Research Article

Forensic analysis of open-source XMPP/Jabber multi-client instant messaging apps on Android smartphones

Alex Akinbi[1] · Ehizojie Ojie[2]

Received: 10 August 2020 / Accepted: 24 February 2021 / Published online: 6 March 2021

Abstract
In the quest for a panacea to ensure digital privacy, many users have switched to using decentralized open-source Extensible Messaging and Presence Protocol multi-client instant messaging (IM) apps for secure end-to-end communication. In this paper, we present a forensic analysis of the artefacts generated on Android smartphones by Conversations and Xabber apps. We identified databases maintained by each app and external Secure Digital card directories that store local copies of user metadata. We analysed each app's storage locations for forensic artefacts and how they can be used in a forensic investigation. The results in this paper show a detailed analysis of forensic files of interest which can be correlated to identify the local user's multiple IM accounts and contact list, contents of messages exchanged with contacts, deleted files, time, and dates in the order of their occurrence. The contributions of this research include a comprehensive description of artefacts, which are of forensic interest, for each app analysed.

Keywords Mobile forensics · Android forensics · Jabber · Conversations · Xabber · Social networking · End-to-end encryption

1 Introduction

Most instant messaging (IM) apps provide a free social networking service to communicate with friends, family, and colleagues [1]. However, many users are concerned that their private messages could be read by the service providers that own these apps, as well as other third parties and even governments who like to gather their citizen's private data. This has led to the wide adoption of secure IM apps that provide end-to-end encryption (E2EE), a method of encrypting data that only allows the sender and receiver of the message to decrypt and read messages passed between them [1]. Examples include *Telegram, Signal, iMessage, Viber, WhatsApp, Wire, Wickr,* etc. Given their popularity, these E2EE IM apps and services are being increasingly used not only for legitimate activities but also for illicit ones [2, 3]. Therefore, forensic analysis of these apps continues to pose challenges to law enforcement involved with criminal investigations where such apps have been used as a means of secure communication in a crime.

Although privacy can be guaranteed using these E2EE IM apps, anonymity is still considered daunting for its users. These E2EE IM apps provide their users with a high degree of privacy, and the app providers cannot read the contents of the messages. However, the providers operate both centralized ecosystem services that enable them to still have access to information such as user identity, the identity of user contacts, and IP addresses. It is also difficult to create and discard accounts and often impossible to run accounts simultaneously, or switch between them [4]. This issue of maintaining both privacy and anonymity has led to the growing adoption of open-source

✉ Alex Akinbi, o.a.akinbi@ljmu.ac.uk; Ehizojie Ojie, ehizojie.ojie@york.ac.uk | [1]School of Computer Science and Mathematics, Liverpool John Moores University, Byrom Street, Liverpool L3 3AF, UK. [2]Department of Computer Science, University of York, Heslington, York YO10 5DD, UK.

SN Applied Sciences (2021) 3:430 | https://doi.org/10.1007/s42452-021-04431-9

Extensible Messaging and Presence Protocol (XMPP) client instant messaging apps. XMPP was originally developed in the Jabber open-source community to provide open, decentralized alternative instant messaging services and offers several key advantages over centralized or closed ecosystem services [5].

These key advantages include open-source code (so it is possible to audit its code), it provides message encryption, partner authentication, deniability, and perfect forward secrecy through the use of the Off-The-Record (OTR), a cryptographic protocol that provides encryption for instant messaging conversations [6]. Another advantage is the use of Multi-End Message and Object Encryption (OMEMO), an open standard based on a Double Ratchet and Personal Eventing Protocol (PEP) for secure multi-client end-to-end encryption [7]. Users can maintain anonymity on OTR by using multiple accounts, connecting over a virtual private network (VPN)/The Onion Router (TOR), or connecting through private XMPP servers. Therefore, with such characteristics of providing both E2EE and anonymity, interest in the forensic analysis of decentralized open-source client instant messaging apps built using XMPP/Jabber and supports OTR or OMEMO is apparent.

In this paper, we deal with the forensic analysis of *Conversations* and *Xabber* apps. Two popular decentralized open-source XMPP/Jabber client instant messaging apps on Android smartphones with 100,000+ and 1,000,000+ downloads respectively from the Google Play Store at the time of writing this paper. Smartphones running the Android operating system held an 87% share of the global market in 2019 compared to the mobile operating system developed by Apple (iOS), which had a 13% share of the market [8]. Therefore, by focusing on the Android platform, we make the most of the potential investigative impact of our work. There has been no published research addressing forensic analysis on *Conversations* and *Xabber* apps on Android smartphones.

Our original contributions in this paper are the exploration and analysis of Conversations and Xabber apps as summarised below:

- From our study and findings, we identified local copies of messages and files exchanged (forensic artefacts) between the user and other contacts that are stored in the main databases and of both apps and file system of the Android device.
- We demonstrate how these forensic artefacts can be correlated together and interpreted to infer various types of information which include the timeline of messages and files exchanged in the order of their occurrence and highlight any evidence of deletion.
- We reveal that the Conversations app which uses OMEMO for E2EE communication, store information associated with the local user, contacts and body of messages sent or received using the app as plaintext in the main database maintained by the app.
- We reveal that Xabber app which supports OTR for E2EE communication, stores information associated with the local user, contacts, and body of messages sent or received using the app as plaintext in the app's main databases.
- We reveal how to map and interpret the data stored by both apps to the local user's actions that generated them.

