmicbak -> /dev/block/sde15
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 raw_resources -> /dev/block/sde8
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 raw_resourcesbak -> /dev/block/sde9
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 rct -> /dev/block/sda10
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 recovery -> /dev/block/sde2
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 recoverybak -> /dev/block/sde3
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 reserve -> /dev/block/sdd2
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 rpm -> /dev/block/sde10
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 rpmbak -> /dev/block/sde11
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 sec -> /dev/block/sde19
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 sns -> /dev/block/sda5
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 ssd -> /dev/block/sda11
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 system -> /dev/block/sda14
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 tz -> /dev/block/sde4
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 tzbak -> /dev/block/sde5
lrwxrwxrwx 1 root root 16 2017-01-01 02:06 userdata -> /dev/block/sda17
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 xbl -> /dev/block/sdb1
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 xbl2 -> /dev/block/sdc1
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 xbl2bak -> /dev/block/sdc2
lrwxrwxrwx 1 root root 15 2017-01-01 02:06 xblbak -> /dev/block/sdb2
```

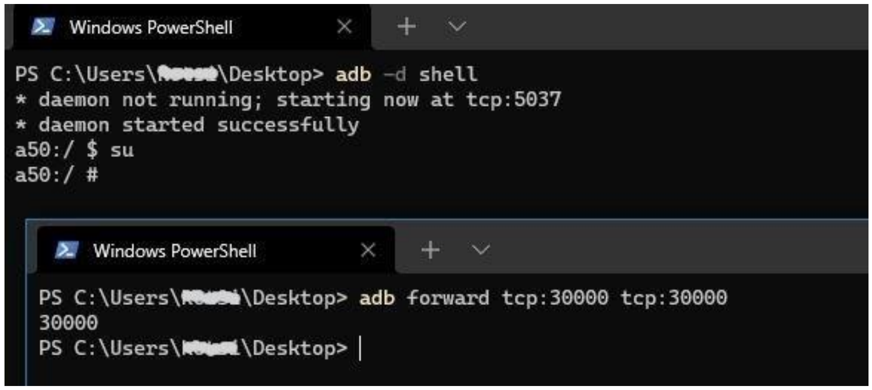
Figure 11. Partition table symlinks view of LG G6 device via ADB command.

This could be very useful when a targeted acquisition is needed (e.g., userdata partition).

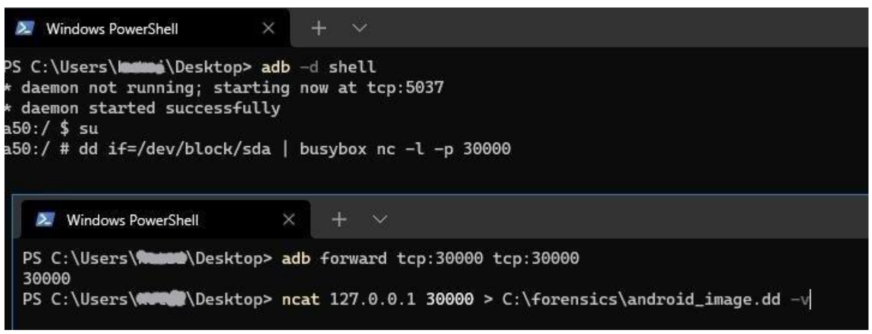
```
lrwxrwxrwx 1 root root 16 2017-01-27 01:00 userdata -> /dev/block/sda17
```

- Acquiring data. The procedure is divided in three parts:
  - Firstly, in a terminal a bridge must be established between the host computer and the device in order for the acquired data to be copied back to the host:
  - adb forward tcp:30000[4]

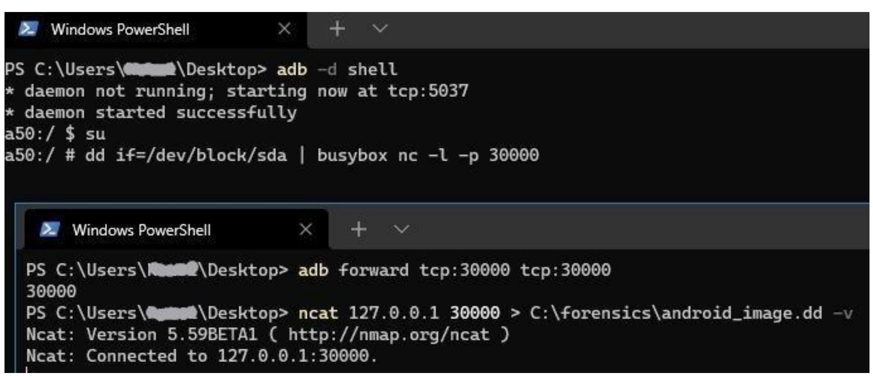
[4] The usage of a high port number is recommended because windows 10 OS tends to block any connection in lower ones.



- Secondly, in another terminal which is already running inside the shell of the android device the partition that must be acquired can be copied through the desired port to the host computer. By choosing the sda partition in Samsung A50 device, a full physical image acquisition is conducted:
- # dd if=/dev/block/sda | nc -l -p 30000



- Thirdly, the acquired image can be "pushed" to a chosen path in the first terminal via the netcat[5] executable:
- ncat 127.0.0.1 30000 > android_image.dd -v (-v for verbose)



On some occasions, when specific folders and files are aimed, a targeted extraction can be applied. A forensic copying could be done, to a SD card[6], by a combination of "adb pull" and copy (cp) commands. The main concern in this scenario is to preserve the source's timestamps.

[5] "nc" in Linux or "ncat" in Windows.

[6] An external SD card was used in order not to ensure the integrity of the internal storage of the examined device.

This can be achieved by adding the “-a” parameter in the adb pull command. For instance, the commands used for the forensic extraction of a specific Viber application’s folder were:

- cp -pvR /data/data/com.viber.voip /sdcard0/forensics/ (-p for preserve, -v for verbose, -R for recursively)

or

- cp -R --preserve=timestamps /data/data/com.viber.voip/databases/* sdcard0/forensics/ (coping only the databases’ folder while preserving the timestamps)
- adb pull -a /sdcard0/forensics/com.viber.voip C:\forensics\viber_data_package (-a preserves file timestamp and mode)

The usage of two different commands was chosen because, when trying to execute the command “adb pull -a” directly from the «/data/data» folder it was unsuccessful, since the “adb root” command was not working. This feature is by default disabled on production build devices and in order to enable it the “ro. secure” file (Skulkin et al., 2018) has to be modified.

2.3. Extracting the data via software

In contrast to the command line approach (adb), several user-friendly forensic tools can be used for the acquisition of an android device. Some of them are presented below.

➢ **Magnet Acquire**.

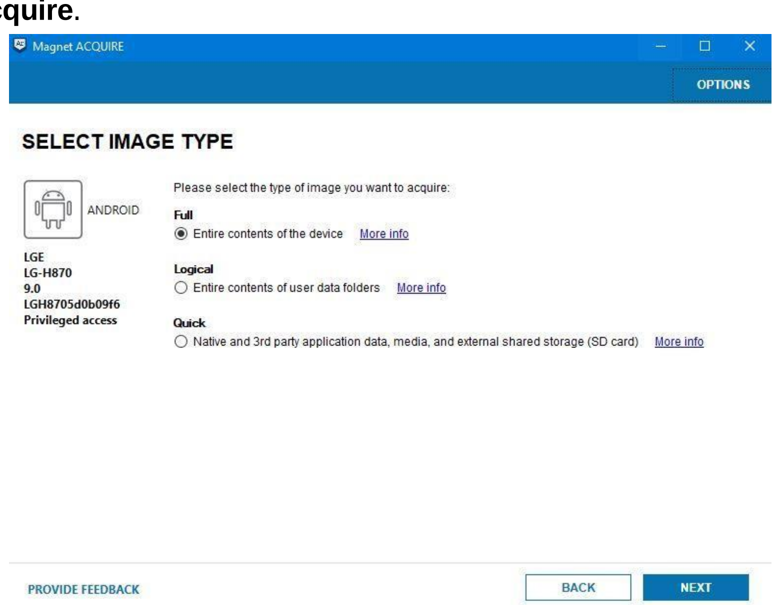
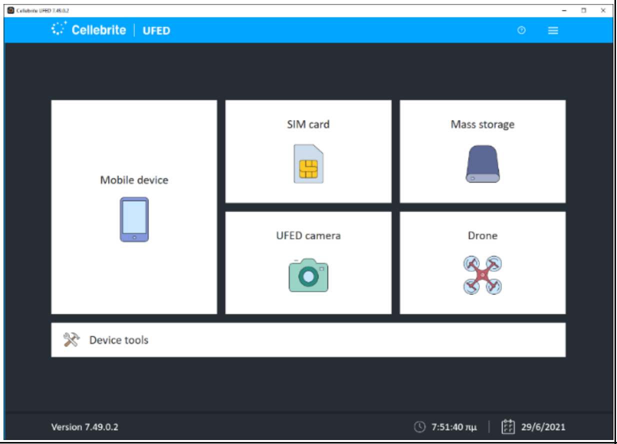
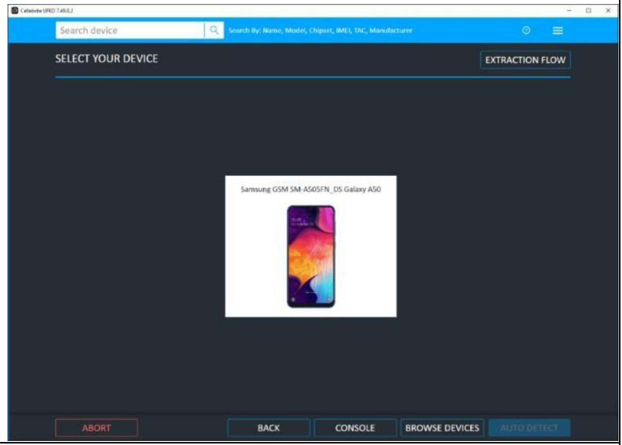


Figure 12. Magnet Acquire user interface while LG G6 is connected.

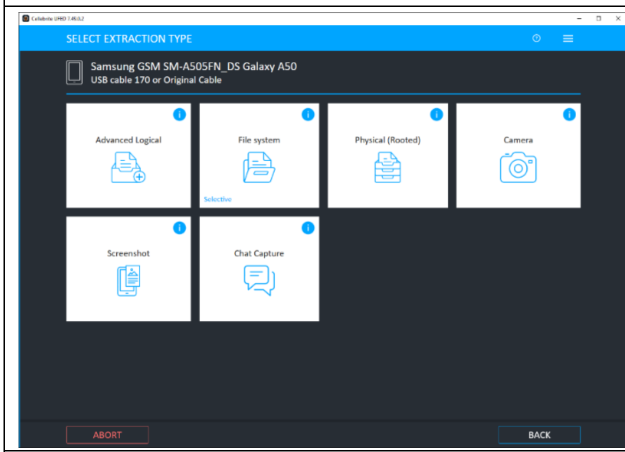
- **UFED 4PC.**

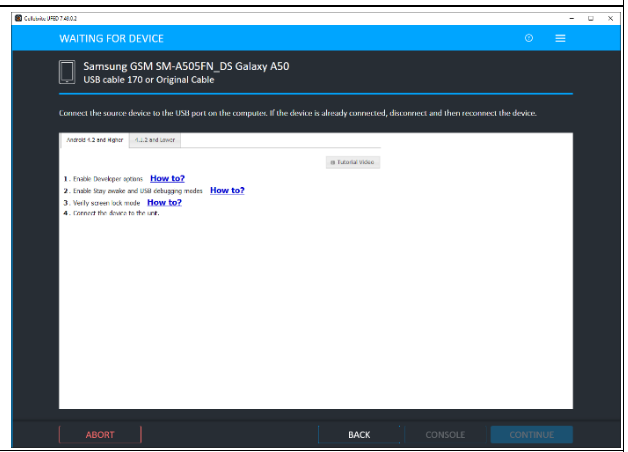
Software GUI	When the device is connected and auto detected by the software



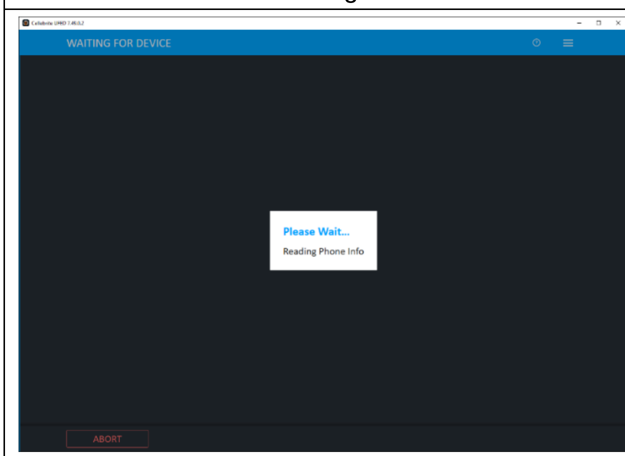


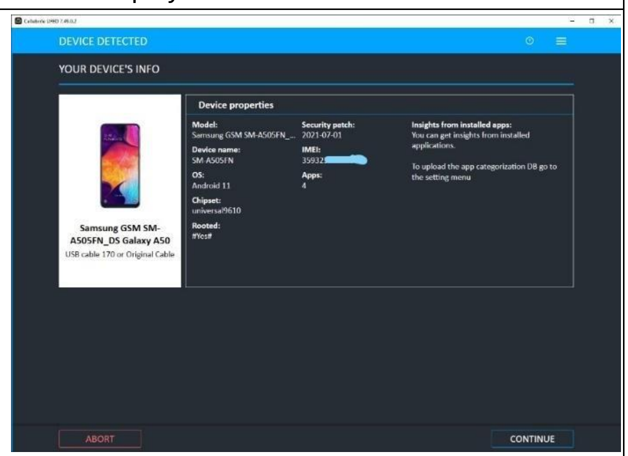
The options are more or less the same for every android device	An Advanced Logical Acquire is feasible by enabling some parameters under the Developer Options Settings



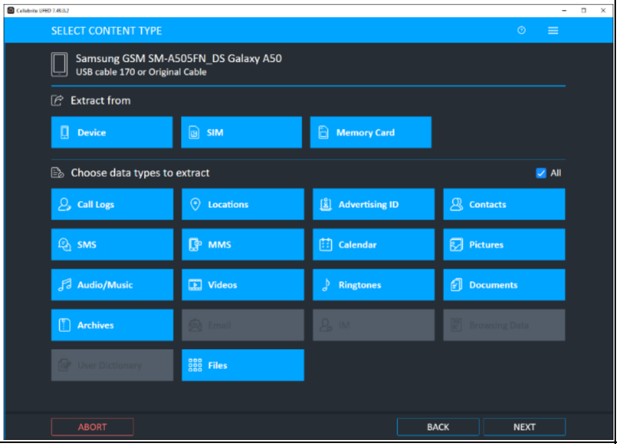


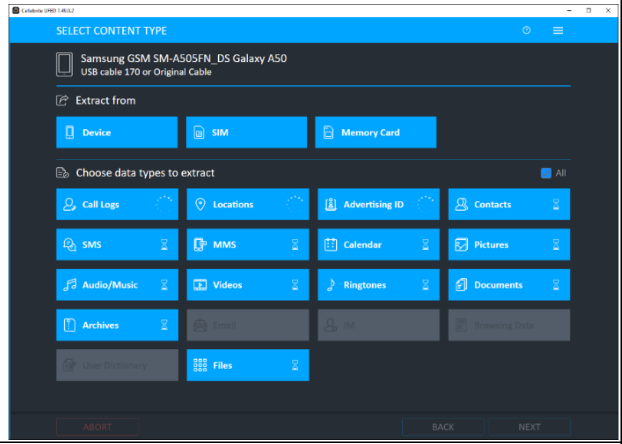
After doing that the device info is read and displayed to the examiner





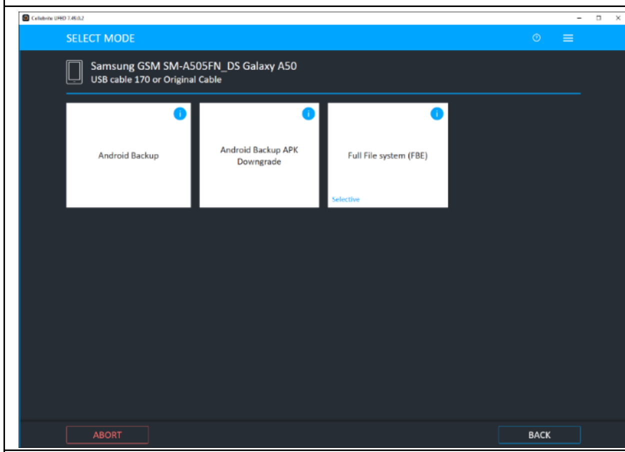
Next step is choosing the artifacts and data extracted

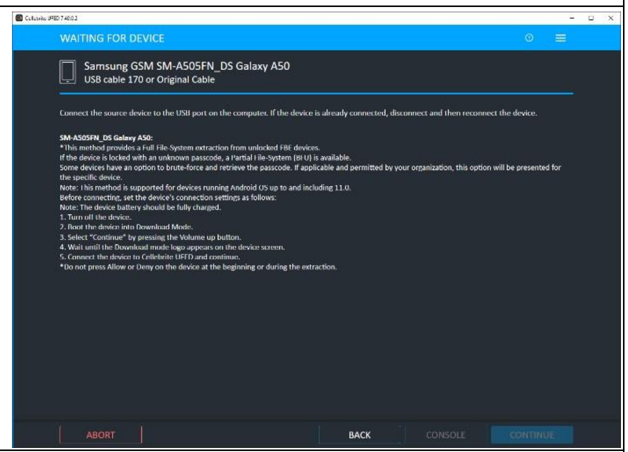




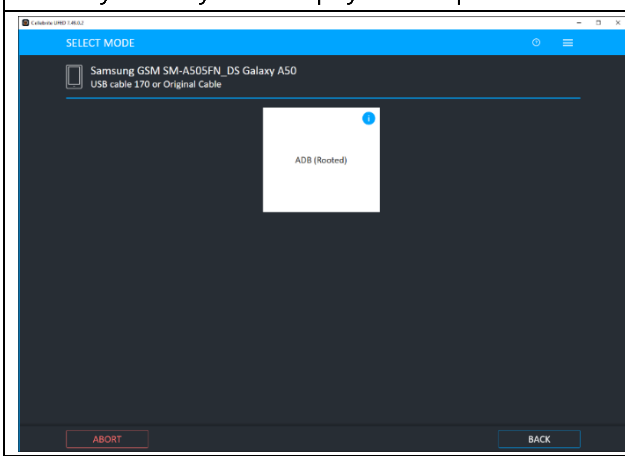
There is also a File System acquisition choice to extract data from the device's File System

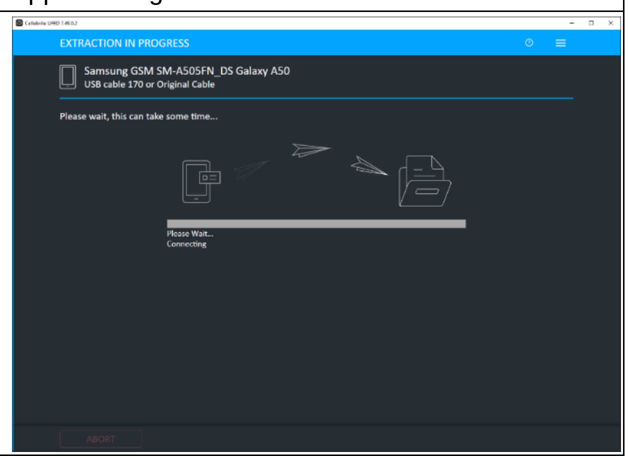
Most cases require to boot the device into Download Mode

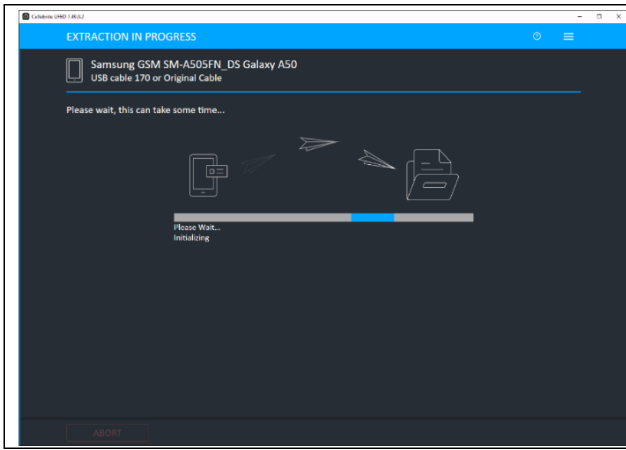




Finally in many cases a physical acquisition is also supported regardless the root status of the device







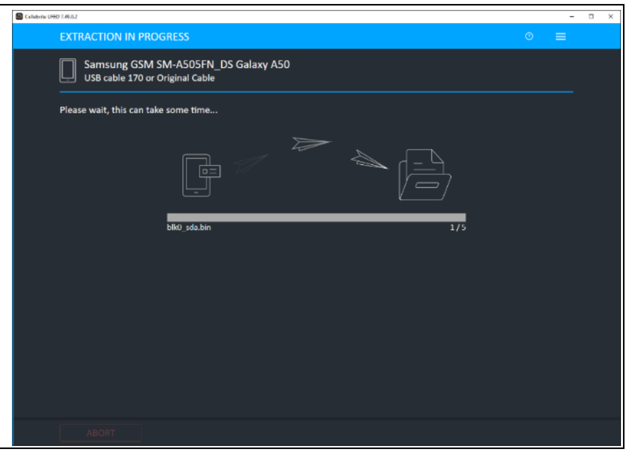
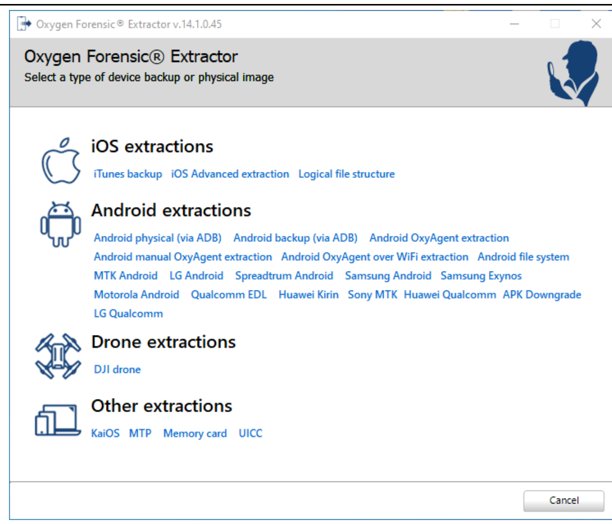
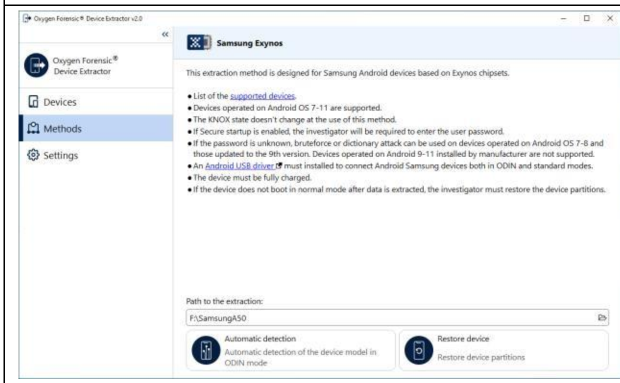


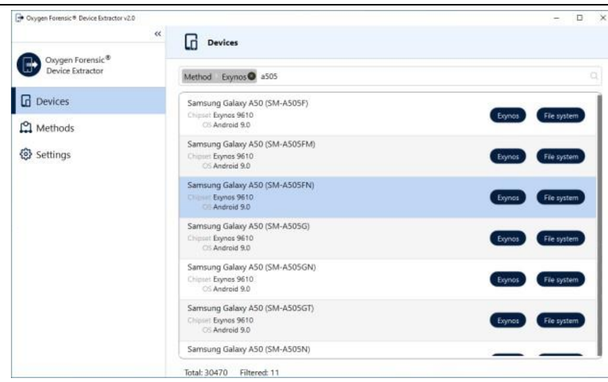
Table 2. Cellebrite UFED 4PC user interface. Acquisition methods while the Samsung A50 device is connected.

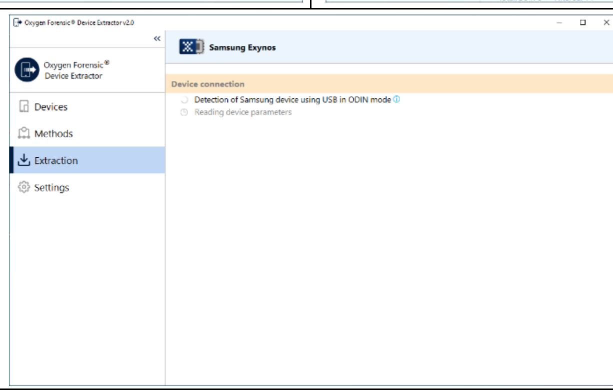
➢ OXYGEN

<table>
<tr><th colspan="2">Software GUI</th></tr>
<tr><td colspan="2">
</td></tr>
<tr><td>Samsung Android Option</td><td>Selection of Supported Samsung Devices in Samsung Android Option
(searching model “A50”)</td></tr>
</table>

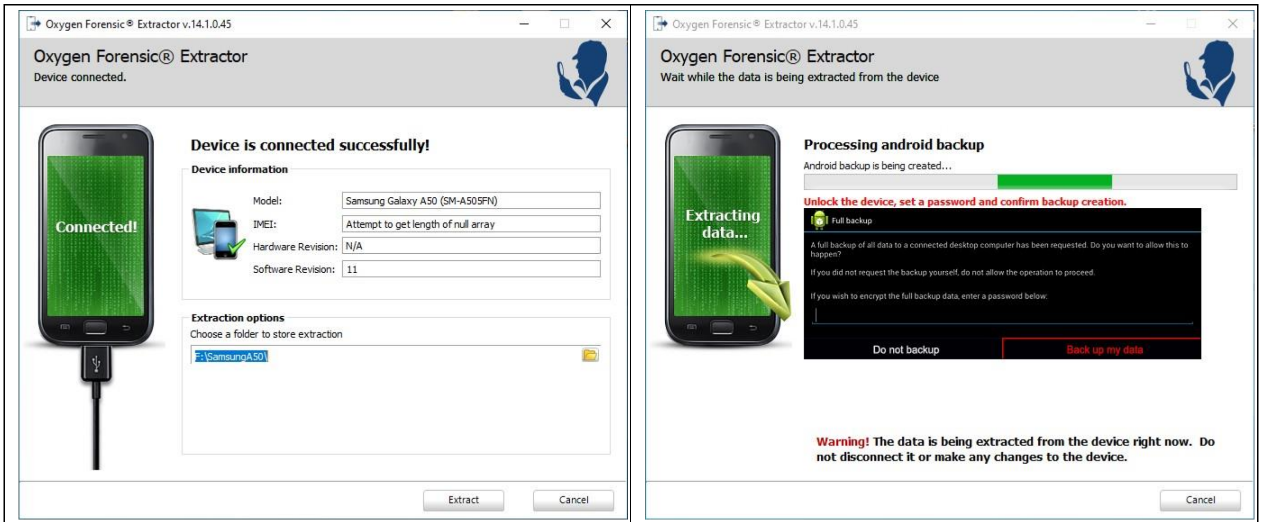
Samsung Exynos Option







Android Backup (via ADB)

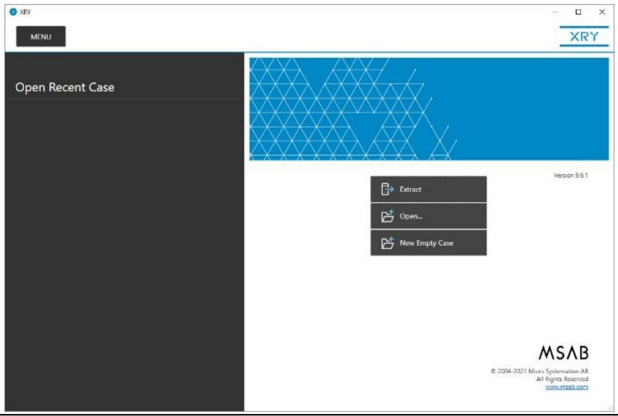


Android Physical (via ADB)

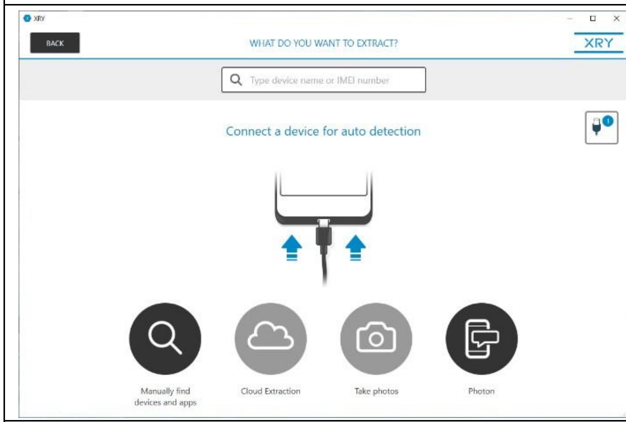
Table 3. OXYGEN Forensic Extractor user interface. Acquiring methods while the Samsung A50 device is connected.

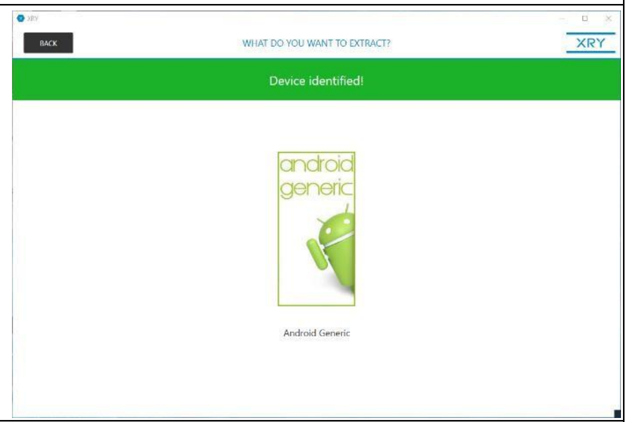
- **XRY**

Software GUI

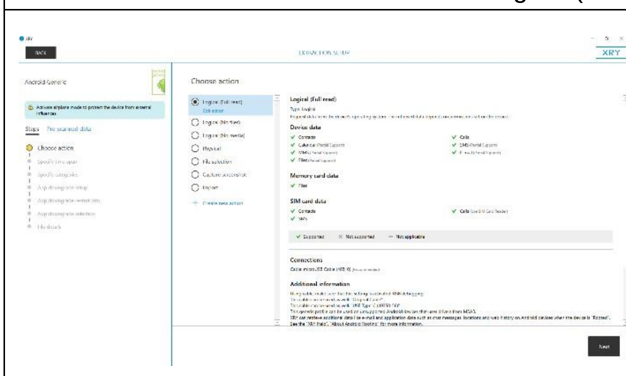


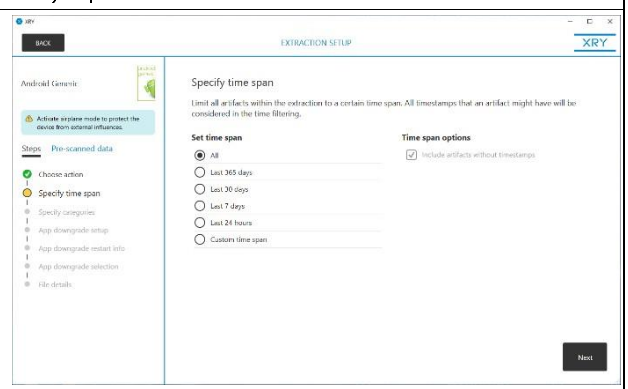
Connecting the device

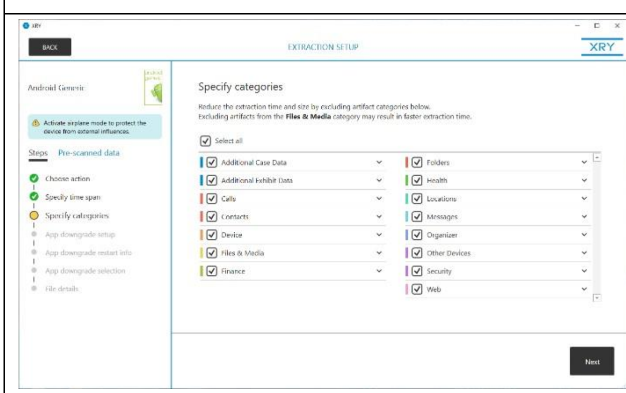


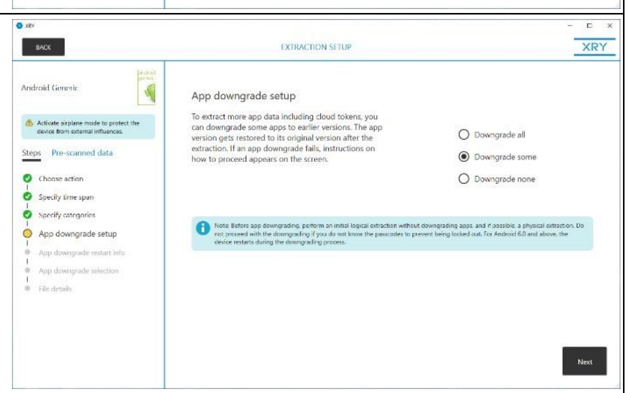


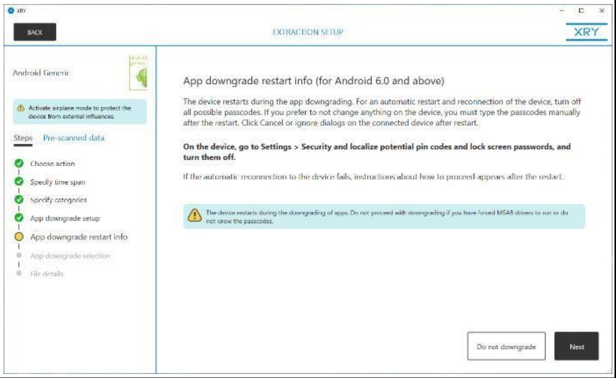
Logical (Full read) Option

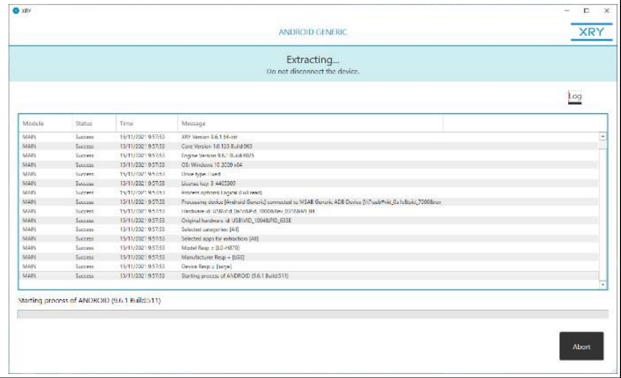


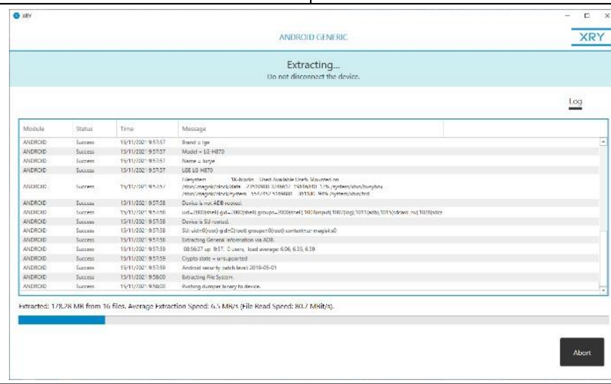




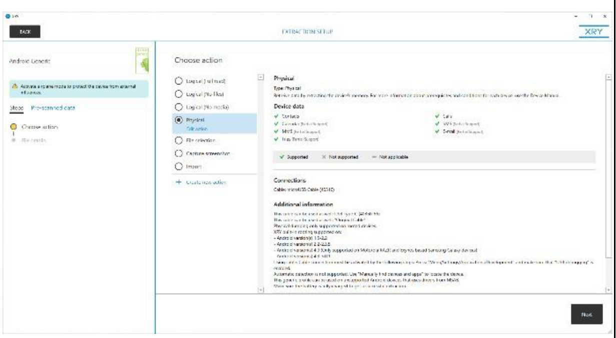








Physical Option



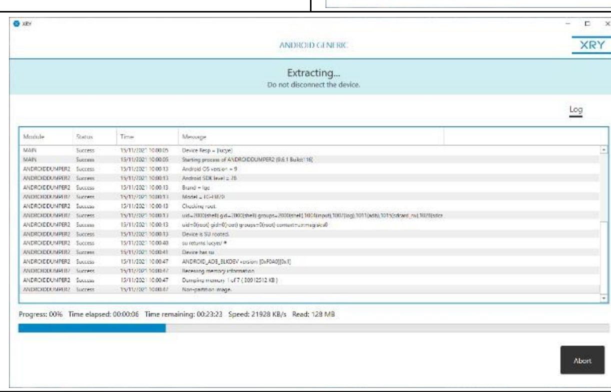


Table 4. MSAB XRY user interface. Acquiring methods while the LG G6 device is connected.

2.3.1. Speed of imaging and transferring data

The speed of data extraction of a device's storage is depended on the quality and length of used cables and of course the device itself. Some vendors, as a cost reduction method, implement an older USB 2.0 chipset on the devices irrespective the existence of a high-speed USB – C port on them. To take a glance of how the transfer speed is affected by these two factors in two different devices with the same acquiring setup (same PC, cable and software), an example between Samsung A50 and LG G6 is presented below (**Figures 13** and **14**). As shown the write speed is dramatically lower in Samsung, with value of 43MB/sec, compared with LG' s 108MB/s while using the Magnet Acquire forensic tool.

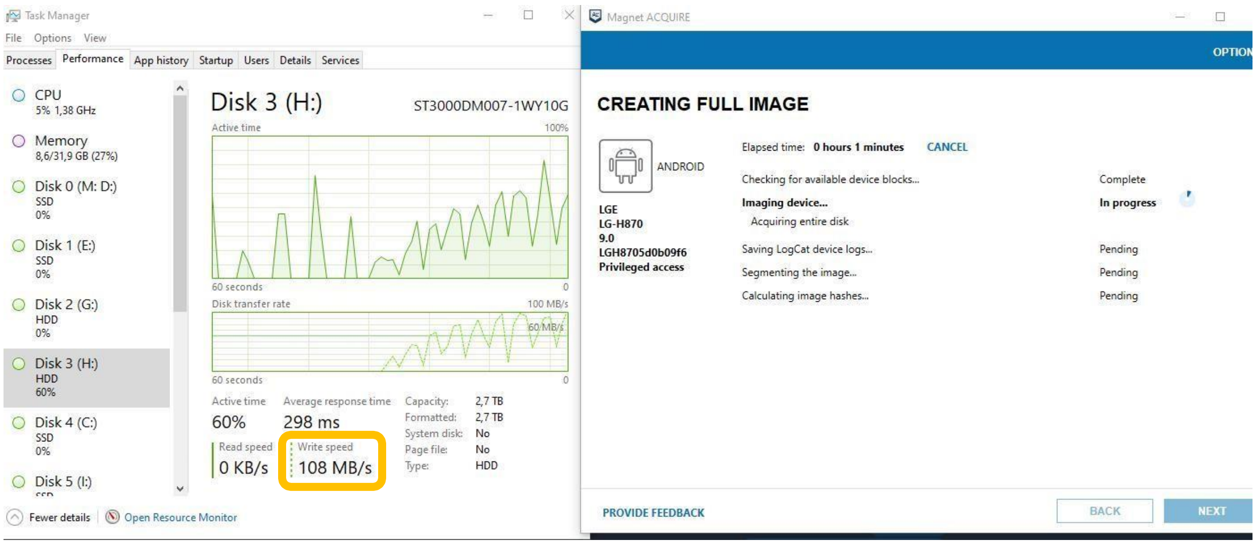


Figure 13. Speed monitoring while acquiring LG G6 device with MAGNET Acquire.

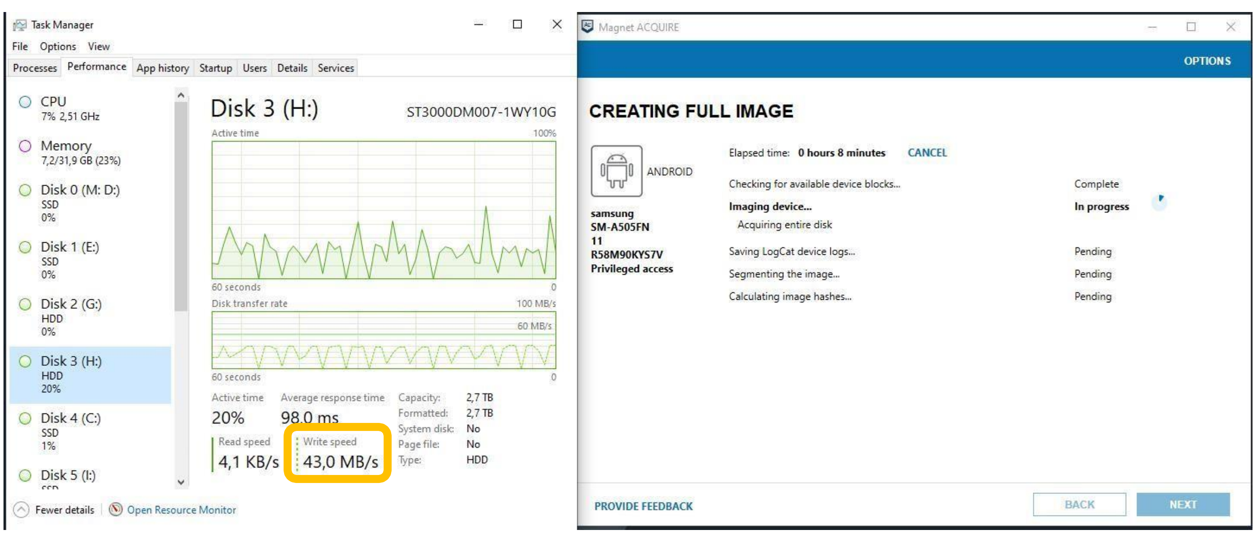


Figure 14. Speed monitoring while acquiring Samsung A50 device with MAGNET Acquire.

2.4. Examination of the acquired data

There is a wide range of options for software and tools specialized in the examination and analysis of the acquired databases. The criteria of choice are the examiners' experience and the available budget. While non-commercial tools may lack of special features, technical support and a user-friendly interface, the commercial ones have a significant cost that tend to be in the form of an annual subscription. Below are depicted some of the user interfaces of the abovementioned tools that are referred in paragraph §**1.4.2**. It has to be mentioned that the following used tools are for demonstration reasons and in-depth exploring their capabilities is out of scope of this Thesis. The choice of the used tool was depended on the examination and the analysis of the instant messaging application's databases.

- **aLEAPP**.

Figure 15. aLEAPP user interface.

- **DB Browser for SQLite**.

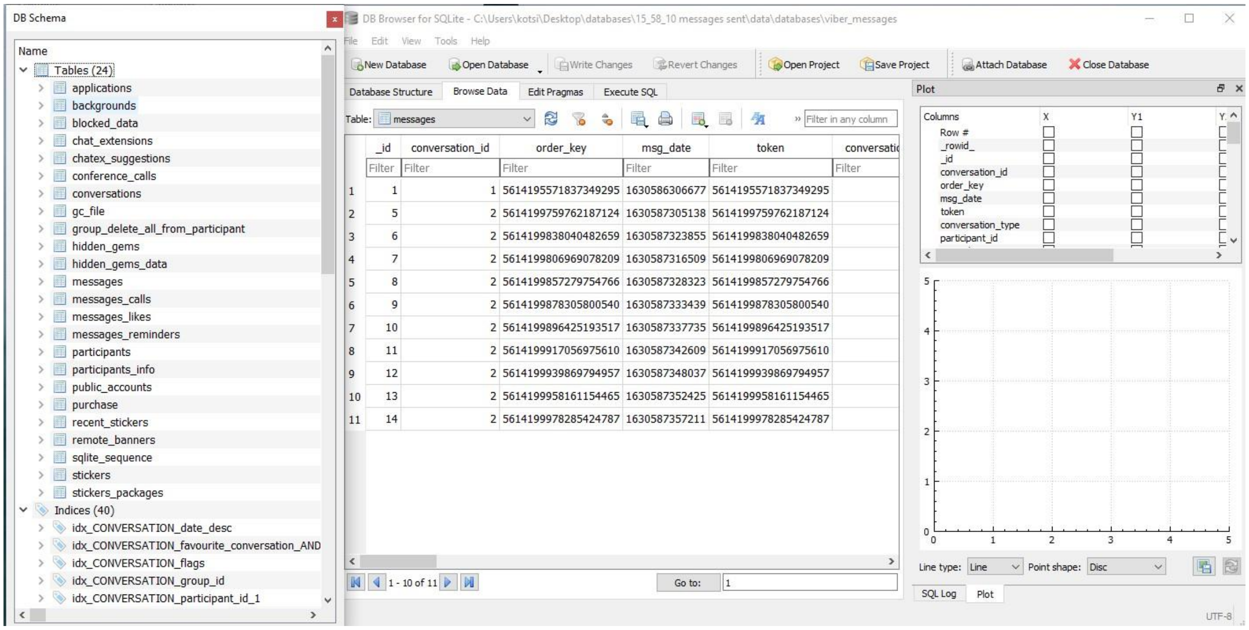


Figure 16. DB Browser for SQLite user interface (database: "viber_messages.db").

- **SQlite Expert**.

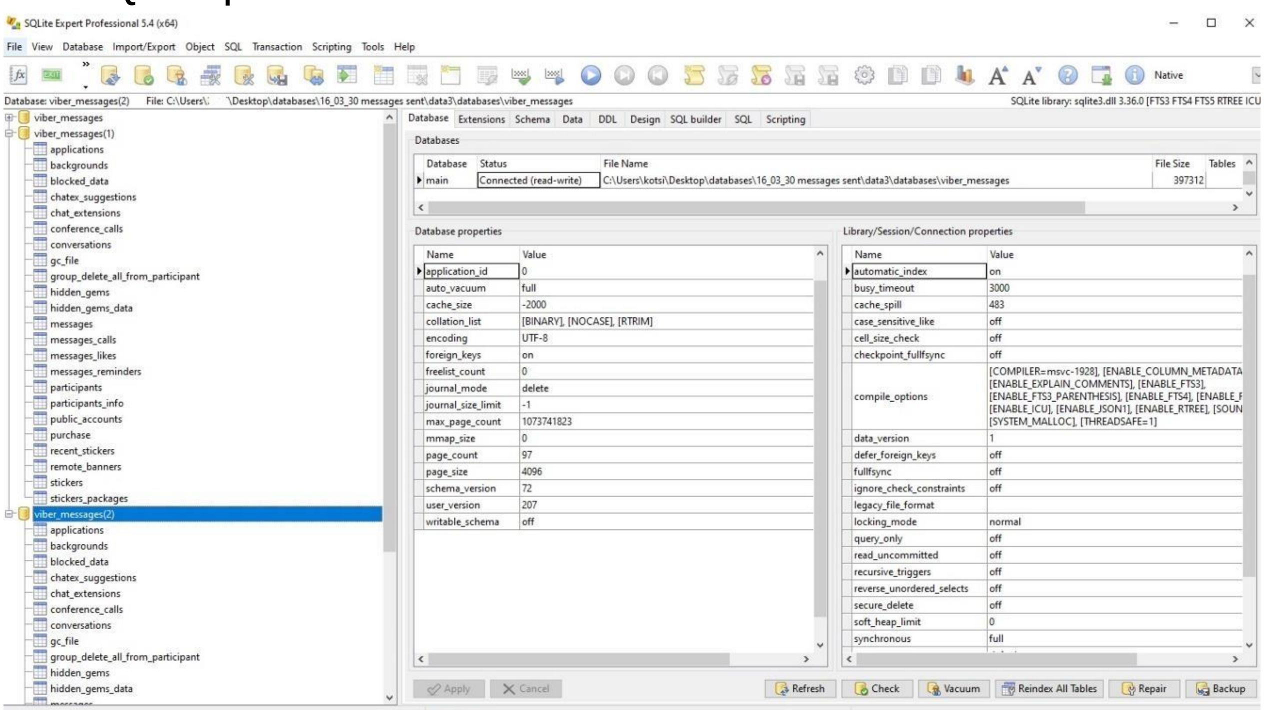


Figure 17. SQlite Expert user interface (database: "viber_messages.db").

- **Oxygen forensic SQlite Viewer.**

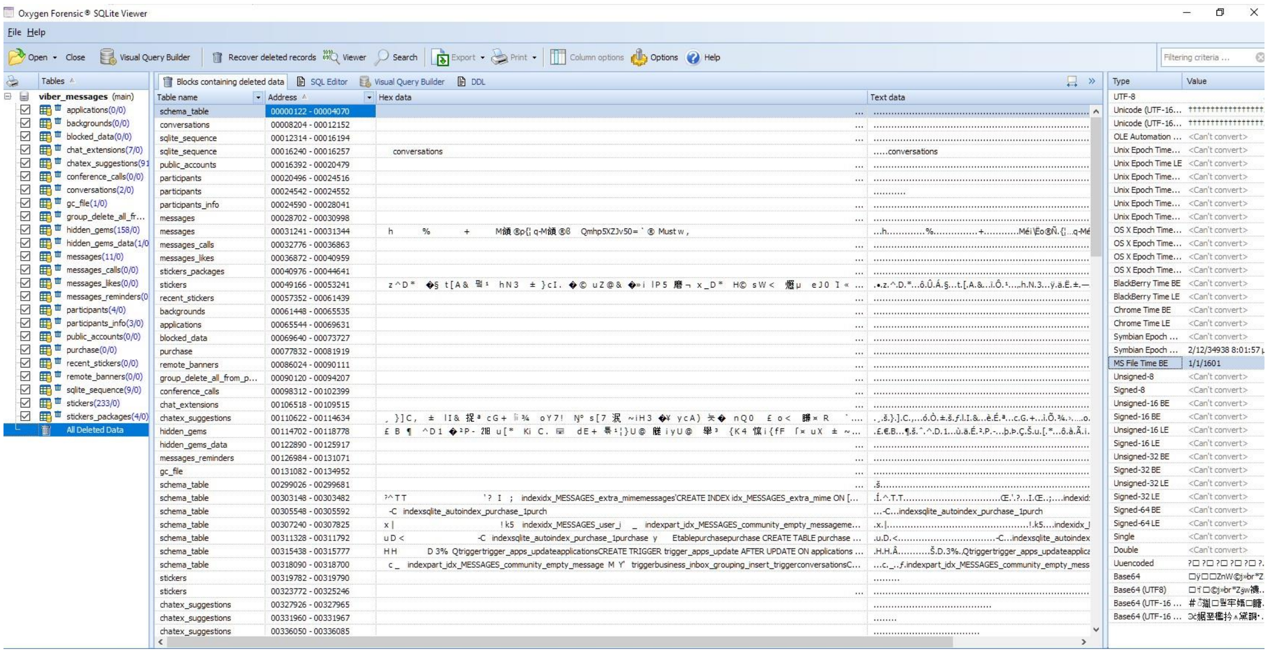


Figure 18. Oxygen forensic SQlite Viewer user interface (database: “viber_messages.db”).

- **Magnet Axiom Forensics.**

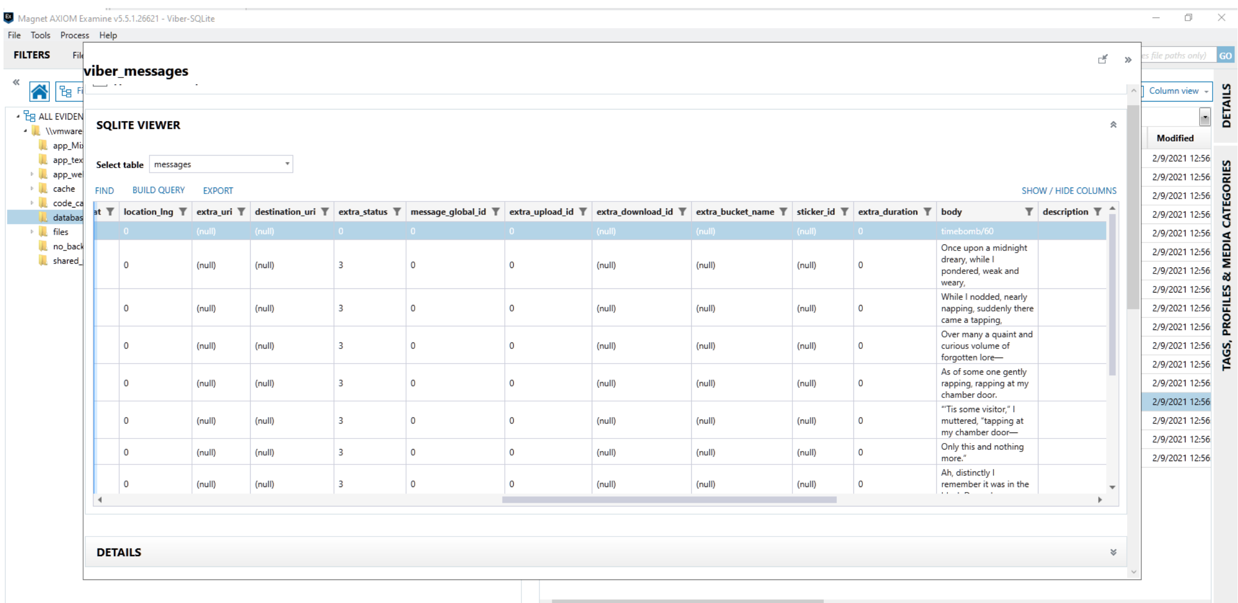


Figure 19. Magnet Axiom Forensics user interface (database: “viber_messages.db”).

➢ **UFED Physical Analyzer**.

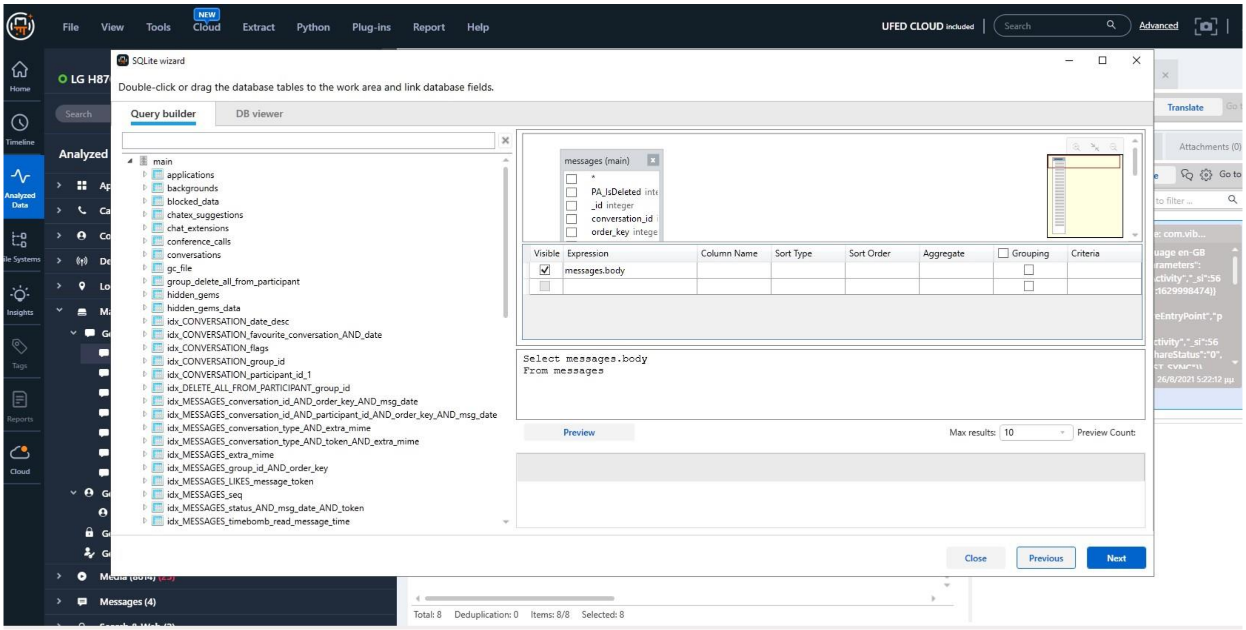


Figure 20. UFED Physical Analyzer user interface (database: “viber_messages.db”).

➢ **Autopsy Suite**.

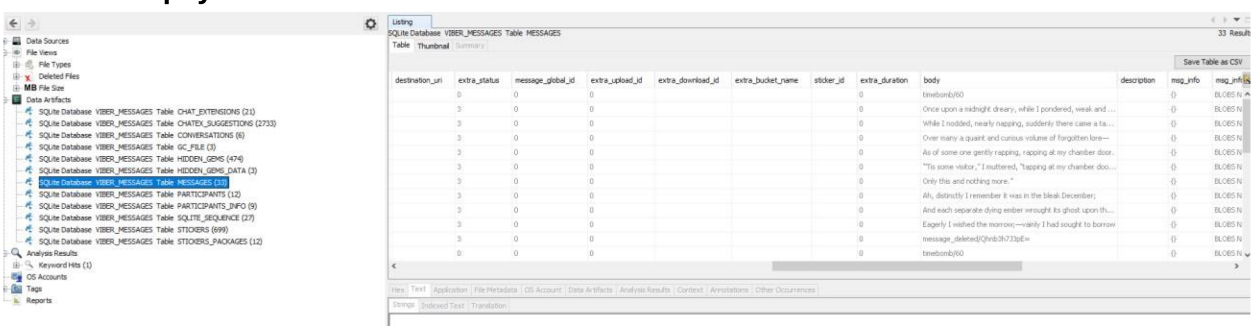


Figure 21. Autopsy Suite user interface (database: “viber_messages.db”).

➢ **Paraben E3**.

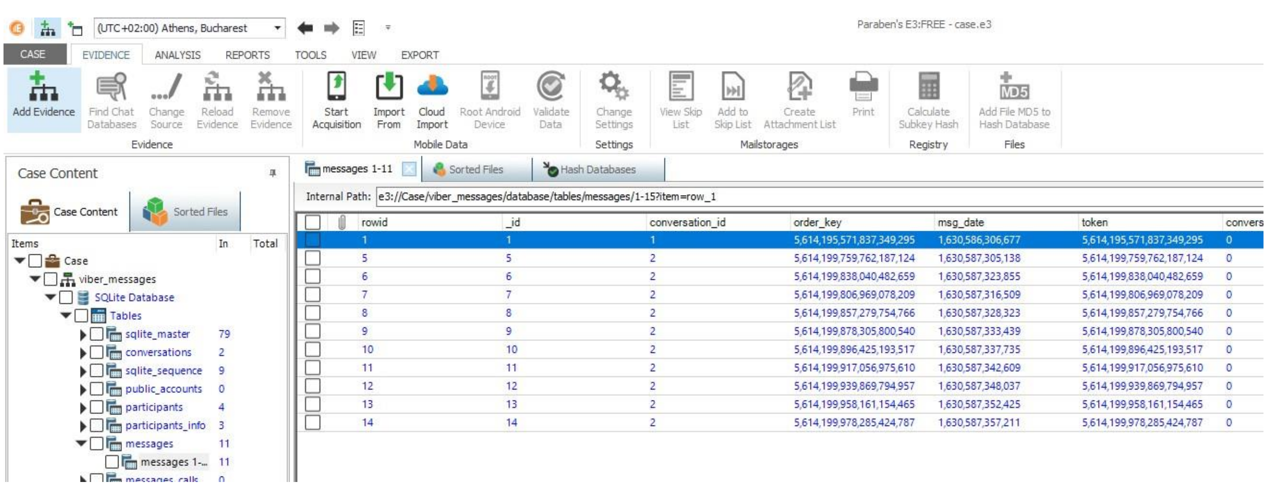


Figure 22. Paraben E3 user interface (database: “viber_messages.db”).

- **Belkasoft X.**

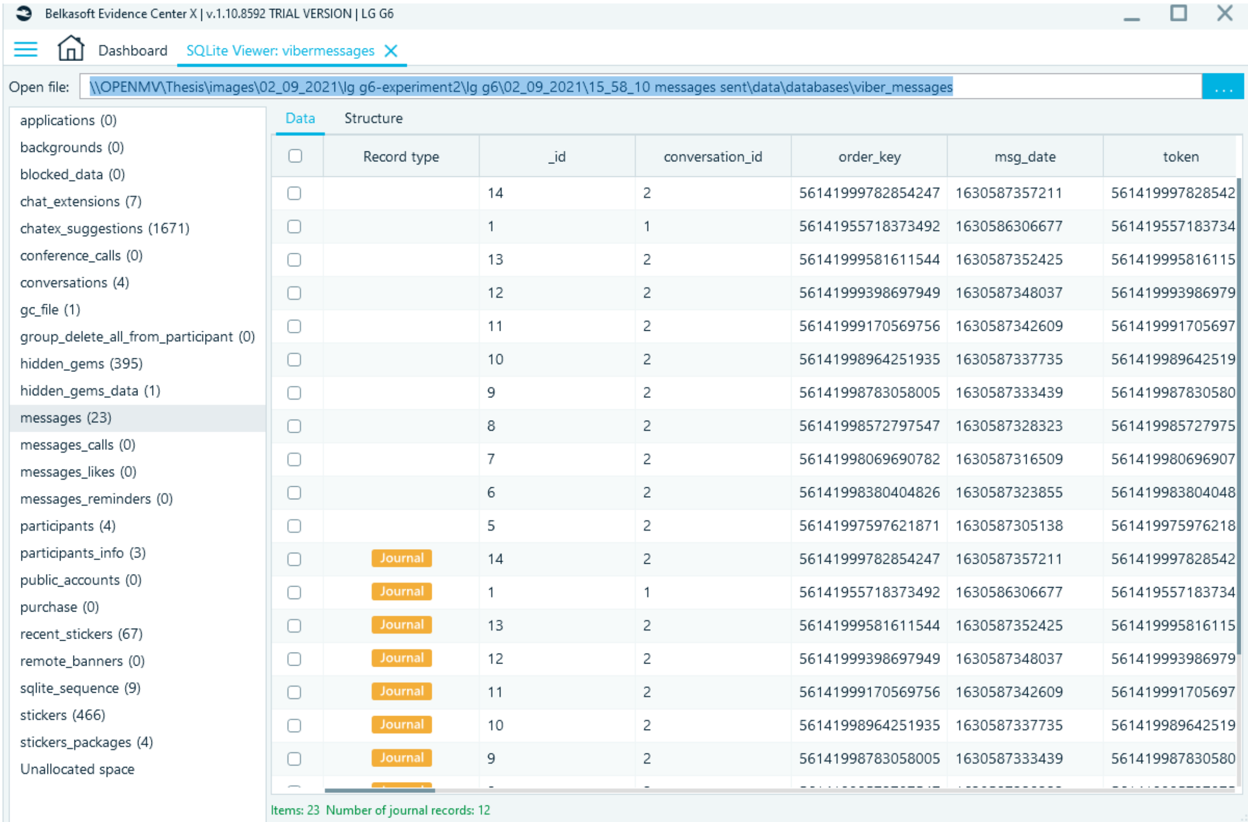


Figure 23. Belkasoft X user interface (database: "viber_messages.db").

- **X-ways Forensics.**

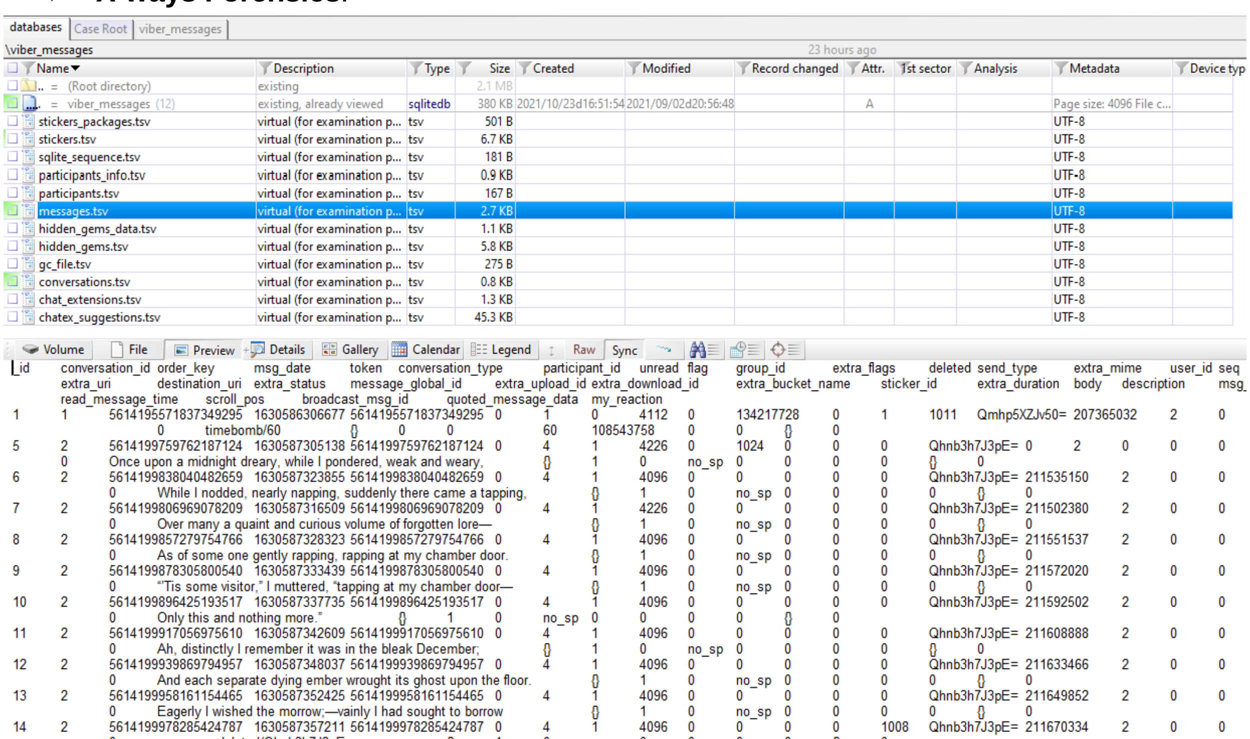


Figure 24. X-ways Forensics user interface (database: “viber_messages.db”).

- **SQLite Forensic Explorer**.

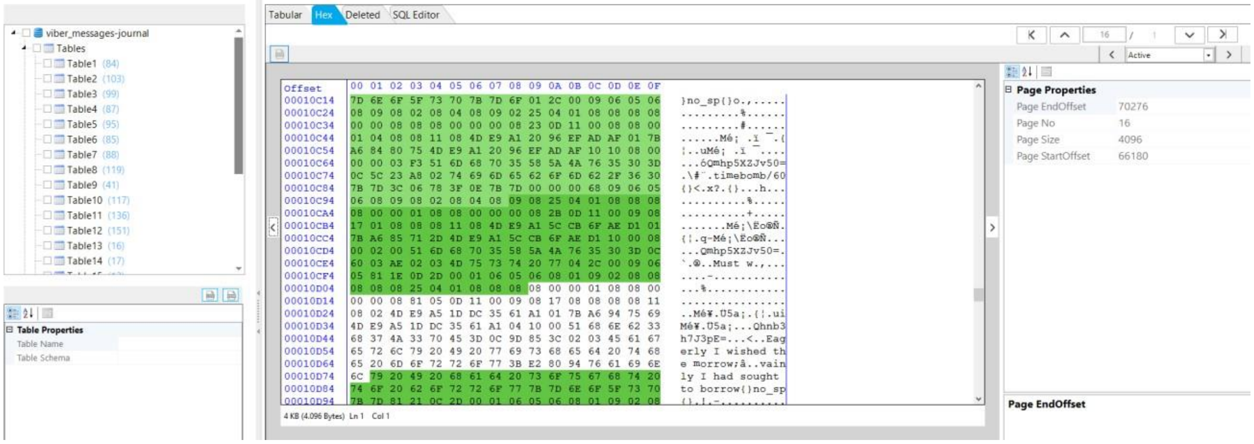


Figure 25. SQLite Forensic Explorer user interface (database: “viber_messages.db”).

- **FQLite**.

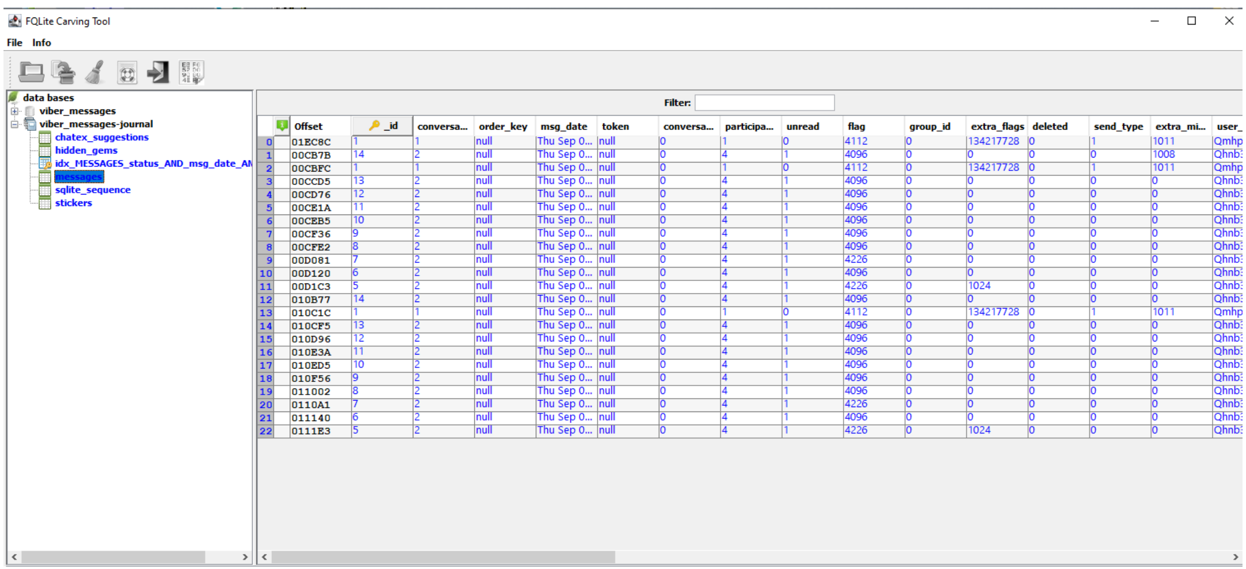


Figure 26. FQLite user interface (database: “viber_messages.db”).

- **Andriller**.

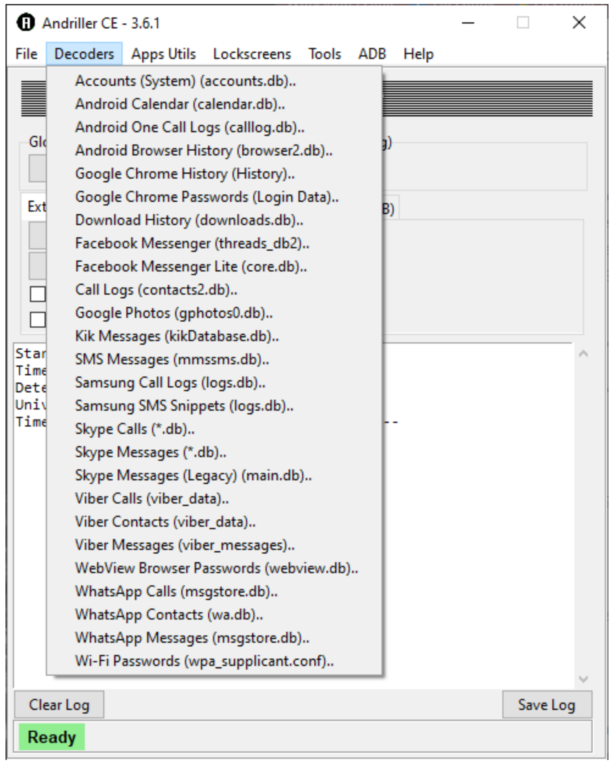


Figure 27. Andriller user interface (default decoders for databases).

- **Forensic Toolkit for SQLite**.

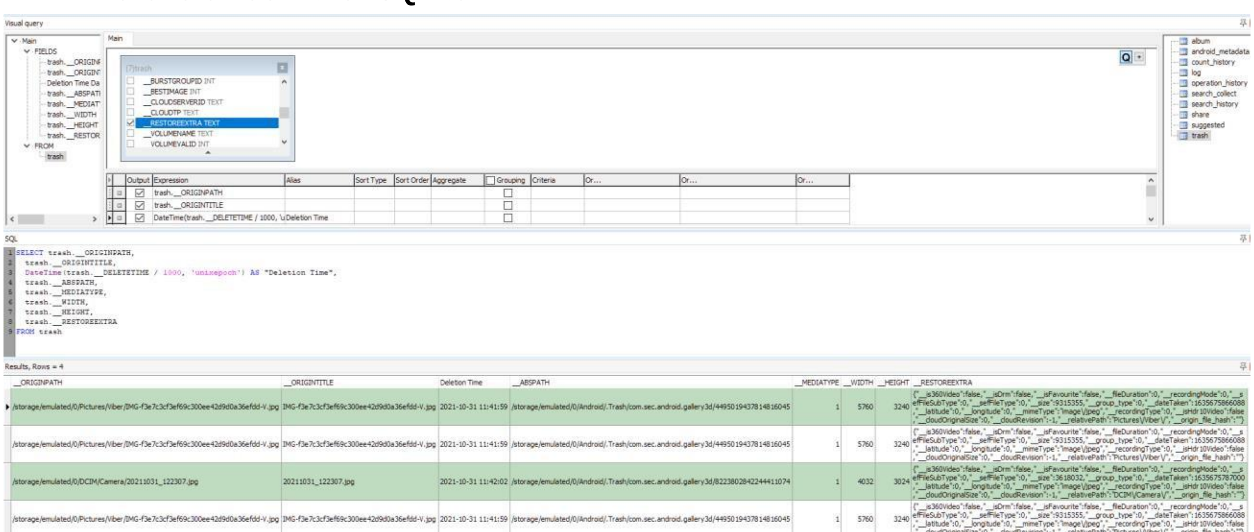


Figure 28. Forensic Toolkit for SQLite user interface.

Chapter 3.

Android Incident Response Scenario

This Thesis focused on DFIR actions and their consequences. This was achieved by creating and executing a possible realistic scenario. In this chapter are referenced the retained scenario, the desired goals of the experiments, the methodology and the procedure on which the choice of instant messaging applications was based. General information about the Viber, WhatsApp and Telegram applications and their communication encryption are mentioned. What is more, the location of the applications communication servers for Greece (Athens) territory is presented via an IDS solution implementation (ntopng with geoip plugin) and the difficulty to retrieve content from the companies' cloud servers based on an FBI document.

3.1. The scenario

Some organizations contain and manage much, sensitive, information for example, nuclear power plant or a military intelligence facility. A person inside the organization facilities wants to send a classified piece of information which saw on a desk or on a desktop. This person could be either an insider (Industrial espionage) or a random visitor. So, this individual uses a mobile phone and via a social chatting application sends a message (saved or non-saved-contact) with the intel-photo included. After having sent the message, the in-question chat thread is deleted by the user, before the incident response team has the chance to spot and seize the mobile device. The device is either a LG G6 or a Samsung A50 which were detailed in the previous chapter. In these scenarios both hypothetical situations were included where the shared images were captured directly using the chat application or/and the device's default camera application.

3.2. Experiments Goals

The volatility or the persistence of evidence is a crucial subject in forensics. Specially in mobile forensics one common question which often arises is related to the handling of the device after its seizure and the actions that must be taken by the IFR. For this purpose, a simple scenario was developed, which is not far from reality. In this scenario an employee (insider), or a random visitor, of an organization/industry using a smartphone's instant messaging application, takes pictures containing classified information from a screen of an unlocked workstation and sends them to a third party alongside some text messages. After that, the suspect immediately deletes the in-question chat thread. These actions come to security team's attention, who apprehends the suspect and seizes the device.

Three key aspects regarding the seized device should be examined to determine the best course of actions from an IFR perspective for the retrieval of the incriminating messages and the receiver's identity:

- The time that has passed from the seizure of the device and until its forensic acquisition
- The status of the device

- The interaction with the messaging application either by the user before the seizure or by the IFR after it.

The main goal of these experiments is to determine under which conditions and actions the in-question data are preserved.

3.3. Messaging Application Choice

For the scenario two devices (LG G6 and Samsung A50), running a different Android OS Version, were prepared according to the procedures described in **Chapter 3** of this Thesis. Different sim cards were activated. Nine of the most popular messaging application were installed using the same version of apk (through “adb -d install” command), a new user was added, and chat messages were exchanged. In **Table 5** is presented the popularity of chosen applications, as it is referenced on the official’s digital distributor site and Android Market (Google Play Store). Furthermore, no additional applications were installed (e.g., google accounts) except the under-examination ones to mitigate the level of “noise” and unrelated data.

Application	Total Downloads
Facebook Messenger	5B+
WhatsApp	5B+
Viber	1B+
Telegram	1B+
Skype	1B+
LINE	500M+
imo	500M+
WeChat	100M+
kik	100M+
Signal	50M+

Table 5. Chat applications’ total downloads (Source: Google Play Store).

All the user data including contacts, messages and call logs are stored in SQLite databases inside the applications’ folders in the device’s filesystem. As it was analyzed in paragraph §**1.3.1.2** the key factor for recovering deleted records (e.g., deleted messages in the scenario’s case) depends on the settings and features of these databases. With the usage of forensic tools, described in paragraph §**1.4**, those nine applications’ databases were compared. The forensically interesting settings of these applications are presented in **Table 6**.

Application	Version	Journaling Mode	Encryption	Auto Vacuum mode	Messages Databases
Facebook Messenger	339.0.0.18.118	Journal	No	Full	com.facebook.orca/databases/threads.db2
imo	2021.11.2051	WAL	No	Full	com.imo.android.imoim/databases/imofriends_xxxxxx.db
kik	15.38.1.25235	WAL	No	Full	kik.android/databases/xxxxxx.kikDatabase.db

LINE	11.19.1	Journal	No	Full	jp.naver.line.android/databases/naver_line
Signal	5.26.11	WAL	Yes	Unknown	org.thoughtcrime.securesms/databases/signal.db
Skype	8.78.0.164	WAL	No	Full	com.skype.raider/databases/s4l-live:cid.xxxxx.db
Telegram	7.9.3	WAL	No	None	org.telegram.messenger/files/cache4.db
Viber	16.4.0.8	Journal	No	Full	com.viber.voip/databases/"viber_messages.db"
WeChat	8.0.16	WAL	Yes	None	com.tencent.mm/files/Micromsg/xxxxxx/EnMicroMsg.db
WhatsApp	2.21.22.27	WAL	No	Full	com.whatsapp/databases/"msgstore.db"

Table 6. Chat application databases' characteristics (sort alphabetically).

From the above nine applications, three of them were chosen based on the differences of their databases' settings such as the type of journaling they use (rollback journal/wal). Those apps are Viber, WhatsApp and Telegram, which have also been correlated in the past as all together by Botha et al. (2019) and Sutikno et al. (2016), or with other instant messaging applications (Onovakpuri P., 2018). Even though the encryption of the WeChat app was bypassed (Rathi et al., 2018) its examination would not add any further value, since by itself alone does not affect under which conditions the database is modified.

The above methodology is depicted in **Figure 29**.

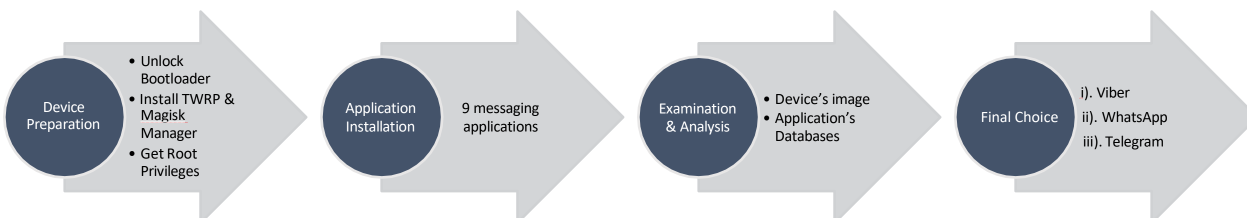


Figure 29. Workflow of the applications' choice.

3.4. Followed Methodology

Before the main scenario's experiments there was a reconnaissance phase involving:

- The understanding of the folder structure, file size and parameters of the databases inside the application after a user has been added. This was achieved by initializing this new user, removing the application, and repeating the process several times.
- The analysis of the interested databases regarding to their tables and columns.
- The understanding of the behavior of the database after examining a single record (text message) in several states (not sent, sent, not delivered, delivered, received etc.) and pinpointing the modified columns.
- The identification of the affected files and databases in the operating system besides the ones residing in the applications' folders when a media or a text message is exchanged.

The main phase of the scenario's experiments involved the examination of the two devices regarding the three key factors, which were previously mentioned:

- The time that has elapsed until and after the device's seizure.
- The status of the device before and after the seizure.
- The application's database modification by sending, receiving, reading or deleting additional messages during the above-mentioned time.

There is a correlation among these three key factors. For example, the longer the device is connected to a network the more likely an application's database modification might occur. The media files (pictures) and the messages which were sent/received and must be retrieved are mentioned in **Appendix A2** and are exactly the same for all the experiments. For the creation of a realistic populated database, the exchange of those messages occurred among the exchange of other, unrelated, text messages between four (4) active users as it depicted in **Appendix A1**. On some occasions (especially in Telegram) the number of the messages of Appendix A1 were not enough to draw any definite conclusions. In this case there was an exchange among the four (4) users of more messages in the form "This is message X", where X represents a sequential numbering (i.e., "This is message 200"). The experimentation phases had to be adjusted according to the unique features and characteristics of each tested application. For example, some of the messages of Appendix A2 in the Viber application included "autodelete" feature (timebomb), whereas in WhatsApp application this feature was ignored since autodeletion feature was activated after 24-hours. Apart of that, forensic images of the devices were acquired in several time frames and in devices' different statuses and conditions to determine if any modification of the databases can occur based solely on these two factors. The main time frames and under examination conditions are represented in **Table 7**.

Status (Always after deleting the messages of Appendix A2)	Time Frame	Action
Standby	As soon as possible	None
Standby	After 2/6/24 hours	None
Standby	As soon as possible	Removing Sim
Airplane mode	After 2/6/24 hours	None
After reboot	24 hours	Removed Sim
After shutdown	48 hours	None
Faraday box	After 2/6/24 hours	None

Table 7. Chat application databases' main characteristics (sort alphabetically).

After the scenario's main phase was completed, more experiments were conducted, which were unique for each tested application regarding some of their distinctive features. For example, in the Viber application the behavior of the journal file was examined regarding its size and correlation with the number of active users. In WhatsApp application was examined the behavior of the database after a local backup has been created and the ability to decrypt it. The above-mentioned methodology was organized into a workflow consisted of distinct phases (**Figure 30**). The experiments that were conducted were identical for both the devices but differed based on the examined application. Their total number for both the devices is presented on **Table 8**.

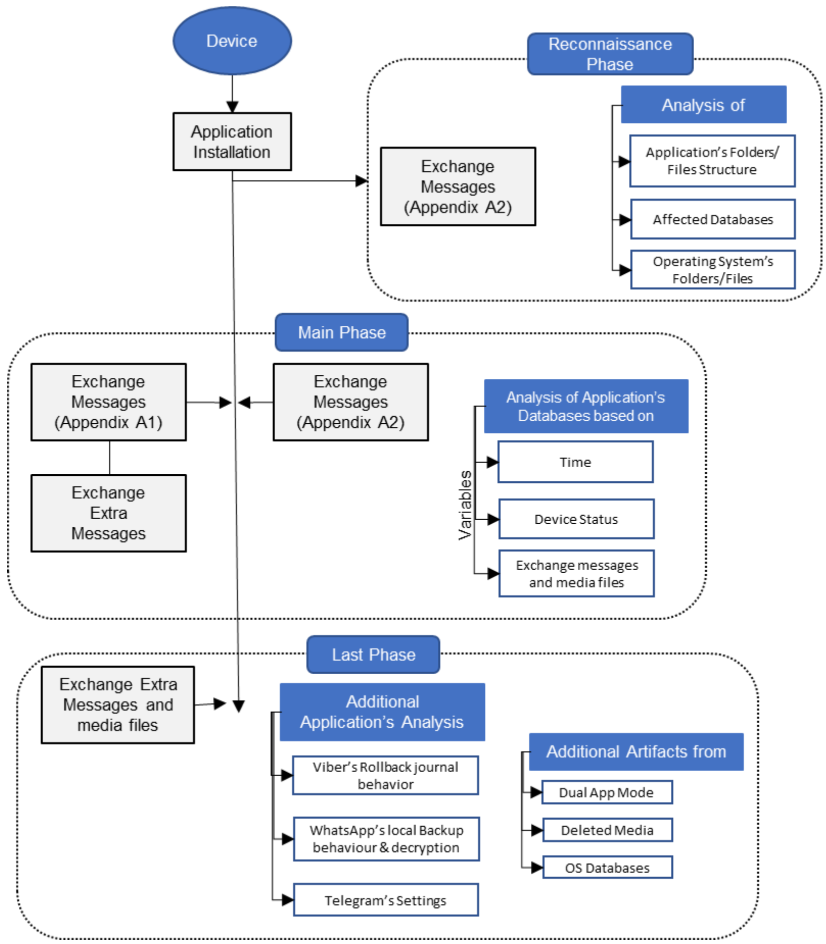


Figure 30. Workflow of methodology.

Application	Physical Images	Acquiring Database's folder			Exchanged Messages
