



## LJMU Research Online

**Akinbi, A and Ojie, E**

**Forensic analysis of open-source XMPP multi-client social networking apps on iOS devices**

<http://researchonline.ljmu.ac.uk/id/eprint/18828/>

### Article

**Citation** (please note it is advisable to refer to the publisher's version if you intend to cite from this work)

**Akinbi, A and Ojie, E (2021) Forensic analysis of open-source XMPP multi-client social networking apps on iOS devices. Forensic Science International: Digital Investigation, 36. ISSN 2666-2817**

LJMU has developed **LJMU Research Online** for users to access the research output of the University more effectively. Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. Users may download and/or print one copy of any article(s) in LJMU Research Online to facilitate their private study or for non-commercial research. You may not engage in further distribution of the material or use it for any profit-making activities or any commercial gain.

The version presented here may differ from the published version or from the version of the record. Please see the repository URL above for details on accessing the published version and note that access may require a subscription.

For more information please contact [researchonline@ljmu.ac.uk](mailto:researchonline@ljmu.ac.uk)

<http://researchonline.ljmu.ac.uk/>

# Forensic analysis of open-source XMPP multi-client social networking apps on iOS devices

Alex Akinbi <sup>a</sup>, Ehizojie Ojie <sup>b</sup>

<sup>a</sup> Department of Computer Science, Liverpool John Moores University, 3 Byrom Street, Liverpool, L3 3AF, UK

<sup>b</sup> Department of Computer Science, University of York, Heslington, York, YO10 5DD, UK

## Abstract

In this paper, we present forensic analysis of Monal and Siskin IM, two decentralized open-source XMPP multi-client social networking apps on iOS devices that provide anonymity and privacy using OMEMO end-to-end encryption. We identified databases maintained by each app and storage locations within the iOS file system that stores the local copies of user information and metadata. We analyzed the databases and storage locations for evidential data of forensic value. The results in this paper show a detailed analysis and correlation of data stored in each app's database to identify the local user's multiple IM accounts and contact list, contents of messages exchanged with contacts, and chronology of conversations. The focus and main contributions of this study include a detailed description of artifacts of forensic interest that can be used to aid mobile forensic investigations.

**Keywords:** Mobile forensics; iOS forensics; Monal; Siskin IM; Instant messaging; XMPP; OMEMO; social networking

## 1. Introduction

The popularity of social networking and instant messaging (IM) apps available to smartphone users to communicate online with friends, family, and colleagues continues to grow as they provide more features and functionalities such as real-time messaging, voice, and video calls. Based on the number of active users in 2019, the most widely used apps include WhatsApp, Facebook Messenger, WeChat, QQ Mobile, Telegram, and Snapchat (Clement, 2019). Since Edward Snowden's revelations, the popularity of messaging apps such as Signal and Telegram that offer end-to-end encryption (E2EE) to the general public has been prominent. To better protect users from marketing companies who increasingly try to track users, deny access to private exchanged messages by governments and malicious hackers who could hack and leak chat sessions stored on provider servers online through a successful hack, the use of E2EE messaging apps have also increasingly gained popularity among smartphone users (David Nield and Brian Turner, 2020).

Although the reasons listed provide legitimate uses for E2EE IM apps, they are increasingly being used by extremist groups and organized criminals to communicate and coordinate without interception (Hexegic, 2020). This has created challenges for law enforcement authorities as they have little means of intercepting such communications unless the user's smartphone is obtained, and IM apps installed on the smartphone are analyzed for traces of forensic artifacts. Moreover, the storage capacity of smartphones is increasing significantly, and there are generally many apps installed on smartphones, it is increasingly difficult for investigators to identify data related to crimes for investigation (Kim and Lee, 2020). Although secure private communication can be guaranteed with the use of E2EE IM apps, maintaining anonymity is still a challenge for its users. Secure messaging apps like Signal and Telegram still require a user to provide a valid phone number to sign up, route encrypted messages over centralized servers and it is also difficult to hide IP addresses through anonymity services like VPNs or Tor when using them (Express VPN, 2019). These mean users can still be identified as well as those they are communicating with. Therefore, open-source Extensible Messaging and Presence Protocol (XMPP) multi-client IM applications combined with E2EE protocols that can provide both anonymity and privacy using a decentralized architecture is considered a better option amongst many users. XMPP was originally developed in the Jabber open-source community to provide an open, decentralized alternative to the closed instant messaging services at that time. The decentralized architecture of the XMPP network is like email; as a result, anyone can run their own private XMPP server, enabling individuals and organizations to take control of their communications experience (XMPP Standards Foundation (XSF), 2020a).

Since its inception, many improvements have been implemented to the protocol and development of mobile apps (Gultsch, 2016). There are several notable XMPP multi-client IM apps available on iOS, macOS, Android, Linux, and Windows platforms. Two popular ones on iOS are *Monal* and *Siskin IM* apps with 3.4 and 3.5 ratings respectively on the App store (at the time of writing). They are both lightweight XMPP clients that support the use of Multi-End Message and Object Encryption(OMEMO), an open standard based on a Double Ratchet and PEP for secure multi-client end-to-end encryption (XMPP Standards Foundation (XSF), 2020b). They are also public projects, which means their source code can be audited, can be routed over Tor network, support the use of decentralized authoritative private servers, and support the use of multiple accounts by a single user. Therefore, given these characteristics, interest in forensic analysis of these applications is imperative (Anglano et al.,

2017; Kim and Lee, 2020; Wu et al., 2017). This prompts this study into the identification, recovery, and analysis of artifacts relating to the usage of Monal and Siskin XMPP multi-client IM apps on iOS.

In this paper, the focus and contributions include dealing with the forensic analysis of both Monal and Siskin apps running on an iOS (iPhone) device. We demonstrate the recovery of forensic artifacts relating to local copies of exchanged messages and files, evidence of deleted messages and files, local user's multiple IM accounts and contact lists, and chronology associated with these events from the main storage locations on each app. Other contributions in our analysis include the recovery of artifacts associated with deleted contacts and associated metadata. To ensure results from our analysis are realistic and can be reproduced, we describe our analysis methodology which includes the creation of user-profiles and the exchange of messages in an investigative scenario using physical iOS devices that are not jail-broken.

This paper is organized as follows. In Section 2, we discuss existing literature, while in Section 3, we describe the analysis methodology and the tools used in our experiments. In Section 4, we discuss the investigative scenario. Forensics analysis and findings of Monal and Siskin IM apps including artifacts recovered are presented in Section 5 and Section 6 respectively. In Section 7, we conclude the paper.