This paper is organized as follows. In Sect. 2, we discuss related works. In Sect. 3, we discuss the experiment design, analysis methodology and tools used in our experiments. In Sect. 4, we discuss the investigative scenario used in our experiments. Forensics analysis and findings of Conversation and Xabber apps including artefacts recovered are presented in Sect. 5 and Sect. 6 respectively. Finally, in Sect. 7, we discuss our findings and in Sect. 8 we conclude the paper.

2 Related works

Most studies of smartphone forensics have focused on the extraction and analysis of data obtained from the device flash memory. Forensic analysis of social networking applications has been conducted on Android devices as demonstrated in several studies [9–11]. Recently, there have been limited works that have focused on forensic analysis of open-source decentralized XMPP/Jabber client IM apps on Android smartphones. The most notable one is Anglano et al. [3] forensic analysis of *ChatSecure* on Android platforms to recover forensic artefacts related to the chronology and contents of chat messages and decrypt the *SQLCipher* databases. Akinbi and Ojie [12] conducted forensic analysis on *Monal* and *Siskin IM* decentralized open-source XMPP apps on iOS devices. Before both studies, Wouter S. van Dongen [13] conducted a forensic analysis of *Pidgin Messenger 2.0* on the Linux platform. The *ChatSecure* app on Android supports OTR for E2EE and has since been deprecated on the Android platform and is currently only available to iOS users [14]. Many studies have focused on popular centralized IM apps on Android smartphones and demonstrated in studies such as *Wickr* [15], *WhatsApp* [16], *Telegram* [17], and *IMO* [18]. However, forensic analysis of decentralized open-source XMPP/Jabber E2EE multi-client IM apps on Android devices have been largely ignored, even as they are currently being used to send encrypted messages by many users. It is not clear why there are limited studies on forensic analysis of decentralized open-source XMPP/Jabber E2EE multi-client

IM apps. However, a recent breakthrough by law enforcement agents in the arrest of organised criminals using EncroChat secure phones and cryptographic messaging systems which implements OTR [19], shows the forensic analysis of these messaging systems are necessary. To the best of our knowledge, no recent studies have focused on forensic analysis of Conversations and Xabber decentralized open-source XMPP/Jabber E2EE multi-client IM apps on Android (with over 100,000 and 1,000,000 downloads on Google Play Store respectively) which are interoperable with other XMPP clients that support OMEMO or OTR encryption protocols.

3 Design of experiment, analysis methodology and tools

The design of the experiment required a set of controlled actions which were conducted separately for the Conversations and Xabber apps. The analysis methodology is focused on the identification and recovery of forensic artefacts generated by both apps and stored on the Android device's internal memory and external Secure Digital (SD) card. In each experiment, we installed and ran the current versions available on the Google Play Store (at the time of writing) which were Conversations v. 2.7.1+ pcr and Xabber v. 2.6.6.645. We then proceeded to create IM accounts for a local user, several contacts' IM accounts and carried out a set of actions to generate forensic artefacts. The order of actions performed in our experiments for both Conversations and Xabber apps to generate forensic artefacts and to create a realistic scenario for a typical user include the following:

1. Disable the use of E2EE encryption for communication.
2. Exchange regular chat messages and files between the user and all contacts.
3. Delete some messages and files sent and received by the local user.
4. Enable the use of E2EE encryption for communication.
5. Verify contacts' encryption keys which ensure forward secrecy and secure message communication.
6. Exchange regular chat messages and files between the user and all contacts.
7. Delete some messages and files sent and received by the local user.
8. Block and delete one contact.

These set of actions in our experiments were played manually over a period to generate forensic traces [20, 21], which can later be analysed based on our investigative scenario (Sect. 4). These actions also ensure the experiments can be generalized, comprehensive and reproduced by a third party under the same operational conditions to achieve the same results [17]. In our analysis, most of the files and artefacts generated are stored on the internal device memory which is normally inaccessible to users. To access the internal device memory and recover evidential data, we used *Cellebrite UFED 4PC v*. 7.32 to obtain a physical image and analysed the evidential data using *Cellebrite Physical Analyzer v*. 7.31 [22]. Both tools are suitable commercial forensic tools used to maintain forensic soundness. At present, Cellebrite supports digital investigation on various third-party Android applications and can extract data from unrooted Android smartphones by exploiting certain bootloader vulnerabilities that exist in many devices running operating system versions up to Android 9 (Pie). To achieve results that are close to realistic scenarios, we used two unrooted Samsung Galaxy S8+ Android devices running Android 9 to conduct our experiments and analysis.