		Reconnaissance Phase	Main Phase	Last Phase	
Viber	48	10	164	30	336
WhatsApp	12	6	168	-	660
Telegram	12	6	180	-	2.200

Table 8. Total conducted experiments' number.

3.5. General information about Viber, WhatsApp & Telegram

3.5.1. Viber

3.5.1.1 General Information

It was founded by Israelian Talmon Marco back in 2010 and joined Google Play Store two years after, in 2012. In 2014 Viber was bought by the Japanese company Rakuten and in 2016 the end-to-end encryption as a default setting was implemented. Viber has implemented the feature of automatically disappearing messages after a brief period of time that they have been read by the receiver. This period can vary from 10 seconds to 24 hours. Viber application can be installed in an Android or an iOS device and only after this initial setup the user can be install a version of it on a personal computer running Windows, Mac, or Linux operating system. According to Viber statistics (99Content, 2021) nowadays Viber market has been expanded to over 190 countries and in March 2020 there were almost 1.17 billion registered users worldwide. Among the countries in which Viber has significant popularity are Ukraine, Greece, Belarus, Russia, and Philippines.

3.5.1.2 Viber' s Communication Encryption

The Viber app provides an end-to-end encryption, including calls, one-on-one or group messages and media sharing. Encryption keys are stored only on the communicating "clients", and no one can access them, including Viber servers.

For exchanging messages for example, as official Viber's site refers (Rakuten Viber, 2021): Every time a device A sends a message to a device B, an ephemeral one-time 128-bit symmetric key is being generated and used for the encryption of the message body, using the Salsa20 encryption algorithm. «Then the ephemeral message key is encrypted using each recipient's session key. The sender device sends a unified message to the server, containing one encrypted cyphertext and a set of encrypted ephemeral keys. A server-side fan-out slices this message and delivers the relevant parts to each target device. The two devices take turns at advancing the session keys in a process called ratcheting. Each time the direction of the conversation changes, the device whose turn it is randomly generates a new Ratchet key pair, and once again performs the following sequence: TempKey = HMAC_SHA256(RootKey, DH (RatchetA, RatchetB)) New RootKey = HMAC_SHA256(TempKey, "root") SessionKey = HMAC_SHA256(TempKey, "mesg") With Ratchetthis_device being the private part of the newly derived key-pair. Alongside each message, the public part of the Ratchetthis_device is also sent. The recipient runs DH with its last private ratchet together with the sender's public ratchet» (Rakuten Viber, 2021). By implementing this double-ratchet method there is a forward and backwards secrecy of and even if there is a compromise of keys, previous or/and future messages cannot be decrypted. Moreover, peer's authentication is being maintained by the algorithm, because the root keys' DH (Diffie–Hellman) chain began with both devices' ID Keys. So, the entire chain is considered as trusted, if the peer's ID key confirmed as trusted at any point. Calls are being encrypted through an ephemeral 256-bit Curve25519 key-pair and the conversion of the RTP stream to SRTP via Salsa20 algorithm. Media sharing encryption uses an ephemeral, symmetric Salsa20 key, an HMAC (Hash-based message authentication code) signature and an MD5 signature for the file encryption in the Viber's storage server.

3.5.2. WhatsApp

3.5.2.1 General Information

It was founded, by Brian Acton and Jan Koum, former employees of Yahoo!, in 2009. In 2013 Google made an offer of ten (10$) billion US dollars, but the offer was rejected. In 2014 it was acquired by Facebook, currently known as Meta Platforms. The acquisition has risen some privacy concerns among the users causing them to migrate at different messaging platforms. WhatsApp has a different approach to implementing the feature of automatically disappearing messages where there is a longer lasting period of time for these messages. This period of time can be either 24 hours,7 days or 90 days. WhatsApp is a cross platform app and can be installed and used on Android or iOS mobile devices and on Windows PC, Mac or Linux desktop or laptop. According to WhatsApp statistics (99Content, 2021) in India the application has over half a billion users. Overall, there more than two billion users accessing the application every month. In several countries including China, the United Arab Emirates, Iran, Syria, North Korea and Cuba the usage of the application is blocked or restricted. The application has often been accused, especially in India, for the spread of misinformation, propaganda and fake news. For this reason, WhatsApp in 2020 forced a limit of chat forwarding (Singh S., 2020).

3.5.2.2 WhatsApp's Communication Encryption

WhatsApp features an end-to-end encryption to all types of communications. The term "end-to-end encryption" is defined by app's programmers (WhatsApp, 2021) «as communications that remain encrypted from a device controlled by the sender to one controlled by the recipient, where no third parties (not directly participants), not even WhatsApp or parent company Facebook, can access the content in between». As official WhatsApp's site refers for exchanging and transmitting messages, media files, calls etc., when a session is established, the content is encrypted because "clients" exchange messages with a Message Key using AES256 in CBC mode for encryption and HMAC-SHA256 for authentication. The Message Key for each message is an ephemeral and the one-time key, is derived from a sender's Chain Key that "ratchets" forward with every sent message. In addition, a new Chain Key is created by a new ECDH agreement which performed with each message roundtrip. The combination of both an immediate "hash ratchet" and a round trip "DH ratchet" provides more secrecy. Another step forward to security, is that all communications between application's clients and servers use Noise Pipes with Curve25519, AES-GCM, and SHA256 from the Noise Protocol Framework for long running interactive connections. Thus, there are a superfast connection setup and resume, a metadata encryption, and no private authentication keys/credentials storing on servers (only public authentication keys).

3.5.3. Telegram

3.5.3.1 General Information

It was founded by the Russian brothers Pavel and Nikolai Durov, who had already lunched an online Russian social media and networking service called "VK" (V Kontakte). It is a cloud-based messaging app which was launched in 2013. The main characteristics of the application are the open-source code and the ability to create secret, end-to-end encrypted and self-destructing messages and chats (Telegram, 2021). Today, Telegram is available in 155 countries with more than 550 million active users per month. The country with the most installations (over 210 million) is India, followed by Russia and Indonesia (Dean B., 2021). Telegram can be installed on every platform such as Android, iOS, Windows PC, Mac or Linux.

3.5.3.2 Telegram's Communication Encryption

Unlike Viber and WhatsApp, Telegram does not enable, by default, end-to-end encryption. Once a chat message arrives at the Telegram servers, it is encrypted using MTProto (v2), while at rest on the servers. However, Telegram could read chat data since it handles encryption and decryption of the messages at its (Telegram) servers. E2E encryption is supported only if the secret chat feature is chosen. MTProto is a custom mobile protocol designed by Telegram and it has been criticized many times by cryptography experts (Saribekyan & Margvelashvili, 2017, Lee et al., 2017, Jakobsen & Orlandi, 2016). MTProto (v2) uses Diffie-Hellman (DH) key exchange, Secure Hash Algorithm 256 (SHA-256), Key Derivation Function (KDF), and AES-256 in IGE mode as cryptographic primitive (End-to-End Encryption, Secret Chats) (telegram.org). Even if it is differentiated from the first version with features like extra padding bytes in the computation of msg_key, researchers from the University of London and ETH Zurich have documented significant problems in terms of Telegram's current encryption system security (Albrecht et al., 2021).

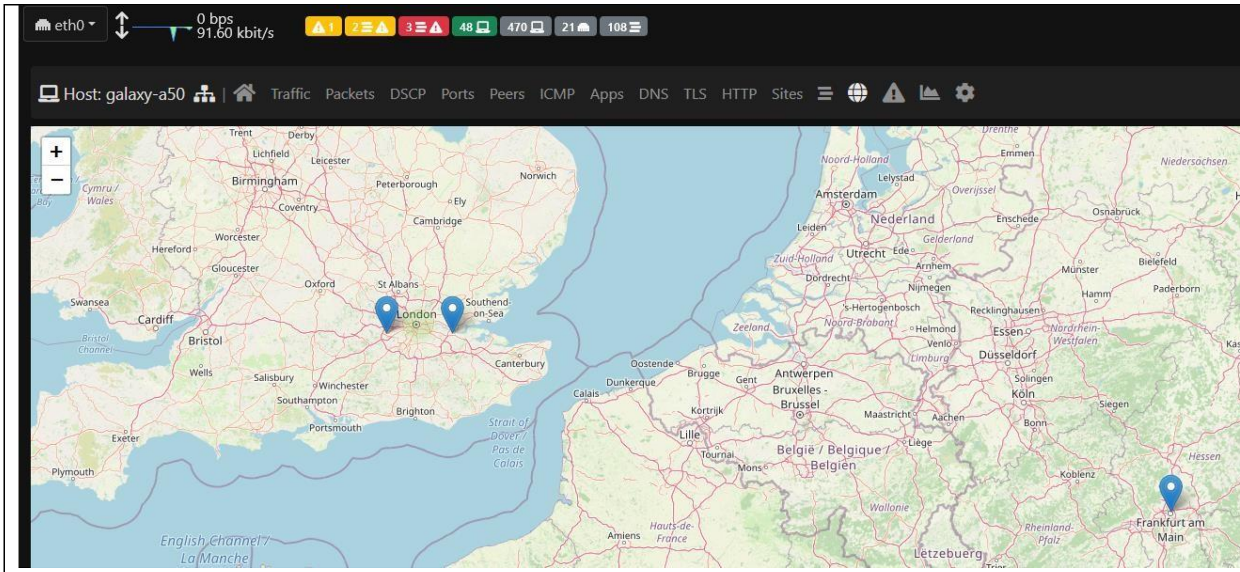
3.5.3.3 Telegram's Privacy Concerns

The by default lack of E2E encryption has risen several privacy concerns. According to app's privacy policy it is possible to «…collect metadata such as your IP address, devices, and Telegram apps you have used, history of username changes, etc.». This in conjunction with the fact that the messages are being encrypted on Telegram servers raise some red flags regarding users' privacy.

3.6. Applications' Cloud Servers

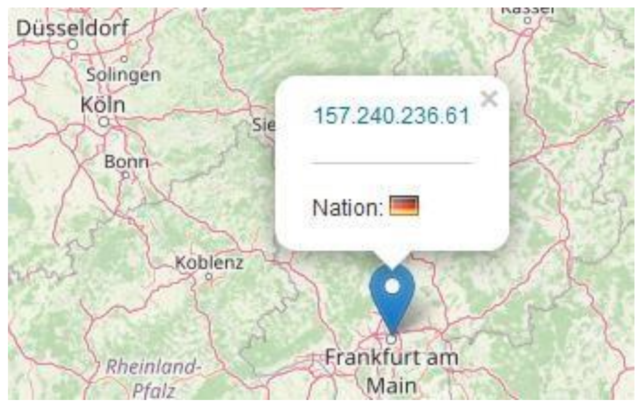
Obtaining data such as account information or exchanged messages, from messaging applications' private company servers is a difficult and complex process. The two main reasons are:

1. The locations of both servers (**Table 8**) and companies where different legislation is implemented. To put this in perspective:
   - Viber is owned by a Japanese company, based in Luxembourg. The messages originating from Greece were relayed/routed from servers in U.S.A.
   - WhatsApp is owned by an American company. The messages originating from Greece were relayed/routed from servers in Germany.
   - Telegram is registered as an American LLC and UK LLP, with headquarters in Dubai. The messages originating from Greece were relayed/routed from servers in England.

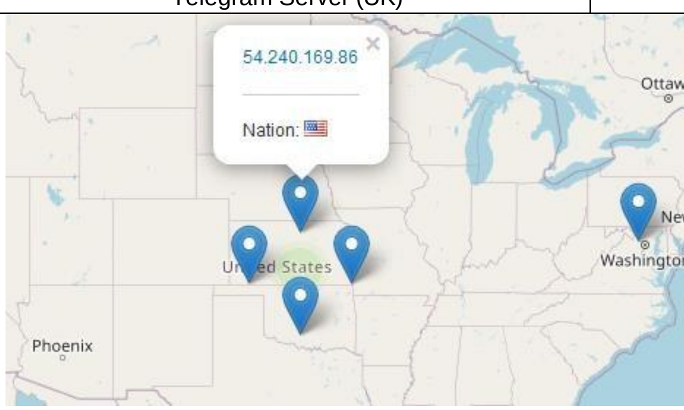


Telegram (UK) and WhatsApp (Germany) servers (AWS)

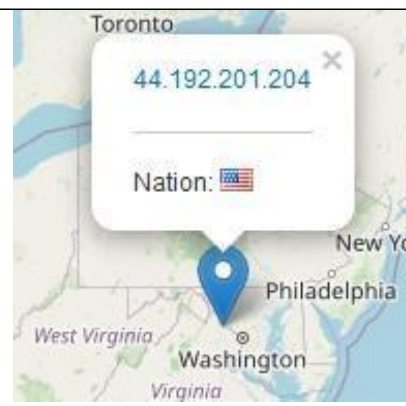
Telegram Server (UK)



WhatsApp Server (Germany)



Viber media server (USA)



Viber text messages server (USA)

Table 9. WhatsApp, Telegram and Viber applications' cloud servers where the devices connected (tool used: ntopng with geoip plugin, device examined: Samsung A50).

2. Even if this Kafkaesque legal system is bypassed and a court order is executed, as an FBI training document indicates, there is little that could be done to gain access to the content of encrypted messages from secure messaging services (Cimpanu C., 2021). In **Table 9** is presented the contents of the above-mentioned document for the messaging applications that

are examined in this Thesis concerning the data that can be retrieved from the company's servers.

App	Legal process & additional details
Viber	* **No message content.** * Provides account (i.e., phone number) registration data and IP address at time of creation. * Message history: time, date, source number, and destination number.
WhatsApp	* **Message content limited.** * **Subpoena:** can render basic subscriber records. * **Court order:** Subpoena return as well as information like blocked users. * **Search warrant:** Provides address book contacts and WhatsApp users who have the target in their address book contacts. * **Pen register:** Sent every 15 minutes, provides source and destination for each message. * If target is using an iPhone and iCloud backups enabled, iCloud returns may contain WhatsApp data, to include message content.
Telegram	* **No message content.** * No contact information provided for law enforcement to pursue a court order. As per Telegram's privacy statement, for confirmed terrorist investigations, Telegram may disclose IP and phone number to relevant authorities.

Table 10. FBI' ability legally content access of Viber, WhatsApp and Telegram. Content of FBI's training document released on Jun. 2021 (Cimpanu C., 2021).

It follows from all the foregoing that, if the data (messages) are not preserved and forensically retrieved from the devices it will be extremely difficult to do so from anywhere else.

Chapter 4.

Viber application Scenario

In this chapter the Viber application's examination and the related results are presented regarding text and media exchanged messages and the volatility of the database in which they are stored. At first, there is a reference on the application database's location, it's features and the differences between an Android 9 and an Android 11 examined device. After that there is an analysis of the tables and columns of the database and the queries that were built in order to view the desired ones. In addition, the volatility of deleted text and media messages is examined through multiple experiments. Furthermore, some unique characteristics of the application are examined, and final conclusions are drawn.

4.1. Viber' s Examination

The used .apk for both the devices was the Viber v.16.4.0.8 which was released in October 2021. The application' s file and folder structure is well known and documented over the years for both educational/research (Hermawan et al., 2021, Anglano et al., 2016) and forensic purposes (Belkasoft, 2021). In this Thesis the focus is on the messaging database and the modifications which occur after sending, receiving, reading and deleting messages or/and elapsed time and the device's status. The goal is to examine under which conditions deleted records can be recovered. Since this is not a black box experiment and as matter of fact, there is a total control of the device it is much safer to draw some conclusions. The above-mentioned "messaging database" is named "viber_mesages.db" and is located, among others, in the path: «data/data/com.viber.voip/databases». The access to this path is normally prohibited, so the examiner must either have "root" access to the device or acquire a physical image of it.

4.1.1. Android 9 vs Android 11 Comparison

The structure of the database's folder («data/data/com.viber.voip/databases») in the 16.4.0.8 version of the application in the two devices had some minor differences:

- Firstly, the google API ("com.google.android.datatransport.events.db") and "cdr.db" (Call Details Records - Metadata) databases in the Android 9 device used the newer WAL journaling mode. On the contrary the newer Android 11 device used the legacy journal mode. Those two databases do not have any participation in the experiment or differentiation on the results and therefore were not taken into consideration.

- Secondly, the initial size of the "viber_messages" databases and the overall size of their corresponding journal files differentiated in the two devices. The initial database (right after the first login of a user), in five different occasions on each device, had a small difference in size. This was depended on the volume of the initial pushed data by the company's server to the "chat_extension", "chat_suggestions" etc. tables.

4.1.2. General information about the under-examination database

Generally, the initial size of the "viber_mesages" database, before any transactions, was approximately 380 kb and the journal's size was 80 kb. After having created several transactions (sending/receiving) among four (4) users, it was observed that the maximum size of the journal file, was related to the number of the active users and the type of transaction that had occurred. The size of the main database is constantly altered as a result of the "auto_vacuum" PRAGMA statement (**Figure 31**).

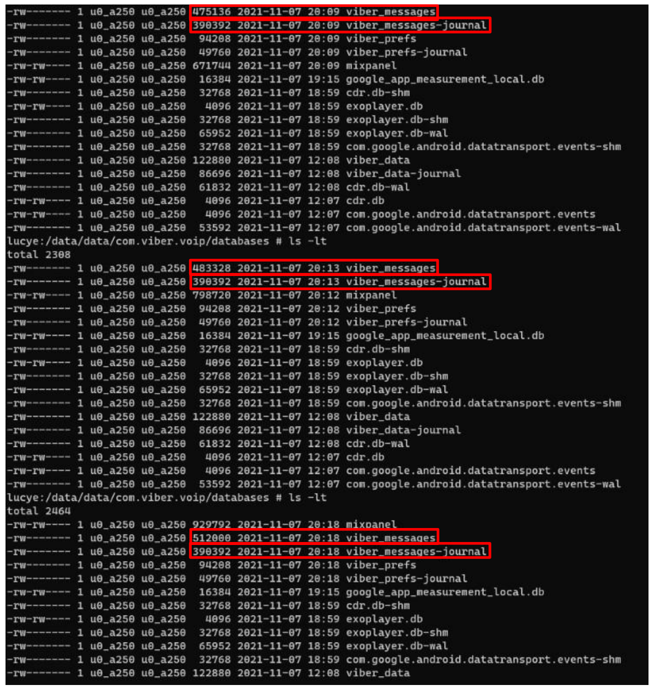


Figure 31. Comparison of the "viber_messages-journal" file size while increasing the size of main database with transaction between two (2) active users. The journal file is always 381 kb (Examined Device: LG Device).

The way the size of journal file was increased or decreased followed a certain pattern. For four (4) active users, after the transmission and receival of several text, photos and video messages, the size of the journal file was always 409 kb for the Samsung device and 393 kb for the LG Device. After a total deletion of the exchanged messages, the size of the file grown (721 kb for Samsung and 777 kb for the LG) but remained stable through the rest of the experiments even after sending and receiving more content on the two devices (**Figures 32** and **33**).

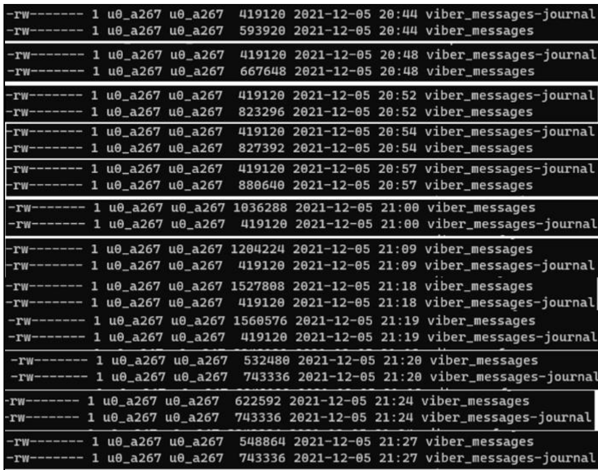


Figure 32. Comparison of "viber_messages.db" journal file's size while increasing the size of main database with transaction among four (4) active users on the Samsung A50 device.

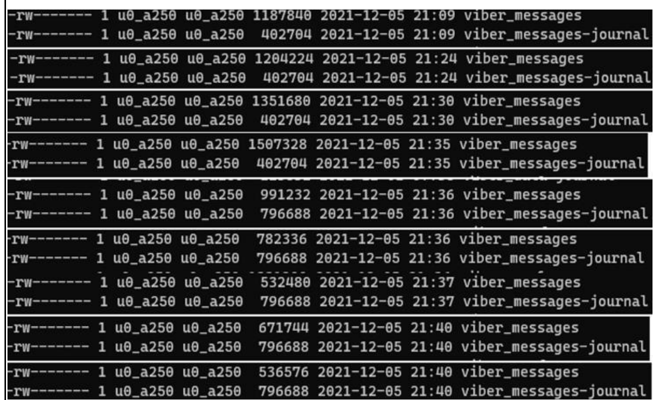


Figure 33. Comparison of "viber_messages.db"-journal file's size while increasing the size of main database with transaction among four (4) active users on the LG G6 device.

There was a clear correlation between the size of the journal file and the number of the active users. A conclusion that could be drawn from this is that the size of the journal file of "viber_messages" database, does not necessarily correlate with the volume of the recovered deleted content from one user/chat thread. The active pragma statements for the databases in the two devices can be examined from the file headers. The examination and comparison which is depicted in **Figures 34** and **35** involves copies of the examined databases by using the non-forensic tools: "SQLite Expert" and "DB Browser for SQLite". When a non-forensic tool is used there is a probability that the data of main database could be modified, especially if the database coexists in the same path as its journal/wal file.

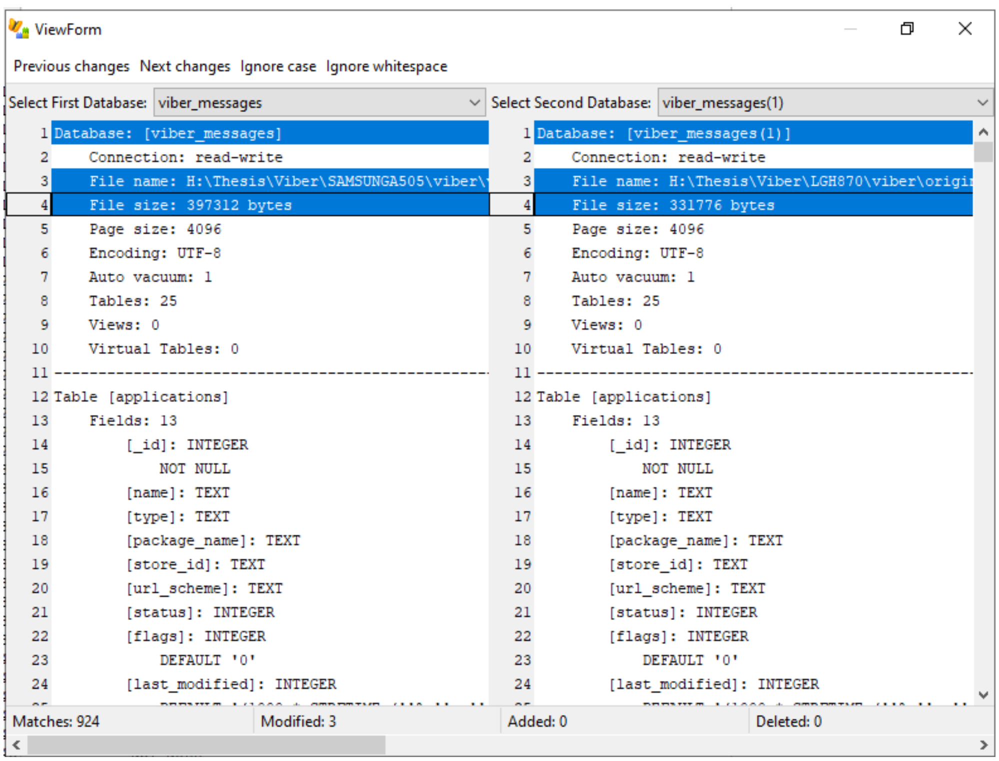


Figure 34. Comparison of "viber_messages" database in Samsung A50 (left) and in LG G6 (right) (tool used: SQLite Expert).

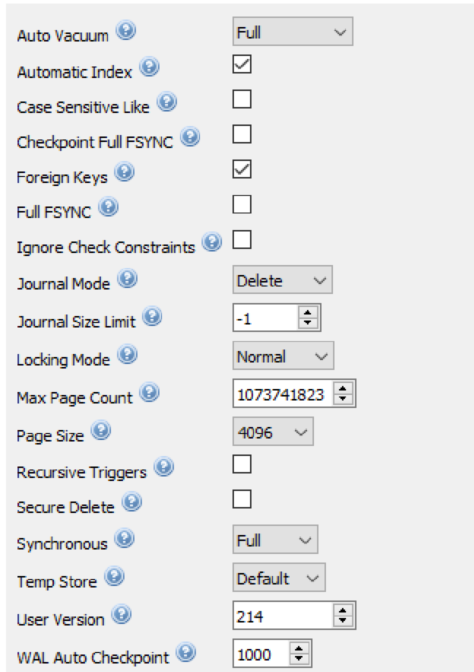


Figure 35. SQLite Pragma Statements of "viber_messages.db" in edit mode (tool used: DB Browser for SQLite).

The analysis of the active pragma statements from the database-file headers yield that both databases:

- ✓ had the Full auto_vacuum enabled and did not create freelist pages,
- ✓ used rollback journal as a journaling mode with unlimited size (-1), in theory.

The meaning and forensic value of these pragma statements (enabled options) have been analyzed in paragraph §**1.3.1.2**. All these settings render the data, especially the deleted ones, extremely volatile. The settings of the "viber_messages" database have been changed over the years. In older versions there were some key differences in the database which allowed an examiner to recover records with a greater success rate. At least until the 2017 versions of the application, freelist pages were created and kept in the database (**Table 10**).

Name	Value (Version 49- September 2017)	Value (Version 72- June 2021)	Value (Version 72- October 2021)
application_id	0	0	0
auto_vacuum	none	full	full
cache_size	-2000	-2000	-2000
collation_list	[BINARY], [NOCASE], [I	[BINARY], [NOCASE], [RT	[BINARY], [NOCASE], [RTR
encoding	UTF-8	UTF-8	UTF-8
foreign_keys	on	on	on
freelist_count	47	0	0
journal_mode	delete	delete	delete
journal_size_limit	-1	-1	-1
max_page_count	1073741823	1073741823	1073741823
mmap_size	0	0	0
page_count	254	95	125
page_size	4096	4096	4096
schema_version	49	72	72
user_version	99	207	214
writable_schema	off	off	off

Table 11. Comparing settings of "viber_messages.db" on different versions of the application (tool used: SQL Expert).

4.1.3. Analysis of the "viber_messages.db"

The "viber_messages" database was composed by twenty-four (24) tables (**Figure 36**). The two (2) most forensically interesting tables, for the examined scenario were the "messages" (**Figure 37**) and "participants_info".

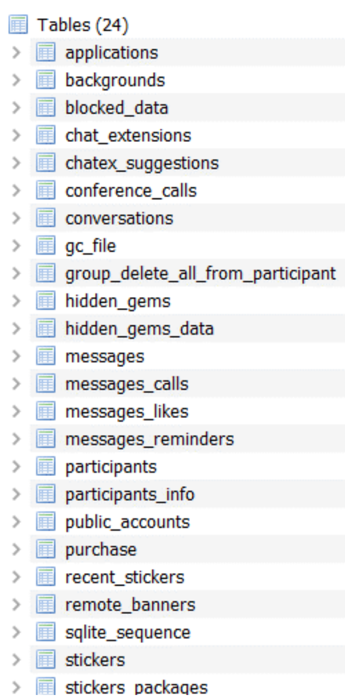


Figure 36. Depiction of the tables tree of "viber_messages.db".

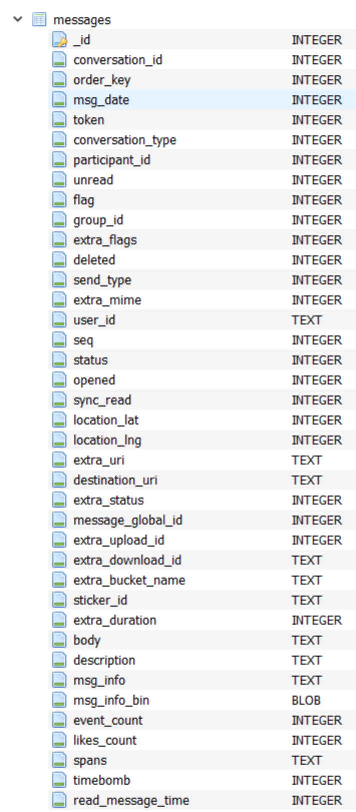


Figure 37. Some of the columns that are consisted in "messages" table.

The “**messages**” table contained the most interesting columns. Some of the column names are self-explanatory, such as the “**body**” which contains the body of the message. On the contrary others need to be explained regarding their type of content and form:

- The “**msg_date**” column contained the timestamp when a message was created, and it was encoded in Unix Millisecond (Java Time).
- The “**send_type**” column’s value indicated the direction of the message. When the value was “1” meant that it was an outgoing message. Any other value indicated that it was an incoming one.
- The “**unread**” column’s value was “1” or “0”. The value “1” indicated that the message had not yet been read and value “0” that the message had been viewed.
- The “**status**” column contained information about whether the message was not sent (“0”) (e.g., problem in the device’s network connection), sent but not delivered (“1”), or delivered (“2”).
- The “**read_message_time**” column contained information about the timestamp of the read time of a message by its receiver.
- The “**msg_info**” column contained several types of information e.g., the metadata of a transmitted media file or if a message has been edited by the user.

The “**participants_info**” table contained information about the participants. In this table, the phone number and the contact’s name could be found. The data in the columns of this table were combined with the data of the columns of the “**messages**” table in order to create new tables which contained easily understandable results. This was achieved through simple SQL queries (**Figures 38** and **39**) and by joining the “**member_id**” and “**user_id**” columns of these two tables.