## **2. Related Literature**

Forensic analysis of social networking and IM apps including those that support E2EE has been extensively studied especially on Android compared with iOS platforms (Akbal et al., 2019; Alyahya and Kausar, 2017; Anglano, 2014; Anglano et al., 2017; Dargahi et al., 2017; Hintea et al., 2018; Jadhav Bhatt et al., 2018; Mahajan et al., 2013; Norouzizadeh Dezfouli et al., 2016; Ovens and Morison, 2016; Shortall and Azhar, 2015; Sudozai et al., 2018; Wu et al., 2017). There is limited study on the forensic analysis of open-source XMPP multi-client IM apps that supports E2EE. One recent study is the forensic analysis of *ChatSecure* instant messaging application (Anglano et al., 2016). However, the study focused on the analysis of the app on Android and not iOS. Before this study, Wouter S. van Dongen (van Dongen, 2007) conducted a forensic analysis of Pidgin Messenger 2.0 on the Linux platform. Based on the recent market share on Android devices (Statista, 2020), Android's popularity has attracted several researchers to focus on investigating different aspects of Android which includes social networking apps' forensics (Dargahi et al., 2017; Norouzizadeh Dezfouli et

al., 2016) creating the need for more forensic analysis of social networking and IM apps on iOS platforms. Therefore, by focusing on XMPP multi-client apps that support E2EE on the iOS platform, we make the most of the potential investigative impact of our work. To the best of our knowledge, no study has focused on Monal and Siskin IM apps on iOS.

### 3. Analysis methodology and tools

The forensic analysis methodology required a set of separate controlled experiments carried out for both Monal and Siskin IM apps. According to Anglano et. al (Anglano et al., 2017, 2016): “Given that the goal of any forensic analysis is to allow the analyst to obtain the digital evidence generated by the applications under consideration, the methodology used to carry it out must exhibit three unique properties”, which are described as follows.

- **completeness** : the identification of all the data generated by the application under analysis by carrying out all relevant functionalities of the application;
- **repeatability** : the possibility for a third party to replicate experiments under the same operational conditions, and to obtain the same results;
- **generality**: the results hold for many (possibly all) iOS smartphones and versions.

Therefore, we created an investigative scenario followed by subsequent phases, “*Installation of application*” and “*Design of experiments*” respectively for each application. In the “*Installation of Application*” phase, we installed and ran both *Monal v. 4.5* and *Siskin IM v. 5.8.1* apps (recent versions at the time of writing) on three iOS devices (not jailbroken) and created multiple user profiles (roles). We then created multiple XMPP accounts and assigned roles to each device (e.g. sender or recipient of a message, group chat leader, etc.). Since both apps only support iOS platforms, the realistic scenario was to install both apps on two of the devices to serve as the test local user’s contacts in the experiments. Table 1 summarizes the profiles and test devices that were used. In the “*Design of experiments*” phase, we define a set of experiments that includes one-to-one chats between a local user and other users in the contacts list, exchange of multimedia files, blocking and deleting contacts, group creation and chats etc. to generate as many forensic artifacts as possible, and demonstrates a typical user’s realistic interaction has happened. The device representing the test local user’s profile was then analyzed for forensic artifacts by first conducting an advanced logical acquisition of the iOS internal memory which was followed by the analysis of the logical image using *Cellebrite UFED Physical Analyzer v. 7.31* (Cellebrite, 2020).

Table 1. Test Devices

Device	OS Version	Model	Profile
iPhone 6s	13.3.1	A1688	Local user
iPhone 6s Plus	13.3.1	A1634	Contact
iPhone 6s Plus	13.3.1	A1634	Contact

*Cellebrite UFED Physical Analyzer's* iOS advanced logical acquisition relies on the iTunes backup and uses Apple's backup infrastructure to create a logical image that includes a robust copy of data including SMSs, text messages with attachments, multimedia files, call logs contact lists, application data, etc from the iOS device. It can decode the extractions from the logical image in a simple graphical user interface which makes it easy to identify various files of interests relating to the installed applications' data. However, *Cellebrite UFED Physical Analyzer v. 7.31* could not decode app activity for both the Monal and Siskin IM apps. Therefore, once these files of interest were identified, we then conducted a manual in-depth analysis of each one for information of interest related to our corresponding experiments. Alternative advanced logical acquisition of the iOS device was conducted using *Cellebrite's* file system extraction that fully integrates the *checkm8* and *checkra1n* exploits to extract a file system image of the iOS device in a forensically sound manner. The results from the forensic extraction, files of interests, and evidential data recovered were the same as the iTunes backup. Therefore, we decided to stick to the advanced logical acquisition which relies on iTunes backup because *checkm8* and *checkra1n* exploits currently support iOS devices (iPhone 5s through iPhone X ) which use the A9 64-bit ARM-based system-on-chip and lower ("checkra1n," 2020). This may not be the case for many iOS devices examined by forensic analysts in the field.

Hence, our analysis methodology provides the possibility for the experiments to be generalized, replicable and reproduced by a third party under the same operational conditions, and to obtain the same results regardless of the iOS version and iPhone model.

#### 4. Investigative Scenario

The test scenario which was used to demonstrate the forensic analysis in this paper is described as follows: A user (local user) with an iOS device as presented in Table 1 is being investigated. Forensic examiners found both Monal and Siskin IM apps installed on the seized device and are keen to answer the following questions:

- i. How many distinct XMPP IM accounts associated with the local user was configured and used with the Monal and Siskin IM apps?
- ii. Who are the local user's contacts on each app? Are there indications of any blocked or deleted contacts?
- iii. What messages have been exchanged with the above contacts and when did each communication occur on each app?
- iv. Did the local user exchange files with contacts and can encrypted messages, deleted data, and files be recovered from each app?

In the following sections, we present the forensic analysis of Monal and Siskin IM apps respectively.

## **5. Forensic analysis of Monal app**

Monal is a decentralized open-source XMPP multi-client IM app designed for iOS and macOS platforms. It uses OMEMO by default for encrypting conversations between verified users only (Anu, 2019). Users need to verify, and trust encryption keys used by both parties before encryption is enabled. To encrypt conversations, a user needs to add a contact by accepting a notification request, verify the identity, and trust the contact's encryption key. A user is also able to send messages to another user's IM account without verifying and trusting keys. However, the conversations will not be encrypted. Users can also toggle between sending messages unencrypted or encrypted using a security feature on the app. In our investigative scenario, we exchange several messages encrypted and unencrypted to conduct a comprehensive forensic analysis of artifacts that can be recovered.

*5.1. Installation of Monal app and account configuration* On installation of Monal, the app places user data in the directory `"/private/var/mobile/Containers/Data/Application/G7YU7X7KRJ.SworIM"` on the iOS device. The directory contains the artifacts that would be of most interest to a forensic investigator. The local user is required to register an account or use an existing XMPP IM account to sign in. Registering an account with a chosen username creates an XMPP IM account with the `@yam.ix` domain. A user can then add multiple accounts to communicate with contacts. In our scenario, we registered an account and then added a second existing XMPP IM account (`@xabber.org`) to the local user's profile. Similar configurations were repeated for the other contact profiles. Details of these profiles and accounts are presented in Table 2.

Table 2. Monal local user's and contacts' XMPP IM accounts

Local user's XMPP IM accounts on analyzed iPhone 6s	<i>behemoth@yax.im</i> <i>behemoth02@xabber.org</i>
1 <sup>st</sup> Contact's XMPP IM accounts on other iPhone 6s Plus devices added to local user's contacts list	<i>alice.behemoth@yax.im</i> <i>alice.behemoth@tigase.im</i>
2 <sup>nd</sup> Contact's XMPP IM accounts on iPhone 6s Plus devices added to local user's contacts list	<i>eve01@yax.im</i> <i>bobuser@xabber.org</i>
XMPP IM account of the deleted contact	<i>bobuser@xabber.org</i>

### 5.2. Design of experiments

As discussed in Section 5.1, once the local user and contact accounts have been created, the next step of the investigative scenario consists in the design of a set of experiments aimed at reconstructing the actions and activities carried out by the local user and contacts using the Monal app. In Table 3, we present descriptions of the actions and steps performed in our experiments. The goal of the design of these experiments was aimed at answering the questions set out in the investigative scenario by recovering relevant forensic artifacts to achieve completeness, repeatability and generality from the detailed forensic analysis.