```sql
SELECT
messages.body As 'Message',
datetime(messages.msg_date/1000,'unixepoch','localtime') AS 'Message Date',
messages.user_id,
participants_info.display_name,
participants_info.number,
CASE messages.send_type
    WHEN '1' THEN 'Outgoing'
    ELSE 'Incoming'
END AS 'Direction',
CASE messages.unread
    WHEN '0' THEN 'True'
    WHEN '1' THEN 'False'
    ELSE messages.opened
END AS 'Opened',
CASE messages.status
     WHEN '0' THEN 'Message was not sent'
     WHEN '1' THEN 'Message was not received'
     when '2' THEN 'Message received'
     ELSE 'Unknown Status'
END AS status,
CASE messages.read_message_time
    WHEN '0' THEN 'Message was not opened'
    ELSE  'Message was Opened'
END AS 'Message Read Status'
FROM messages
JOIN participants_info ON messages.user_id = participants_info.member_id
ORDER BY messages.msg_date ASC
```

	Message	Message Date	user_id	display_name	number	Direction	Opened	status	Message Read Status
1	This is a test	2021-11-10 10:43:48	DrmrchQhcUo=	Bad Actor 1	+3069407 [illegible]	Outgoing	True	Message received	Message was Opened
2	This is also a test	2021-11-10 10:48:32	DrmrchQhcUo=	Bad Actor 1	+3069407 [illegible]	Incoming	False	Message received	Message was not opened
3	This is a test 2	2021-11-10 10:49:21	Qhnb3h7J3pE=	Μιχάλης	+3069472 [illegible]	Outgoing	True	Message received	Message was not opened
4	This is third test	2021-11-10 10:49:46	Qhnb3h7J3pE=	Μιχάλης	+306947 [illegible]	Outgoing	True	Message received	Message was not opened
5	This is test number 4	2021-11-10 10:50:56	CfunDWmeF1o=	Edgar Allan Poe	+306986 [illegible]	Outgoing	True	Message was not received	Message was not opened
6	This is test 5	2021-11-10 10:51:17	CfunDWmeF1o=	Edgar Allan Poe	+306986 [illegible]	Outgoing	True	Message was not received	Message was not opened
7	This is test 6	2021-11-10 10:51:41	Qhnb3h7J3pE=	Μιχάλης	+306947 [illegible]	Outgoing	True	Message was not sent	Message was not opened
8	Thisbis test 7	2021-11-10 10:51:51	Qhnb3h7J3pE=	Μιχάλης	+306947 [illegible]	Outgoing	True	Message was not sent	Message was not opened

Figure 38. SQL query combining some of the columns and tables.