Table 3. Experiments concerning dialogues, files exchanges and deletions between contacts.

Description	Steps
Regular chat	1. The local user creates several regular chat messages (OMEMO encryption disabled) with 1 <sup>st</sup> and 2 <sup>nd</sup> contacts
	2. The local user creates several regular chat messages (OMEMO encryption enabled) with 1 <sup>st</sup> and 2 <sup>nd</sup> contacts
	3. 1 <sup>st</sup> and 2 <sup>nd</sup> contacts send messages to the local user
	4. The local user deletes some chat messages exchanged with the 1 <sup>st</sup> and 2 <sup>nd</sup> contacts
	5. The local user deletes one of the 2 <sup>nd</sup> contact's IM account
File exchange	1. The local user sends an image to 1 <sup>st</sup> contact
	2. The local user receives an image from 1 <sup>st</sup> contact
	3. Local user deletes image received from 1 <sup>st</sup> contact



### 5.3. Location of Monal app artifacts

Data of conversation records and configuration files when using the app are stored in two main subdirectories (“*Documents*” and “*Library*”) of the “*G7YU7X7KRJ.SworIM*” main folder ( see Figure 1).

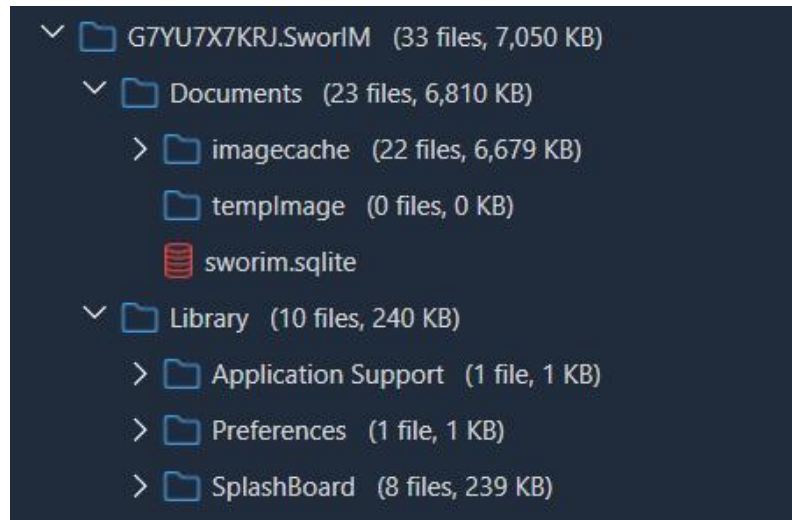


Figure 1: The main folder structure of the Monal app

The main database “*sworim.sqlite*” and other forensic artifacts of interests are in *Documents* subdirectory. Raw copies of images are stored in the “*imagecache*” subfolder while the *Library* subdirectory contains various default configuration and property list(plist) files. All files in the *Library* subdirectory which include the plist files were not significant artifacts. The *sworim.sqlite* SQLite database is the most important evidential data of forensic interest. This database maintains a record of messages exchanged, information associated with the local user and contacts’ XMPP IM accounts, the chronology of messages and files exchanged, etc. (See Figure 2). From our findings, only 11 out of these 23 tables contain information of forensic interest namely, *account*, *activechats*, *blockList*, *buddy\_resources*, *buddylist*, *imageCache*, *message\_history*, *muteList*, *signalContactIdentity*, *sqlite\_sequence*, and *subscriptionRequests*. To answer the questions from our investigative scenario, we discuss the details of the contents of these tables, their analysis, and interpretation as follows.

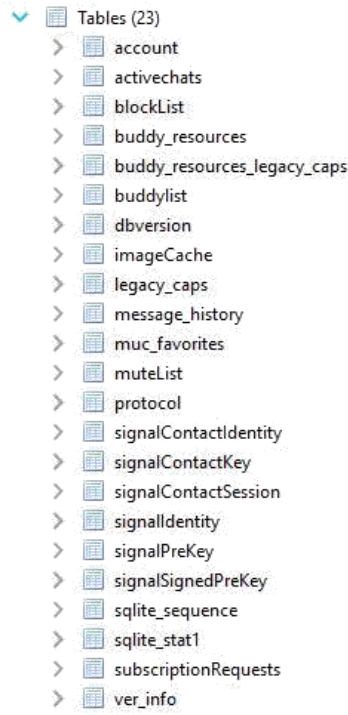


Figure 2: Structure of the main *sworim.sqlite* database

#### 5.4. Recovering account information (Monal app)

- i. The **accounts** table contains information associated with the local user’s distinct XMPP IM accounts. Each account is assigned a unique identifier and stored in the “*account\_id*” field which is the ( primary key) for this table. The two accounts associated with the local user from our scenario named *behemoth@yax.im* and *behemoth02@xabber.org* are stored in the table. Each account username and XMPP server domain name are stored in the “*account\_name*” and “*server*” fields respectively (See Figure 3).

Table: account				
	account_id	account_name	protocol_id	server
	Filter	Filter	Filter	Filter
1	1	behemoth	1	yax.im
2	2	behemoth02	1	xabber.org

Figure 3: *accounts* table

ii. The *activechats* and *buddylist* tables (See Figure 4) jointly store records associated with the local user’s contacts. The *activechats* table stores record of contacts that are active (not deleted or blocked) in the “*buddy\_name*” field. It also stores a record of the date and time the last message was exchanged between the local user and each contact in the “*lastMessageTime*” field. However, information associated with both active and deleted contacts is stored in the *buddylist* table. The XMPP IM account associated with the deleted contact *bobuser@xabber.org* was seen in this table. The “*account\_id*” field appears to be a foreign key in the *buddylist* table which shows the relationship between the *activechats* and *buddylist* tables. The field indicates the distinct local user IM account that was used to exchange messages with a specific contact. Information associated with blocked and muted contacts is stored in the *blockList* and *muteList* tables respectively. Pending request notifications by new contacts are stored in the *subscriptionsRequests* table, the *signalContactIdentity* table shows all active verified and trusted contacts, while the *buddy\_resources* table shows the iOS device names of all active contacts.

Table: activechats

	account_id	buddy_name	lastMessageTime
	Filter	Filter	Filter
1	1	alice.behemoth@yax.im	2020-04-29 21:43:46
2	1	eve01@yax.im	2020-04-29 22:29:09
3	1	alice.behemoth@tigase.im	2020-04-30 05:37:38

Table: buddylist

	buddy_id	account_id	buddy_name	full_name
	Filter	Filter	Filter	Filter
1	1	1	alice.behemoth@yax.im	alice.behemoth@yax.im
2	5	1	bobuser@xabber.org	bobuser@xabber.org
3	8	1	eve01@yax.im	eve01@yax.im
4	9	2	bobuser@xabber.org	Bob
5	10	1	alice.behemoth@tigase.im	alice.behemoth@tigase.im

Figure 4: *activechats* and *buddylist* tables

5.5. *Recovering chronology of chat logs, message contents, and deleted files (Monal app)*

iii. Information associated with messages and multimedia files exchanged between the local user and contacts (buddylist) is stored in the *message\_history* table. This table contains the record of messages sent and received including the textual content of each message body, the contact associated with each exchanged message, date and time a message was exchanged, and a status flag which indicates whether the message exchanged is encrypted (encrypted =1) or unencrypted (encrypted =0). Each message is assigned with a unique identifier stored in the “*message\_history\_id*” field and the local user account associated with each exchanged message can be identified by the “*account\_id*” (foreign key). All messages sent (encrypted and unencrypted) is stored in plaintext in the table. However, information associated with deleted messages exchanged with both active and deleted contacts is not stored. The timeline of messages exchanged can be reconstructed using the information stored in this table. In Table 4, we present a detailed interpretation of the relevant fields from the *message\_history* table of the *sworim.sqlite* database.

Table 4. Structure of the *message\_history* table

Field	Role	Type	Meaning
message_history_id	Primary Key	int	unique identifier of the message
message_from		string	contact user/ local user IM account message was sent from
message_to		string	contact user/ local user IM account message was sent to
timestamp		string	the date and time a message was sent or received
message		string	body of the message
unread		boolean	flag indicating whether content of a message is unread (unread=0) or read (unread=1)
delivered		boolean	flag indicating whether content of a message is delivered (delivered =1) or undelivered (delivered=0)
messageType		string	flag indicating whether a message type (messageType = Text) or (messageType =Image)
received		boolean	flag indicating whether a message was received (received=1) or not received (received=NULL)
encrypted		boolean	flag indicating whether a message is encrypted (encrypted=1) or unencrypted (encrypted=0)

In Figure 5, we show a record of 21 messages exchanged. From the figure, we see the first record in the field named “*message*” corresponds to an unencrypted message

(encrypted = 0) sent from the contact *alice.behemoth@yax.im* on the 29<sup>th</sup> Apr. 2020 at 3:40 pm UTC and the “*messageType*” field indicates the body of the message is text (*messageType*=Text). The 16<sup>th</sup> record from this figure also shows an encrypted (encrypted = 1) message (*messageType* = Image) sent from the local user IM account *behemoth@yax.im* to the contact *alice.behemoth@yax.im* on the 29<sup>th</sup> Apr. 2020 at 9:35 pm UTC. The *sqlite\_sequence* table log and store information associated with the total number of messages exchanged (including deleted ones), number subscription requests by contacts ( including rejected ones), and the total number of contacts (including deleted ones). It is well known that remnants of deleted data from SQLite databases are kept in unallocated cells in the file corresponding to the database, from which they can be recovered (Anglano et al., 2016; Jeon et al., 2012). However, our attempts to recover deleted data from the database using *Undark v 0.6* (Daniels, 2015) and *Cellebrite Physical Analyzer SQLite* recovery tools were unsuccessful as the fields containing deleted data had been overwritten with null bytes upon deletion. This confirms that the VACUUM command is used within the main *sworim.sqlite* database of the Monal app.

	message_from	message_to	timestamp	message	actual_from	messageType	encrypted
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	alice.behemoth@yax.im	behemoth@yax.im	2020-04-29 15:40:31	Hello	alice.behemoth@yax.im	Text	0
2	behemoth@yax.im	alice.behemoth@yax.im	2020-04-29 15:40:54	Hi Bee. How are y...	behemoth@yax.im	Text	0
3	alice.behemoth@yax.im	behemoth@yax.im	2020-04-29 15:41:13	I am fine Alice an...	alice.behemoth@yax.im	Text	0
4	behemoth@yax.im	alice.behemoth@yax.im	2020-04-29 15:41:41	Not much you can ...	behemoth@yax.im	Text	0
5	behemoth@yax.im	alice.behemoth@yax.im	2020-04-29 16:08:01	https://upload.yax...	behemoth@yax.im	Image	0
6	behemoth@yax.im	alice.behemoth@yax.im	2020-04-29 16:11:38	Are you working fr...	behemoth@yax.im	Text	0
7	alice.behemoth@yax.im	behemoth@yax.im	2020-04-29 16:12:20	Yes	alice.behemoth@yax.im	Text	0
8	alice.behemoth@yax.im	behemoth@yax.im	2020-04-29 19:36:39	This is a secure m...	alice.behemoth@yax.im	Text	0
9	behemoth@yax.im	alice.behemoth@yax.im	2020-04-29 19:37:40	Keys verified so yes	behemoth@yax.im	Text	0
10	behemoth@yax.im	alice.behemoth@yax.im	2020-04-29 20:04:24	https://upload.yax...	behemoth@yax.im	Image	0
11	behemoth@yax.im	alice.behemoth@yax.im	2020-04-29 20:04:47	Just sent an imag...	behemoth@yax.im	Text	0
12	alice.behemoth@yax.im	behemoth@yax.im	2020-04-29 20:04:58	Got it	alice.behemoth@yax.im	Text	0
13	alice.behemoth@yax.im	behemoth@yax.im	2020-04-29 20:06:27	Delete this message	alice.behemoth@yax.im	Text	0
14	alice.behemoth@yax.im	behemoth@yax.im	2020-04-29 21:34:07	Secure message s...	alice.behemoth@yax.im	Text	1
15	behemoth@yax.im	alice.behemoth@yax.im	2020-04-29 21:34:34	Secure message r...	behemoth@yax.im	Text	1
16	behemoth@yax.im	alice.behemoth@yax.im	2020-04-29 21:35:04	aesgcm://upload....	behemoth@yax.im	Image	1
17	alice.behemoth@yax.im	behemoth@yax.im	2020-04-29 21:43:30	Sent your a nice i...	alice.behemoth@yax.im	Text	1
18	behemoth@yax.im	alice.behemoth@yax.im	2020-04-29 21:43:46	Got it thanks	behemoth@yax.im	Text	1
19	behemoth@yax.im	eve01@yax.im	2020-04-29 22:03:11	Message was not ...	behemoth@yax.im	Text	1
20	behemoth@yax.im	eve01@yax.im	2020-04-29 22:06:51	Message was not ...	behemoth@yax.im	Text	1
21	eve01@yax.im	behemoth@yax.im	2020-04-29 22:07:05	There was an err...	eve01@yax.im	Text	1

Figure 5: *message\_history* table

- iv. A record of each multimedia (image) file exchanged is stored as a relative URL path in the *message* field. If a file is shared with OMEMO encryption turned off, a copy

file can be recovered directly from the XMPP server by entering the URL link in a browser, for example (*https://upload.yax.im/upload/<file-hash>.jpg*). However, if a file is shared with encryption turned on, a random key and IV are generated and the file is then encrypted with AES-256 in Galois/Counter Mode (GCM) (XMPP, 2018). This converts the HTTPS URL to an *aesgcm://* URL as shown in the 16<sup>th</sup> record in Figure 5. A copy of the encrypted file is stored on the XMPP server and when received is decrypted locally on the device using a symmetric key sent with the secure message body. In our investigative scenario, we were able to recover all files exchanged including raw copies of deleted ones. Each file is stored with a hash value file name and information for all files exchanged including URLs and relative paths is stored in the *imageCache* table (Figure 6). Raw copies of each file are stored in the *imageCache* subdirectory of the app's main folder located in the */private/var/mobile/Containers/Data/Application/G7YU7X7KRJ.SworIM/Documents/imagecache* subdirectory in the iOS file system. It is worth noting that if a contact uses an XMPP IM account such as Xabber (*@xabber.org*) which supports only end-to-end message encryption with Off-the-Record Messaging (OTR) but notOMEMO, the messages and files exchanged using the Monal client app are not encrypted.