```sql
CASE
    WHEN messages.msg_info LIKE '%edit%' THEN 'Message has been edited'
END AS 'Message info'
FROM messages
JOIN participants_info ON messages.user_id = participants_info.member_id
ORDER BY messages.msg_date ASC
```

	Message	Message Date	user_id	display_name	number	DIRECTION	Opened	status	Message Opened	Message info
1	This is a test and test 12	2021-11-11 23:48:25	DrmrchQhcUo=	Bad Actor 1	+306940713831	Incoming	True	RECEIVED	Message Opened	Message has been edited
2	This a test 2 and 11	2021-11-11 23:48:58	DrmrchQhcUo=	Bad Actor 1	+306940713831	Outgoing	True	RECEIVED	Message Opened	Message has been edited
3	This is a test 4	2021-11-11 23:54:10	DrmrchQhcUo=	Bad Actor 1	+306940713831	Incoming	True	RECEIVED	Message Opened	NULL
4	This is test 5	2021-11-11 23:56:34	DrmrchQhcUo=	Bad Actor 1	+306940713831	Outgoing	True	RECEIVED	Message Opened	NULL
5	This is test 6	2021-11-11 23:57:31	DrmrchQhcUo=	Bad Actor 1	+306940713831	Incoming	True	RECEIVED	Message Opened	NULL
6	This is test 7	2021-11-11 23:57:39	DrmrchQhcUo=	Bad Actor 1	+306940713831	Incoming	True	RECEIVED	Message Opened	NULL
7	This is test 8	2021-11-11 23:57:47	DrmrchQhcUo=	Bad Actor 1	+306940713831	Incoming	True	RECEIVED	Message Opened	NULL
8	This is test 9	2021-11-12 00:01:38	DrmrchQhcUo=	Bad Actor 1	+306940713831	Outgoing	True	RECEIVED	Message Opened	NULL
9	This is test 10 and 11	2021-11-12 00:01:52	DrmrchQhcUo=	Bad Actor 1	+306940713831	Outgoing	True	RECEIVED	Message Opened	Message has been edited
10	This a test 2 and 11	2021-11-13 10:00:42	DrmrchQhcUo=	Bad Actor 1	+306940713831	Outgoing	True	RECEIVED	Message not opened	Message has been edited

Figure 39. Adding another case statement in the previous SQL query, it can be viewed if the message has been edited or not.

At this point it must be mentioned that the columns' names inside the tables and the meaning of the corresponding flags/data are modified among versions. For example, in the Viber v.16.4.7.2, the "**unread**" column was absent and was replaced with the "**open**" column. Those two columns differ not only in name but also in meaning. In the "**unread**" column the value of "1" meant that the message was not read by the user, in contrast with the column "**open**" where the flag "1" meant the exact opposite (that the message was read by the user). This means that the SQL queries that were executed in this Thesis might not be applicable in other versions of these applications.

4.1.4. Volatility of the "viber_messages.db"

The experiments confirmed that the crucial data in the "viber_messages" database and its corresponding journal file are not susceptible to any of device status changes or time, including:

- ✓ Leaving the device powered on with active network. Take note that in the experiments that were executed it was not taken under consideration the possibility of a remote wipe of the device. Forensically is highly recommended to disable all the network connections of the device.
- ✓ Leaving the device in a faraday box with network.
- ✓ Putting the device in airplane mode.
- ✓ Rebooting the device.
- ✓ Shutting down the device.
- ✓ Removing the SIM card (which was the only thing associated with account), while the device was live and after it was deactivated and then powered on without the SIM.

All the above experiments were repeated in several time frames and did not modify by any means the two files. During the experiments there was only one case, in which "killing" (closing) the application manually from the background and reopening it, modified the files. That exception had to do with "bad timing" because at that moment Viber's server happened to push some new data in the "**chatex_suggestions**" table in "viber_messages.db". As long as there were not any interactions with the application, the examined database was not modified even after several hours (**Figure 40**) or even days.

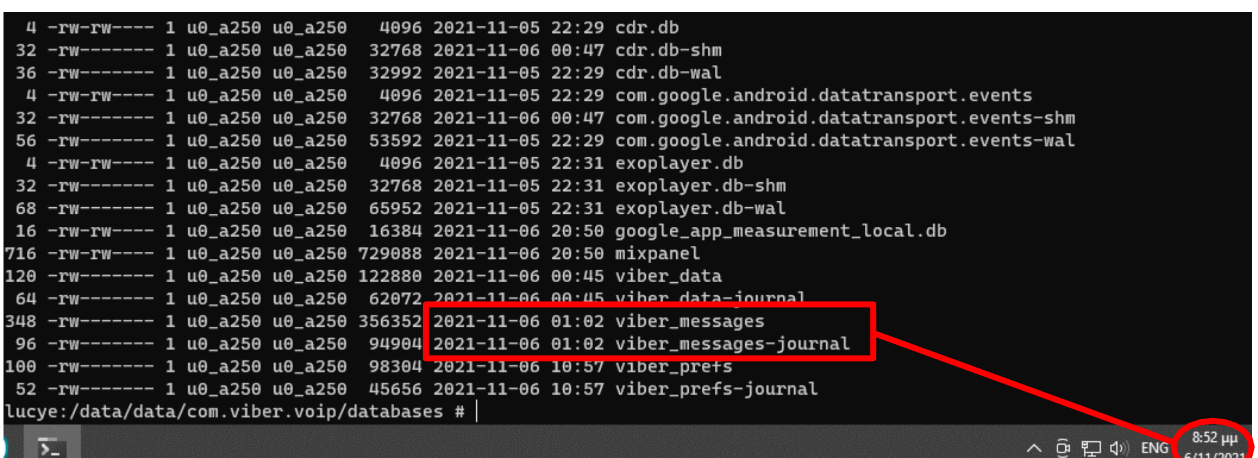
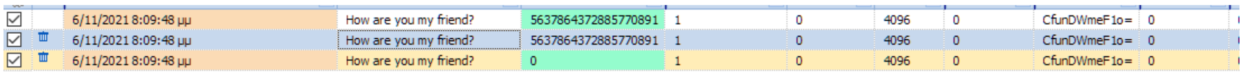


Figure 40. The device's "viber_messages.db" was not modified after 20 hours, even changing its state several times.

As mentioned before, the schema which Viber has implemented in the last versions, renders the data extremely volatile. By sending just one message the record was present three times in the main database and its journal file. Every time the record changes it's state (from sent to delivered, or from delivered to have been read by the receiver) its previous version is moved to the journal file. Only the most recent status of the record resides in the main database while the previous states of the same record reside in the journal file (**Figure 41**).



☑		6/11/2021 8:09:48 μμ	How are you my friend?	5637864372885770891	1	0	4096	0	CfunDWmeF1o=	0
☑	🗑	6/11/2021 8:09:48 μμ	How are you my friend?	5637864372885770891	1	0	4096	0	CfunDWmeF1o=	0
☑	🗑	6/11/2021 8:09:48 μμ	How are you my friend?	0	1	0	4096	0	CfunDWmeF1o=	0

Figure 41. How "viber_messages.db" and its journal file are altered after one message was sent (tool used: Oxygen Sqlite Viewer, Examined Device: Samsung A50).

This was verified through the FQLite tool. In **Figure 42** it can be observed that the same message (record) existed two times in the journal file and one time in the main database (latest state). The same rule applied when more messages were sent as it can be viewed in **Figures 43** to **45**.

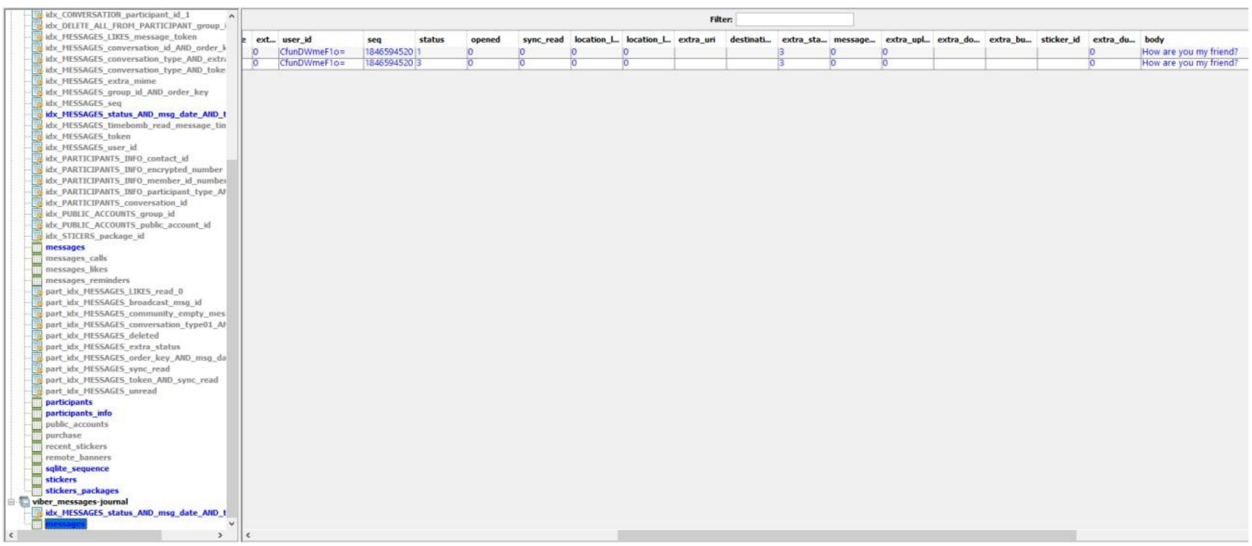
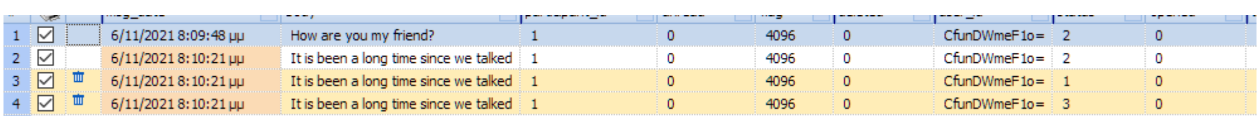
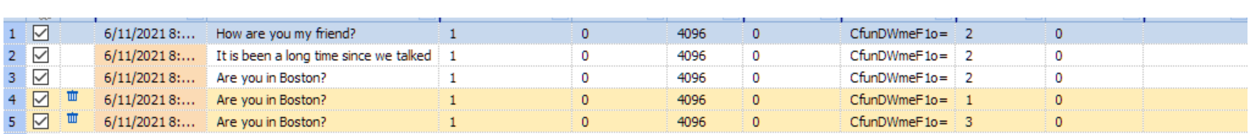


Figure 42. How "viber_messages.db" journal file was altered after one message was sent (tool used: FQLite, Examined Device: Samsung A50).



1	☑		6/11/2021 8:09:48 μμ	How are you my friend?	1	0	4096	0	CfunDWmeF1o=	2	0
2	☑		6/11/2021 8:10:21 μμ	It is been a long time since we talked	1	0	4096	0	CfunDWmeF1o=	2	0
3	☑	🗑	6/11/2021 8:10:21 μμ	It is been a long time since we talked	1	0	4096	0	CfunDWmeF1o=	1	0
4	☑	🗑	6/11/2021 8:10:21 μμ	It is been a long time since we talked	1	0	4096	0	CfunDWmeF1o=	3	0

Figure 43. How "viber_messages.db" journal file was altered after two messages were sent (tool used: Oxygen Sqlite Viewer, Examined Device: Samsung A50).



1	☑		6/11/2021 8:...	How are you my friend?	1	0	4096	0	CfunDWmeF1o=	2	0
2	☑		6/11/2021 8:...	It is been a long time since we talked	1	0	4096	0	CfunDWmeF1o=	2	0
3	☑		6/11/2021 8:...	Are you in Boston?	1	0	4096	0	CfunDWmeF1o=	2	0
4	☑	🗑	6/11/2021 8:...	Are you in Boston?	1	0	4096	0	CfunDWmeF1o=	1	0
5	☑	🗑	6/11/2021 8:...	Are you in Boston?	1	0	4096	0	CfunDWmeF1o=	3	0

Figure 44. How "viber_messages.db" journal file was altered after three messages were sent (tool used: Oxygen Sqlite Viewer, Examined Device: Samsung A50).

#			msg_date	participant_id	body	unread	flag	deleted	user_id	status	opened	read_message_time
1	☑		6/11/2021 8:09:48 μμ	1	How are you my friend?	0	4096	0	CfunDWmeF1o=	2	0	651788737
2	☑	🗑	6/11/2021 8:09:48 μμ	1	How are you my friend?	0	4096	0	CfunDWmeF1o=	2	0	0
3	☑		6/11/2021 8:10:21 μμ	1	It is been a long time since we talked	0	4096	0	CfunDWmeF1o=	2	0	651788737
4	☑	🗑	6/11/2021 8:10:21 μμ	1	It is been a long time since we talked	0	4096	0	CfunDWmeF1o=	2	0	0
5	☑		6/11/2021 8:10:38 μμ	1	Are you in Boston?	0	4096	0	CfunDWmeF1o=	2	0	651788737
6	☑	🗑	6/11/2021 8:10:38 μμ	1	Are you in Boston?	0	4096	0	CfunDWmeF1o=	2	0	0
7	☑	🗑	6/11/2021 8:10:38 μμ	1	Are you in Boston?	0	4096	0	CfunDWmeF1o=	1	0	0
8	☑	🗑	6/11/2021 8:10:38 μμ	1	Are you in Boston?	0	4096	0	CfunDWmeF1o=	3	0	0

Figure 45. How “viber_messages.db” journal file was altered after three messages were opened by the receiver. By changing the “read_message_time” flag, all records which were previously unread are moved to the journal (tool used: Oxygen Sqlite Viewer, Examined Device: Samsung A50).

So, the database and its journal file were modified when:

- a message was created. It did not matter whether was successfully sent or not.
- a message was received.
- a message was opened. This was regardless of if it was an outgoing message read by the receiver or an incoming read by the user.
- a message or a thread was deleted by the user or by timebomb.
- a bot message (ads) was received, opened, or deleted.
- a message notification was removed (**Figures 46** and **47**).
- if new data were pushed by Viber’ s server (e.g., download new emoticons suggestion).

backgrounds (0)
blocked_data (0)
chat_extensions (12)
chatex_suggestions (1671)
conference_calls (0)
conversations (16)
gc_file (2)
group_delete_all_from_participant (0)
hidden_gems (395)
hidden_gems_data (2)
messages (34)
messages_calls (0)
messages_likes (0)
messages_reminders (0)
participants (8)
participants_info (5)
public_accounts (1)
purchase (0)
recent_stickers (67)
remote_banners (0)
sqlite_sequence (20)
stickers (444)
stickers_packages (4)
Unallocated space

☐	Record type	_id	conversation_id	order_key	msg_date	participant_id	send_type	user_id
☐	Journal	16	4	56378672129207774	6/11/2021 8:21:06 PM	8	0	DrmrchQhcUo=
☐	Journal	17	4	56378672366563432	6/11/2021 8:21:12 PM	8	0	DrmrchQhcUo=
☐	Journal	18	4	56378681589329967	6/11/2021 8:24:51 PM	7	1	DrmrchQhcUo=
☐	Journal	19	4	56378682672928407	6/11/2021 8:25:17 PM	7	1	DrmrchQhcUo=
☐	Journal	20	4	56378686799457097	6/11/2021 8:26:56 PM	8	0	DrmrchQhcUo=
☐	Journal	21	4	56378686994743896	6/11/2021 8:27:00 PM	8	0	DrmrchQhcUo=
☐	Journal	22	4	56378687224843408	6/11/2021 8:27:06 PM	8	0	DrmrchQhcUo=
☐	Journal	23	4	56378687647377592	6/11/2021 8:27:16 PM	8	0	DrmrchQhcUo=
☐	Journal	24	4	56378688089625016	6/11/2021 8:27:26 PM	8	0	DrmrchQhcUo=
☐	Journal	13	4	56378671401663797	6/11/2021 8:20:49 PM	8	0	DrmrchQhcUo=
☐	Journal	14	4	56378671685072925	6/11/2021 8:20:55 PM	8	0	DrmrchQhcUo=
☐	Journal	15	4	56378671926245399	6/11/2021 8:21:01 PM	8	0	DrmrchQhcUo=
☐	Journal	36	1	56378721672450117	6/11/2021 8:40:47 PM	2	0	CfunDWmeF1o=
☐	Journal	37	1	56378722149258594	6/11/2021 8:40:59 PM	2	0	CfunDWmeF1o=
☐	Journal	38	1	56378722346516705	6/11/2021 8:41:03 PM	2	0	CfunDWmeF1o=
☐	Journal	39	1	56378722537818914	6/11/2021 8:41:08 PM	2	0	CfunDWmeF1o=
☐		16	4	56378672129207774	6/11/2021 8:21:06 PM	8	0	DrmrchQhcUo=
☐		17	4	56378672366563432	6/11/2021 8:21:12 PM	8	0	DrmrchQhcUo=

Figure 46. Before removing Viber Notification: eighteen (18) messages existed in “viber_messages.db” and 16 messages existed in journal (tool used: Belkasoft-X, Examined Device: Samsung A50).

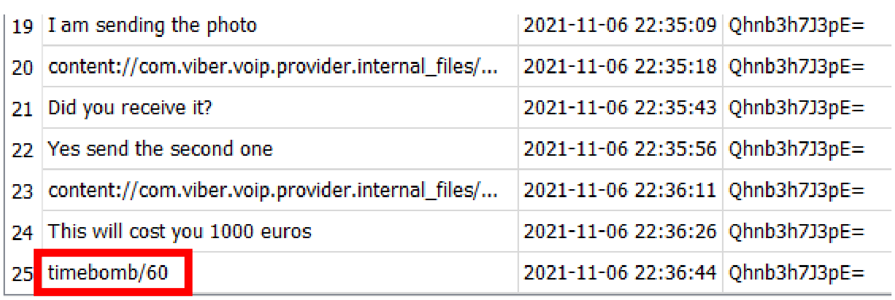
backgrounds (0)
blocked_data (0)
chat_extensions (12)
chatex_suggestions (1671)
conference_calls (0)
conversations (12)
gc_file (2)
group_delete_all_from_participant (0)
hidden_gems (395)
hidden_gems_data (2)
messages (51)
messages_calls (0)
messages_likes (0)
messages_reminders (0)
participants (8)
participants_info (5)
public_accounts (1)
purchase (0)
recent_stickers (67)
remote_banners (0)
sqlite_sequence (20)
stickers (444)
stickers_packages (4)
Unallocated space

Record type	_id	conversation_id	order_key	msg_date	participant_id	send_type	user_id
Journal	16	4	56378672129207774	6/11/2021 8:21:06 PM	8	0	DrmrchQhcUo=
Journal	17	4	56378672366563432	6/11/2021 8:21:12 PM	8	0	DrmrchQhcUo=
Journal	18	4	56378681589329967	6/11/2021 8:24:51 PM	7	1	DrmrchQhcUo=
Journal	19	4	56378682672928407	6/11/2021 8:25:17 PM	7	1	DrmrchQhcUo=
Journal	20	4	56378686799457097	6/11/2021 8:26:56 PM	8	0	DrmrchQhcUo=
Journal	21	4	56378686994743896	6/11/2021 8:27:00 PM	8	0	DrmrchQhcUo=
Journal	22	4	56378687224843408	6/11/2021 8:27:06 PM	8	0	DrmrchQhcUo=
Journal	23	4	56378687647377592	6/11/2021 8:27:16 PM	8	0	DrmrchQhcUo=
Journal	24	4	56378688089625016	6/11/2021 8:27:26 PM	8	0	DrmrchQhcUo=
Journal	13	4	56378671401663797	6/11/2021 8:20:49 PM	8	0	DrmrchQhcUo=
Journal	14	4	56378671685072925	6/11/2021 8:20:55 PM	8	0	DrmrchQhcUo=
Journal	15	4	56378671926245399	6/11/2021 8:21:01 PM	8	0	DrmrchQhcUo=
Journal	36	1	56378721672450117	6/11/2021 8:40:47 PM	2	0	CfunDWmeF1o=
Journal	37	1	56378722149258594	6/11/2021 8:40:59 PM	2	0	CfunDWmeF1o=
Journal	38	1	56378722346516705	6/11/2021 8:41:03 PM	2	0	CfunDWmeF1o=
Journal	39	1	56378722537818914	6/11/2021 8:41:08 PM	2	0	CfunDWmeF1o=
Journal	16	4	56378672129207774	6/11/2021 8:21:06 PM	8	0	DrmrchQhcUo=
Journal	17	4	56378672366563432	6/11/2021 8:21:12 PM	8	0	DrmrchQhcUo=
Journal	18	4	56378681589329967	6/11/2021 8:24:51 PM	7	1	DrmrchQhcUo=

Figure 47. After having Viber Notification removed: eighteen (18) messages existed in "viber_messages.db" and thirty-three (33) messages existed in it's journal file. Removing messages notification can potentially push something important out from the journal file by replacing it (tool used: Belkasoft-X, Examined Device: Samsung A50).

4.1.5. Retrieving the messages in question and the timebomb feature

During the experiments some of the exchanged messages of the scenario included the timebomb feature. Timebomb is a message autodeletion setting. The user predefines an autodeletion timer countdown which is activated after a message has been read by the receiver. After that the exchanged message is deleted from the chat thread. The Viber application treats these kinds of messages like normal ones, and the only element that differentiates them, is the flag "autodelete" which is displayed as "timebomb/X" (where X= 10,60,3600 seconds) in the body of the message (**Figure 48**).



19	I am sending the photo	2021-11-06 22:35:09	Qhnb3h7J3pE=
20	content://com.viber.voip.provider.internal_files/...	2021-11-06 22:35:18	Qhnb3h7J3pE=
21	Did you receive it?	2021-11-06 22:35:43	Qhnb3h7J3pE=
22	Yes send the second one	2021-11-06 22:35:56	Qhnb3h7J3pE=
23	content://com.viber.voip.provider.internal_files/...	2021-11-06 22:36:11	Qhnb3h7J3pE=
24	This will cost you 1000 euros	2021-11-06 22:36:26	Qhnb3h7J3pE=
25	timebomb/60	2021-11-06 22:36:44	Qhnb3h7J3pE=

Figure 48. Use of timebombs and how they are depicted in the "viber_messages.db" (tool used: DB Browser for SQLite, Examined Device: Samsung A50).

By examining the database and its journal file with the X-Ways Forensics, or any other HEX Editor, and searching it by the value inside the "**user_id**" column, which contains the user that sent the messages, it could be observed that the messages with enabled timebomb setting still existed and could be retrieved as long as no other commits were executed to the database (**Figure 49**). To make things clearer: the timebomb feature added an extra state to the record when it deleted it. Since the journal file is the invalid previous version of the database it could still be retrieved.

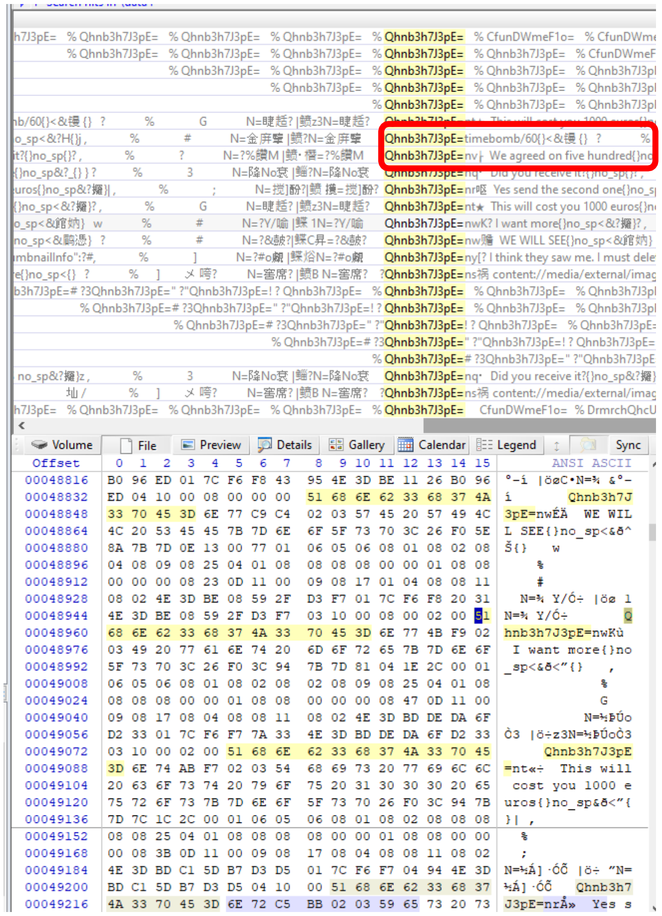


Figure 49. Searching timebombs messages content using "user_id" in "viber_messages.db" (tool used: X-Ways Forensics, Examined Device: Samsung A50).

The problem lies that with the speed that the applications are updating their settings and databases, it is very difficult for forensic tools, free or commercial, to keep up with the correct parsing of the application's databases and journal/wal files. The verification by using multiple tools could help, with the retrieval of most of the records, but at the end, it is necessary to dive straight into file itself in a hex level to ensure that nothing is missed. The experiments showed that the usage of the contents of the column "user_id" as a searchable string can preview some fast results. It is understandable that viewing the data of the file as hexadecimal is a hard task, but unfortunately necessary and vital in many cases (AskClees, 2020). Several forensic tools (free and commercial) and the way they parse the deleted timebomb records are depicted in (**Figures 50** to **52**).

On some occasions a forensic tool returned one message of the deleted ones after the timebomb feature has been activated (**Figure 50**), whereas examining the same database with different tools retrieved the most if not all of the erased, by timebomb, messages (**Figures 51** and **52**). This comparison is not made to point out the advantage of one software/tool over another but to point out that multiple ones have to be used in order to draw a final and detailed conclusion.

conversa...	order_key	msg_date	body	user_id	
2	null	Sat Nov 06 22:35:43 EET 2021	Did you receive it?	Qhnb3h7J...	r
2	null	Sat Nov 06 22:35:56 EET 2021	Yes send the second one	Qhnb3h7J...	r
2	null	Sat Nov 06 22:36:11 EET 2021	content://com.viber.voip.provider.internal_files/message/lo...	Qhnb3h7J...	r
2	null	Sat Nov 06 22:36:26 EET 2021	This will cost you 1000 euros	Qhnb3h7J...	r
2	null	Sat Nov 06 22:36:44 EET 2021	timebomb/60	Qhnb3h7J...	r
2	null	Sat Nov 06 22:37:41 EET 2021	I think they saw me. I must delete those	Qhnb3h7J...	r

Figure 50. How "viber_messages" journal file was modified after messages with timebomb were deleted (tool used: FQLite Examined Device: Samsung A50).

☐	Journal	2	6/11/2021 8:36:12 PM	Qhnb3h7J3pE=	content://com.viber.voip.provider.internal_files/message/lo
☐	Journal	2	6/11/2021 8:35:44 PM	Qhnb3h7J3pE=	Did you receive it?
☐	Journal	2	6/11/2021 8:37:41 PM	Qhnb3h7J3pE=	I think they saw me. I must delete those
☐	Journal	2	6/11/2021 8:36:26 PM	Qhnb3h7J3pE=	This will cost you 1000 euros
☐	Journal	2	6/11/2021 8:35:56 PM	Qhnb3h7J3pE=	Yes send the second one
☐	Journal	2	6/11/2021 8:36:45 PM	Qhnb3h7J3pE=	timebomb/60

Figure 51. How "viber_messages" journal file was modified after timebomb messages were deleted (tool used: Belkasoft-X**,** Examined Device: Samsung A50).

10	☑	🗑	6/11/2021 8:37:00 μμ	4	We agreed on five hundred	0	Qhnb3h7J3pE=
11	☑	🗑	6/11/2021 8:37:08 μμ	3	I want more	0	Qhnb3h7J3pE=
12	☑	🗑	6/11/2021 8:37:08 μμ	3	I want more	0	Qhnb3h7J3pE=
13	☑	🗑	6/11/2021 8:37:17 μμ	4	WE WILL SEE	0	Qhnb3h7J3pE=
14	☑	🗑	6/11/2021 8:37:41 μμ	3	I think they saw me. I must delete those	0	Qhnb3h7J3pE=

Figure 52. How "viber_messages" file was modified after timebomb messages were deleted (tool used: Oxygen Forensic SQLite, Examined Device: Samsung A50).

After the manual deletion of the conversation thread which also contained timebomb messages, the forensic tools, without tweaking them, were still able to recover several records, with different levels of success (**Figures 53** and **54**).

☐	Journal	6/11/2021 10:36:12 P	Qhnb3h7J3pE=	content://com.viber.voip.provider.internal_files/message/local/thumb/b68ab198524939961cbb801a0a	66 69 6C 65 49 6E 66
☐	Journal	6/11/2021 10:35:44 P	Qhnb3h7J3pE=	Did you receive it?	
☐	Journal	6/11/2021 10:36:26 P	Qhnb3h7J3pE=	This will cost you 1000 euros	
☐	Journal	6/11/2021 10:35:56 P	Qhnb3h7J3pE=	Yes send the second one	
☐	Journal	6/11/2021 10:36:45 P	Qhnb3h7J3pE=	timebomb/60	

Figure 53. How "viber_messages" journal file was modified five minutes after the conversation thread was deleted (tool used Belkasoft-X, Examined Device: Samsung A50).

17	6/11/2021 8:35:09 μμ	I am sending the photo	3
18	6/11/2021 8:35:18 μμ	content://com.viber.voip.provider.internal_files/message/local/thumb/155012...	3
19	6/11/2021 8:35:18 μμ	content://com.viber.voip.provider.internal_files/message/local/thumb/155012...	3
20	6/11/2021 8:35:43 μμ	Did you receive it?	3
21	6/11/2021 8:35:43 μμ	Did you receive it?	3
22	6/11/2021 8:35:56 μμ	Yes send the second one	4
23	6/11/2021 8:36:11 μμ	content://com.viber.voip.provider.internal_files/message/local/thumb/b68ab1...	3
24	6/11/2021 8:36:11 μμ	content://com.viber.voip.provider.internal_files/message/local/thumb/b68ab1...	3
25	6/11/2021 8:36:11 μμ	content://com.viber.voip.provider.internal_files/message/local/thumb/b68ab1...	3
26	6/11/2021 8:36:26 μμ	This will cost you 1000 euros	3
27	6/11/2021 8:36:26 μμ	This will cost you 1000 euros	3
28	6/11/2021 8:36:26 μμ	This will cost you 1000 euros	3
29	6/11/2021 8:36:44 μμ	timebomb/60	4
30	6/11/2021 8:36:44 μμ	timebomb/60	4
31	6/11/2021 8:37:00 μμ	We agreed on five hundred	4
32	6/11/2021 8:37:08 μμ	I want more	3
33	6/11/2021 8:37:08 μμ	I want more	3
34	6/11/2021 8:37:17 μμ	WE WILL SEE	4
35	6/11/2021 8:37:41 μμ	I think they saw me. I must delete those	3

Figure 54. How "viber_messages" journal file was modified five minutes after the conversation thread was deleted (tool used: Oxygen Forensic SQlite, Examined Device: Samsung A50).

It has to be mentioned that in the latter example the timebomb messages had not yet been vacuumed. In a realistic scenario the acquisition of the data after five minutes it is considered impossible. The full "auto_vacuum" feature of the database irretrievably removes the timebomb records after a few minutes, making it impossible to recover them, while the records/messages which were manually deleted (the ones that existed in the previous state of the database) are still recoverable. This procedure is independent of the device's brand or OS version. The exact timeframe in which full "auto_vacuum" is triggered could not be pinpointed. **Figures 55** to **58** depicts the way that the messages table in the Viber database in the LG device is modified through time using several forensic tools and software.

#	msg_date	participant_id	user_id	status	opened	body	extra_uri
1	31/10/2021 1...	1	Qhnb3h7J3pE=	2	0	I am sending the photo	
2	31/10/2021 1...	1	Qhnb3h7J3pE=	2	0	content://com.viber.voip.provider.internal_files/message...	content://media/external/images/media/949
3	31/10/2021 1...	1	Qhnb3h7J3pE=	2	0	content://com.viber.voip.provider.internal_files/message...	content://media/external/images/media/949
4	31/10/2021 1...	1	Qhnb3h7J3pE=	2	0	Did you receive it?	
5	31/10/2021 1...	2	Qhnb3h7J3pE=	2	0	Yes send the second one	
6	31/10/2021 1...	1	Qhnb3h7J3pE=	2	0	content://com.viber.voip.provider.internal_files/message...	content://com.viber.voip.provider.internal_files/message...
7	31/10/2021 1...	1	Qhnb3h7J3pE=	2	0	content://com.viber.voip.provider.internal_files/message...	content://com.viber.voip.provider.internal_files/message...
8	31/10/2021 1...	1	Qhnb3h7J3pE=	2	0	This will cost you 1000 euros	
9	31/10/2021 1...	1	Qhnb3h7J3pE=	2	0	This will cost you 1000 euros	
10	31/10/2021 1...	2	Qhnb3h7J3pE=	0	0	timebomb/60	
11	31/10/2021 1...	2	Qhnb3h7J3pE=	2	0	We agreed on five hundred	
12	31/10/2021 1...	1	Qhnb3h7J3pE=	2	0	I think they saw me. I must delete these	
13	31/10/2021 1...	1	Qhnb3h7J3pE=	2	0	I think they saw me. I must delete these	

Figure 55. How "viber_messages" journal file was modified five minutes after the timebomb messages were deleted (tool used: Oxygen Forensic SQlite, Examined Device: LG G6).

31/10/2021 10:08:05 πμ	1	I am sending the photo	Qhnb3h7J3pE=
31/10/2021 10:08:05 πμ	1	I am sending the photo	Qhnb3h7J3pE=
31/10/2021 10:08:12 πμ	1	content://com.viber.voip.provider.internal_files/message/local/thumb/f61a2b5b541014969df6695d...	Qhnb3h7J3pE=
31/10/2021 10:08:12 πμ	1	content://com.viber.voip.provider.internal_files/message/local/thumb/f61a2b5b541014969df6695d...	Qhnb3h7J3pE=
31/10/2021 10:08:26 πμ	1	Did you receive it?	Qhnb3h7J3pE=
31/10/2021 10:08:26 πμ	1	Did you receive it?	Qhnb3h7J3pE=
31/10/2021 10:08:48 πμ	2	Yes send the second one	Qhnb3h7J3pE=
31/10/2021 10:08:48 πμ	2	Yes send the second one	Qhnb3h7J3pE=
31/10/2021 10:09:24 πμ	1	content://com.viber.voip.provider.internal_files/message/local/thumb/e27c001ac1c34e8c6dff95419...	Qhnb3h7J3pE=
31/10/2021 10:09:24 πμ	1	content://com.viber.voip.provider.internal_files/message/local/thumb/e27c001ac1c34e8c6dff95419...	Qhnb3h7J3pE=
31/10/2021 10:09:44 πμ	1	This will cost you 1000 euros	Qhnb3h7J3pE=
31/10/2021 10:09:44 πμ	1	This will cost you 1000 euros	Qhnb3h7J3pE=
31/10/2021 10:10:03 πμ	2	timebomb/60	Qhnb3h7J3pE=
31/10/2021 10:10:03 πμ	2	timebomb/60	Qhnb3h7J3pE=

Figure 56. How “viber_messages” journal file was modified 1, 6, 24 and 48 hours after the conversation thread was deleted with the device in several states. The timebomb messages were deleted permanently (tool used: Oxygen Forensic SQLite, Examined Device: LG G6).

#	Deleted	Attachment	Timestamp	From		Body
4	×	📎	31/10/2021 10:08:12 πμ(UTC...	+3069407	Bad Actor 1 (owner)	
5			31/10/2021 10:08:26 πμ(UTC...	+3069407	Bad Actor 1 (owner)	Did you receive it?
6	×		31/10/2021 10:08:26 πμ(UTC...	+3069407	Bad Actor 1 (owner)	Did you receive it?
7			31/10/2021 10:08:48 πμ(UTC...	+3069472	Μιχάλης	Yes send the second one
8	×		31/10/2021 10:08:48 πμ(UTC...	+3069472	Μιχάλης	Yes send the second one
9		📎	31/10/2021 10:09:24 πμ(UTC...	+3069407	Bad Actor 1 (owner)	This is the second one
10	×	📎	31/10/2021 10:09:24 πμ(UTC...	+3069407	Bad Actor 1 (owner)	This is the second one
11			31/10/2021 10:09:44 πμ(UTC...	+3069407	Bad Actor 1 (owner)	This will cost you 1000 euros
12	×		31/10/2021 10:09:44 πμ(UTC...	+3069407	Bad Actor 1 (owner)	This will cost you 1000 euros
13			31/10/2021 10:10:03 πμ(UTC...	+3069472	Μιχάλης	timebomb/60
14	×		31/10/2021 10:10:03 πμ(UTC...	+3069472	Μιχάλης	timebomb/60
15	×		31/10/2021 10:10:29 πμ(UTC...	+3069472	Μιχάλης	We agreed on five hundred
16	×		31/10/2021 10:11:05 πμ(UTC...	+3069407	Bad Actor 1 (owner)	I think they saw me. I must delete thes

Figure 57. How “viber_messages” journal file was modified five minutes after the timebomb messages were deleted (tool used: UFED PA, Examined Device: LG G6)

#	Deleted	Attachment	Timestamp	From		Body
1	×		31/10/2021 10:08:05 πμ(UTC...	+3069407	Bad Actor 1 (owner)	I am sending the photo
2	×	📎	31/10/2021 10:08:12 πμ(UTC...	+3069407	Bad Actor 1 (owner)	
3	×		31/10/2021 10:08:26 πμ(UTC...	+3069407	Bad Actor 1 (owner)	Did you receive it?
4	×		31/10/2021 10:08:48 πμ(UTC...	+3069472	Μιχάλης	Yes send the second one
5	×	📎	31/10/2021 10:09:24 πμ(UTC...	+3069407	Bad Actor 1 (owner)	This is the second one
6	×		31/10/2021 10:09:44 πμ(UTC...	+3069407	Bad Actor 1 (owner)	This will cost you 1000 euros
7	×		31/10/2021 10:10:03 πμ(UTC...	+3069472	Μιχάλης	timebomb/60

Figure 58. How “viber_messages” journal file was modified 1, 6, 24 and 48 hours after the conversation thread was deleted with the device in several states. The timebomb messages were deleted permanently (tool used: UFED PA, Examined Device: LG G6).

As already shown even one commit to the database was enough to lose the majority off the “incriminating messages”. By receiving just five (5) messages from a third participant almost all the previous commits were vacuumed and thus permanently deleting the messages of interest (**Figures 59** and **60**).

11	3	6/11/2021 8:35:09 μμ	I am sending the photo	Qhnb3h7J3pE=	2
12	3	6/11/2021 8:35:18 μμ	content://com.viber.voip.provider.internal_files/message/local/thumb/155012ebd1c93729a0dce8dc9...	Qhnb3h7J3pE=	2
13	3	6/11/2021 8:36:11 μμ	content://com.viber.voip.provider.internal_files/message/local/thumb/b68ab198524939961cbb801a0...	Qhnb3h7J3pE=	2

Figure 59. How “viber_messages” journal file is modified after five (5) additional messages were received and not opened by the user (tool used: Oxygen Forensic SQLite, Examined Device: Samsung A50). Only the messages that contained records from the “msg_info” column remained.

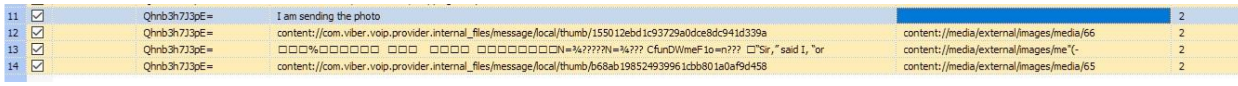


Figure 60. How "viber_messages" journal file is modified after the notification, of the previously received messages, is removed from the device. The data are further corrupted introducing false positive records (tool used: Oxygen Forensic SQLite, Examined Device: Samsung A50).

In conclusion, just three actions-commits, involving receiving messages, reading incoming messages, or having outcoming messages, read by another participant (receiver) etc., were enough to erase every trace of the records of interest.

4.2. Viber's Media Files Attachments

In the experiments two media files, were sent, one which created through the default camera application of the device and the other, was created via the internal camera feature of the Viber application. The media file was stored at «media/0/Pictures/Viber» if it was captured by the internal feature of the application or at «media/0/DCIM/Camera/» if it was captured by the default camera application of the device. Since the application did not automatically remove the media file from the filesystem when the messages which included it, was deleted, it was removed manually. In addition, the trash bin folder of the photo gallery application was emptied. The trash bin folder is an OS feature where after a media file's deletion, it is moved and kept separately for a period of time. This period varies, and for the LG device is 7 days as for the SAMSUNG device is 30 days. For media files except the columns which were mentioned in paragraph §**4.1.3**, there were also two more, which were extremely useful. The "**msg_info**" (**Figure 61**) which contained the metadata of the sent files in JSON format and the "**extra_uri**" (**Figure 62**) in which the internal storage path of the file is mentioned.

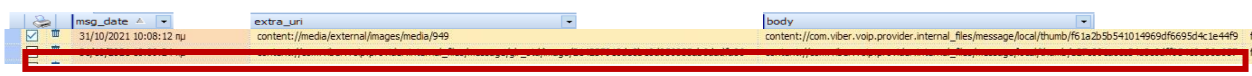


Figure 61. The "extra_uri" and "body" columns of the table "messages" in "viber_messages.db" (Examined Device: LG G6).

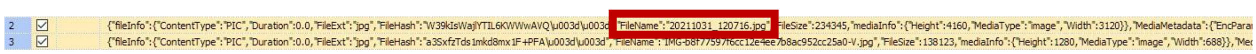


Figure 62. The "msg_info" column of the table "messages" in "viber_messages.db" (Examined Device: LG G6).

The thumbnails of the media files, which were created during the transmission, were stored at the path «/media/0/Android/data/com.viber.voip/files/.thumbnails/» (**Figure 63**). In order to correlate the content of the transmitted message with a media file (picture) with a picture's created thumbnail, a comparison was made between their timestamps. One other path where artifacts can be found is the cache folder of the Viber application at «/media/0/Android/data/com.viber.voip/cache/ImageFetcherThumb» (**Figure 64**).

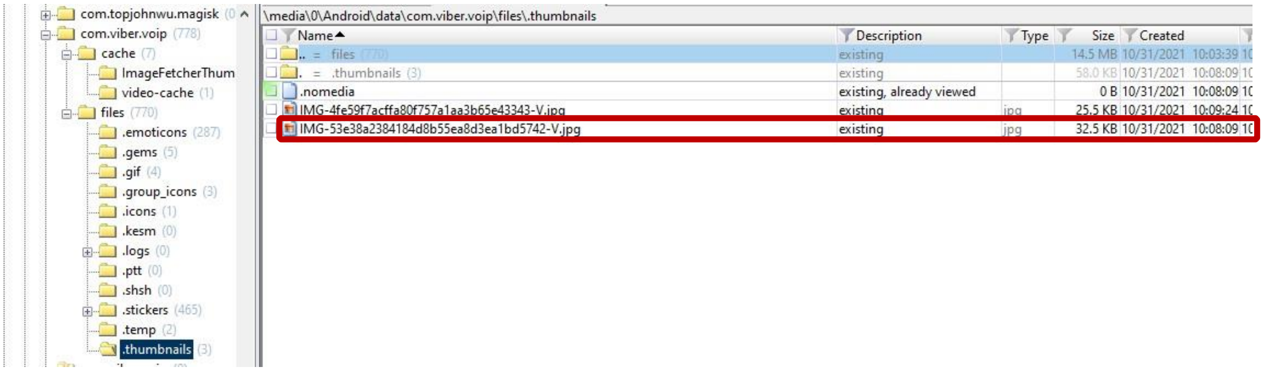


Figure 63. Media files thumbnail's location (Examined Device: LG G6).

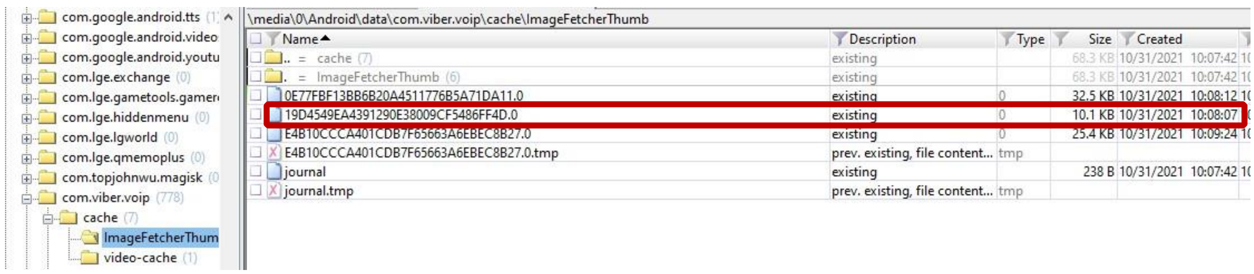


Figure 64. Media files cache location (Examined Device: LG G6).

4.2.1. LG device's media artifacts

In the LG device the thumbnails of the media gallery were stored at path: «media/0/DCIM/.thumbnails». The timestamp when the user deleted the files and when the trash bin was emptied could be retrieved from the contents of the "trashcan.db" database and it's WAL file located at the path: «/data./data/com.android.gallery3d/databases/» (**Figure 65**).

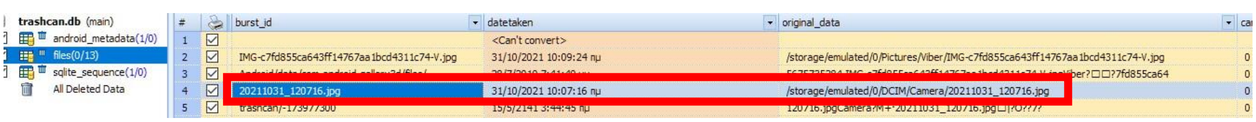


Figure 65. LG G6's "trashcan.db" WAL file record (tool used: Oxygen Forensic SQlite).

Furthermore, the conversion of the blob file named "imagecache.0", which is located at the path: «/media/0/Android/data/com.android.gallery3d/cache», with a free tool (slo-sleuth, 2021) to a SQLite database and then parsing it with DB Browser for SQLite, provides valuable information about the captured media files (**Figure 66**).

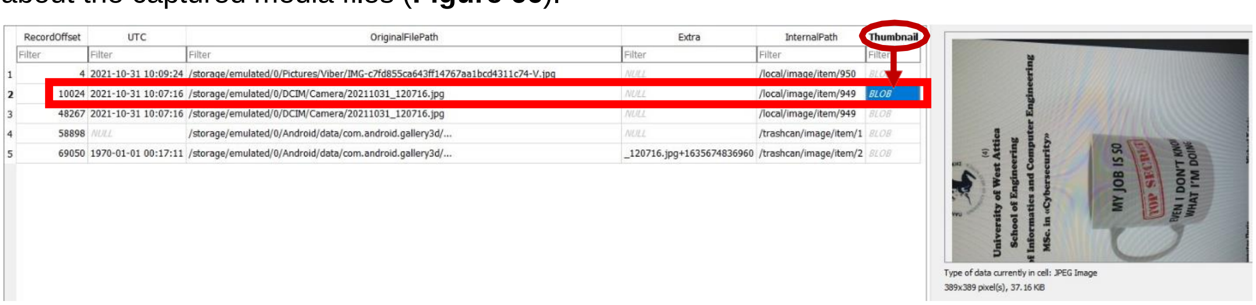


Figure 66. Parsing "imagecache0.sqlite" (tool used: DB Browser for SQLite, Examined Device: LG G6).

To summarize all the above-mentioned artifacts, they are presented in chronological order:

- At 31-10-2021 **10:07:16** UTC a media file named «20211031_120716» was captured by the device's camera application and stored at path: «../DCIM/Camera/» as the item «**949**» in the device's "InternalPath". The file's content is presented at the "Thumbnail" column (**Figure 66**).
- At 31-10-2021 **10:08:07** UTC the file is imported at the application for transmission and a cache file is created (**Figure 64**).
- At 31-10-2021 **10:08:09** UTC the file's thumbnail is created and stored permanently at path: «media/0/Android/data/com.viber.voip/files/.thumbnails/» (**Figure 63**).
- At 31-10-2021 **10:08:12** UTC the file is sent as a message. The "extra_uri" and "body" columns depict the original internal path of the media (**Figure 61**), and the media's filename is shown inside the JSON file (**Figure 62**).
- The file is deleted and removed from the trash bin of the device's gallery application (**Figure 65**).

Finally, if the location tag' s setting was enabled in the examined device, the geolocation of the captured pictures could be found in the file «/media/0/Android/data/com.android.gallery3d/cache/**rev_geocoding.0**».

4.2.1. Samsung device's media artifacts

The Samsung device has a modified version of stock android. It does not use the Android's default camera or photo gallery applications. The pictures' thumbnails are located in the Samsung gallery application at path: «/data/com.sec.android.gallery.3d/cache/». The analysis of the "local.db" database located at the path: «/data/com.sec.android.gallery.3d/databases/» and specifically the "trash" table of this database provided the timestamps of media deletion (**Figure 67**). Most of the column names of this table were self-explanatory.

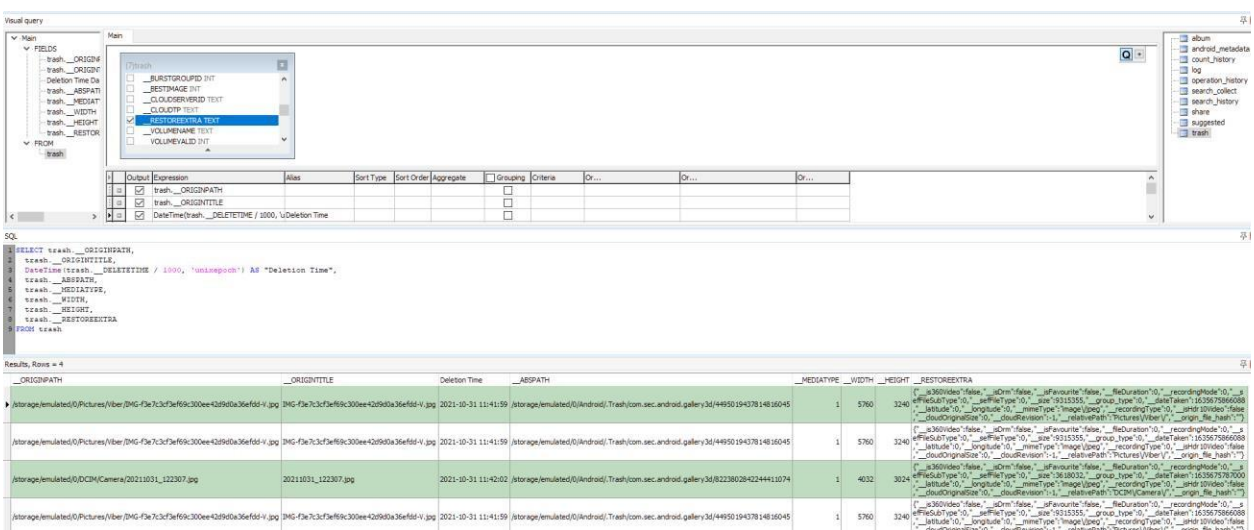


Figure 67. The "trash" table of Samsung's "local.db" (tool used: Forensic Toolkit for SQLite, Examined Device: Samsung A50).

Two more databases which contained interesting artifacts were:

- The “graph.db” located at path: «/data/com.samsung.**mlp**/databases» (**Figure 68**). MLP stands for Mobile Location Protocol, and it can be used, among others, to track the media files which were sent as messages.

☑	/storage/emulated/0/DCIM/Camera/20211031_122307.jpg	65537		31/10/2021 10:23:56 πμ	31/10/2021 10:23:07 πμ
☑	/storage/emulated/0/Pictures/Viber/IMG-f3e7c3cf3ef69c300ee42d9d0a36e…	65537		31/10/2021 10:25:04 πμ	31/10/2021 10:24:26 πμ

Figure 68. The “MediaAttribute” table of Samsung “graph.db” (tool used: Oxygen Forensic SQLite, Examined Device: Samsung A50).

- The “core2.db” at path: «/data/com.sec.android.app.camera/databases/» (**Figure 69**), contained information only about the pictures created via the camera application.

☑	2147483649	/storage/emulated/0/DCIM/Camera/20210905_124444.jpg	external_primary	5/9/2021 9:44:44 πμ	20210905_124444	20210905_124444.jpg
☑	2147483650	/storage/emulated/0/DCIM/Camera/20210905_124454.jpg	external_primary	5/9/2021 9:44:54 πμ	20210905_124454	20210905_124454.jpg
☑	2147483651	/storage/emulated/0/DCIM/Camera/20211031_104226.jpg	external_primary	31/10/2021 8:42:26 πμ	20211031_104226	20211031_104226.jpg
☑	2147483653	/storage/emulated/0/DCIM/Camera/20211031_122307.jpg	external_primary	31/10/2021 10:23:07 πμ	20211031_122307	20211031_122307.jpg

Figure 69. The “files” table of Samsung “core2.db” (tool used: Oxygen Forensic SQLite, Examined Device: Samsung A50).

4.3. Final Conclusions on Viber

To sum up, DFIR Team members should avoid interacting with the Viber Application on android devices (remove notifications, read messages etc). In addition, they must isolate the device from networks, so that the Viber server cannot push any new data (i.e., read a sent message from a third participant/receiver). The experiments showed that the elapsed time or the state of the device (turned on/off, reboot, removing SIM etc.) did not alter the in-question data (deleted). Information about the in-question messages, especially the ones that contain media files, might be retrieved from artifacts residing in files and databases of other OS applications.

Chapter 5.

WhatsApp application Scenario

In this chapter the WhatsApp application's examination and the related results are presented regarding text and media exchanged messages and the volatility of the database in which they are stored. Like in the previous chapter there is a reference on the application database's location, it's features and the differences between an Android 9 and an Android 11 examined device. After that there is an analysis of the tables and columns of the database and the queries that were built in order to view the desired ones. In addition, the volatility of deleted text and media messages is examined through multiple experiments. Furthermore, some unique characteristics of the application are examined, and final conclusions are drawn.

5.1. WhatsApp's Examination

The used .apk for both the devices was the WhatsApp v.2.21.22.27 and its release date was on 8th November 2021. Application' s data's, file and folder structure is well known and documented over the years for both educational/research (Rathi et al., 2018, Zhang et al., 2018) and forensic purposes (Belkasoft, 2021). There is an interesting approach made by Mirza et al. (2020) in which artifacts can be retrieved, under specific circumstances from the notification system of the application. As it has already been mentioned this Thesis is focused on the actions of an DFIR and the methodology best suited to be followed regarding the preservation of deleted data (messages). There were two databases that contained forensically interesting artifacts: the “msgstore.db” which included the messages, and the “wa.db” which included information about the user contacts. Both databases were located at the path: «/data/data/com.whatsapp/databases». Furthermore, the private encryption key was retrieved from the path: «/data/data/com.whatsapp/files» and used for the decryption of the user's chat backup. Once again, it must be pointed out that access to these paths is normally prohibited, so either the examiner must have root access to the device or acquire a physical image of it to extract them.

5.1.1. Android 9 vs Android 11 Comparison

The initial size of the “msgstore.db” on both devices, running different Android version, was 988kb. However, the WAL file's initial size differed. On the LG Device (Android 9) its size was 12kb, growing to the maximum of 477kb, and on the Samsung Device (Android 11) it had a steady size of 512kb.

5.1.2. General information about the under-examination database

The header and PRAGMA statements of the “msgstore.db” database analysis, by using the non-forensic tools “SQL Expert” and “DB Browser for SQLite” showed that (**Figure 70**):

- ✓ had the Full auto_vacuum enabled and did not create freelist pages,
- ✓ used WAL as a journaling mode,

- ✓ the WAL Auto Checkpoint had been set at one thousand (1.000) commits (default size).

The meaning and forensic value of these pragma statements (enabled options) have been analyzed in paragraph §**1.3.1.2**. All these settings render the data, especially the deleted ones, volatile. However, the use of WAL as a journaling mode provides some leniency in comparison to the use of the rollback journal in the Viber application.

Name	Value
application_id	0
auto_vacuum	full
cache_size	-2000
collation_list	[BINARY], [NOCASE], [RTRIM]
encoding	UTF-8
foreign_keys	on
freelist_count	0
journal_mode	wal
journal_size_limit	-1
max_page_count	1073741823
mmap_size	0
page_count	249
page_size	4096
schema_version	315
user_version	1
writable_schema	off

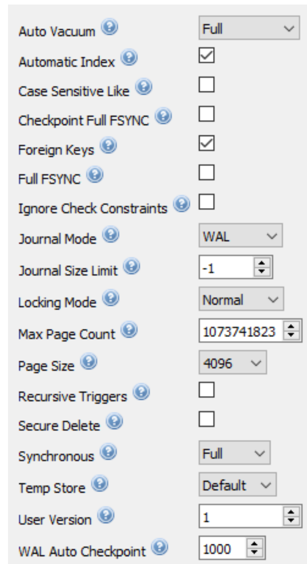


Figure 70. Analysis of "msgstore.db" properties (tools used: SQLite Expert (left) & DB Browser for SQLite (right)).

The "wa.db" database contained information about the contacts of the user and participants in the chat threads. Since it is not frequently modified (a user does not add or delete contacts every minute) it can be characterized as a "steadier" database and even though it will be utilized in the SQL queries, its properties, will not be analyzed any further.

5.1.3. Analysis of the "msgstore.db" and the "wa.db"

The "msgstore" database was composed by one hundred and forty-two (142) tables (**Figure 69**). The two (2) most forensically interesting tables for the examined scenario were the "messages" and "message_media". The "wa" database was composed by twenty-four (24) tables (**Figure 70**) with the most forensically interesting table to be the "wa_contacts".

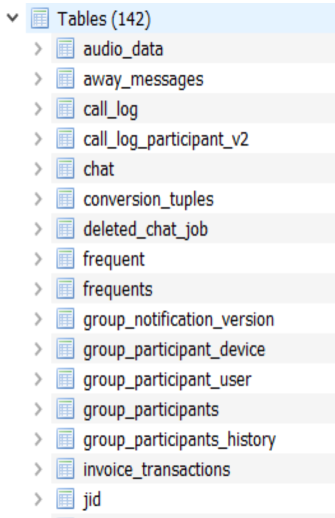


Figure 71. Depiction of the tables tree of "msgstore.db".

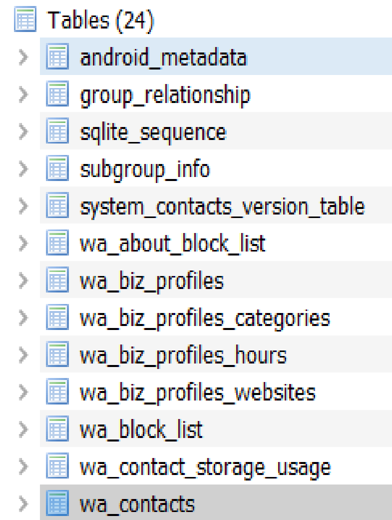


Figure 72. Depiction of the tables tree of "wa.db".

Among the most interesting columns of the "msgstore" database's table "**messages**" were:

- The "**key remote jid**" contained the WhatsApp ID of the contact.
- The "**timestamp**" contained the timestamp of the message creation, and it was encoded in Unix Millisecond (Java Time).
- The "**key_from_me**" contained information about the message direction.
- The "**media_wa_type**" identified the type of message (i.e., text, media, audio, video).
- The "**data**" contained the body of the text message.
- The "**media_url**" was the web location of the encrypted media in WhatsApp servers.
- The "**media_caption**" contained the title of the transmitted media.

Among the most interesting columns of the "msgstore" database's table "**message_media**" were:

- The "**file_path**" contained the original file path of the media file in the device's filesystem.
- The "**message_url**" was the web location of the encrypted media' s in WhatsApp servers.
- The "**media_key**" contained the decryption key in binary form (BLOB).

Among the most interesting columns of the "wa" database's table "**wa_contacts**" were:

- The "**number**" contained the phone number associated to the contact.
- The "**display name**" contained the contact's display name.

At this point it must be mentioned that the columns' names inside the tables and the meaning of the corresponding flags/data are modified among versions. A further analysis for most of the above-mentioned tables and their columns is made by Anglano C. (2014). To get an overview of the user's activity the two different databases ("msgstore" and "wa") had to be attached and the three tables had to be combined (**Figure 73**). The creation of the queries was a bit more complicated (**Figure 74**).

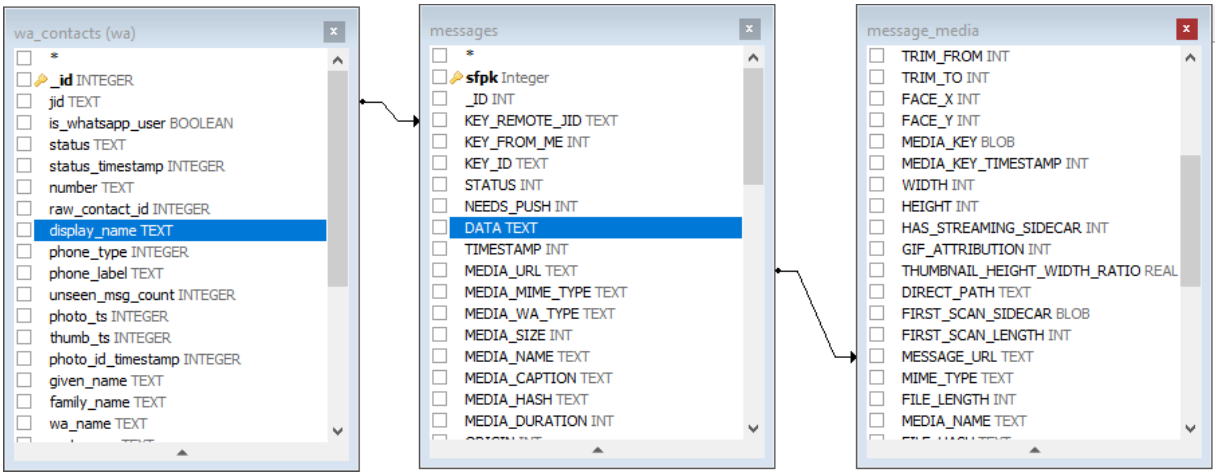


Figure 73. SQL Diagram of the joined tables (tool used: Forensic Browser for SQLite)