8	<a href="https://upload.yax.im/upload/64jq81gDmvLUK...">https://upload.yax.im/upload/64jq81gDmvLUK...</a>	418C7E62-0794-4947-8602-82A9CB3CD40E
9	<a href="https://upload.yax.im/upload/64jq81gDmvLUK...">https://upload.yax.im/upload/64jq81gDmvLUK...</a>	E7385F7C-278A-417A-B56F-EFFF0C10889B
10	<a href="aesgcm://upload.yax.im/upload/IBeuV-Uz71a...">aesgcm://upload.yax.im/upload/IBeuV-Uz71a...</a>	DC398660-9F65-49A0-8188-89198D311F72
11	<a href="aesgcm://upload.yax.im/upload/Ssz3puptWt...">aesgcm://upload.yax.im/upload/Ssz3puptWt...</a>	7943E672-5817-4D5A-A9EE-A59CC3AE2F41

↓ URL
↓ File Name

Figure 6: *imageCache* table

## 6. Forensic analysis of Siskin IM app

Siskin IM app by Tigase, Inc. is a lightweight and decentralized open-source XMPP multi-client for iOS platforms (Tigase, 2020). Like many social networking apps, Siskin IM allows users to exchange text messages and multimedia files, make audio and video calls using Voice over IP (VoIP). It supports the use of OMEMO for secure end-to-end-encryption

communication. However, the feature is not enabled by default. The user can toggle between sending messages unencrypted or encrypted in the app’s chat settings. Users can also create groups, join public and private chat rooms.

### 6.1. Installation of Siskin IM app and account configuration

The Siskin IM app follows a different file structure in the iOS file system compared to most social networking apps. Identifying the location of these files and folders along with their usage can be important for an investigator (Sudozai et al., 2018). On installation, user data is placed in three main folders “/private/var/mobile/Containers/Data/Application/org.tigase.messenger.mobile”, “/private/var/mobile/Containers/Shared/AppGroup/group.siskinim.shared” and “/private/var/mobile/Containers/Shared/AppGroup/group.TigaseMessenger.Share” on the iOS file system. Like many XMPP multi-client apps, a user can sign in using an existing XMPP account or register one on the Tigase domain (@tigase.im) and then proceed to add multiple accounts. In our scenario, we registered an account and added an existing Xabber (@xabber.org) IM account. Similar configurations were repeated for the other contact profiles. Details of these profiles and accounts are presented in Table 5. We then proceeded to add the contacts by sending and accepting requests, verifying user identities, and trusting encryptions keys for each user. It is worth noting that for a comprehensive forensic analysis, we exchanged some messages without OMEMO encryption enabled. Also, the local user and contact’s Xabber accounts only support OTR end-to-end encryption. Therefore, messages exchanged between these accounts are not encrypted.

Table 5. Siskin local user’s and contacts’ XMPP IM accounts

Local user’s XMPP IM accounts on analyzed iPhone 6s	<i>behemoth@tigase.im</i> <i>behemoth02@xabber.org</i>
1 <sup>st</sup> Contact’s XMPP IM accounts on other iPhone 6s Plus devices added to local user’s contacts list	<i>alice.behemoth@yax.im</i> <i>alice.behemoth@tigase.im</i>
2 <sup>nd</sup> Contact’s XMPP IM accounts on iPhone 6s Plus devices added to local user’s contacts list	<i>eve01@tigase.im</i> <i>bobuser@xabber.org</i>
XMPP IM account of the deleted contact	<i>alice.behemoth@yax.im</i>
XMPP IM account of blocked contact	<i>bobuser@xabber.org</i>

## 6.2. Design of experiments

Similar to the Monal app, once the local user and contact accounts have been created on the Siskin IM app, the next step of the investigative scenario consists in the design of a set of experiments aimed at reconstructing the actions and activities carried out by the local user and contacts. In Table 6, we present descriptions of actions and steps performed in our experiments. The goal of the design of these experiments was aimed at answering the questions set out in the investigative scenario by recovering relevant forensic artifacts to achieve completeness, repeatability and generality from the detailed forensic analysis.

Table 6. Experiments concerning dialogues, files exchanges and deletions between contacts.

Description	Steps
Regular chat	1. The local user creates several regular chat messages (without encryption enabled) with 1 <sup>st</sup> contact and 2 <sup>nd</sup> contact
	2. The local user creates several regular chat messages (encryption enabled) with 1 <sup>st</sup> contact and 2 <sup>nd</sup> contact
	3. 1 <sup>st</sup> and 2 <sup>nd</sup> contacts send messages to the local user
	4. The local user deletes some chat messages exchanged with the 1 <sup>st</sup> and 2 <sup>nd</sup> contacts
	5. The local user deletes one of the 2 <sup>nd</sup> contact's IM accounts
	6. The local user blocks one of the 1 <sup>st</sup> contact's IM accounts
File exchange	1. The local user sends an image to 1 <sup>st</sup> contact
	2. The local user receives an image from 1 <sup>st</sup> contact
	3. Local user deletes image received from 1 <sup>st</sup> contact
Private chat group	1. The local user creates a private group
	2. The local user invites 2 <sup>nd</sup> contact to join the group
	3. 2 <sup>nd</sup> contact accepts invite and joins the group
	4. The local user sends a textual message to group members
	5. 2 <sup>nd</sup> contact send message to group members
	6. The local user removes 2 <sup>nd</sup> contact
	7. The local user deletes the group

## 6.3. Location of Siskin IM app artifacts

As mentioned earlier, Siskin IM app store user data in three main folders

(*org.tigase.messenger.mobile* , *group.siskinim.shared* and *group.TigaseMessenger.Share*) on



the iOS file system. The most crucial forensic artifacts of interest are stored in the *siskinim\_main.db* SQLite database located in the *group.siskinim.shared* folder. Forensic artifacts associated with raw copies of images received by the local user are the only significant artifacts in the *org.tigase.messenger.mobile* folder, while the *group.TigaseMessenger.Share* folder contains several configurations and plist files with no forensic significance (see Figure 7).