```sql
SELECT
wa.wa_contacts.display_name AS 'User Name',
wa.wa_contacts.number AS 'Phone Number',
datetime(messages.timestamp/1000, 'unixepoch') AS 'Message Date',
 CASE messages.KEY_FROM_ME
    WHEN '0' THEN 'Message Received'
    ELSE 'The Message was sent'
END AS 'Message Direction',
datetime(messages.READ_DEVICE_TIMESTAMP/1000, 'unixepoch') AS 'Date was Read',
messages.DATA AS Message,
CASE messages.media_wa_type
    WHEN '0' THEN 'text'
    WHEN '1' THEN 'image'
    WHEN '2' THEN 'audio'
    WHEN '3' THEN 'video'
    WHEN '4' THEN 'contact card'
    WHEN '5' THEN 'geo location'
END AS 'Message Type',
messages.MEDIA_CAPTION AS 'Media Message',
messages.MEDIA_URL AS 'Whatsapp server media url',
hex(message_media.MEDIA_KEY) as 'Media Decryption key',
message_media.FILE_PATH AS 'File in the Device'

FROM messages

   LEFT JOIN message_media ON message_media.MESSAGE_URL = messages.MEDIA_URL
   LEFT JOIN wa.wa_contacts ON wa.wa_contacts.jid = messages.KEY_REMOTE_JID
  WHERE messages.key_remote_jid NOT like '-1'
  ORDER BY messages.timestamp ASC
```

	Jser Name	one Numb	Message Date	Message Direction	Date was Read	Message	Message Type	Media Message	Whatsapp server media url	Media Decryption key
1	ιχάλης	+30694...	2021-11-14 18:25:40	The Message was sent	NULL	NULL	text	NULL	NULL	
2	ιχάλης	+30694...	2021-11-14 18:46:59	Message Received	NULL	This is the third ...	text	NULL	NULL	
3	ιχάλης	+30694...	2021-11-14 18:48:23	The Message was sent	2021-11-14 18:49:00	This is the fourth ...	text	NULL	NULL	
4	ad Actor 1	+30694...	2021-11-17 10:47:27	Message Received	NULL	NULL	image	This is an elephant	https://mmg.whatsapp.net/d/f/...	167F89F24F39FC687EE149DEF7915AA10E15C29...
5	ad Actor 1	+30694...	2021-11-17 10:47:29	The Message was sent	NULL	NULL	text	NULL	NULL	
6	ad Actor 1	+30694...	2021-11-17 10:49:31	The Message was sent	2021-11-17 10:51:43	NULL	image	This is a crocodile	https://mmg.whatsapp.net/d/f/...	23AFE1071D7A23C8191325616913520B12C9BE...
7	ad Actor 1	+30694...	2021-11-17 10:52:02	Message Received	NULL	This is the fifth ...	text	NULL	NULL	

Figure 74. SQL query combining several columns from three different tables in two different databases.

A misperception is that since the WAL's checkpoint is at one thousand (1.000) commits there is "plenty of room" for many messages to be stored. Examinations showed that by sending one plain text message there were fifteen (15) different commits (**Figure 75**) to the "msgstore" database and by sending one media file fifty (50) commits occurred (**Figure 76**).

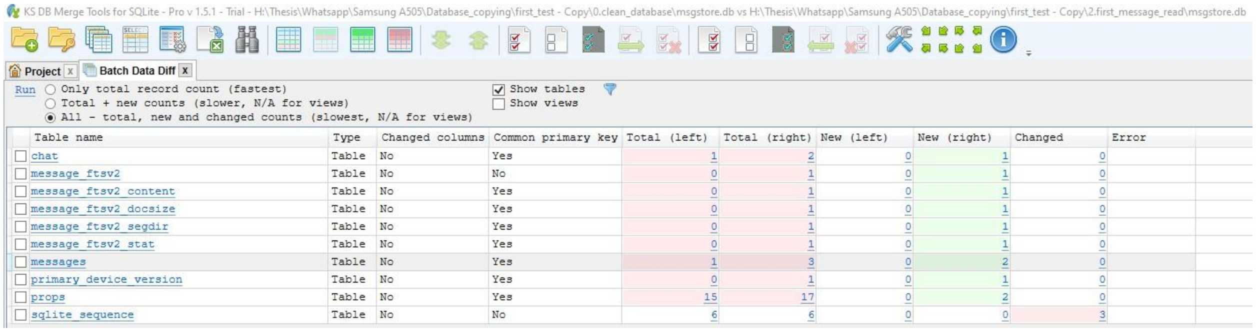


Table name	Type	Changed columns	Common primary key	Total (left)	Total (right)	New (left)	New (right)	Changed	Error
chat	Table	No	Yes	1	2	0	1	0	
message_ftsv2	Table	No	No	0	1	0	1	0	
message_ftsv2_content	Table	No	Yes	0	1	0	1	0	
message_ftsv2_docsize	Table	No	Yes	0	1	0	1	0	
message_ftsv2_segdir	Table	No	Yes	0	1	0	1	0	
message_ftsv2_stat	Table	No	Yes	0	1	0	1	0	
messages	Table	No	Yes	1	3	0	2	0	
primary_device_version	Table	No	Yes	0	1	0	1	0	
props	Table	No	Yes	15	17	0	2	0	
sqlite_sequence	Table	No	No	6	6	0	0	3	

Figure 75. Differences in the “msgstore.db” after the receipt of one message (tool used: KS DB Merge Tools).

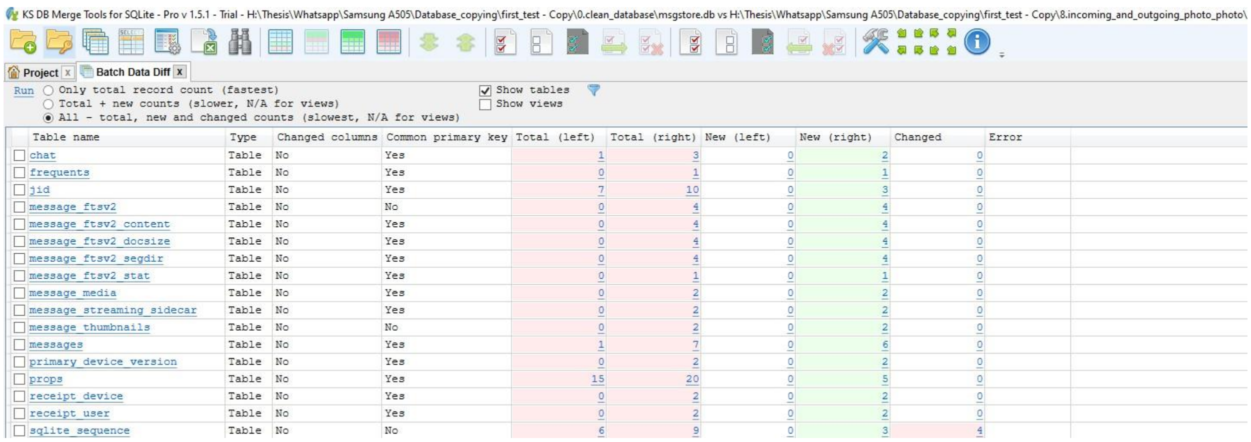


Table name	Type	Changed columns	Common primary key	Total (left)	Total (right)	New (left)	New (right)	Changed	Error
chat	Table	No	Yes	1	3	0	2	0	
frequents	Table	No	Yes	0	1	0	1	0	
jid	Table	No	Yes	7	10	0	3	0	
message_ftsv2	Table	No	No	0	4	0	4	0	
message_ftsv2_content	Table	No	Yes	0	4	0	4	0	
message_ftsv2_docsize	Table	No	Yes	0	4	0	4	0	
message_ftsv2_segdir	Table	No	Yes	0	4	0	4	0	
message_ftsv2_stat	Table	No	Yes	0	1	0	1	0	
message_media	Table	No	Yes	0	2	0	2	0	
message_streaming_sidecar	Table	No	Yes	0	2	0	2	0	
message_thumbnails	Table	No	No	0	2	0	2	0	
messages	Table	No	Yes	1	7	0	6	0	
primary_device_version	Table	No	Yes	0	2	0	2	0	
props	Table	No	Yes	15	20	0	5	0	
receipt_device	Table	No	Yes	0	2	0	2	0	
receipt_user	Table	No	Yes	0	2	0	2	0	
sqlite_sequence	Table	No	No	6	9	0	3	4	

Figure 76. Differences in the “msgstore.db” after sending one media file (tool used: KS DB Merge Tools).

5.1.4. Volatility of the “msgstore.db”

The experiments confirmed that both the “msgstore” database and its WAL file were extremely volatile. However, certain actions/statuses/time did not seem to alter the data inside the “messages” or “message_media” tables. Those tested circumstances were:

- ✓ Leaving the device powered on and having enabled network connections.
- ✓ When device’s boot sequence finished (reboot/restart) and the application auto starts.
- ✓ At opening of the application after closing it.
- ✓ Every few hours even in airplane mode.
- ✓ Leaving the device in a faraday box with or without network.

The database’s “messages” or “message_media” tables were modified when:

- ! a message was created. It did not matter if it was successfully sent or not.
- ! a message was received.
- ! a message was opened. This was regardless of whether it was an outgoing message which was read by the receiver or an incoming which was read by the user.

! a message or a thread was deleted.

During the experiments the devices did not receive any bot messages (ads). Another finding was that the incoming message notifications did not modify the “messages” table as it did on the examination of the Viber application. The record of one message was present in the WAL file multiple times due to its state changes (sent, delivered, read) or it’s status alterations (live, deleted). So, there were cases where a message record was created, sent and deleted and never committed to the main database. All these records resided only in the WAL file.

5.1.5. Retrieving the messages in question

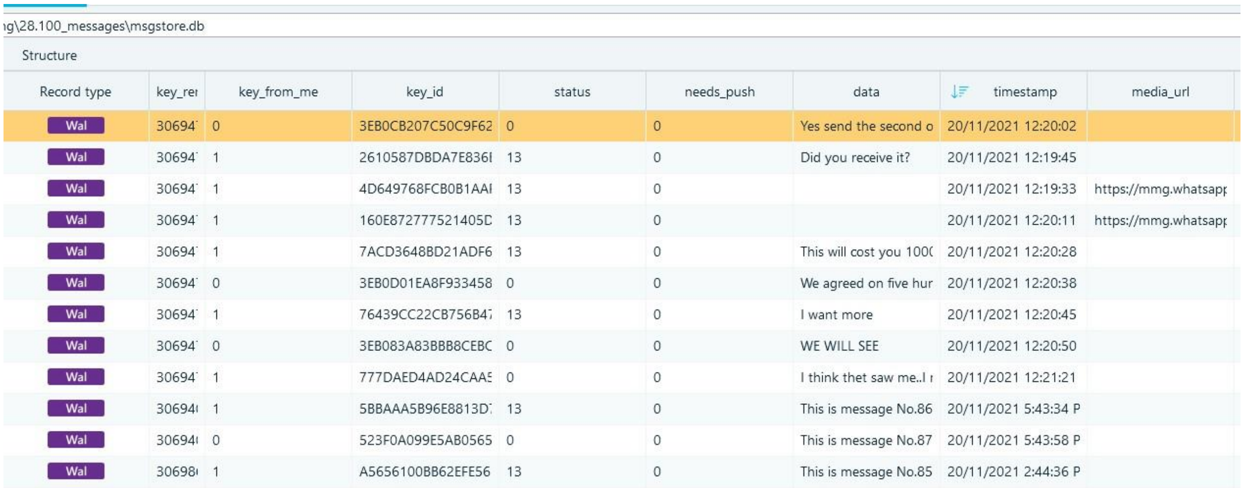
At the time of the experimentation phase WhatsApp’s implementation for messages autodeletion was a predefined period of time of seven (7) days. This was the only available option and because of the long duration of this feature it was not taken under consideration. The reason was that in an DFIR scenario the procedures and actions are supposed to occur in a short period of time. As already mentioned, it is very difficult for the forensic tools, free or commercial, to keep up with the applications’ settings and databases updating speed. The automated parsing, of the forensics tools, specifically regarding the WAL file, still had different levels of success as is depicted in **Figures 77** to **79**. In general, in order to verify and preview results, related to the communication with a specific contact, the searchable string of its WhatsApp ID (column “key_remote_jid”) was used in a Hex Viewer (e.g., X-Ways Forensics) (**Figure 80**).

	Offset	commit	dbpage	walframe	salt1	salt2	_id	key_rem...	key_from...	key_id	status	needs_pu...	data	timestamp
7	01E011	true	249	29	772401287	2520792042	149	306986605...	0	3EB0A149...	0	0	This is message No.64	Sat Nov 20 11:50:26 EET 2021
9	02D179	true	249	44	772401287	2520792042	149	306986605...	0	3EB0A149...	0	0	This is message No.64	Sat Nov 20 11:50:26 EET 2021
2	03C2E1	true	249	59	772401287	2520792042	149	306986605...	0	3EB0A149...	0	0	This is message No.64	Sat Nov 20 11:50:26 EET 2021
8	0443A1	true	249	67	772401286	1649287693	149	306986605...	0	3EB0A149...	0	0	This is message No.64	Sat Nov 20 11:50:26 EET 2021
5	053509	true	249	82	772401286	1649287693	149	306986605...	0	3EB0A149...	0	0	This is message No.64	Sat Nov 20 11:50:26 EET 2021
...	05F629	true	249	94	772401286	1649287693	149	306986605...	0	3EB0A149...	0	0	This is message No.64	Sat Nov 20 11:50:26 EET 2021
...	06E791	true	249	109	772401286	1649287693	149	306986605...	0	3EB0A149...	0	0	This is message No.64	Sat Nov 20 11:50:26 EET 2021
...	00EEA9	true	249	14	772401287	2520792042	149	306986605...	0	3EB0A149...	0	0	This is message No.64	Sat Nov 20 11:50:26 EET 2021
6	01DF8D	true	249	29	772401287	2520792042	150	306986605...	0	3EB02DCA...	0	0	This is message No.65	Sat Nov 20 11:50:29 EET 2021

Figure 77. Parsing messages’ content in “msgstore.db”. The tool failed to parse any of the records in question (tool used: FQLite, Examined Device: Samsung A50).

20/11/2021 11:50:49 A...	20/11/2021 11:50:49 A...			This is message No.70	text
20/11/2021 12:20:02 PM	20/11/2021 12:20:02 PM			Yes send the second...	text
20/11/2021 12:20:38 PM	20/11/2021 12:20:38 PM			We agreed on five hu...	text
20/11/2021 12:20:50 PM	20/11/2021 12:20:50 PM			WE WILL SEE	text
20/11/2021 2:05:42 PM	20/11/2021 2:05:42 PM			This is message No.81	text
20/11/2021 2:05:42 PM	20/11/2021 2:05:42 PM			This is message No.81	text
20/11/2021 2:17:35 PM	20/11/2021 2:17:35 PM			This is message No.82	text
20/11/2021 2:17:35 PM	20/11/2021 2:17:35 PM			This is message No.82	text
20/11/2021 2:44:36 PM		20/11/2021 2:44:38 PM	20/11/2021 2:44:36 PM	This is message No.85	text
20/11/2021 5:43:34 PM		20/11/2021 5:43:35 PM	20/11/2021 5:43:34 PM	This is message No.86	text
20/11/2021 5:43:58 PM	20/11/2021 5:43:59 PM			This is message No.87	text
20/11/2021 7:12:53 PM	20/11/2021 7:12:54 PM			This is messagge No.91	text
20/11/2021 7:12:53 PM	20/11/2021 7:12:54 PM			This is messagge No.91	text

Figure 78. Parsing messages’ content in “msgstore.db”. The tool managed to retrieve/parse some of the records in question (tool used: Magnet Axiom, Examined Device: Samsung A50).



Record type	key_rer	key_from_me	key_id	status	needs_push	data	timestamp	media_url
Wal	30694	0	3EB0CB207C50C9F62	0	0	Yes send the second o	20/11/2021 12:20:02	
Wal	30694	1	2610587DBDA7E836I	13	0	Did you receive it?	20/11/2021 12:19:45	
Wal	30694	1	4D649768FCB0B1AAI	13	0		20/11/2021 12:19:33	https://mmg.whatsapp
Wal	30694	1	160E872777521405D	13	0		20/11/2021 12:20:11	https://mmg.whatsapp
Wal	30694	1	7ACD3648BD21ADF6	13	0	This will cost you 100(	20/11/2021 12:20:28	
Wal	30694	0	3EB0D01EA8F933458	0	0	We agreed on five hur	20/11/2021 12:20:38	
Wal	30694	1	76439CC22CB756B4	13	0	I want more	20/11/2021 12:20:45	
Wal	30694	0	3EB083A83BBB8CEBC	0	0	WE WILL SEE	20/11/2021 12:20:50	
Wal	30694	1	777DAED4AD24CAA	0	0	I think thet saw me..I	20/11/2021 12:21:21	
Wal	30694	1	5BBAAA5B96E8813D	13	0	This is message No.86	20/11/2021 5:43:34 P	
Wal	30694	0	523F0A099E5AB0565	0	0	This is message No.87	20/11/2021 5:43:58 P	
Wal	30698	1	A5656100BB62EFE56	13	0	This is message No.85	20/11/2021 2:44:36 P	

Figure 79. Parsing messages' content in "msgstore.db". The tool was able to retrieve all the records in question (tool used: Belkasoft-X, Examined Device: Samsung A50).

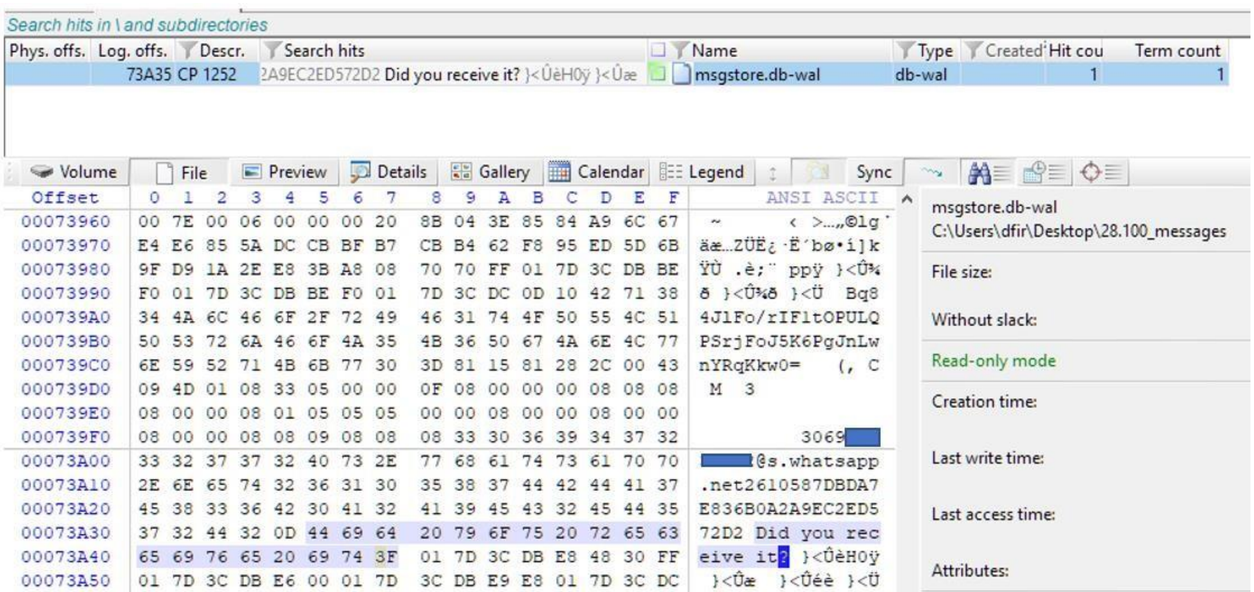


Figure 80. Searching for messages content using "key_remote_jid" in "msgstore.db" (tool used: X-Ways Forensics, Examined Device: Samsung A50).

To monitor the experiments, course the "msgstore" database and its WAL file were combined into a new database. After having done that, several queries were executed. At first, the undeleted records could be detected in the WAL file after the messages were sent/received (**Figure 81**). The messages were characterized as "committed". This meant that when the one thousand (1000) commits-checkpoint was reached they would be moved to the main database.