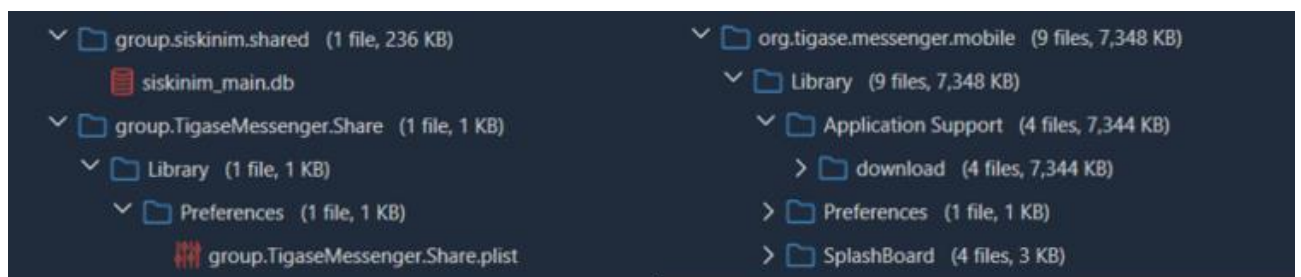



Figure 7: Main folders of the Siskin IM app

The *siskinim\_main.db* store records and information of chat messages, user and contact account information, metadata of files exchanged include time and dates in the order of their occurrence. The database contains 15 different tables that store the listed information. However, from our findings, only 8 out of these 15 tables contain artifacts of forensic interests namely *chat\_history*, *chats*, *omemo\_identities*, *omemo\_sessions*, *omemo\_signed\_pre\_keys*, *roster\_items*, *sqlite\_sequence*, and *vcards\_cache*. We proceed to discuss the contents of these tables in the database to answer questions from our investigative scenario.

#### 6.4. Recovering account information (Siskin IM app)

- i. Information associated with the distinct local user's accounts created during installation and used to communicate with contacts are stored in the *omemo\_signed\_pre\_keys* table (Figure 8). However, the table appears to have duplicate copies of one of the IM accounts with no unique identifier (primary key) defined in the table.

Table:  omemo\_signed\_pre\_keys

	account	id
	Filter	Filter
1	behemoth02@xabber.org	1
2	behemoth@tigase.im	1
3	behemoth02@xabber.org	2

Figure 8: *omemo\_signed\_pre\_keys* table

- ii. The *omemo\_sessions* table store records of conversation sessions with active contacts (not deleted or blocked) that support and use OMEMO encryption. Account information of all active IM accounts which includes that of the local user is stored in the *omemo\_identities* table. Both tables are linked together through the “*device\_id*” which is the unique identifier associated with encryption keys used to exchange messages between the local user and specific contact (See Figure 9). Information associated with subscription requests to add and verify a contact identity is stored in the *roster\_items* table. The table shows the date and time each request was made. However, information associated with deleted contacts is not stored in the table. The *vcards\_cache* table holds information about each active contact. A request query can be sent to the hosting XMPP server that hosts the account information for details of the contact store in the vCard entry (XMPP, 2013). The *sqlite\_sequence* table logs and store information associated with the total number of chat messages exchanged (including deleted ones), and the total number of contacts vcards.

Table: omemo\_identities

	account	name	device_id
	Filter	Filter	Filter
1	behemoth02@xabber.org	behemoth02@xabber.org	338460127
2	behemoth@tigase.im	behemoth@tigase.im	1546313320
3	behemoth@tigase.im	behemoth@tigase.im	1047466181
4	behemoth@tigase.im	eve01@tigase.im	2130835371
5	behemoth@tigase.im	alice.behemoth@tigase.im	71302407
6	behemoth@tigase.im	alice.behemoth@tigase.im	1324021390

Table: omemo\_sessions

	account	name	device_id
	Filter	Filter	Filter
1	behemoth@tigase.im	alice.behemoth@tigase.im	71302407
2	behemoth@tigase.im	behemoth@tigase.im	1047466181
3	behemoth@tigase.im	eve01@tigase.im	2130835371
4	behemoth@tigase.im	alice.behemoth@tigase.im	1324021390

Figure 9: Tables *omemo\_identities* and *omemo\_sessions*

### 6.5. Recovering chronology of chat logs, message contents, and deleted files (Siskin IM app)

- i. Information association with messages exchanged with all contacts including deleted chat messages ( active, blocked, and deleted) are stored in the *chats*, *chats\_read*, and *chat\_history* tables. The *chats* table store the date and time when a message was first exchanged with each contact while the *chats\_read* table shows the last time a message received was read. The *chat\_history* table is the main table in the *siskinim\_main.db* database. It contains a detailed record of all textual messages(encrypted and unencrypted) exchanged, group chats, chronological timeline of each conversation, and associated metadata. Each record in the table is assigned a unique identifier stored in the “*id*” field. The local user IM account name is stored in the “*account*” field and corresponding contact’s IM account name is stored in the “*jid*” field. In Table 7, we present a detailed interpretation of the relevant fields from the *chat\_history* table of the *siskinim\_main.db*.

Table 7. Structure of the *chat\_history* table

Field	Role	Type	Meaning
id	Primary Key	int	unique identifier of the message
account		string	local user IM account
jid		string	contact user IM account message is exchanged with
author_jid		string	chat group administrator's IM account
author_nickname		string	IM account used to send messages within the chat group
timestamp		int	the date message was been sent or received (13-digits Unix epoch format)
item_type		boolean	flag indicating whether content a message is text (item_type=0) or image file (item_type=1)
data		string	body of the message
state		int	flag indicating whether a message was received (state= 0) or sent (state=9 or 1)
encryption		boolean	flag indicating whether a message is encrypted (encryption=1) or unencrypted (encryption=0)
appendix		json	contains file size, mime type and name of file attachments sent or received.

To demonstrate the chronology of messages exchanged with each one of the contacts from our experiments, we present some of the contents of the first 22 messages exchanged and stored in this table as shown in Figure 10.

From the figure we see all messages exchanged between each distinct local user's IM account with each contact. From the first record in the figure, an encrypted (encryption=1) text message (item\_type=0) stored in the "data" field was sent (state =9) from *behemoth@tigase.im* to contact *eve01@tigase.im* on the 29<sup>th</sup> Apr. 2020 11:29:08 pm UTC ( Unix time stamp = '1588202948571'). The 4<sup>th</sup> record shows an image file (item\_type =1) was received (state = 0) from the contact *eve01@tigase.im* on the 29<sup>th</sup> Apr. 2020 at 11:31:31 pm UTC ( Unix time stamp = '1588203091298'). Both the 9<sup>th</sup> and 19<sup>th</sup> records in the figure show information associated with unencrypted messages exchanged with the blocked contact (*bobuser@xabber.org*) and deleted contact (*alice.behemoth@yax.im*) respectively.