```sql
SELECT
messages._Id AS 'A/A',
wa.wa_contacts.display_name AS 'User Name',
messages.sfIsLive AS 'Live Record',
messages.sfSource AS 'Record Location',
messages.sfPage AS ' Leaf Page',
datetime(messages.timestamp/1000, 'unixepoch', 'localtime') AS 'Message Date',
 CASE messages.KEY_FROM_ME
    WHEN '0' THEN 'Message Received'
    ELSE 'The Message was sent'
END AS 'Message Direction',
messages.DATA AS Message,
CASE messages.media_wa_type
    WHEN '0' THEN 'text'
    WHEN '1' THEN 'image'
    WHEN '2' THEN 'audio'
    WHEN '3' THEN 'video'
    WHEN '4' THEN 'contact card'
    WHEN '5' THEN 'geo location'
END AS 'Message Type'

FROM messages

   LEFT JOIN message_media ON message_media.MESSAGE_URL = messages.MEDIA_URL
   LEFT JOIN wa.wa_contacts ON wa.wa_contacts.jid = messages.KEY_REMOTE_JID
  WHERE messages.key_remote_jid NOT like '-1'  ---WHERE wa.wa_contacts.display_name LIKE 'Μιχάλης'
  ORDER BY messages.timestamp ASC
```

	A/A	User Name	Live Record	Record Location	Leaf Page	Message Date	Message Direction	Message	Message Type
49	166	Μιχάλης	True	DB	250	2021-11-20 12:14:36	The Message was sent	This is message No.76	text
50	167	Μιχάλης	True	DB	250	2021-11-20 12:14:39	The Message was sent	This is message No.77	text
51	168	Μιχάλης	True	DB	250	2021-11-20 12:14:43	The Message was sent	This is message No.78	text
52	169	Μιχάλης	True	DB	250	2021-11-20 12:14:47	The Message was sent	This is message No.79	text
53	170	Μιχάλης	True	DB	250	2021-11-20 12:14:50	The Message was sent	This is message No.80	text
54	171	Μιχάλης	True	DB	250	2021-11-20 12:22:01	The Message was sent	I am sending the photo	text
55	172	Μιχάλης	True	WAL-10-commit	251	2021-11-20 12:22:09	The Message was sent	NULL	image
56	173	Μιχάλης	True	WAL-10-commit	251	2021-11-20 12:22:24	The Message was sent	Did you receive it?	text
57	174	Μιχάλης	True	WAL-10-commit	251	2021-11-20 12:22:43	Message Received	Yes send the second one	text
58	175	Μιχάλης	True	WAL-10-commit	251	2021-11-20 12:22:50	The Message was sent	NULL	image
59	176	Μιχάλης	True	WAL-10-commit	251	2021-11-20 12:23:06	The Message was sent	This will cost you 1000 euros	text
60	176	Μιχάλης	False	WAL-20-commit	251	2021-11-20 12:23:06	The Message was sent	This will cost you 1000 euros	text
61	176	Μιχάλης	False	WAL-21-commit	251	2021-11-20 12:23:06	The Message was sent	This will cost you 1000 euros	text

Figure 81. The undeleted records after transmission using SQL queries in a reconstructed database made from "msgstore.db" and its WAL file (tools used: Forensic Browser for SQLite & DB Browser for SQLite, Examined Device: Samsung A50).

When the messages thread was deleted, and after 2 hours with no activity by the user, it was still feasible to detect them. Their status had been altered ("Live Record Status" and "Record Location" columns) (**Figure 82**).

	A/A	User Name	Live Record	Record Location	Leaf Page	Message Date	Message Direction	Message	Message Type
5	172	Μιχάλης	False	WAL-10-commit	251	2021-11-20 12:22:09	The Message was sent	NULL	image
6	173	Μιχάλης	False	WAL-10-commit	251	2021-11-20 12:22:24	The Message was sent	Did you receive it?	text
7	174	Μιχάλης	False	WAL-10-commit	251	2021-11-20 12:22:43	Message Received	Yes send the second one	text
8	175	Μιχάλης	False	WAL-10-commit	251	2021-11-20 12:22:50	The Message was sent	NULL	image
9	176	Μιχάλης	False	WAL-10-commit	251	2021-11-20 12:23:06	The Message was sent	This will cost you 1000 euros	text
0	177	Μιχάλης	False	WAL-10-commit	251	2021-11-20 12:23:19	Message Received	We agreed on five hundred	text
1	178	Μιχάλης	False	WAL-10-commit	251	2021-11-20 12:23:26	The Message was sent	I wan more	text
2	178	Μιχάλης	False	WAL-66-commit	251	2021-11-20 12:23:26	The Message was sent	I wan more	text
3	178	Μιχάλης	False	WAL-67-commit	251	2021-11-20 12:23:26	The Message was sent	I wan more	text
4	178	Μιχάλης	False	WAL-68-commit	251	2021-11-20 12:23:26	The Message was sent	I wan more	text
5	178	Μιχάλης	False	WAL-75-commit	251	2021-11-20 12:23:26	The Message was sent	I wan more	text
6	179	Μιχάλης	False	WAL-10-commit	251	2021-11-20 12:23:42	Message Received	WE WILL SEE	text
7	180	Μιχάλης	False	WAL-10-commit	251	2021-11-20 12:23:58	The Message was sent	I think they saw me. I must delete those	text
8	180	Μιχάλης	False	WAL-0-commit	251	2021-11-20 12:23:58	The Message was sent	I think they saw me. I must delete those	text
9	180	Μιχάλης	False	WAL-1-commit	251	2021-11-20 12:23:58	The Message was sent	I think they saw me. I must delete those	text
'0	180	Μιχάλης	False	WAL-8-commit	251	2021-11-20 12:23:58	The Message was sent	I think they saw me. I must delete those	text
'1	180	Μιχάλης	False	DB	251	2021-11-20 12:23:58	The Message was sent	I think they saw me. I must delete those	text

Figure 82. How “msgstore.db” and its WAL file were modified after 2 hours with no activity by the user (tools used: Forensic Browser for SQLite & DB Browser for SQLite, Examined Device: Samsung A50).

After several messages were received, sent, read and/or deleted, the status of the in-question messages inside the WAL had been changed (“Record Location” column) (**Figure 83**).

A/A	User Name	Live Record	Record Location	Leaf Page	Message Date	Message Direction	Message	Message Type
175	Μιχάλης	False	WAL-114	251	2021-11-20 12:22:50	The Message was sent	NULL	image
176	Μιχάλης	False	WAL-114	251	2021-11-20 12:23:06	The Message was sent	This will cost you 1000 euros	text
177	Μιχάλης	False	WAL-114	251	2021-11-20 12:23:19	Message Received	We agreed on five hundred	text
178	Μιχάλης	False	WAL-114	251	2021-11-20 12:23:26	The Message was sent	I wan more	text
179	Μιχάλης	False	WAL-114	251	2021-11-20 12:23:42	Message Received	WE WILL SEE	text
180	Μιχάλης	False	WAL-114	251	2021-11-20 12:23:58	The Message was sent	I think they saw me. I must delete those	text
181	Edgar Allan Poe	True	DB	248	2021-11-20 14:05:35	Message Received	This is message No.81	text
182	Edgar Allan Poe	True	DB	248	2021-11-20 14:17:43	Message Received	This is message No.82	text
184	Μιχάλης	True	DB	248	2021-11-20 14:33:05	Message Received	This is message No.83	text
183	Μιχάλης	True	DB	248	2021-11-20 14:33:05	The Message was sent	NULL	text
185	Μιχάλης	True	DB	248	2021-11-20 14:33:08	Message Received	This is message No.84	text
186	Edgar Allan Poe	True	DB	248	2021-11-20 14:44:58	The Message was sent	This is messagge No.85	text
187	Bad Actor 2	True	DB	248	2021-11-20 17:43:34	Message Received	This is message No.86	text
188	Bad Actor 2	True	WAL-82-commit	249	2021-11-20 17:43:58	The Message was sent	This is message No.87	text
188	Bad Actor 2	False	DB	249	2021-11-20 17:43:58	The Message was sent	This is message No.87	text
188	Bad Actor 2	False	WAL-0-commit	249	2021-11-20 17:43:58	The Message was sent	This is message No.87	text
188	Bad Actor 2	False	WAL-98-commit	249	2021-11-20 17:43:58	The Message was sent	This is message No.87	text
189	Μιχάλης	True	WAL-82-commit	249	2021-11-20 18:03:27	The Message was sent	This is message No.88	text
189	Μιχάλης	False	WAL-26-commit	249	2021-11-20 18:03:27	The Message was sent	This is message No.88	text
189	Μιχάλης	False	WAL-27-commit	249	2021-11-20 18:03:27	The Message was sent	This is message No.88	text
189	Μιχάλης	False	WAL-28-commit	249	2021-11-20 18:03:27	The Message was sent	This is message No.88	text
189	Μιχάλης	False	WAL-35-commit	249	2021-11-20 18:03:27	The Message was sent	This is message No.88	text
190	Μιχάλης	True	WAL-82-commit	249	2021-11-20 18:03:44	The Message was sent	This is message No.89	text
190	Μιχάλης	False	WAL-54-commit	249	2021-11-20 18:03:44	The Message was sent	This is message No.89	text
190	Μιχάλης	False	WAL-55-commit	249	2021-11-20 18:03:44	The Message was sent	This is message No.89	text
190	Μιχάλης	False	WAL-56-commit	249	2021-11-20 18:03:44	The Message was sent	This is message No.89	text
190	Μιχάλης	False	WAL-63-commit	249	2021-11-20 18:03:44	The Message was sent	This is message No.89	text
191	Μιχάλης	True	WAL-82-commit	249	2021-11-20 18:03:58	Message Received	This is message No.90	text

Figure 83. The “msgstore.db” and its corresponding WAL file were modified after several messages received/sent/read/deleted (tools used: Forensic Browser for SQLite & DB Browser for SQLite, Examined Device: Samsung A50).

When a second deletion of a chat thread related to the same user was made, the two devices behaved differently because of their WAL file size. As mentioned in paragraph §**5.1.1** the Samsung device’s WAL size was able to retain the in-question messages (**Figure 84**), unlike the LG device where the same messages were removed (**Figure 85**). The difference in size and therefore the capacity for temporarily storing data in the WAL file of the two devices produces varied results. In the Samsung device the deleted records were preserved, even after the exchange of ten more messages than in the LG device. Only after twenty more messages were exchanged the in-question messages were also removed.

One more thing to notice is that in the Samsung device even though the in-question messages were retained (*records 167 to 175*), the was a removal of two more recently deleted records (*records 183 to 185*). So, there was an assumption about WAL file behavior, that confirmed several times. The assumption was that the application chose to remove as little data as possible from its database.

22	155	Edgar Allan Poe	True	WAL-59-commit	249	2021-11-20 11:50:49	Message Received	This is message No.70	text
23	167	Μιχάλης	False	WAL-114	251	2021-11-20 12:19:32	The Message was sent	NULL	imag
24	168	Μιχάλης	False	WAL-114	251	2021-11-20 12:19:44	The Message was sent	Did you receive it?	text
25	169	Μιχάλης	False	WAL-114	251	2021-11-20 12:20:02	Message Received	Yes send the second one	text
26	170	Μιχάλης	False	WAL-114	251	2021-11-20 12:20:10	The Message was sent	NULL	imag
27	171	Μιχάλης	False	WAL-114	251	2021-11-20 12:20:28	The Message was sent	This will cost you 1000 euros	text
28	172	Μιχάλης	False	WAL-114	251	2021-11-20 12:20:38	Message Received	We agreed on five hundred	text
29	173	Μιχάλης	False	WAL-114	251	2021-11-20 12:20:44	The Message was sent	I want more	text
30	174	Μιχάλης	False	WAL-114	251	2021-11-20 12:20:50	Message Received	WE WILL SEE	text
31	175	Μιχάλης	False	WAL-114	251	2021-11-20 12:21:21	The Message was sent	I think thet saw me..I must delete those	text
32	176	Edgar Allan Poe	True	WAL-59-commit	249	2021-11-20 14:05:42	Message Received	This is message No.81	text
33	177	Edgar Allan Poe	True	WAL-59-commit	249	2021-11-20 14:17:35	Message Received	This is message No.82	text
34	181	Edgar Allan Poe	True	WAL-59-commit	249	2021-11-20 14:44:36	The Message was sent	This is message No.85	text
35	182	Bad Actor 1	True	WAL-59-commit	249	2021-11-20 17:43:34	The Message was sent	This is message No.86	text
36	183	Bad Actor 1	True	WAL-59-commit	249	2021-11-20 17:43:58	Message Received	This is message No.87	text
37	187	Edgar Allan Poe	True	WAL-59-commit	249	2021-11-20 19:12:53	Message Received	This is messagge No.91	text
38	188	Edgar Allan Poe	True	WAL-59-commit	249	2021-11-20 19:12:56	Message Received	This is messagge No.92	text
39	189	Edgar Allan Poe	True	WAL-59-commit	249	2021-11-20 19:12:59	Message Received	This is messagge No.93	text
40	190	Edgar Allan Poe	True	WAL-59-commit	249	2021-11-20 19:13:02	Message Received	This is messagge No.94	text
41	191	Edgar Allan Poe	True	WAL-59-commit	249	2021-11-20 19:13:05	Message Received	This is messagge No.95	text
42	193	Μιχάλης	True	WAL-59-commit	249	2021-11-20 19:15:32	Message Received	This is message No.96	text
43	192	Μιχάλης	True	WAL-59-commit	249	2021-11-20 19:15:32	The Message was sent	NULL	text
44	194	Μιχάλης	True	WAL-59-commit	249	2021-11-20 19:15:35	Message Received	This is message No.97	text
45	195	Μιχάλης	True	WAL-59-commit	249	2021-11-20 19:15:37	Message Received	This is message No.98	text
46	196	Μιχάλης	True	WAL-59-commit	249	2021-11-20 19:15:40	Message Received	This is message No.99	text
47	197	Μιχάλης	True	WAL-59-commit	249	2021-11-20 19:15:43	Message Received	This is message No.100	text

Figure 84. How “msgstore.db” and its WAL file are modified after a second deletion of a chat thread. The in-question messages were still retrievable. The messages 83 and 84 were removed (tools used: Forensic Browser for SQLite & DB Browser for SQLite, Examined Device: Samsung A50).

157	Edgar Allan Poe	True	DB	249	2021-11-20 11:51:38	Message Received	This is message No.67	text
158	Edgar Allan Poe	True	DB	249	2021-11-20 11:51:40	Message Received	This is message No.68	text
159	Edgar Allan Poe	True	DB	249	2021-11-20 11:51:43	Message Received	This is message No.69	text
160	Edgar Allan Poe	True	DB	249	2021-11-20 11:51:47	Message Received	This is message No.70	text
181	Edgar Allan Poe	True	DB	249	2021-11-20 14:05:35	Message Received	This is message No.81	text
182	Edgar Allan Poe	True	DB	249	2021-11-20 14:17:43	Message Received	This is message No.82	text
184	Μιχάλης	False	WAL-84-commit	248	2021-11-20 14:33:05	Message Received	This is message No.83	text
183	Μιχάλης	False	WAL-84-commit	248	2021-11-20 14:33:05	The Message was sent	*NULL*	text
185	Μιχάλης	False	WAL-84-commit	248	2021-11-20 14:33:08	Message Received	This is message No.84	text
186	Edgar Allan Poe	True	DB	249	2021-11-20 14:44:58	The Message was sent	This is messagge No.85	text
187	Bad Actor 2	True	DB	249	2021-11-20 17:43:34	Message Received	This is message No.86	text
188	Bad Actor 2	True	DB	249	2021-11-20 17:43:58	The Message was sent	This is message No.87	text
189	Μιχάλης	False	WAL-26-commit	249	2021-11-20 18:03:27	The Message was sent	This is message No.88	text
189	Μιχάλης	False	WAL-27-commit	249	2021-11-20 18:03:27	The Message was sent	This is message No.88	text
189	Μιχάλης	False	WAL-28-commit	249	2021-11-20 18:03:27	The Message was sent	This is message No.88	text
189	Μιχάλης	False	WAL-35-commit	249	2021-11-20 18:03:27	The Message was sent	This is message No.88	text
189	Μιχάλης	False	WAL-67-commit	249	2021-11-20 18:03:27	The Message was sent	This is message No.88	text
190	Μιχάλης	False	WAL-54-commit	249	2021-11-20 18:03:44	The Message was sent	This is message No.89	text
190	Μιχάλης	False	WAL-55-commit	249	2021-11-20 18:03:44	The Message was sent	This is message No.89	text
190	Μιχάλης	False	WAL-56-commit	249	2021-11-20 18:03:44	The Message was sent	This is message No.89	text
190	Μιχάλης	False	WAL-63-commit	249	2021-11-20 18:03:44	The Message was sent	This is message No.89	text
190	Μιχάλης	False	WAL-65-commit	249	2021-11-20 18:03:44	The Message was sent	This is message No.89	text
191	Μιχάλης	False	WAL-82-commit	249	2021-11-20 18:03:58	Message Received	This is message No.90	text

Figure 85. How “msgstore.db” and its WAL file alters after a second deletion of a chat thread. The messages of interest (between 70 and 81) were removed. The deleted messages 83 and 84 were still retrievable (tools used: Forensic Browser for SQLite & DB Browser for SQLite, Examined Device: LG G6).

As demonstrated the size, the type, and the complexity of the database, in comparison with Viber’s one, works in favor of an DFIR team rather against it. The in-question messages were deleted after all the below actions were executed:

i. receiving approximately thirty-three (33) new messages (forty-three (43) at the Samsung device),

ii. reading fourteen (14) messages,

iii. sending two (2) messages and

iv. deleting five (5) additional messages.

These numbers are not absolute since different conditions and user actions could, slightly, modify these results. For example, the reading of an incoming message has a smaller impact (less commits) in the WAL file than creating and sending one. This means that there is a bigger margin for reading incoming messages than sending ones, until a WAL checkpoint is reached and so the in question deleted messages are permanently removed.

5.1.6. WhatsApp Backup

One of WhatsApp application' s feature is the ability to store a local or cloud backup of the user's messages database in an encrypted form. The backups have file names ending with ".cryptNN" extension, where "NN" is the number of the encryption version such as "crypt5", "crypt7", "crypt12" (Ghannam H. A., 2018). The latest is "crypt14". The decryption of the backup requires the usage of the private key which is stored at the path «/data/data/com.whatsapp/files». There are many tools which have the ability to decrypt these backups, but not all of them can decrypt the latest versions. One solution for decrypting the "crypt14" backup version is depicted below (**Figure 86**).

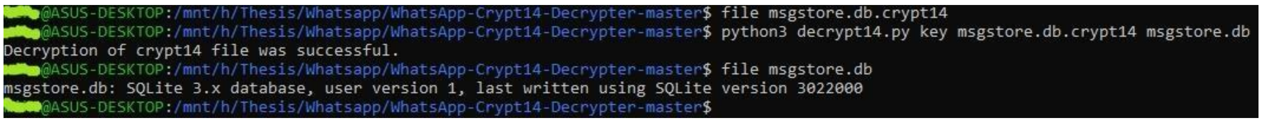


Figure 86. Decrypting a crypt14 backup with free tool «WhatsApp Crypt14 Database Decrypter» (ElDavoo D., 2021).

The backup can be executed either manually by the user or at a time interval. Before the backup the application committed all changes residing in the WAL file to the main database and then executed the "auto_vacuum" PRAGMA statement, leading to the permanent removal of all the deleted records. A more detailed reference on WhatsApp backup was made by Orr & Castro (2018), which includes the usage of a specific tool, named WhatsApp Key/DB Extractor (2016), to extract the key of the device's RAM. Unfortunately, this tool is obsolete and does not support newer versions of Android or WhatsApp. A downgrade procedure of the application is required.

5.2. WhatsApp's Media Files Attachments

The media transmission procedure was the same as it was explained in paragraph §**4.2**. In WhatsApp application when a chat thread including media, is deleted, these media files and their temporary created ones are also erased from the device's filesystem. WhatsApp transmitted files were stored at path: «/media/0/Android/media/com.whatsapp/WhatsApp/Media/WhatsApp Images» (**Figure 87**), while their hashes were stored at path: «/media/0/Android/media/com.whatsapp/WhatsApp/.Shared» (**Figure 88**). There was also a folder at path: «/media/0/Android/media/com.whatsapp/WhatsApp/.trash», where deleted files were located (**Figure 89**). As depicted in below figures, the files are erased and only their record traces could be found.

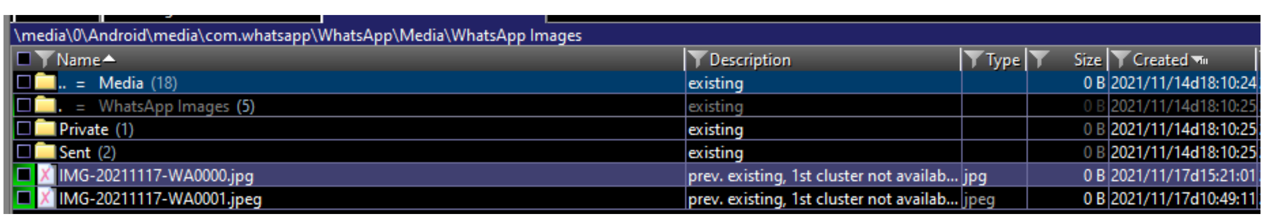


Figure 87. The path that the WhatsApp transmitted files were stored (tool used: X-WAYS Forensic, Examined Device: LG G6).

Figure 88. The path that the WhatsApp transmitted files' hashes were stored (tool used: X-WAYS Forensic, Examined Device: LG G6).

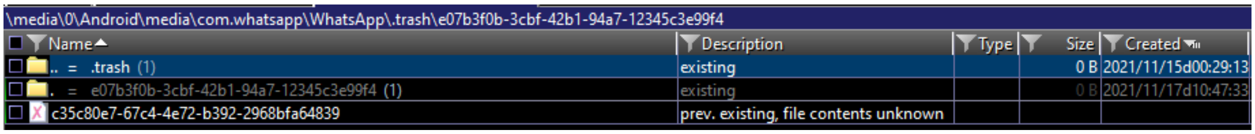


Figure 89. The WhatsApp media files trash folder (tool used: X-WAYS Forensic, Examined Device: LG G6).

5.2.1. Media persistence on WhatsApp Server

Even though the media files cannot be recovered locally, there is a possibility to do it remotely. After their transmission and encryption with a unique key, media are stored at WhatsApp media server to be received by any participant. The duration of the persistence on the server varies. Angus M. (2018) have made the hypothesis that if the encrypted media on the server are to be forwarded from user to user, the deletion timer is being reset. The files can be downloaded via a "wget" script or simply by using a web browser. In the conducted examinations it was possible to download files via WhatsApp server even after 36 hours of the original transmission time (**Figure 90**). As claimed in 2018 when the above paper was published, the media were saved at an unencrypted format. Nowadays they are encrypted with AES-256. The decryption key was found at the "message_media" table under the "media_key" column (**Figure 91**). This column was characterized as blob (thumbnails). By converting the contents of the column to a hex value and with the usage of a python script (i.e., "whatsapp-media-decrypt") the media was able to be retrieved (**Figure 92**). Unfortunately, even though the decryption key could be retrieved, after the deletion of the in-question messages it had a smaller persistence than the other records. The record was wiped after the exchange of just five (5) messages in any case (notice that the other rules about the non-alteration of the database still applied, like reboot, elapsed time, etc).

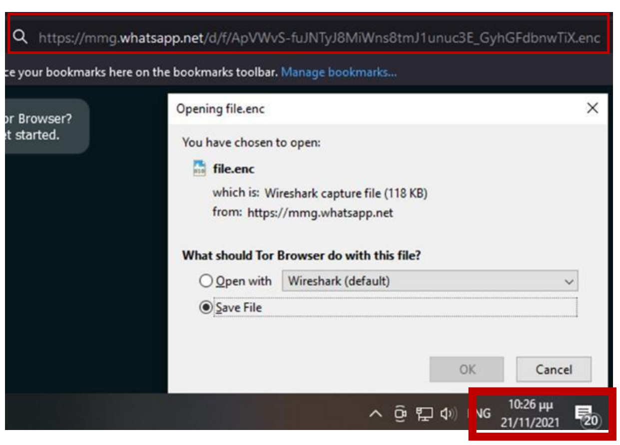


Figure 90. Download transmitted files via WhatsApp server.

Message Date	Whatsapp server media url	Media Decryption key
2021-11-20 10:19:23	NULL	
2021-11-20 10:19:32	https://mmg.whatsapp.net/d/f/ApVWvS-fuJNTyJ8MiWns8tmJ1unuc3E_GyhGFdbnwTiX.enc	8B043E8584A96C67E4E6855ADCCBBFB7CBB462F...
2021-11-20 10:19:44	NULL	
2021-11-20 10:20:02	NULL	
2021-11-20 10:20:10	https://mmg.whatsapp.net/d/f/AkgxHetqP2sU5jF5SNV554yA1gB8z0KK4C6ArDMMLWJE.enc	267D3EC39E5B0696E4E96CB9ED171F4AEB9EFBA...

Figure 91. Media urls and decryption keys were stored on "message_media" table of "msgstore.db".

```
[redacted]@ASUS-DESKTOP:/mnt/h/Thesis/Whatsapp/whatsapp-media-decrypt-main$ python3 decrypt.py -j "8B043E8584A96C67E4E6855ADCCBBFB7CBB462F895ED5D6B9FD91A2EE83BA808" file.enc
Decrypted
[redacted]@ASUS-DESKTOP:/mnt/h/Thesis/Whatsapp/whatsapp-media-decrypt-main$ file file.enc
file.enc: data
[redacted]@ASUS-DESKTOP:/mnt/h/Thesis/Whatsapp/whatsapp-media-decrypt-main$ file file.jpg
file.jpg: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, progressive, precision 8, 960x1280, components 3
[redacted]@ASUS-DESKTOP:/mnt/h/Thesis/Whatsapp/whatsapp-media-decrypt-main$
```

Figure 92. Decrypting downloaded from WhatsApp server media files, by using python script named «whatsapp-media-decrypt».

5.2.2. LG device's media artifacts

In the LG device, even if there was no content of the media that were captured through the WhatsApp application's camera feature, thumbnails of those captured through the device's camera application and then sent via WhatsApp application could still be found and parsed at the path: «media\0\Android\data\com.android.gallery3d\cache\imgcache.0» (**Figure 93**).

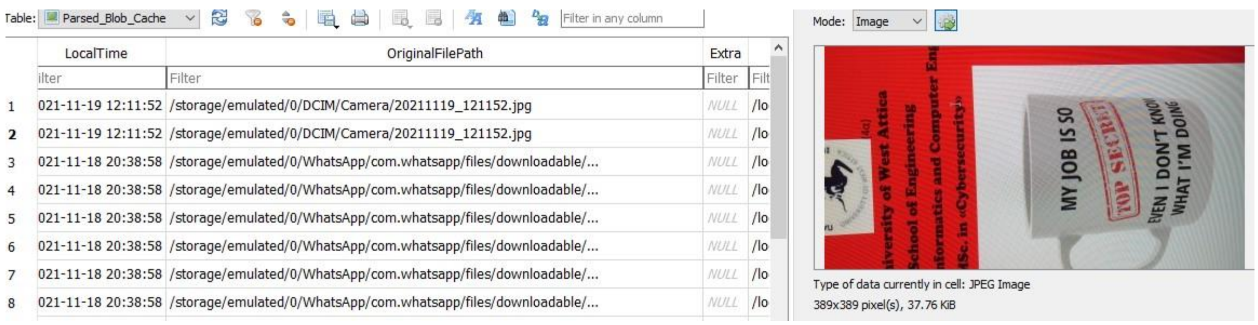


Figure 93. Viewing the "imgcache.sqlite" file (BLOB) (tool used: DB Browser for SQLite, Examined Device: LG G6).

5.2.3. Samsung device's media artifacts

The Samsung device has a modified version of stock android. It does not use the Android's default camera or photo gallery applications, but not the ones that were captured through the WhatsApp application's camera feature. The pictures' thumbnails are located in the Samsung gallery application at path: «/data/com.sec.android.gallery.3d/cache/». As presented in paragraph §**4.2.1** the analysis of the "local" database located at the path: «/data/com.sec.android.gallery.3d/databases/» and specifically the "trash" table could provide the timestamps of media deletion. In "graph.db" database located at «/data/com.samsung.mlp/databases», it was still possible to track the media files which were sent as messages via the WhatsApp application (**Figure 94**).

ID	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8
7	8		1	/storage/emulated/0/Pictures/Viber/IMG-81eac0eb87d2e3e132a5900fcd59593b-V.jpg	65537		2021/11/05 18:35:01	2021/11/05 18:34:28
5	7		1	/storage/emulated/0/Pictures/Viber/IMG-d80dfc7b3d8a0349e93cb1148ff4dbb4-V.jpg	65537		2021/11/05 18:34:04	2021/11/05 18:33:19
1	3		1	/storage/emulated/0/DCIM/Camera/20211031_104226.jpg	65537		2021/10/31 08:43:06	2021/10/31 08:42:26
4	6		1	/storage/emulated/0/Pictures/Viber/IMG-f3e7c3cf3ef69c300ee42d9d0a36efdd-V.jpg	65537		2021/10/31 10:25:04	2021/10/31 10:24:26
3	5		1	/storage/emulated/0/DCIM/Camera/20211031_122307.jpg	65537		2021/10/31 10:23:56	2021/10/31 10:23:07
18	18		1	/storage/emulated/0/Android/media/com.whatsapp/WhatsApp/Media/WhatsApp Images/IMG-20211118-WA0001.jpeg	65537		2021/11/18 21:11:48	2021/11/18 21:11:06
15	17		1	/storage/emulated/0/DCIM/Camera/20211118_230817.jpg	65537		2021/11/18 21:08:58	2021/11/18 21:08:17
14	16		1	/storage/emulated/0/Android/media/com.whatsapp/WhatsApp/Media/WhatsApp Images/IMG-20211117-WA0001.jpeg	65537		2021/11/17 10:50:01	2021/11/17 10:49:11
13	14		1	/storage/emulated/0/DCIM/Screenshots/Screenshot_20211115-194127_One UI Home.jpg	65537		2021/11/15 20:40:26	2021/11/15 17:41:27
12	12		1	/storage/emulated/0/Pictures/Viber/IMG-133563d0e654758e1b2602a36af42559-V.jpg	65537		2021/11/15 13:51:14	2021/11/15 13:50:40
10	11		1	/storage/emulated/0/Pictures/Viber/IMG-71e92ae13d4db3ef81b9e5926049c3f3-V.jpg	65537		2021/11/15 13:49:11	2021/11/15 13:48:22
9	10		1	/storage/emulated/0/Pictures/Viber/IMG-a081b57754a8936d0401434b27bfcd3b-V.jpg	65537		2021/11/10 20:30:29	2021/11/10 10:47:11
8	9		1	/storage/emulated/0/Pictures/Viber/IMG-6759053542d2ed2a449eb370dd09f6a2-V.jpg	65537		2021/11/10 10:44:11	2021/11/10 10:43:43

Figure 94. Exploring “MediaAttribute” table of Samsung “graph.db” (tool used: Oxygen Forensic SQlite, Examined Device: Samsung A50).

5.3. Final Conclusions on WhatsApp

To sum up, it is impossible to avoid the modification of “msgstore” database’ s WAL file, however certain types of actions (e.g., reboot, closing/opening the application) does not seem to alter the data of the database’s critical “messages” table. The usage of the WAL file as a journaling mode provides some leniency to the DFIR Team members, because as the experiments proved it was able to retain a sufficient amount of deleted data. However, there is a time pressure regarding a potential application’s auto backup execution and the deletion of the media files from the media server.

Chapter 6.

Telegram application Scenario

In this chapter the Telegram application's examination and the related results are presented regarding text and media exchanged messages and the volatility of the database in which they are stored. Like in the two previous chapters there is a reference on the application database's location, it's features and the differences between an Android 9 and an Android 11 examined device. After that there is an analysis of the tables and columns of the database and the queries that were built in order to view the desired ones. In addition, the volatility of deleted text and media messages is examined through multiple experiments. Furthermore, some unique characteristics of the application are examined, and final conclusions are drawn.

6.1. Telegram's Examination

The used .apk for both the devices was the Telegram v.7.9.3 with released date in August of 2021. Application' s data, file and folder structure are not as well-known as to other applications. Anglano et al. (2017) have made an excellent job in terms of educational and research spectre. The examined database for this application was the "cache4.db" located at the path: «data/data/org.telegram.messenger/files». In this scenario's experiments more than one thousand and one hundred (1.100) messages were exchanged in total for each device, among four users. The different approach which was taken this time, was related to the creation and existence of freelist pages in the database and the need to study them. This was achieved by creating a large amount of data traffic inside the application's database. The access to these files and folders is normally prohibited to the user, so either the examiner must have root access to the device or acquire a physical image of it.

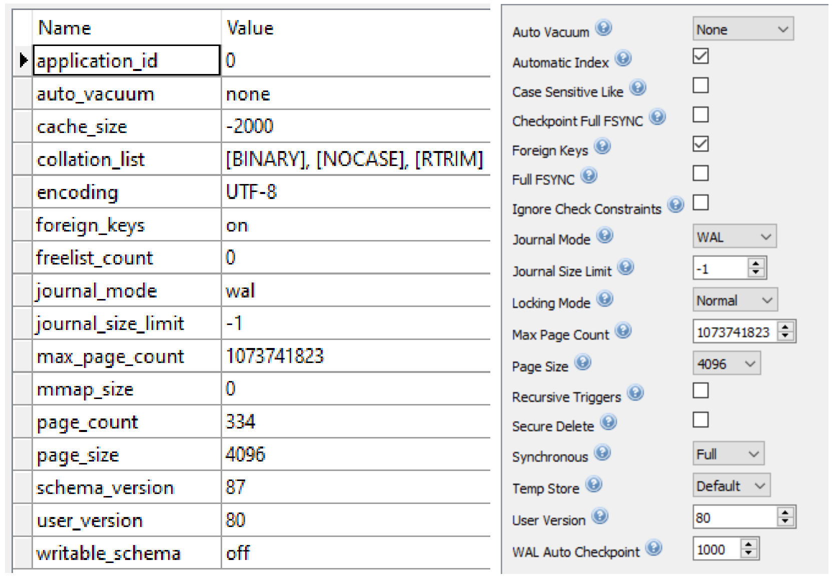
6.1.1. Android 9 vs Android 11 Comparison

There were no differences in the folder structure or the files of the application on both examined devices. In any case the initial size of "cache4.db" was 1.336 kb and its corresponding WAL file was 4.475kb.

6.1.2. General information about the under-examination database

The header and PRAGMA statements of the "cache4" database analysis, by using the non-forensic tools "SQL Expert" and "DB Browser for SQLite" showed that (**Figure 95**):

- ✓ created freelist pages since the auto_vacuum feature was disabled,
- ✓ used WAL as a journaling mode,
- ✓ the WAL Auto Checkpoint had been set at one thousand (1.000) commits (default size).



Name	Value
application_id	0
auto_vacuum	none
cache_size	-2000
collation_list	[BINARY], [NOCASE], [RTRIM]
encoding	UTF-8
foreign_keys	on
freelist_count	0
journal_mode	wal
journal_size_limit	-1
max_page_count	1073741823
mmap_size	0
page_count	334
page_size	4096
schema_version	87
user_version	80
writable_schema	off

Figure 95. Properties of “cache4.db” (tools used: SQLite Expert (left) & DB Browser for SQLite (right)).

6.1.3. Analysis of the “cache4.db”

The “cache4” database was composed by fifty-two (52) tables (**Figure 96**). The six (6) most forensically interesting tables for the examined scenario were the “**dialogs**”, “**enc_chats**”, “**messages**”, “**user_contacts_v7**”, “**user_phones_v7**” and “**users**”.

Among the most interesting columns of the “**dialogs**” table were:

- The “**did**”, contained the dialog id of the secret or normal chat.
- The “**date**”, contained the timestamp of a chat-thread’s last message in Unix Seconds.
- The “**unread count**”, contained information about the unread messages of the dialog.
- The “**last_mid**”, “**inbox_max**” and “**outbox_max**” which in conjunction could clarify the size of the dialog.

Under the “**enc_chats**” table the “**uid**”, “**user**” and “**name**” columns were the most valuable. The records included in these columns had to be compared with some of the records on the “**messages**” table to get information about the participants of a secret chat.

Among the most interesting columns of the “**messages**” table were:

- The “**uid**”, contained the participant of a normal or a secret chat.
- The “**read_state**”, had information about the message’s read status.
- The “**send_state**”, clarified whether the outgoing message was actually sent (i.e., network unavailability).
- The “**date**”, contained the timestamp of the message in Unix Seconds.

- The “**body**” contained the body of the text message, along with several other information, in a binary serialized form. From now on, these types of data will be referred as “TDS” (Telegram Data Structure). This term was introduced by Anglano et al. (2017).
- The “**out**”, contained information about the message direction.
- The “**TTL**”, kept the timer set by the user for autodeleting the message.

The “**user_contacts_v7**”, “**user_phones_v7**” and “**users**” tables contained information about the users participating in chat threads.

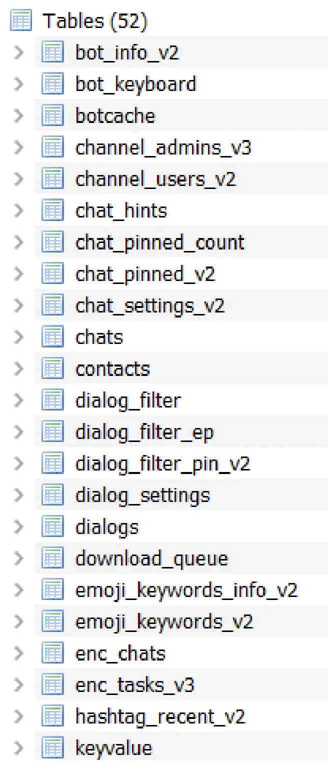


Figure 96. Depiction of the tables tree of “cache4_db”.

The form of data inside the TDSs was an obstacle to preview results from a created SQL query. (**Figure 97**).

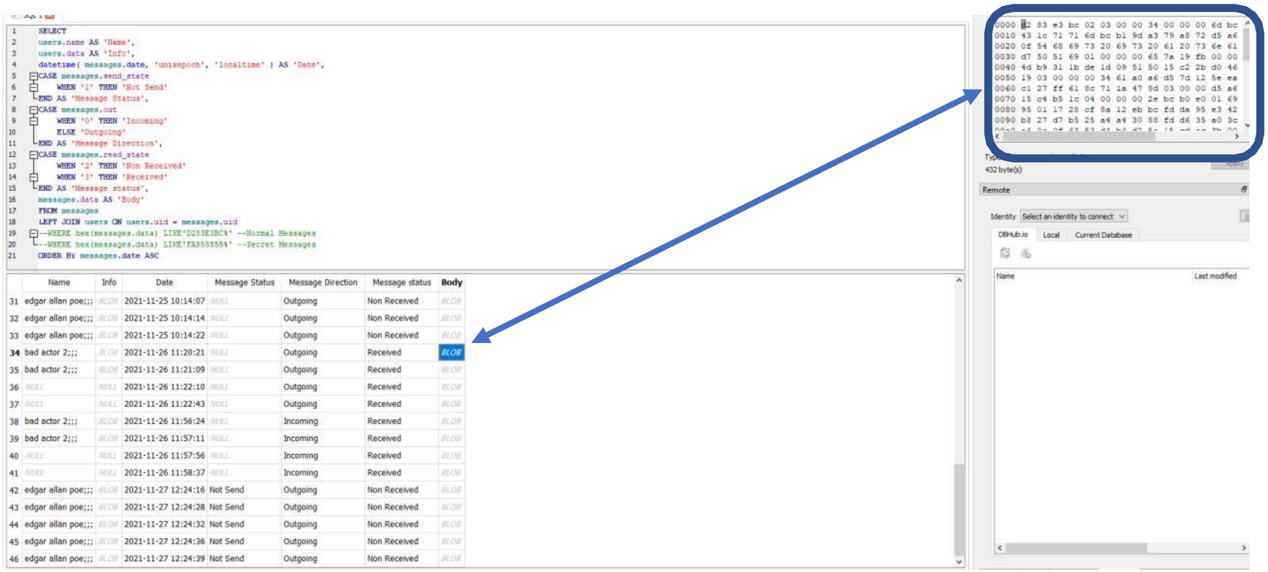


Figure 97. SQL query combining several columns from different tables from "cache4.db".

6.1.3.1 Analysis of TDSs

Anglano et al. (2017) explain that the serialized data are structured in the Telegram Language as composite types (include integers and strings) and defined by a 4-byte integer number in little Indian. Each 4-bytes integer comprise one object type. In April 2020 through the examination of 42.000 lines of Telegram code, 1.340 objects were found (dfirfpi, 2020). Telegram Team adds, removes or redefines these objects by adding or removing application's features. Boiko M. (2018) explains some of these objects, even though many of them have been redefined. Analysis of some of the TDSs and objects found in the examined application's version are presented below throughout examples. In the first example there is an exchange of two messages (one incoming and one outgoing) of a normal chat thread (records 162 and 163). In every TDS specific 4-byte integer (in little Indian) have a specific meaning (**Table 11**).

	Name	Info	Date	Message Status	Message Direction	Message status	Body
162	bad actor 2;;;	BLOB	2021-11-27 23:52:33	NULL	Incoming	Received	BLOB
163	bad actor 2;;;	BLOB	2021-11-27 23:53:31	NULL	Outgoing	Received	BLOB

Records including normal chat messages.

Table: users

	uid	name	status	data
1	1903238211	zugni feyda;;;	1638046912	BLOB
2	1923643811	bad actor 2;;;	-100	BLOB
3	2135101684	μιχάλης;;;	-100	BLOB
4	2138980292	edgar allan poe;;;	-100	BLOB

The "users" table of the "cache4.db". The record 1 (device's user) and record 2 are the participants.