id	account	jid	timestamp	item_type	data	state	encryption
1	behemoth@tigase...	eve01@tigase.im	1588202948571	0	Good day Eve	9	1
2	behemoth@tigase...	eve01@tigase.im	1588202967901	0	Hello how are you?	0	0
3	behemoth@tigase...	eve01@tigase.im	1588203030251	0	Fine thanks	9	1
4	behemoth@tigase...	eve01@tigase.im	1588203091298	1	aesgcm://sure.im:8443/upload/904...	0	1
5	behemoth@tigase...	eve01@tigase.im	1588203192394	0	Nice app	9	1
6	behemoth@tigase...	eve01@tigase.im	1588203230714	0	Encryption off	9	0
7	behemoth@tigase...	eve01@tigase.im	1588203241615	0	Yes now	0	0
9	behemoth02@xab...	bobuser@xabber.org	1588203437682	0	Hello Bob	9	0
10	behemoth02@xab...	bobuser@xabber.org	1588203469661	0	Hi Behemoth	0	0
11	behemoth@tigase...	alice.behemoth@tigase...	1588203665928	0	Hello Alice	9	1
12	behemoth@tigase...	alice.behemoth@tigase...	1588203674366	0	Hi mate	0	1
13	behemoth@tigase...	alice.behemoth@tigase...	1588203682468	0	Is rather very late	9	1
14	behemoth@tigase...	alice.behemoth@tigase...	1588203704166	0	Yes bored?	0	1
15	behemoth@tigase...	alice.behemoth@tigase...	1588203720055	1	aesgcm://sure.im:8443/upload/9a2...	9	1
16	behemoth@tigase...	alice.behemoth@tigase...	1588203889202	1	aesgcm://sure.im:8443/upload/d4ff...	0	1
17	behemoth@tigase...	eve01@tigase.im	1588204022839	1	aesgcm://sure.im:8443/upload/e2d...	0	1
18	behemoth@tigase...	eve01@tigase.im	1588204102389	1	https://sure.im:8443/upload/ec2d4...	0	0
19	behemoth@tigase...	alice.behemoth@yax.im	1588204173406	0	Hi Alice	9	0
20	behemoth@tigase...	eve01@tigase.im	1588204362092	0	Going to bed now	9	0
21	behemoth@tigase...	eve01@tigase.im	1588204400817	0	Ok see you soon	9	1
22	behemoth@tigase...	alice.behemoth@yax.im	1588204439688	0	Testing your other account	0	0

Figure 10: *chat\_history* table

Moreover, the 51<sup>st</sup> - 53<sup>rd</sup> records as shown in Figure 11, show contents of encrypted messages in plaintext (encryption=1) sent (state = 9) to the deleted contact (*alice.behemoth@yax.im*) on the 1<sup>st</sup> May 2020 at 11:35:55 pm UTC, 11:36:05 pm UTC and 11:36:49 pm UTC respectively ( Unix time stamp = ‘1588376155096’, ‘1588376165590’ and ‘1588376209615’) stored in the table.

id	account	jid	timestamp	item_type	data	state	encryption
51	behemoth@tigase.im	alice.behemoth@yax.im	1588376155096	0	Hi	9	1
52	behemoth@tigase.im	alice.behemoth@yax.im	1588376165590	0	Cool it works	9	1
53	behemoth@tigase.im	alice.behemoth@yax.im	1588376209615	0	Good	9	1

Figure 11: *chat\_history* table ( encrypted messages stored in plaintext)

- ii. Record of group creation and messages exchanges with members within a group are stored in *chat\_history* table. To show records of group creation, the chronology of messages exchanged and participants in group chats, we present some of the contents of the messages exchanged and stored in this table as shown in Figure 12. From the 61<sup>st</sup> record in the figure, we see the local user

(*behemoth@tigase.im*) creates a private group on the 18<sup>th</sup> of Dec 2020 at 8:39:21 pm UTC ( Unix time stamp = ‘1608323961348’) with the message (data=‘Welcome! You created new Multi User Chat Room. Room is locked now. Configure it please!’). From the 63<sup>rd</sup>, the local user/administrator( author\_nickname = PrivateGroup1) sends (state=1) a message (item\_type=0) to members in the group on the 18<sup>th</sup> of Dec 2020 at 8:39:30 pm UTC (Unix time stamp= ‘1608323970852’). From the 64<sup>th</sup> record in the table, local user receives (state= 0) a message (item\_type=0) from group member (author\_nickname=alice.behemoth@yax.im) on the 18<sup>th</sup> of Dec 2020 at 8:41:53 pm UTC (Unix time stamp= ‘1608324113848’). These records remain in the database upon deletion of messages, group members and the chat group.

Table: chat\_history

	account	jid	author_nickname	timestamp	item_type	data	state
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
61	behemoth@tigase.im	484C7B83-A811-419D...	NULL	1608323961348	0	Welcome! You created new Multi User Cha...	0
62	behemoth@tigase.im	484C7B83-A811-419D...	NULL	1608323962442	0	Room is now unlocked	0
63	behemoth@tigase.im	484C7B83-A811-419D...	PrivateGroup1	1608323970852	0	Hi	1
64	behemoth@tigase.im	8AC9AC6F-936A-4BA...	alice.behemoth	1608324113848	0	@alice.behemoth hi	0
65	behemoth@tigase.im	8AC9AC6F-936A-4BA...	behemoth	1608325348522	0	Hello all and welcome to the group	1
66	behemoth@tigase.im	8AC9AC6F-936A-4BA...	behemoth	1608325383235	0	Good to have you here for a private chat	1
67	behemoth@tigase.im	8AC9AC6F-936A-4BA...	alice.behemoth	1608325400543	0	Thanks	0

Figure 12: chat\_history table ( group messages exchanged)

iii. Record of multimedia files exchanged is stored in the “data” field of the chat\_history table. Like the Monal app, the relative URL path of an image exchanged without OMEMO encryption turned on can be retrieved by just entering the URL link stored in the field in a browser. For example, (<https://sure.im:8443/upload/<hash-value>/image.jpg>). However, only the relative path of files sent using encryption is stored in this field and cannot be retrieved. Raw copies of files received and downloaded on the iOS device by the local user are stored in the “/private/var/mobile/Containers/Data/Application/org.tigase.messenger.mobile/Library/Application Support/download” directory of the iOS file system. However, if the local user clears the download cache by clicking on the “Clear download cache” feature on the app, all downloaded files are deleted from this directory.

## 6.6. Limitations

Siskin IM app provides VoIP features that allow users to make audio and video calls. In our experiment, we initiated both audio and video calls between the local user and contacts. However, from our analysis, there was no record of call logs stored in the main *siskinim\_main.db* database or in any of the user data locations on the iOS file system associated with the app. An indication of this was observed in the app's chat window as there was no record of calls made, received, or missed displayed. We conducted a further analysis using Cellebrite's file system extraction that fully integrates the checkm8 exploit to extract a file system image of the iOS device. However, data recovered from the database and file system did not include records of call logs. Similar to the Monal app, attempts to recover missing data from the database using *Undark v 0.6* (Daniels, 2015) and *Cellebrite Physical Analyzer SQLite* recovery tools were unsuccessful as the fields did not show records of call logs within the main *siskinim\_main.db* database of the Siskin IM app.

## 7. Conclusion

In this study, mobile device forensics analysis on the Monal and Siskin IM secure XMPP apps installed and running on iOS was conducted. The focus of the study was to identify and analyze artifacts of forensic interests stored on the iOS file system and application's main folders. We were able to identify artifacts left behind on each app and have presented how they can provide detailed information of investigative value.

In our analysis of the Monal app, we were able to access the database maintained by the app. All useful information associated with the local user and contacts stored in plaintext were recovered. Critical information which includes all messages exchanged (encrypted and unencrypted) is stored in plaintext, a chronology of these messages and URL links containing all images exchanged without OMEMO encryption were shown. However, information associated with deleted messages exchanged with both active and deleted contacts was not found or recovered from the app's main database. We were able to recover all multimedia files exchanged including raw copies of deleted ones stored in the image cache folder of the application's main folder.

In our analysis of the Siskin IM app, we were able to access the main database maintained by the app. We identified, analyzed, and recovered information associated with the local user's multiple IM accounts, information associated with all contact IM accounts (active and

deleted), and textual content of all messages both exchanged whether they were sent encrypted or not. Multimedia files that have not been cleared from the app's image cache were also shown. As discussed in Section 6.6, there was no record of call logs in any of the database entries.