```
0000 d2 83 e3 bc 00 01 00 00 c5 00 00 00 6d bc b1 9d   ............m...
0010 a3 79 a8 72 6d bc b1 9d a3 79 a8 72 a1 a8 a2 61   .y.rm....y.r...a
0020 16 54 68 69 73 20 69 73 20 6d 65 73 73 61 67 65   .This is message
0030 20 4e 6f 2e 31 36 30 00 01 20 00 00               No.160.. ..
```

Objects in record 162 (Normal Incoming Message)	
BCE383D2	Integer flagging this TDS as a normal message between two users
72A879A3	The hex value of the user’s id sending the message in the conversation (1923643811)
72A879A3	The hex value of the user’s id participating in the conversation (1923643811)
61A2A8A1	The timestamp of the message (1638049953) in Unix Seconds

```
0000 d2 83 e3 bc 02 03 00 00 c6 00 00 00 6d bc b1 9d  ............m...
0010 43 1c 71 71 6d bc b1 9d a3 79 a8 72 db a8 a2 61  C.qqm....y.r...a
0020 16 54 68 69 73 20 69 73 20 6d 65 73 73 61 67 65  .This is message
0030 20 4e 6f 2e 31 36 31 00 20 63 ed 3d 01 20 00 00   No.161. c.=. ..
0040
```

Objects in record 163 (Normal Outgoing Message)	
BCE383D2	Integer flagging this TDS as a normal message between two users
71711C43	The hex value of the user’s id sending the message in the conversation (1903238211)
72A879A3	The hex value of the user’s id participating in the conversation (1923643811)
61A2A8DB	The timestamp of the message (1638050011) in Unix Seconds

Table 12. TDSs (Telegram Data Structure) objects in Normal Chat thread

In the second example there is an exchange of two messages (one incoming and one outgoing) of a secret chat thread (records 225 and 226). In every TDS specific 4-byte integer (in little Indian) have a specific meaning (**Table 12**).

225	NULL	NULL	2021-11-29 20:24:26	NULL	Outgoing	Received	BLOB
226	NULL	NULL	2021-11-29 20:24:39	NULL	Incoming	Received	BLOB

Records including secret chat messages.

Table: users

	uid	name	status	data
	Filter	Filter	Filter	Filter
1	1903238211	zugni feyda;;;	1638046912	BLOB
2	1923643811	bad actor 2;;;	-100	BLOB
3	2135101684	μιχάλης;;;	-100	BLOB
4	2138980292	edgar allan poe;;;	-100	BLOB

The “users” table of the “cache4.db”. The record 1 (device’s user) and record 3 are the participants.

```
0000 fa 55 55 55 03 03 00 00 fd ca fc ff 00 00 00 00  .UUU............
0010 43 1c 71 71 6d bc b1 9d f4 10 43 7f da 1a a5 61  C.qqm.....C....a
0020 1d 54 68 69 73 20 77 69 6c 6c 20 63 6f 73 74 20  .This will cost 
0030 79 6f 75 20 31 30 30 30 20 65 75 72 6f 73 00 00  you 1000 euros..
0040 20 63 ed 3d 15 c4 b5 1c 00 00 00 00 00 00 00 00   c.=............
0050
```

Objects in record 225 (Secret Outgoing Message)	
555555FA	Integer flagging this TDS as a secret message between two users
71711C43	The hex value of the user’s id sending the message in the conversation (1903238211)
7F4310F4	The hex value of the user’s id receiving the message in the conversation (2135101684)
61A51ADA	The timestamp of the message (1638210266) in Unix Seconds

```
0000  fa 55 55 55 01 03 00 00 fc ca fc ff 00 00 00 00   .UUU............
0010  f4 10 43 7f 6d bc b1 9d 43 1c 71 71 e7 1a a5 61   ..C.m...C.qq...a
0020  19 57 65 20 61 67 72 65 65 64 20 6f 6e 20 66 69   .We agreed on fi
0030  76 65 20 68 75 6e 64 72 65 64 00 00 20 63 ed 3d   ve hundred.. c.=
0040  15 c4 b5 1c 00 00 00 00 00 00 00 00               ............
```

Objects in record 226 (Secret Incoming Message)

555555FA	Integer flagging this TDS as a secret message between two users
7F4310F4	The hex value of the user's id sending the message in the conversation (2135101684)
71711C43	The hex value of the user's id receiving the message in the conversation (1903238211)
61A51AE7	The timestamp of the message (1638210279) in Unix Seconds

Table 13. TDSs (Telegram Data Structure) objects in Secret Chat thread.

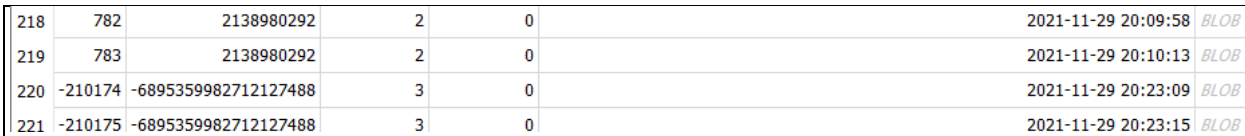
In **Table 13** the objects of an outgoing message containing media file is presented. As occurred the TDS structure had the same logic.

```
0000  fa 55 55 55 03 03 02 00 cb ca fc ff 00 00 00 00   .UUU............
0010  43 1c 71 71 6d bc b1 9d f4 10 43 7f a7 07 a7 61   C.qqm.....C....a
0020  1c 54 68 69 73 20 69 73 20 61 20 67 75 6e 20 6f   .This is a gun o
0030  75 74 67 6f 69 6e 67 20 70 68 6f 74 6f 00 00 00   utgoing photo...
0040  d7 50 51 69 03 00 00 00 65 7a 19 fb 00 00 00 00   .PQi....ez......
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0060  00 00 00 00 a7 07 a7 61 15 c4 b5 1c 02 00 00 00   .......a........
0070  1b b6 bf 77 01 73 00 00 cd c6 7f bc 00 00 00 80   ...w.s..........
0080  ff ff ff ff 8d cb fc ff 5a 00 00 00 32 00 00 00   ........Z...2...
0090  98 06 00 00 1b b6 bf 77 01 79 00 00 54 55 55 55   .......w.y..TUUU
00a0  04 00 00 00 fe 09 00 00 88 8f 38 51 37 91 3c 44   ..........8Q7.<D
00b0  a7 39 51 a2 bc 49 13 c2 20 0a 9b c5 7b 83 8c b7   .9Q..I.. ...{...
00c0  b5 86 29 6e 29 d0 08 9a 95 d5 29 45 55 6a ab ba   ..)n).....)EUj..
00d0  50 e8 3d 3d 7d b6 ca 3b c1 00 00 00 20 9c d6 2e   P.==}..;.... ...
00e0  85 46 12 05 8a 0c 57 41 28 d4 1a 99 63 95 fc a7   .F....WA(...c...
00f0  5e ff a2 8d 33 da 8c ea 42 01 4c b2 f0 00 00 00   ^...3...B.L.....
0100  00 05 00 00 d0 02 00 00 8b 0f 02 00 00 00 00 00   ................
0110  15 c4 b5 1c 00 00 00 00 00 00 00 00 00 00 00 00   ................
0120  ee 7c 7c 6f 72 69 67 69 6e 61 6c 50 61 74 68 7c   .||originalPath|
0130  3d 7c 2f 73 74 6f 72 61 67 65 2f 65 6d 75 6c 61   =|/storage/emula
0140  74 65 64 2f 30 2f 41 6e 64 72 6f 69 64 2f 64 61   ted/0/Android/da
0150  74 61 2f 6f 72 67 2e 74 65 6c 65 67 72 61 6d 2e   ta/org.telegram.
0160  6d 65 73 73 65 6e 67 65 72 2f 63 61 63 68 65 2f   messenger/cache/
0170  49 4d 47 5f 32 30 32 31 31 32 30 31 5f 30 37 32   IMG 20211201 072
0180  36 34 34 5f 35 34 35 2e 6a 70 67 31 30 31 37 30   644 545.jpg10170
0190  30 37 5f 31 36 33 38 33 33 36 34 30 34 30 30 30   07 1638336404000
01a0  7c 7c 66 69 6e 61 6c 7c 3d 7c 31 7c 7c 67 72 6f   ||final|=|1||gro
01b0  75 70 49 64 7c 3d 7c 30 7c 7c 2f 73 74 6f 72 61   upId|=|0||/stora
01c0  67 65 2f 65 6d 75 6c 61 74 65 64 2f 30 2f 41 6e   ge/emulated/0/An
01d0  64 72 6f 69 64 2f 64 61 74 61 2f 6f 72 67 2e 74   droid/data/org.t
01e0  65 6c 65 67 72 61 6d 2e 6d 65 73 73 65 6e 67 65   elegram.messenge
01f0  72 2f 63 61 63 68 65 2f 2d 32 31 34 37 34 38 33   r/cache/-2147483
0200  36 34 38 5f 2d 32 31 30 30 33 36 2e 6a 70 67 00   648 -210036.jpg.
0210
```

555555FA	Integer flagging this TDS as a secret message between two users
71711C43	The hex value of the user's id sending the message in the conversation (1903238211)
7F4310F4	The hex value of the user's id receiving the message in the conversation (2135101684)
61A707A7	The timestamp of the message (1638336423) in Unix Seconds
695150D7	End of the media message caption
The original path of the media file and the temporary path in the cache folder of the application are readable in ASCII characters.	

Table 14. Media file TDS (Telegram Data Structure) objects in Secret Chat thread.

Another interesting artifact was that in secret chats the user participant's "**uid**" in the conversation is hidden, unlike in the normal ones. The way to decode this number is by converting it to a hex value, and then converting the first eight (8) bytes back to decimal.



218	782	2138980292	2	0	2021-11-29 20:09:58	BLOB
219	783	2138980292	2	0	2021-11-29 20:10:13	BLOB
220	-210174	-6895359982712127488	3	0	2021-11-29 20:23:09	BLOB
221	-210175	-6895359982712127488	3	0	2021-11-29 20:23:15	BLOB

Record from the "messages" table where the "uid" of the chat participant is encoded

The value(dec) 6895359982712127488 is converted to (hex) **5FB1 3D5F** 0000 0000

The value (hex) **5FB1 3D5F** is converted back a decimal value 1605451103 and can be located in the "enc_chats" table.

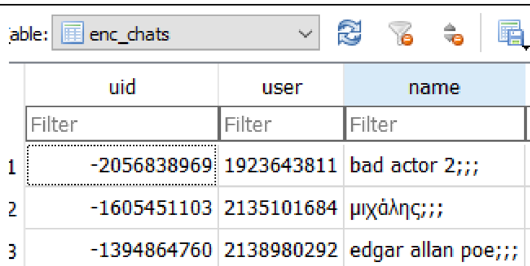


Table: enc_chats

	uid	user	name
1	-2056838969	1923643811	bad actor 2;;;
2	-1605451103	2135101684	μιχάλης;;;
3	-1394864760	2138980292	edgar allan poe;;;

Records from the table "enc_chats". The record 2 indicates the "uid" of the chat's thread participant.

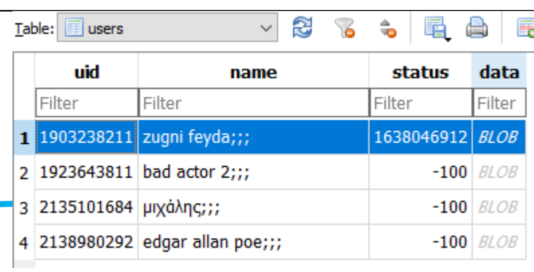


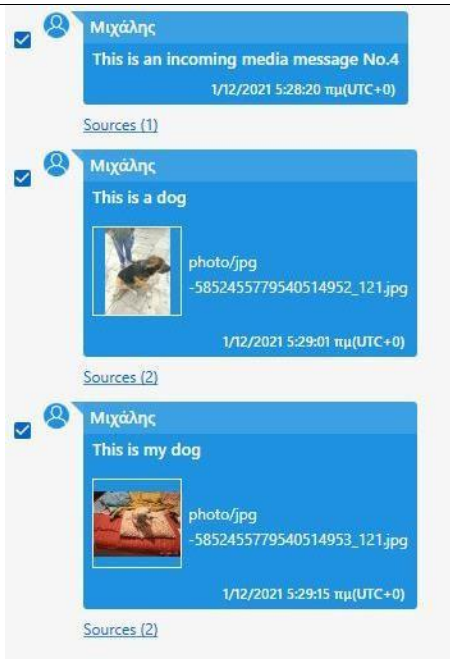
Table: users

	uid	name	status	data
1	1903238211	zugni feyda;;;	1638046912	BLOB
2	1923643811	bad actor 2;;;	-100	BLOB
3	2135101684	μιχάλης;;;	-100	BLOB
4	2138980292	edgar allan poe;;;	-100	BLOB

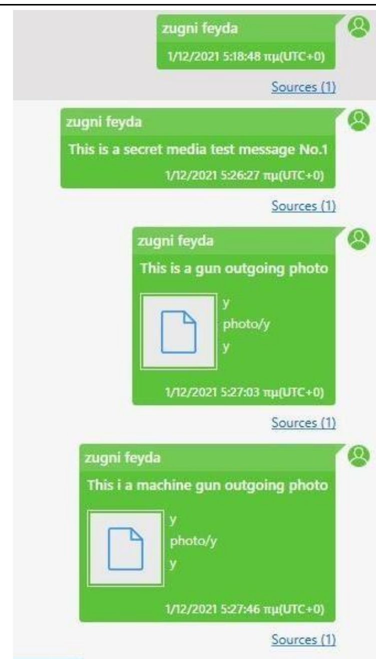
Records from the table "users".

Table 15. How to decode participant' s "uid" from a Telegram's Secret Chat.

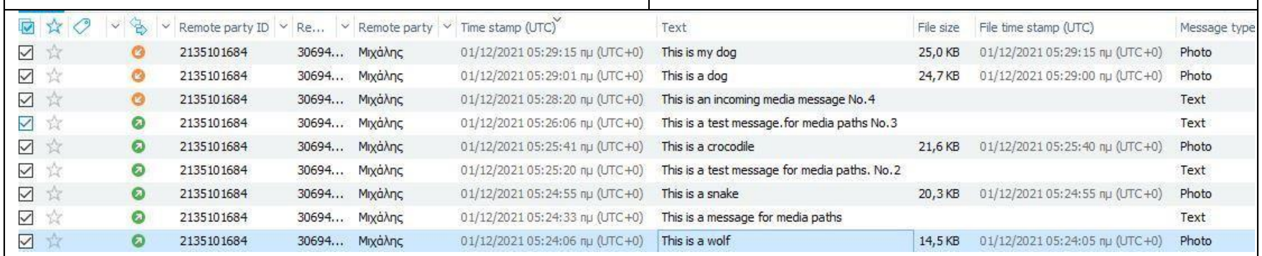
With the volume of objects being added, removed and modified, forensic tools are unable to decode a part of them. A simple example is that the latest versions, of two popular forensic tools, at the time of this Thesis, could decode several types of TDSs.



UFED PA v7.50 decoded normal chat media messages

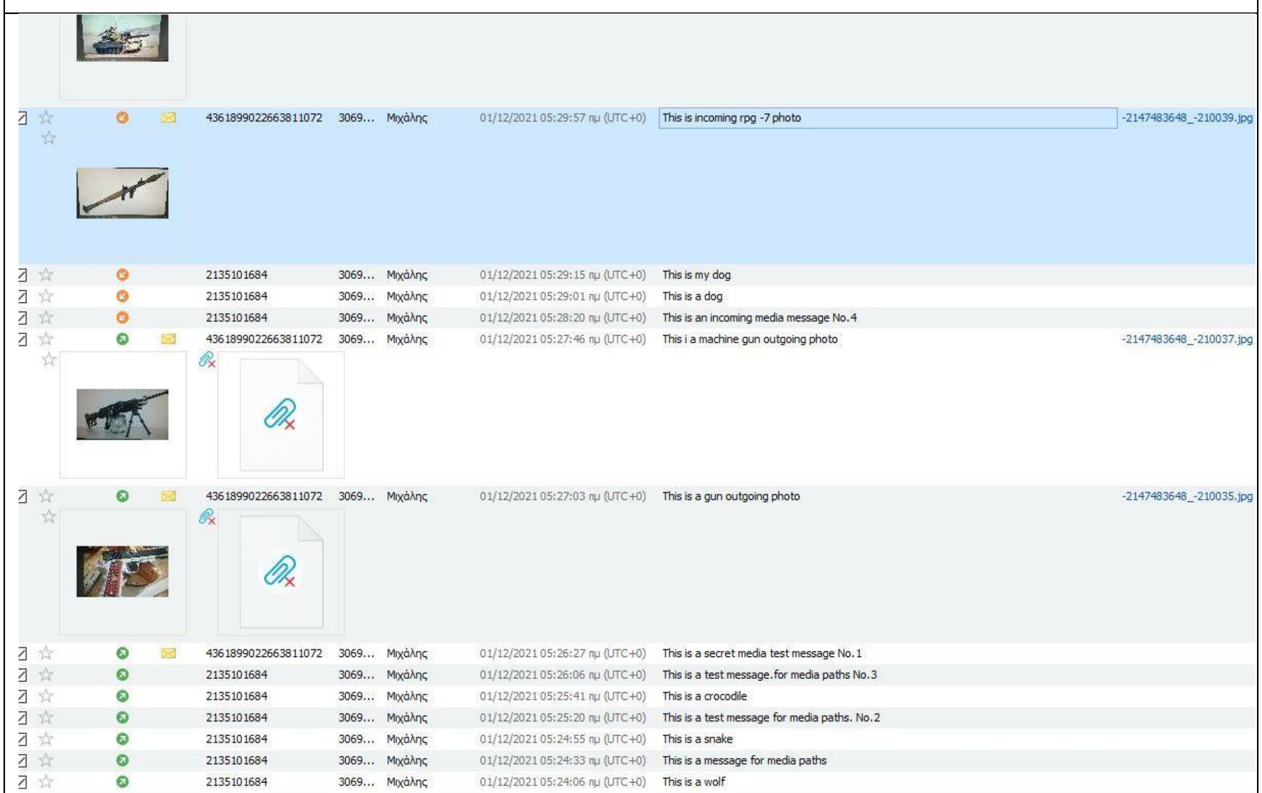


UFED PA v7.50 did not decode secret media messages



Remote party ID	Re...	Remote party	Time stamp (UTC)	Text	File size	File time stamp (UTC)	Message type
2135101684	30694...	Μιχάλης	01/12/2021 05:29:15 πμ (UTC+0)	This is my dog	25,0 KB	01/12/2021 05:29:15 πμ (UTC+0)	Photo
2135101684	30694...	Μιχάλης	01/12/2021 05:29:01 πμ (UTC+0)	This is a dog	24,7 KB	01/12/2021 05:29:00 πμ (UTC+0)	Photo
2135101684	30694...	Μιχάλης	01/12/2021 05:28:20 πμ (UTC+0)	This is an incoming media message No.4			Text
2135101684	30694...	Μιχάλης	01/12/2021 05:26:06 πμ (UTC+0)	This is a test message.for media paths No.3			Text
2135101684	30694...	Μιχάλης	01/12/2021 05:25:41 πμ (UTC+0)	This is a crocodile	21,6 KB	01/12/2021 05:25:40 πμ (UTC+0)	Photo
2135101684	30694...	Μιχάλης	01/12/2021 05:25:20 πμ (UTC+0)	This is a test message for media paths. No.2			Text
2135101684	30694...	Μιχάλης	01/12/2021 05:24:55 πμ (UTC+0)	This is a snake	20,3 KB	01/12/2021 05:24:55 πμ (UTC+0)	Photo
2135101684	30694...	Μιχάλης	01/12/2021 05:24:33 πμ (UTC+0)	This is a message for media paths			Text
2135101684	30694...	Μιχάλης	01/12/2021 05:24:06 πμ (UTC+0)	This is a wolf	14,5 KB	01/12/2021 05:24:05 πμ (UTC+0)	Photo

Oxygen Forensics v14.1 did not decode normal chat media messages



Oxygen Forensics v14.1 decoded secret media messages

Table 16. Comparison of how UFED PA (above) and Oxygen Forensics (below) were able to decode different kind of TDS (Telegram Data Structure)

6.1.4. Volatility of the "cache4.db"

The experiments confirmed that both the "cache4" database and its corresponding WAL file were not susceptible to certain device status changes, which did not seem to alter the data inside the "messages" table. Those actions included:

- ✓ Leaving the device powered on and with network.
- ✓ After the finishing boot sequence of the device and the application auto starts when network was available.
- ✓ At opening of the app after closing it.

Under the following circumstances the "cache4" database was not modified:

- ✓ Leaving the device powered on in airplane mode.
- ✓ Rebooting/turning off the device while it was in airplane mode.
- ✓ At closing the app and removing it from the task manager of the android device.
- ✓ Removing the SIM card (which was the only thing associated with account), while the device is live and after it was deactivated and then powered on without the SIM.
- ✓ Removing notifications of incoming messages.

The database's "messages" table was modified when:

- ! a message was created. It does not matter if it was successfully sent or not.
- ! a message was received.
- ! a message was opened. This was regardless of whether it was an outgoing message which was read by the receiver or an incoming which was read by the user.
- ! a message or a thread was deleted.

6.1.5. Retrieving the messages in question and the timebomb feature

Telegram implements a timebomb/autodelete feature but only in the secret chat section (the E2E encrypted section). The autodelete feature could vary, from one (1) second to a few days. Nevertheless, this feature did not affect the ability to recover this kind of messages, since it did not have any other additional properties except the data in column "TTL", which represents the timer until deletion, in seconds. Every time a message's state changed the whole record was moved inside the WAL file creating a new separate record.

```sql
SELECT
users.name AS 'Name',
messages.UID,
messages.MID AS 'Message ID',
users.data AS 'Info',
datetime( messages.date, 'unixepoch', 'localtime' ) AS 'Date',
CASE messages.SEND_STATE
    WHEN '1' THEN 'The message has not reached Server'
    ELSE 'The message was delivered to the Server'
END AS 'Message Local Status',
CASE messages.out
    WHEN '0' THEN 'Incoming'
    ELSE 'Outgoing'
END AS 'Message Direction',
CASE messages.read_state
    WHEN '2' THEN 'Non Received'
    WHEN '3' THEN 'Received'
END AS 'Message status',
messages.data AS 'Body',
messages.TTL AS 'Timer',
messages.sfSource AS 'Source'
FROM messages
LEFT JOIN users ON users.uid = messages.uid
ORDER BY messages.date ASC
```

Query which was executed

	Name	UID	Message ID	Info	Date	Message Local Status	Message Direction	Message status	Body	Timer	Source
168	NULL	-7424347666923913216	-210225	NULL	2021-11-29 22:44:05	The message has not reached Server	Outgoing	Non Received	BLOB	30	WAL-96-commit
169	NULL	-7424347666923913216	-210225	NULL	2021-11-29 22:44:06	The message was delivered to the Server	Outgoing	Non Received	BLOB	30	WAL-111-commit
170	NULL	-7424347666923913216	-210225	NULL	2021-11-29 22:44:06	The message was delivered to the Server	Outgoing	Received	BLOB	30	WAL-115-commit

The same secret message has three different records inside the WAL

Table 17. Records of a Secret Chat in “cache4” database’s WAL file.

Telegram did not have the full “auto_vacuum” PRAGMA statement enabled and thus it creates freelist pages. As mentioned in paragraph §**1.3.1.4** the existence of freelist pages enables the possibility of retrieving deleted messages. However, even if it had not the “secure_delete” PRAGMA statement enabled it had implemented some kind of failsafe for wiping the deleted records. This was tested multiple times in both devices by deleting hundreds of records, creating freelist pages, and attempting to recover them. In every case, immediately after the deletion, most of the messages were recoverable, but after committing some new ones (incoming or outgoing) all the messages residing in the main database were erased (**Table 17**). This was verified by searching specific messages as a string, in the database and WAL file, with a hex editor.

Name	Value
application_id	0
auto_vacuum	none
cache_size	-2000
collation_list	[BINARY], [NOCASE], [RTRIM]
encoding	UTF-8
foreign_keys	on
freelist_count	0
journal_mode	wal
journal_size_limit	-1
max_page_count	1073741823
mmap_size	0
page_count	334
page_size	4096
schema_version	94
user_version	80
writable_schema	off

Initial state of the database (1.336 kb in size)

```sql
SELECT
messages.sfSource AS 'Source',
count(messages.sfSource)
FROM messages
GROUP by messages.sfSource
ORDER BY count(messages.sfSource) DESC
```

	Source	count(messages.sfSource)
1	DB	198
2	WAL-338-commit	29
3	WAL-218-commit	29
4	WAL-422-commit	28
5	WAL-139	11
6	WAL-449	10
7	WAL-153	7
8	WAL-1002-commit	7
9	WAL-986	1
10	WAL-968	1

Records that were parsed from the database and its WAL file before deletion

Name	Value
application_id	0
auto_vacuum	none
cache_size	-2000
collation_list	[BINARY], [NOCASE], [RTRIM]
encoding	UTF-8
foreign_keys	on
freelist_count	22
journal_mode	wal
journal_size_limit	-1
max_page_count	1073741823
mmap_size	0
page_count	383
page_size	4096
schema_version	94
user_version	80
writable_schema	off

State of the database (1.568 kb in size) before deleting all records

```sql
SELECT
messages.sfSource AS 'Source',
count(messages.sfSource)
FROM messages
GROUP by messages.sfSource
ORDER BY count(messages.sfSource) DESC
```

	Source	count(messages.sfSource)
1	DB	225
2	WAL-170-commit	12
3	WAL-1002-commit	7
4	WAL-416-commit	2
5	WAL-410-commit	2
6	WAL-404-commit	2
7	WAL-392-commit	2
8	WAL-386-commit	2
9	WAL-380-commit	2
10	WAL-374-commit	2

Records that were parsed from the database and its WAL file after deletion

Name	Value
application_id	0
auto_vacuum	none
cache_size	-2000
collation_list	[BINARY], [NOCASE], [RTRIM]
encoding	UTF-8
foreign_keys	on
freelist_count	39
journal_mode	wal
journal_size_limit	-1
max_page_count	1073741823
mmap_size	0
page_count	383
page_size	4096
schema_version	94
user_version	80
writable_schema	off

State of the database (1.568 kb in size) after deleting all records

```sql
SELECT
messages.sfSource AS 'Source',
count(messages.sfSource)
FROM messages
GROUP by messages.sfSource
ORDER BY count(messages.sfSource) DESC
```

	Source	count(messages.sfSource)
1	WAL-224-commit	29
2	WAL-170-commit	29
3	WAL-344-commit	28
4	WAL-1002-commit	23
5	WAL-23	20
6	WAL-486	15
7	WAL-61	11
8	WAL-698	5
9	WAL-64	5
10	DB	5

Records that were parsed from the database and its WAL file after committing new records.

Name	Value
application_id	0
auto_vacuum	none
cache_size	-2000
collation_list	[BINARY], [NOCASE], [RTRIM]
encoding	UTF-8
foreign_keys	on
freelist_count	39
journal_mode	wal
journal_size_limit	-1
max_page_count	1073741823
mmap_size	0
page_count	383
page_size	4096
schema_version	94
user_version	80
writable_schema	off

State of the database (1.568 kb in size) after committing twenty incoming messages

Table 18. The state of “cache4.db” and its WAL file in different conditions.

There was a difference in the gravity of the acts. An outgoing message occupied more records and space in the WAL file than reading an incoming message. The element of luck is also involved. A crucial factor is the timing of the in-question messages deletion. Whether it occurs right after a WAL commit to the database, or before it. This is a parameter purely random and cannot be controlled in realistic situations. In **Table 18** is presented a list of the different experiments which were executed (*Test No*), the prior condition before deleting the in-question messages (*Parameters prior to deletion*), the different actions in which they were still recoverable and are green colored (*Incoming/Outgoing/Opened/Deleted Messages*) and the action which finally led to the removal of the chat thread from the main database or the WAL file and is red coloured (*Parameters prior to deletion*).

Test No	Parameters prior to deletion		Incoming Messages	Outgoing Messages	Messages that were opened	Messages that were deleted	Action that removed the messages in-question
1	None	DELETION OF THE IN-QUESTION CHAT THREAD	15	5	15	None	10 outgoing messages
2	All messages of the database were deleted		50	None	20	None	the rest 30 messages were opened
3	60 messages included on the thread alongside with in-question ones were deleted		50	None	None	80 (From other chat thread)	50 messages were opened
4	None		20		60	None	10 incoming messages
5	WAL commit was forced[7]		20	15	None	None	40 messages were opened
6	All messages of the database were deleted, and WAL commit was forced		70	None	None	None	70 read
7	None		50	None	None	None	70 read

Table 19. Progress of experiments’ results on “cache4.db” and its WAL file.

6.1.6. Wiped Databases

In Telegram there were two occasions where no records of the database were recoverable. **i**. When the user logged out of the device, triggering a complete database’s wipe and **ii**. By clearing the local database from the settings menu. In any case the difference between the timestamp of “cache4.db” and the Android’ OS usage history application, might be a strong indicator that one of the above occasions have occurred.

6.2. Telegram’s Media Files Attachments

The media transmission procedure was the same as it was explained in paragraph §**4.2**. Telegram administrated the shared media files, in a different way based on the type of the chat thread (normal/secret).

[7] The database was monitored in real time by exchanging single messages, until a WAL commit occurred.

6.2.1. Normal chat media transmission

In a normal chat thread, when the in-app camera feature was used, the media files were saved to the following paths:

- media\0\Pictures\Telegram. The files stored by their captured file name (e.g., «IMG_20211125_102653_487.jpg») and
- media\0\Telegram\Telegram Images. The files stored by their, by the transmitted, via the Telegram server, name (e.g., «-5834547226205076084_121.jpg»).

After a media file deletion some artifacts which have remained on the android operating system were utilized to retrieve its information. To be more accurate this was done by comparing those artifacts with the records located in "cache4" database. Those artifacts' paths were analyzed in previous corresponding paragraphs (§**4.2** and §**5.2**), for each device. Samsung device's path was «data\com.sec.android.gallery3d\cache», and the LG device's paths were: **i.** «media\0\DCIM\.thumbnails» and **ii.** «media\0\Android\data\com.android.gallery3d\cache\imgcache.0». An example of the latter is presented in **Table 19**.

MID	UID	READ_STATE	SEND_STATE	DATE	DATA	OUT
Filter	Filter	Filter	Filter	Filter	Filter	Filter
49	1923643811	2	0	2021-11-25 10:27:00	BLOB	1

Record in the "cache4.db"

```
0000 d2 83 e3 bc 02 03 02 00 97 cb fc ff 6d bc b1 9d  ............m...
0010 43 1c 71 71 6d bc b1 9d a3 79 a8 72 d4 48 9f 61  C.qqm....y.r.H.a
0020 00 00 00 00 d7 50 51 69 03 00 00 00 65 7a 19 fb  .....PQi....ez..
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040 00 00 00 00 00 00 00 00 d4 48 9f 61 15 c4 b5 1c  .........H.a....
0050 02 00 00 00 1b b6 bf 77 01 73 00 00 cd c6 7f bc  .......w.s......
0060 00 00 00 80 ff ff ff ff b0 cb fc ff 5a 00 00 00  ............Z...
0070 32 00 00 00 04 07 00 00 1b b6 bf 77 01 79 00 00  2..........w.y..
0080 cd c6 7f bc 00 00 00 80 ff ff ff ff af cb fc ff  ................
0090 00 05 00 00 d0 02 00 00 a4 8e 02 00 00 00 00 00  ................
00a0 00 00 00 00 00 00 00 00 d6 7c 7c 6f 72 69 67 69  .........||origi
00b0 6e 61 6c 50 61 74 68 7c 3d 7c 2f 73 74 6f 72 61  nalPath|=|/stora
00c0 67 65 2f 65 6d 75 6c 61 74 65 64 2f 30 2f 50 69  ge/emulated/0/Pi
00d0 63 74 75 72 65 73 2f 54 65 6c 65 67 72 61 6d 2f  ctures/Telegram/
00e0 49 4d 47 5f 32 30 32 31 31 31 32 35 5f 31 30 32  IMG_20211125_102
00f0 36 35 33 5f 34 38 37 2e 6a 70 67 31 31 30 39 36  653_487.jpg11096
0100 38 35 5f 31 36 33 37 38 32 38 38 31 33 30 30 30  85_1637828813000
0110 7c 7c 66 69 6e 61 6c 7c 3d 7c 31 7c 7c 67 72 6f  ||final|=|1||gro
0120 75 70 49 64 7c 3d 7c 30 7c 7c 2f 73 74 6f 72 61  upId|=|0||/stora
0130 67 65 2f 65 6d 75 6c 61 74 65 64 2f 30 2f 41 6e  ge/emulated/0/An
0140 64 72 6f 69 64 2f 64 61 74 61 2f 6f 72 67 2e 74  droid/data/org.t
0150 65 6c 65 67 72 61 6d 2e 6d 65 73 73 65 6e 67 65  elegram.messenge
0160 72 2f 63 61 63 68 65 2f 2d 32 31 34 37 34 38 33  r/cache/-2147483
0170 36 34 38 5f 2d 32 31 30 30 30 31 2e 6a 70 67 00  648_-210001.jpg.
0180
```

Contents of the TDS

1.	a normal media message is sent
2.	by the user 1903238211
3.	to the user 1923643811
4.	at 1637828820
5.	where media file's name is IMG 20211125102653487

UTC	LocalTime ▲1	OriginalFilePath	Extra	InternalPath	Thumbnail
Filter	Filter	Filter	Filter	Filter	Filter
2021-11-25 08:26:53	2021-11-25 10:26:53	/storage/emulated/0/Pictures/Telegram/IMG_20211125_102653_487.jpg	NULL	/local/image/item/4405	BLOB
2021-11-19 10:14:15	2021-11-19 12:14:15	/storage/emulated/0/WhatsApp/Media/WhatsApp Images/IMG-20211119-WA0001.jpeg	NULL	/local/image/item/2255	BLOB
2021-11-19 10:14:15	2021-11-19 12:14:15	/storage/emulated/0/WhatsApp/Media/WhatsApp Images/IMG-20211119-WA0001.jpeg	NULL	/local/image/item/2255	BLOB
2021-11-19 10:11:52	2021-11-19 12:11:52	/storage/emulated/0/DCIM/Camera/20211119_121152.jpg	NULL	/local/image/item/2254	BLOB
2021-11-19 10:11:52	2021-11-19 12:11:52	/storage/emulated/0/DCIM/Camera/20211119_121152.jpg	NULL	/local/image/item/2254	BLOB
2021-11-18 18:38:58	2021-11-18 20:38:58	/storage/emulated/0/WhatsApp/com.whatsapp/files/downloadable/...	NULL	/local/image/item/2553	BLOB
2021-11-18 18:38:58	2021-11-18 20:38:58	/storage/emulated/0/WhatsApp/com.whatsapp/files/downloadable/...	NULL	/local/image/item/2553	BLOB

Viewing the "imgcache.sqlite" file (BLOB)

Table 20. Correlation of a sent image file's record in "cache4.db" with "imgcache.sqlite" file (tool used: DB Browser for SQLite, Examined Device: LG G6).

6.2.2. Secret chat media transmission

In a secret chat when the media were captured with in-app camera feature never resided outside the cache folder of the application. After deleting the secret chat messages, it was impossible to retrieve/determine the content of the files (attachments) which were sent. During the experiments, it was observed that the corresponding thumbnail files had always a minus one in its file name values than the ones in the database. For instance, the thumbnail of the transmitted media named "-2147483648_-2100**2**.jpg", but in the database record its name was "-2147483648_-2100**3**.jpg". This was a recurring pattern, but it was not decoded.

1611	-210029	-3733244930131230 72	3	0		2021-11-25 10:28:00	BLOB	1	0

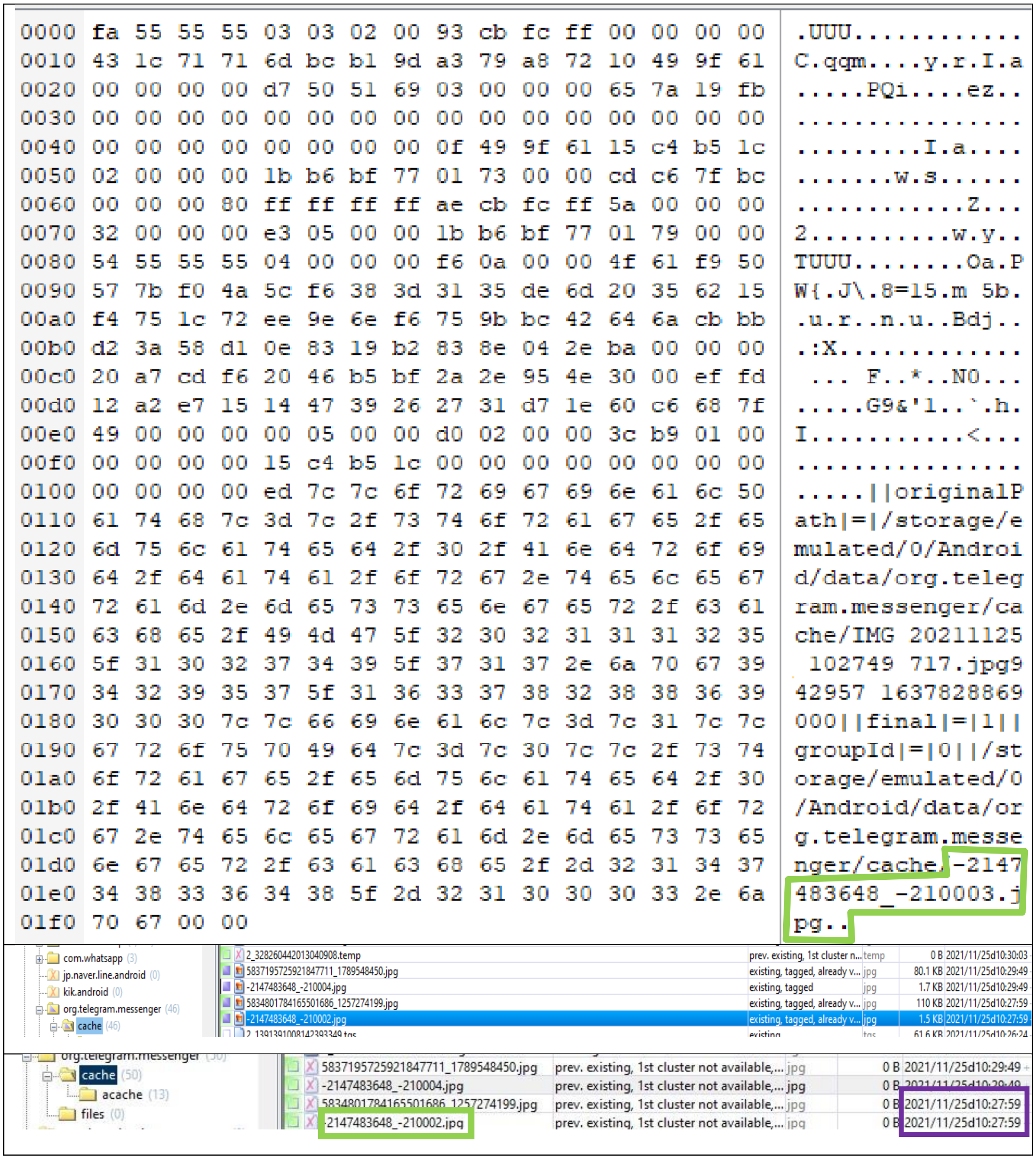
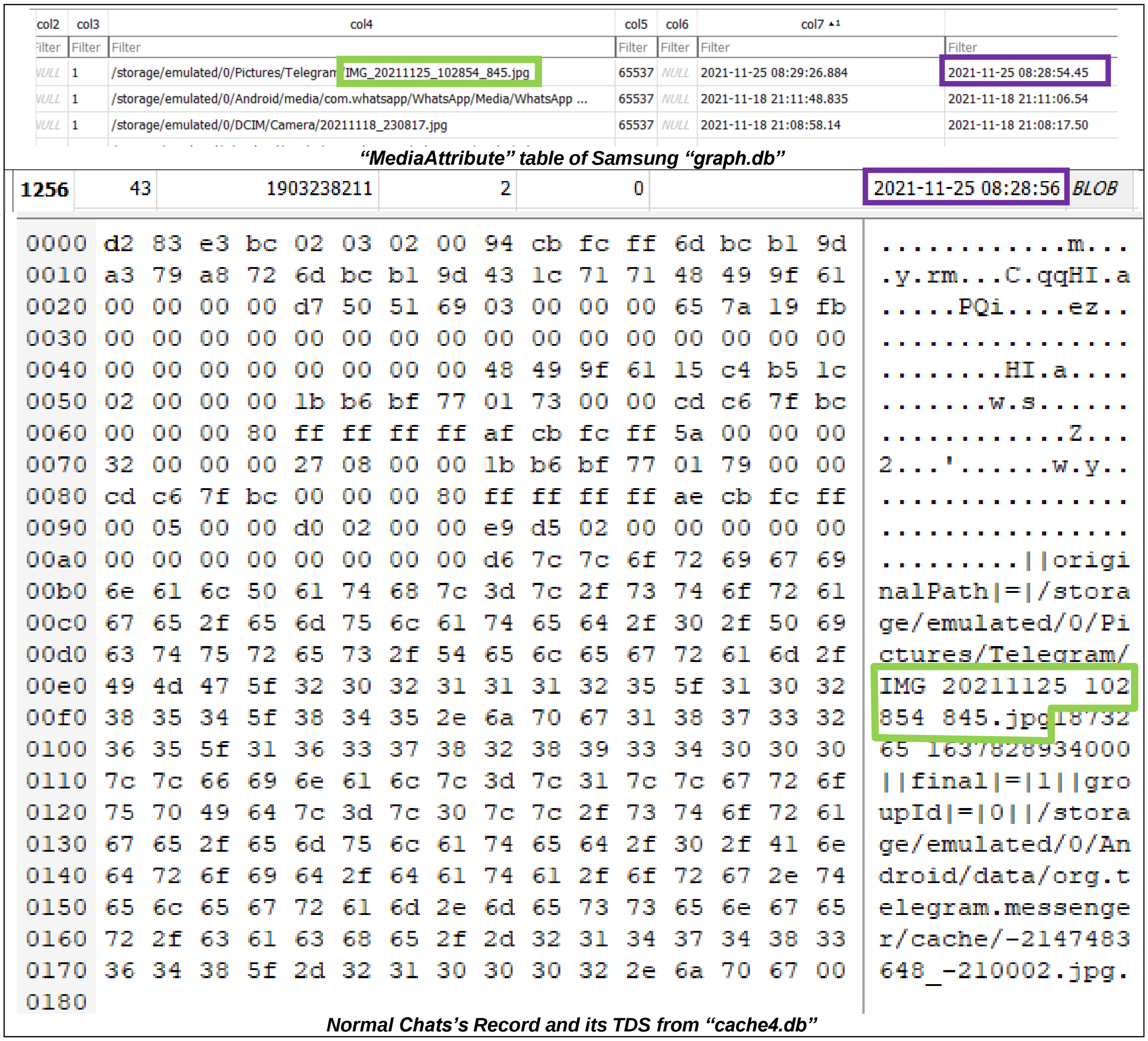


Table 21. Detection of a sent image file in “cache4.db” from a Secret chat thread.

One last interesting fact was that the secret media transmission was the only one that was not recorded in the “graph.db” located in the path «/data/com.samsung.mlp/databases» of the Samsung device. On the contrary the media sent as normal chat message could be retrieved from the “graph” database (**Table 21**).



"MediaAttribute" table of Samsung "graph.db"

Normal Chats's Record and its TDS from "cache4.db"

Table 22. Retrieving artifacts from the "graph.db" for media transfers of Telegram's Normal Chat thread (tool used: DB Browser for SQLite, Examined Device: Samsung A50).

6.2.3. Final Conclusions on Telegram

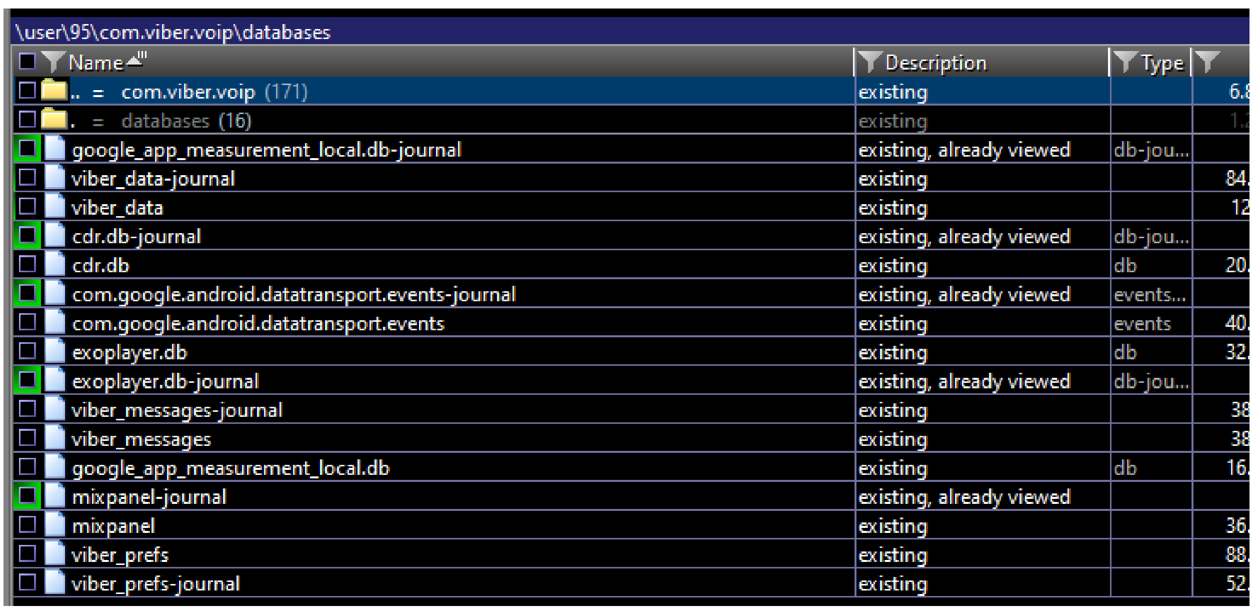
To sum up, certain types of actions modified the "cache4.db" database's WAL file but did not seem to alter data of the database's crucial "messages" table. The usage of the WAL file as journaling mode and its size, almost 5mb, provides a sufficient level of leniency to the DFIR Team members, because as the experiments proved it was able to retain a vast amount of deleted data. In the case of shared media files via an encrypted (secret) chat it was impossible to retrieve any content of them.

Chapter 7.

Dual Applications Feature

In this chapter usage of android's dual application feature was examined. This feature involves the creation of different accounts in the same device's application. It is branded with different names depending on the device like "Dual apps" for LG or "Dual Messenger" for Samsung (**Figure 98**). When this feature is enabled a second user is created in the filesystem with individual folders and files and separate read/execute/write permissions (**Figures 99** and **100**). Specifically for the creation of a Viber's second user's profile it is not mandatory to have a second SIM installed on the device. The second user's profile verification can be done through a separate device.

Figure 98. Use of dual Viber application. (Device used: Samsung A50)



\user\95\com.viber.voip\databases

Name	Description	Type	
.. = com.viber.voip (171)	existing		6.8
. = databases (16)	existing		1.2
google_app_measurement_local.db-journal	existing, already viewed	db-jou...	
viber_data-journal	existing		84.
viber_data	existing		12
cdr.db-journal	existing, already viewed	db-jou...	
cdr.db	existing	db	20.
com.google.android.datatransport.events-journal	existing, already viewed	events...	
com.google.android.datatransport.events	existing	events	40.
exoplayer.db	existing	db	32.
exoplayer.db-journal	existing, already viewed	db-jou...	
viber_messages-journal	existing		38
viber_messages	existing		38
google_app_measurement_local.db	existing	db	16.
mixpanel-journal	existing, already viewed		
mixpanel	existing		36.
viber_prefs	existing		88.
viber_prefs-journal	existing		52.

Figure 99. Depiction of second Viber account's separate databases (path: «/user/95/com.viber.voip/databases», Examined Device: Samsung A50).

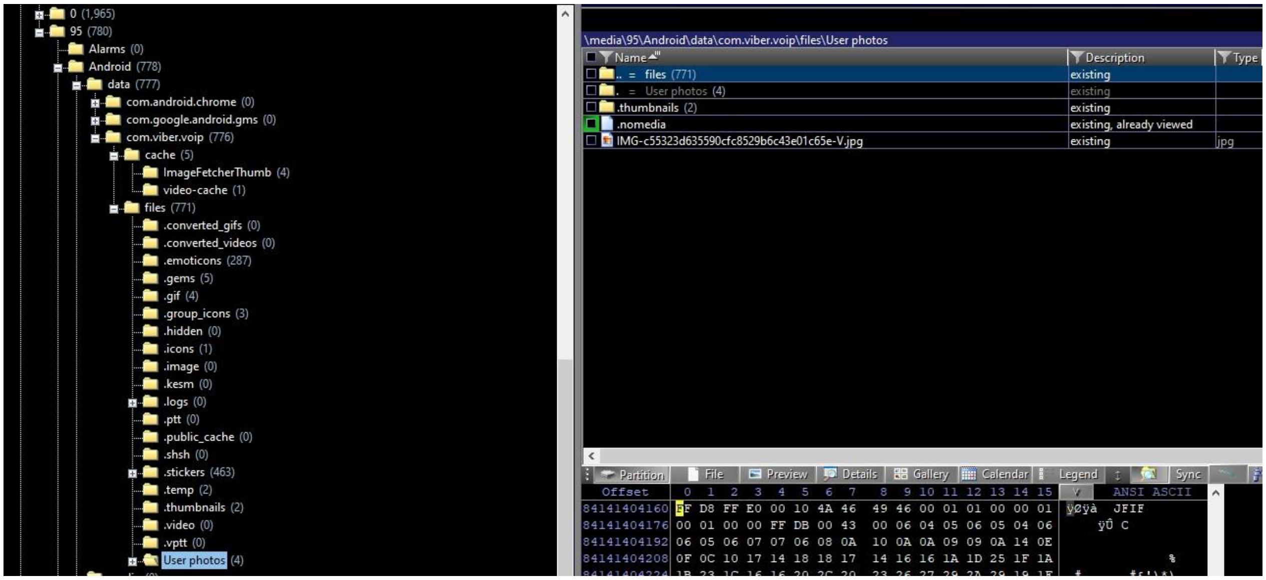


Figure 100. The second Viber user (95) has a separate folder structure for media located in «/media/95/Adroid/data/com.viber.voip/file/User photos». The original user of the device is located in folder (0) (Examined Device: Samsung A50).

The feature adds a new layer of complexity and difficulty to the investigation for two reasons. Firstly, if the user logs in from another device (e.g., the one with the SIM associated with the account) the investigator can only see the initial login screen of the application's second version. This action does not remove the user's data from the filesystem. A second reason is that if the user disables the application's second profile from the device's system settings, the corresponding folder structure is deleted, while the second profile's folder remains. This would be an indicator that additional examination like carving the acquired image for deleted databases must be done.

Chapter 8.

Applications Comparison

In this chapter the results of every examined instant messaging application are summarized and compered, in order to gain an overall view.

The volatility of each application's data in different conditions is presented in **Table 23**. The crucial data is never altered by the elapsed time or the changes at the device's status. The interaction with system notifications affected vital data only in the Viber application. Any message creation, reading and deletion affected all examined applications. WhatsApp and Telegram had some unique features (e.g., WhatsApp local backup) which might modify application's data. Regarding exchanged media files, Viber leaved significant number of artifacts on the device's internal storage. The same applied to certain occasions in Telegram (on normal chat messages). In case of WhatsApp application there was also a possibility of retrieving exchanged media files from its cloud server.

Applications	Time elapsed	Device's status	Interacting with Notifications	Message (Read/Sent/Delete)	Unique features	Media Retrieval from device's storage	Media Retrieval from app's Servers
Viber	✗	✗	✓	✓	✗	✓	✗
WhatsApp	✓*	✓*	✗	✓	✓	✗	✓
Telegram	✗	✓*	✗	✓	✓	✓**	✗

Table 23. The volatility of each application in different conditions.

The usage of rollback journal as a journaling mode in Viber application rendered the data extremely volatile. After exchanging a few messages all the in-question records (deleted messages) were erased. On the other hand, the usage of the WAL as a journaling mode in WhatsApp and Telegram applications, resulted in the preservation of the in-question records after multiple actions (exchange messages/read/delete). The Telegram WAL's file size which was ten times bigger compared to the same file of the WhatsApp application, could retain a larger amount of data, including deleted ones.

All in all, the Viber application is the most volatile one related to retrieving deleted messages records and the least one when it comes to retrieving deleted media files. Telegram application is the least volatile in respect of retrieving deleted messages records, however the recovery of artifacts as for deleted media files depends on the chat thread type (normal/secret). The WhatsApp is the only application that even there were no traces left of a deleted exchanged media file on the examined device, it was possible to retrieve it from its Server.

* Not crucial data.

** Depending on the message's type (normal/secret).

Conclusions, Challenges & Future Work

To sum everything up, throughout the experiments done in this thesis several conclusions could be drawn about the most acceptable and forensic-sound first steps while handling an incident relating to instant messaging applications on an android-based mobile device. First and foremost, the seizure of the device must happen as soon as possible. By leaving the device at the suspect's possession there is a certainty that more messages will be exchanged and thus application's data will be further modified. There is always the element of chance as an unforeseen factor, such as multiple incoming messages, unexpectedly modifying irreversibly vital data (i.e., erase in-question deleted records). Once the device is seized the very first step is to block the incoming data, which will lead to the modification of database's volatile data and hence a potential key evidence removal. This could be achieved by disconnecting the device from the network (e.g., airplane mode, faraday bag or shutdown). The successful recovery of deleted content (messages/media) is determined by two key factor categories.

- The first category is related to IR team's actions regarding the incident. These actions are:
  - Block the device's network connections. The used method is irrelevant (deactivate the device, airplane mode).
  - Not remove any devices notification messages.
  - Not try to create new messages. There are reported occasions in which members of the IR team tried to lure the third-party participants into revealing their identity by exchanges messages.
  - Not read unread messages.

As the experiments showed, once the aforementioned policies are followed the elapsed time does not modify the crucial data. The same applies to device's different statuses (powered on/off, reboot, plugged in, etc.).

- The second category is related to IR team's followed methodology regarding the evidence (data) forensic examination. This methodology must include:
  - Validation of the result. There is no known forensic tool able to decrypt and examine all chat messaging application databases. So, the results must be confirmed and validated by using multiple forensic tools. There are instances where the validation is feasible by diving into the raw data and examining the files in their hexadecimal representation.
  - Experimentation and Analysis. During the development of application versions different database schema/pragma statements and folder structure is implemented. The evidence's specific installed and running application version should be examined through research and development to test the validity of the results.

From the scope of digital evidence forensics, there are always three main challenges on two different areas. From a legal standpoint there are obstacles of bureaucracy which emerge and delay the technical procedures of starting the seizure and the examination of the suspect device. From a technical standpoint there are two main problems. On one hand there is the device's acquisition, which should be a physical one in order to access most of its data. Such acquisition is not always supported all devices' vendors and models. On the other hand, there is the decoding/parsing of the volatile database files is not always up to date or not even supported by forensic tool. It is more than necessary to have updated digital forensic tools and simplify the IR-related legislation.

As future work, more android applications should be examined to find out if there are more volatile data on their databases and how react in different occasions.

References

Albrecht, M. R., Mareková, L., Paterson, K. G. & Stepanovs, I. (2021), Four Attacks and a Proof for Telegram, In 43rd IEEE Symposium on Security and Privacy (IEEE S&P 2022), San Francisco, CA, USA, 22 – 26/05/2022, deposited on: 21/09/2021, DOI: 10.3929/ethz-b-000506353.

Anglano, C., Canonico, M. & Guazzone, M. (2017), Forensic analysis of telegram messenger on android smartphones, Digital Investigation, vol. 23, pp. 31-49, published on: 20/09/2017, DOI: 10.1016/j.diin.2017.09.002.

Anglano, C., Canonico, M., & Guazzone, M. (2016), Forensic analysis of the ChatSecure instant messaging application on android smartphones, Digital Investigation, vol 19, 2016, pp. 44-59, published on: 15/10/2016, DOI: 10.1016/j.diin.2016.10.001, last accessed 13/11/2021.

Anglano, C. (2014), Forensic Analysis of WhatsApp Messenger on Android Smartphones, Digital Investigation, vol. 11(3), pp. 201-213, published on: 27/05/2014, DOI: 10.1016/j.diin.2014.04.003.

Angus, M. (2018), WhatsApp server-side media persistence, Digital Investigation, vol. 25, pp. 114-115, published on: 25/04/2018, DOI: 10.1016/j.diin.2018.04.005.

Bair, J. (2017), Seeking the Truth from Mobile Evidence, Basic Fundamentals, Intermediate and Advanced Overview of Current Mobile Forensic Investigations, Elsevier, London, UK, published on: 13/11/2017, ISBN: 978-0-12-811056-0.

Botha, J., Van 't Wout, C. & Leenen, L. (2019), Proceedings of the 18th European Conference on Cyber Warfare and Security (ECCWS 2019), Hosted by University of Coimbra, Portugal, 4-5/7/2019, Academic Conferences and Publishing International Limited Reading, UK, pp. 55-62, ISBN: 978-1-912764-28-0.

SWGDE (2014), Digital and Multimedia Evidence (Digital Forensics) as a Forensic Science Discipline, Scientific Working Group on Digital Evidence, published on: 05/09/2014, retrieved from: https://drive.google.com/file/d/1OBux0n7VZQe7HSgObwAtmhz5LgwvX0oY/view, last accessed 31/01/2022.

Easttom, C. (2021), An In-Depth Guide to Mobile Device Forensics, CRC Press, Boca Raton, Florida, published on: 22/10/2021, DOI: 10.1201/9781003118718.

Easttom, C. (2021), Digital forensics, investigation, and response Fourth Edition, Information Systems Security & Assurance Series, Jones & Bartlett Learning, Burlington, Massachusetts, USA, published on 24/08/2021, ISBN 9781284226065.

Ghannam, H. A. (2018), Forensic Analysis of Artifacts of Giant Instant Messaging “WhatsApp” in Android Smartphone. Journal of Applied Information, Communication and Technology, vol.5(2), pp. 63-72, published on: 03/10/2018, DOI: 10.33555/ejaict.v5i2.55.

Gogolin, G. (2021), Digital Forensics Explained Second Edition, CRC Press, Boca Raton, Florida, published on: 12/04/2021, ISBN 9780367503437.

Jakobsen, J. & Orlandi, C. (2016), On the CCA (in) security of MTProto, In Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 113-116, published on: 10/2016, retrieved from URL: https://eprint.iacr.org/2015/1177.pdf.

Lee, J., Choi, R., Kim, S. & Kim, K. (2017), Security analysis of end-to-end encryption in Telegram, 2017 Symposium on Cryptography and Information Security (SCIS 2017), Naha, Japan, The Institute of Electronics, Information and Communication Engineers, published on: 01/2017, retrieved from URL: https://caislab.kaist.ac.kr/publication/paper_files/2017/SCIS17_JU.pdf.

Hermawan, T., Suryanto, Y., Alief, F. & Roselina, L. (2021), Android Forensic Tools Analysis for Unsend Chat on Social Media, 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2020, pp. 233-238, published on: 13/01/2021, DOI: 10.1109/ISRITI51436.2020.9315364, last accessed 13/11/2021.

Meng, C., Baier, H. (2019), bring2lite: A Structural Concept and Tool for Forensic Data Analysis and Recovery of Deleted SQLite Records, Digital Investigation, vol. 29, pp. 31-41, published on: 14/07/2019, DOI: 10.1016/j.diin.2019.04.017, last accessed 09/11/2021.

Mahalik, H., Bair, J, Brignoni, A., Coates, S., Dickinson, M., Epifani, M., Hyde, J., Katalov, V., Koenig, S., Lorentz, P., Poirier, C., Reiber, L., Westman, M., Williamson, M., Whiffin, I. & Skulkin, O. (2021), Six Steps To Successful Mobile Validation, SANS Institute Reading Room site.

Mirza, M. M., Salamh, F. E. & Karabiyik, U. (2020), An Android Case Study on Technical Anti-Forensic Challenges of WhatsApp Application, 2020 8th International Symposium on Digital Forensics and Security (ISDFS), pp. 1-6, published on: 15/06/2020, DOI: 10.1109/ISDFS49300.2020f.9116192.

Nemetz, S., Schmitt, S. & Freiling, F. (2018), A standardized corpus for SQLite database forensics, Digital Investigation, vol. 24, pp. 121-130, published on: 21/03/2018, DOI: 10.1016/j.diin.2018.01.015, last accessed 09/11/2021.

Onovakpuri, P. (2018), Forensics Analysis of Skype, Viber and WhatsApp Messenger on Android Platform, International Journal of Cyber-Security and Digital Forensics (IJCSDF), Hong Kong, vol. 7, no. 2, pp. 119-131, published on: 06/2018, DOI: 10.17781/P002369.

Orr, D. A., Castro, A. (2018), Whatsapp Messenger on the Android Platform: A Forensic Examination of a Physical Device, Forensic Science: Methods and Techniques, issue 2018(1), pp. 04-19, published on: 20/09/2018, DOI: 10.33513/FSMT/1801-05.

Pawlaszczyk, D., Hummert, C. (2021), Making the Invisible Visible –Techniques for Recovering Deleted SQLite Data Records, The International Journal of Cyber Forensics and Advanced Threat Investigations under a Creative Commons License, vol. 1(1-3), pp. 27-41, published on: 15/02/2021, DOI: 10.46386/ijcfati.v1i1-3.17, last accessed 09/11/2021.

Raji, M., Wimmer, H. & Haddad, R. J. (2018), Analyzing Data from an Android Smartphone while Comparing between Two Forensic Tools, SoutheastCon 2018, 2018, pp. 1-6, published on: 04/10/2018, DOI: 10.1109/SECON.2018.8478851, last accessed 14/11/2021.

Rathi, K., Karabiyik, U., Aderibigbe, T. & Chi, H. (2018), Forensic analysis of encrypted instant messaging applications on Android, 2018 6th International Symposium on Digital Forensic and Security (ISDFS), pp. 1-6, published on: 07/05/2018, DOI: 10.1109/ISDFS.2018.8355344.

Saribekyan, H. & Margvelashvili, A. (2017), Security Analysis of Telegram 6.857 Final Project, published on 2017, retrieved from URL: https://www.semanticscholar.org/paper/Security-Analysis-of-Telegram-6-.-857-Final-Project-Saribekyan/fabe0871280513af67b23438ce17ca8c6c79a2d8, last accessed on 05/12/2021.

Skulkin, O., Tindall, D. & Tamma, R. (2018), Learning Android Forensics Analyze Android Devices With The Latest Forensic Tools And Techniques, 2nd Edition, published by Packt Publishing Ltd, December 2018, Birmingham UK, pp. 73-94 & 135-148, ISBN: 978-1-78913-101-7.

Sutikno, T., Handayani, L., Stiawan, D., Riyadi, M.A. & Subroto, I. (2016). WhatsApp, Viber and Telegram: which is the Best for Instant Messaging?, International Journal of Electrical and Computer Engineering (IJECE), vol. 6, no. 3, pp. 909-914, published on: 26/05/2016, DOI: 10.11591/ijece.v6i3.10271.

Tamma, R., Skulkin, O., Mahalik, H., Bommisetty, S. (2020), Practical Mobile Forensics Fourth Edition, Forensically investigate and analyze iOS, Android, and Windows 10 devices, Packt Publishing, Birmingham, UK, published on: 04/2020, ISBN 978-1-83864-752-0.

Vasilaras, A. (2016), Android Forensics in Android Smart Devices, MSc Thesis, Advanced Informatics and Computing Systems, Security Management Engineering, Department of Informatics, University of Piraeus, created on 03/2016.

Zhang, H., Chen, L. & Liu, Q. (2018), Digital Forensic Analysis of Instant Messaging Applications on Android Smartphones, 2018 International Conference on Computing, Networking and Communications (ICNC), pp. 647-651, published on: 21/06/2018, DOI: 10.1109/ICCNC.2018.8390330.

Internet Sites/Forums/Blogs:

askclees1711 (2020), SQLite Databases at hex level, AskClees, published on: 20/10/2020, retrieved from URL: https://askclees.com/2020/11/20/sqlite-databases-at-hex-level, last accessed 13/11/2021.

Belkasoft (2021), ANDROID VIBER FORENSICS WITH BELKASOFT X, retrieved from URL: https://belkasoft.com/android_viber_forensics, last accessed 13/11/2021.

Belkasoft (2021), Whitepaper: WhatsApp Forensics on Smartphones, retrieved from URL: https://belkasoft.com/whitepaper_whatsapp_forensics, last accessed 24/11/2021.

Boiko (Бойко), M. (2018), TELEGRAM ELECTRONIC CORRESPONDENCE RESEARCH ON ANDROID DEVICES («ДОСЛІДЖЕННЯ ЕЛЕКТРОННОЇ ПЕРЕПИСКИ TELEGRAM НА ПРИСТРОЯХ ANDROID»), Kiev, 2018, CACHE4.DB FILE OF TELEGRAM FOR ANDROID (PART 1), Digital Forensics (dflab.blogspot), published on: 03/01/2019, retrieved from URL: https://dflab.blogspot.com/2019/01/cache4db-file-of-telegram-for-android.html, last accessed on 05/12/2021.

Cimpanu, C. (2021), FBI document shows what data can be obtained from encrypted messaging apps, created on: 30/11/2021, retrieved from URL: https://therecord.media/fbi-document-shows-what-data-can-be-obtained-from-encrypted-messaging-apps, last accessed 09/12/2021.

Cynet (2021), NIST Incident Response Plan: Building Your Own IR Process Based on NIST Guidelines, last updated on 03/11/2021, retrieved from URL: https://www.cynet.com/incident-response/nist-incident-response, last accessed 09/11/2021.

Cynet (2021), Incident Response SANS: The 6 Steps in Depth, last updated on 03/11/2021, retrieved from URL: https://www.cynet.com/incident-response/incident-response-sans-the-6-steps-in-depth, last accessed 09/11/2021.

Cynet (2021), Incident Response Team: A Blueprint for Success, last updated on 03/11/2021, retrieved from URL: https://www.cynet.com/incident-response/incident-response-team-a-blueprint-for-success, last accessed 09/11/2021.

Cynet (2021), What is Incident Response? People, Processes and Tools [2021 Guide], last updated on 06/2021, retrieved from URL: https://www.cynet.com/incident-response, last accessed 09/11/2021.

Dean, B. (2021), How Many People Use Telegram in 2021? 55 Telegram Stats, BACKLINKO, published on: 14/10/2021, retrieved from URL: https://backlinko.com/telegram-users, last accessed 05/12/2021.

dfirfpi (2020), teleparser, ZENA FORENSICS something about digital forensics and something not, published on: 06/04/2020, retrieved from URL: https://blog.digital-forensics.it/2020/04/teleparser.html, last accessed on 05/12/2021.

ElDavoo, D. (2021), WhatsApp Crypt14 Database Decrypter, GitHub, created on: 29/08/2021, retrieved from URL: https://github.com/ElDavoo/WhatsApp-Crypt14-Decrypter, last accessed 24/11/2021.

González, J. (2012), From PowerOn to Android – The Boot Sequence, My Cellar Door, published on: 24/08/2012, retrieved from URL: https://javigon.com/2012/08/24/from-poweron-to-android-the-boot-sequence, last accessed 14/11/2021.

Macdonald, I. (2020), multidisabler-samsung, GitHub, created on: 30/12/2020, retrieved from URL: https://github.com/ianmacd/multidisabler-samsung, last accessed 09/11/2021.

Magisk Manager (2021), Download Magisk Manager Latest Version 23.0 For Android 2021, retrieved from URL: https://magiskmanager.com, last accessed 09/11/2021.

Rakuten Viber (2021), Viber Encryption Overview, retrieved from URL: https://www.viber.com/app/uploads/viber-encryption-overview.pdf, last accessed 22/11/2021.

Precise Law (2017), What is a Forensic Image?, Precise, published on: 09/11/2017, retrieved from URL: https://discovery.precise-law.com/what-is-a-forensic-image, last accessed 09/11/2021.

Rahmanpour, A. (2019), Preparing For Computer Forensics Investigation (Partitions), Threat Hunting, published on: 31/12/2019, retrieved from URL: https://www.threathunting.se/2019/12/31/preparing-for-computer-forensics-investigation-partitions, last accessed 09/11/2021.

redymedan (2021), [TWRP][Root] Unofficial TWRP 3.5.0 Android 11 for Galaxy A50, created on: 13/03/2021, XDA Developers, retrieved from URL: https://forum.xda-developers.com/t/twrp-root-unofficial-twrp-3-5-0-android-11-for-galaxy-a50.4246581, last accessed 09/11/2021.

SalvationDATA (2020), [Case Study] Mobile Forensics: Forensic Data Extraction from Android Devices Using ADB (Android Debug Bridge) Part II, published on 05/06/2020, retrieved from URL: https://blog.salvationdata.com/2020/06/05/case-study-mobile-forensics-forensic-data-extraction-from-android-devices-using-adb-android-debug-bridge-part-ii, last accessed 09/11/2021.

Singh S. (2020), WhatsApp has a fake news problem: Here’s everything that it’s doing to fight it in wake of Coronavirus outbreak, Financial Express, created on: 07/04/2020, retrieved from URL: https://www.financialexpress.com/industry/technology/whatsapp-has-a-fake-news-problem-heres-everything-that-its-doing-to-fight-it-in-wake-of-coronavirus-outbreak/1921515/, Last accessed 08/12/2021.

SQLite.org (2021), Atomic Commit In SQLite, retrieved from URL: https://www.sqlite.org/atomiccommit.html, last accessed 09/11/2021.

SQLite.org (2021), PRAGMA Statements (journal mode), retrieved from URL: https://www.sqlite.org/pragma.html#pragma_auto_vacuum, last accessed 09/11/2021.

SQLite.org (2021), Database File Format, retrieved from URL: https://www.sqlite.org/fileformat2.html, last accessed 09/11/2021.

SQLite.org (2021), PRAGMA Statements, retrieved from URL: https://www.sqlite.org/pragma.html#pragma_secure_delete, last accessed 09/11/2021.

SQLite.org (2021), PRAGMA Statements (journal mode), retrieved from URL: https://www.sqlite.org/pragma.html#pragma_journal_mode, last accessed 09/11/2021.

SQLite.org (2021), PRAGMA Statements (journal mode), retrieved from URL: https://www.sqlite.org/pragma.html#pragma_journal_size_limit, last accessed 09/11/2021.

O’ Dea, S. (2021), Forecast number of mobile devices worldwide from 2020 to 2025 (in billions), statista, published on 24/09/2021, retrieved from URL: https://www.statista.com/statistics/245501/multiple-mobile-device-ownership-worldwide, last accessed 31/01/2022.

Tahiri, S. (2016), Android forensic logical acquisition, INFOSEC, published on: 16/03/2016, retrieved from URL: https://resources.infosecinstitute.com/topic/android-forensic-logical-acquisition, last accessed 09/11/2021.

Telegram (2021), The Evolution of Telegram, retrieved from URL: https://telegram.org/evolution, last accessed on 05/12/2021.

Telegram (2021), End-to-End Encryption, Secret Chats, retrieved from URL: https://core.telegram.org/api/end-to-end, last accessed on 05/12/2021.

WhatsApp (2021), WhatsApp Encryption Overview, Technical white paper, retrieved from URL: https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf, last accessed on 22/11/2021.

Xiaomi MIUI (2021), Unlock Your Mi Device, retrieved from URL: https://en.miui.com/unlock/download_en.html, last accessed 09/11/2021.

zefie (2017), [RECOVERY][UNOFFICIAL][US997/H870][rel_o2/rel_t2][2018-06-08] Melina TWRP v3.2.1, created on: 20/12/2017, XDA Developers, retrieved from URL: https://forum.xda-developers.com/t/recovery-unofficial-us997-h870-rel_o2-rel_t2-2018-06-08-melina-twrp-v3-2-1.3722199, last accessed 09/11/2021.

99Content (2021), Viber Statistics, 99FIRMS BLOG, retrieved from URL: https://99firms.com/blog/viber-statistics, last accessed 13/11/2021.

99Content (2021), Whatsapp Statistics, 99FIRMS BLOG, retrieved from URL: https://99firms.com/blog/whatsapp-statistics, last accessed 23/11/2021.

Software, Tool and Utilities' Official Sites:

Access Data (FTK): https://accessdata.com/products-services/forensic-toolkit-ftk

Acquire Forensics (SQLite Forensic Explorer): https://www.acquireforensics.com/services/database-forensics.html

aLEAPP: https://github.com/abrignoni/ALEAPP

Andriller: https://0x1.gitlab.io/phone/Andriller

Android Gallery BlobCache Files: https://github.com/slo-sleuth/android_blob_cache

Autopsy: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2

Autopsy-Plugins: https://github.com/markmckinnon/Autopsy-Plugins

Belkasoft: https://belkasoft.com/android

bring2lite: https://github.com/bring2lite/bring2lite

Cellebrite UFED: https://www.cellebrite.com/en/ufed

Cellebrite Physical Analyzer: https://www.cellebrite.com/en/physical-analyzer

Forensic SQLite Data Recovery Tool (FQLite): https://www.staff.hs-mittweida.de/~pawlaszc/fqlite

Forensic Toolkit for SQLite: https://sqliteforensictoolkit.com/sqlite-forensic-toolkit

Hancom (MD-Series): http://www.hancomgmd.com/product & https://www.youtube.com/watch?v=YVkh7qFBVHc&ab_channel=TeelTechnologiesCanada

KS DB Merge Tools for SQLite: https://www.db-merge-tools.net/sqlite-diff-merge-overview.html

Magnet Forensics (Axiom & Acquire): https://www.magnetforensics.com/products/magnet-axiom-cyber & https://www.magnetforensics.com/resources/magnet-acquire

Melina TWRP v3.2.1: https://forum.xda-developers.com/t/recovery-unofficial-us997-h870-rel_o2-rel_t2-2018-06-08-melina-twrp-v3-2-1.3722199

MOBILedit: https://www.mobiledit.com/forensic-express-personal

MSAB (XRY): https://www.msab.com/law-enforcement/forensic-examiners

multidisabler-samsung: https://github.com/ianmacd/multidisabler-samsung

ncat: https://nmap.org/ncat

Oxygen forensics: https://www.oxygen-forensic.com/en/products/oxygen-forensic-detective

Paraben Forensics (E3 Forensic Platform): https://paraben.com/free-dfir-tools

Salvationdata (DBF6300): https://www.salvationdata.com/database-forensic-analysis-system

Sqlitebrowser: https://github.com/sqlitebrowser/sqlitebrowser

SQLite Expert database administration: http://www.sqliteexpert.com

Undark: https://github.com/alitrack/undark

WhatsApp Crypt14 Database Decrypter: https://github.com/ElDavoo/WhatsApp-Crypt14-Decrypter.

WhatsApp Key/DB Extractor: https://github.com/EliteAndroidApps/WhatsApp-Key-DB-Extractor

whatsapp-media-decrypt: https://github.com/sh4dowb/whatsapp-media-decrypt

X-Ways: http://www.x-ways.net/forensics

Appendix

A1.

A/A	Sender	Receiver	Content	Acquisition
1	Bad Actor_2	Edgar	How are you my friend?	data0_0_1
2	Bad Actor_2	Edgar	It is been a long time since we talked	data0_0_2
3	Bad Actor_2	Edgar	Are you in Boston?	data0_1
4	Edgar Reading the messages by (Receiver)			data0_1_1
5	Michael	Bad Actor_2	5 lyrics from the poem “The Crow” (1-5)	data0_2
6	Bad Actor_2	Edgar	Did Mike really send you a great poem? He should send me more Who wrote it?	data0_3
7	Reading a random message bot message			data0_3_1
8	Bad Actor_1	Bad Actor_2	5 lyrics from the poem “The Crow” (6-10)	data0_4
9	Reading the messages sent from Bad Actor_1			data0_4_1
10	Bad Actor_2	Bad Actor_1	Mike sent some lyrics from the same poem Where did you find it?	data1
11	Deleted chat thread with Edgar			data2
12	Bad Actor_1	Bad Actor_2	5 lyrics from the poem “The Crow” (11-15)	data3
13	Reading the messages sent from Michael			data3_1
14	Bad Actor_2	Michael	**Scenario** (10 messages from which 2 photos) (see Appendix A2)	data4 (Non-Deleted) data4_1 (Deleted)
15	Closing the app from background			data5
16	Edgar	Bad Actor_2	5 lyrics from the poem “The Crow” (16-20)	data6
17	Removing the messages notification			data7
18	Rebooting the device			data8
19	Opening the app			data9
20	Bad Actor_2	Michael	WHAT IS WRONG WITH YOU TWO AND THIS POEM	data10
21	Reading the messages sent from Edgard			data11
22	Michael	Bad Actor_2	1 lyric from the poem “The Crow” (21)	data12
23	Rebooting the device			data13
24	Opening the app			data14
25	Michael	Bad Actor_2	4 lyrics from the poem “The Crow” (22-25)	data15
26	Edgar	Bad Actor_2	5 lyrics from the poem “The Crow” (26-30)	data16
27	Bad Actor_1	Bad Actor_2	5 lyrics from the poem “The Crow” (31-35)	data17

A2.

A/A	Sender	Receiver	Action / Content	Parameters
1	Bad Actor_1	Michael	Message: I am sending the photo	

2	Bad Actor_1	Michael	Action: Sending photo from gallery	
3	Bad Actor_1	Michael	Message: Did you receive it?	
4	Michael	Bad Actor_1	Message: Yes send the second one	
5	Bad Actor_1	Michael	Action: Sending photo from app's camera feature	
6	Bad Actor_1	Michael	Message: This will cost you 1000 euros	
7	Michael	Bad Actor_1	Message: We agreed on five hundred	Timebomb at 60 seconds
8	Bad Actor_1	Michael	I want more	Timebomb at 60 seconds
9	Michael	Bad Actor_1	WE WILL SEE	Timebomb at 60 seconds
10	Bad Actor_1	Michael	I think they saw me. I must delete those	Timebomb at 60 seconds