This study can assist forensic investigators to interpret Monal and Siskin IM apps artifacts retrieved from iOS devices. It can also be of value to developers of forensic tools in developing software applications that can decode all relevant data related to these apps and similar multi-client XMPP IM apps to help reconstruct past communications.

## **Acknowledgments**

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

## **Conflict of Interest**

None.

## **References**

- Akbal, E., Baloglu, I., Tuncer, T., Dogan, S., 2019. Forensic analysis of BiP Messenger on android smartphones. *Aust. J. Forensic Sci.* 1–20.  
<https://doi.org/10.1080/00450618.2019.1610064>
- Alyahya, T., Kausar, F., 2017. Snapchat Analysis to Discover Digital Forensic Artifacts on Android Smartphone. *Procedia Comput. Sci.* 109, 1035–1040.  
<https://doi.org/10.1016/j.procs.2017.05.421>
- Anglano, C., 2014. Forensic analysis of WhatsApp Messenger on Android smartphones. *Digit. Investig.* 11, 201–213. <https://doi.org/10.1016/j.diin.2014.04.003>
- Anglano, C., Canonico, M., Guazzone, M., 2017. Forensic analysis of Telegram Messenger on Android smartphones. *Digit. Investig.* 23, 31–49.  
<https://doi.org/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. *Digit. Investig.* 19, 44–59.  
<https://doi.org/10.1016/j.diin.2016.10.001>
- Anu, 2019. Monal Has Omemo [WWW Document]. *monal.im*. URL <https://monal.im/blog/monal-has-omemo/> (accessed 4.29.20).
- Cellebrite, 2020. Cellebrite UFED 4PC and Physical Analyzer.
- checkra1n, 2020.



- Clement, J., 2019. Most popular messaging apps 2019 | Statista [WWW Document]. statista.com.
- Daniels, P., 2015. Undark - a SQLite deleted and corrupted data recovery tool.
- Dargahi, T., Dehghantanha, A., Conti, M., 2017. Forensics Analysis of Android Mobile VoIP Apps, in: Contemporary Digital Forensic Investigations of Cloud and Mobile Applications. Elsevier, pp. 7–20. <https://doi.org/10.1016/B978-0-12-805303-4.00002-2>
- David Nield, Brian Turner, 2020. Best encrypted instant messaging apps 2020 for Android [WWW Document]. Techradar.com. URL <https://www.techradar.com/uk/best/best-encrypted-messaging-app-android#4-threema> (accessed 4.30.20).
- Express VPN, 2019. How to keep your messages private and anonymous [WWW Document]. URL <https://www.expressvpn.com/blog/anonymous-chat-services/> (accessed 4.18.20).
- Gultsch, D., 2016. The State of Mobile XMPP in 2016 [WWW Document]. gultsch.de. URL [https://gultsch.de/xmpp\\_2016.html](https://gultsch.de/xmpp_2016.html) (accessed 4.30.20).
- Hexegic, 2020. The Rise of Instant Messaging Apps and their use by Extremist Groups [WWW Document]. hexegic.com. URL <https://www.hexegic.com/blog/the-rise-of-instant-messaging-apps-and-their-use-by-extremist-groups/> (accessed 4.30.20).
- Hintea, D., Sangins, A., Bird, R., 2018. Forensic analysis of the telegram instant messenger application on android devices, in: European Conference on Information Warfare and Security, ECCWS.
- Jadhav Bhatt, A., Gupta, C., Mittal, S., 2018. Network Forensics Analysis of iOS Social Networking and Messaging Apps, in: 2018 Eleventh International Conference on Contemporary Computing (IC3). IEEE, pp. 1–6. <https://doi.org/10.1109/IC3.2018.8530576>
- Jeon, S., Bang, J., Byun, K., Lee, S., 2012. A recovery method of deleted record for SQLite database. Pers. Ubiquitous Comput. <https://doi.org/10.1007/s00779-011-0428-7>
- Kim, D., Lee, S., 2020. Study of identifying and managing the potential evidence for effective Android forensics. Forensic Sci. Int. Digit. Investig. 200897. <https://doi.org/10.1016/j.fsidi.2019.200897>
- Mahajan, A., S. Dahiya, M., P. Sanghvi, H., 2013. Forensic Analysis of Instant Messenger Applications on Android Devices. Int. J. Comput. Appl. 68, 38–44. <https://doi.org/10.5120/11602-6965>
- Norouzizadeh Dezfouli, F., Dehghantanha, A., Eterovic-Soric, B., Choo, K.-K.R., 2016. Investigating Social Networking applications on smartphones detecting Facebook, Twitter, LinkedIn and Google+ artefacts on Android and iOS platforms. Aust. J. Forensic Sci. 48, 469–488. <https://doi.org/10.1080/00450618.2015.1066854>
- Ovens, K.M., Morison, G., 2016. Forensic analysis of Kik messenger on iOS devices. Digit. Investig. 17, 40–52. <https://doi.org/10.1016/j.diin.2016.04.001>
- Shortall, A., Azhar, M.A.H. Bin, 2015. Forensic Acquisitions of WhatsApp Data on Popular Mobile Platforms, in: 2015 Sixth International Conference on Emerging Security Technologies (EST). IEEE, pp. 13–17. <https://doi.org/10.1109/EST.2015.16>

- Statista, 2020. Share of global smartphone shipments by operating system from 2014 to 2023 [WWW Document]. Statista.com. URL <https://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/> (accessed 4.18.20).
- Sudozai, M.A.K., Saleem, S., Buchanan, W.J., Habib, N., Zia, H., 2018. Forensics study of IMO call and chat app. *Digit. Investig.* 25, 5–23. <https://doi.org/10.1016/j.diin.2018.04.006>
- Tigase, 2020. Siskin IM [WWW Document]. siskin.im. URL <https://siskin.im/> (accessed 4.30.20).
- van Dongen, W.S., 2007. Forensic artefacts left by Pidgin Messenger 2.0. *Digit. Investig.* 4, 138–145. <https://doi.org/10.1016/j.diin.2008.01.002>
- Wu, S., Zhang, Y., Wang, X., Xiong, X., Du, L., 2017. Forensic analysis of WeChat on Android smartphones. *Digit. Investig.* 21, 3–10. <https://doi.org/10.1016/j.diin.2016.11.002>
- XMPP, 2018. OMEMO Media sharing [WWW Document]. xmpp.org. URL <https://xmpp.org/extensions/inbox/omemo-media-sharing.html> (accessed 4.30.20).
- XMPP, 2013. XEP-0292: vCard4 Over XMPP [WWW Document]. xmpp.org. URL <https://xmpp.org/extensions/xep-0292.html> (accessed 4.28.20).
- XMPP Standards Foundation (XSF), 2020a. An Overview of XMPP [WWW Document]. About XMPP. URL <https://xmpp.org/about/technology-overview.html> (accessed 4.18.20).
- XMPP Standards Foundation (XSF), 2020b. XEP-0384: OMEMO Encryption [WWW Document]. URL <https://xmpp.org/extensions/xep-0384.html> (accessed 4.18.20).