4 Investigative scenario

Conversations and Xabber are open-source XMPP multi-client IM applications that allow their users to communicate securely via their existing accounts on IM providers that use the XMPP protocol. To demonstrate the forensic analysis, interpretation of results in this paper, and how it can be applied in the context of a forensic investigation, we derived questions similar to the ones in the forensic analysis of ChatSecure by Anglano et al. [3] and created an investigative scenario which is described as follows:

Both the Conversations and Xabber apps are installed on an Android device which is being examined for evidential data and forensic files of interest. Forensic investigators are keen to extract digital evidence and answer the following questions:

i. How many unique XMPP IM accounts associated with the local users were configured and used with the Conversations app and Xabber app?
ii. What are the identities and XMPP IM accounts associated with contacts of the local user for each app?
iii. What is the timeline of communication with each of the contacts and what messages were exchanged?
iv. Is there evidence of file exchanges between the local user and contacts? If yes, when did these exchanges occur and what is the content of such files?
v. Can encrypted messages be recovered from the databases maintained by both apps?
vi. Can deleted data be recovered from both apps?

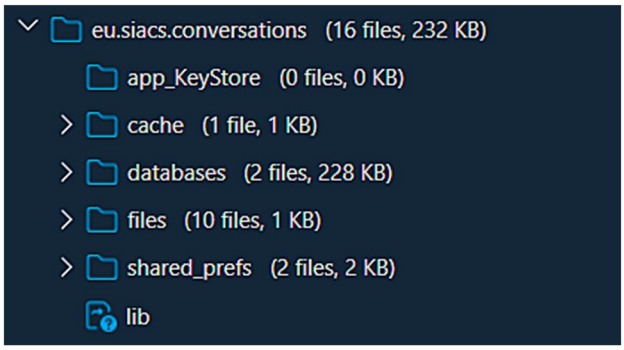

Fig. 1 Main folder structure of the Conversations app

In the following sections, we present the forensic analysis of the Conversations and Xabber apps respectively.

5 Forensic analysis of Conversations app

Conversations is a secure, decentralized, and open-source Jabber/XMPP multi-client IM app for Android 4.0+ smartphones that allow users to communicate securely and does not collect or store user information that could be inferred to identify the user [23]. On installation, Conversations places the application paths in the main folder *"/data/data/eu.siacs.conversations/"* and external storage card *"/storage/emulated/0/Conversations"* on the Android device as shown in Fig. 1. By default, the app requires the user to set up an existing account by specifying an XMPP address (username) and password or register a new account on the Conversation XMPP server and a random password is automatically generated and saved into the user account. Users can also add multiple accounts to an existing account and use them at the same time to communicate with other IM contacts. We created two XMPP IM accounts associated with the local user and three other contact XMPP IM accounts (*Roster or Buddylist*) to exchange messages with the local user. Details on these IM accounts are as follows (Table 1).

The Conversations app uses OMEMO by default for encrypting conversations and the user can toggle between sending messages unencrypted or encrypted using OMEMO or OpenPGP. Other functionalities include contact and account management, notification, privacy, and expert settings (to tunnel all connections through Tor), contact management, and verification of counterpart identity. The data such as chat records, configurations generated during the running of Conversations is stored in five subdirectories of the folder *"eu.siacs.conversations"*, they are *"app_KeyStore" "cache", "databases", "files"* and *"shared_prefs"* (see Fig. 1).

The *shared_prefs* and *files* subdirectories contain several activity files, default preferences settings files, media, and configuration files. For extensiveness, we mention a file in the *shared_prefs* subdirectory named *eu.siacs.conversations_preferences.xml* which stores the app's settings and preferences. The most crucial evidential data of forensic interest are stored in an unencrypted SQLite database named *"history"* in the databases subdirectory under the directory path *"/data/data/eu.siacs.conversations/databases/history"*. The path *"/storage/emulated/0/Conversations"* is used to store multimedia resources such as sent and received images, audio, and video files. This is the directory path for the external SD (memory) card which is accessible by attaching the Android device to a PC using a USB cable (see Table 2).

Table 1 Conversations local user and contact IM accounts

Local user's XMPP IM accounts analysed Android smartphone	behemoth@conversations.im myotherbehemothaccount@conversations.im
Contacts' XMPP IM accounts on other Android devices	Contact 1: bob_behemoth@conversations.im Contact 2: behemoth01@conversations.im Contact 3: alice_behemoth@conversations.im
Deleted and blocked contact's XMPP IM account	alice_behemoth@conversations.im

Table 2 File paths of critical evidence sources of the Conversations app

Directory path	Details
/data/data/eu.siacs.conversations/databases/history	The unencrypted database containing chat messages and local user account information
/storage/emulated/0/Conversations/Media/	External SD card location where local copies of raw multimedia relating to audio and video messages are stored

5.1 Location of Conversations app artefacts

The *history* database stores information generated by the Conversations app and associated user activities. These include user information, the list of the corresponding contacts, and local copies of messages that have been exchanged in plaintext. This unencrypted database contains 18 different tables. From our findings, only 10 out of these 18 tables contain information of forensic interest namely tables *accounts, contacts, conversations, identities, messages, messages_index, messages_index_content, prekeys, signed_prekeys, and sessions* (See Fig. 2). We now discuss the contents of these tables along with their mapping to the user accounts and activities to answer questions from our investigative scenario.

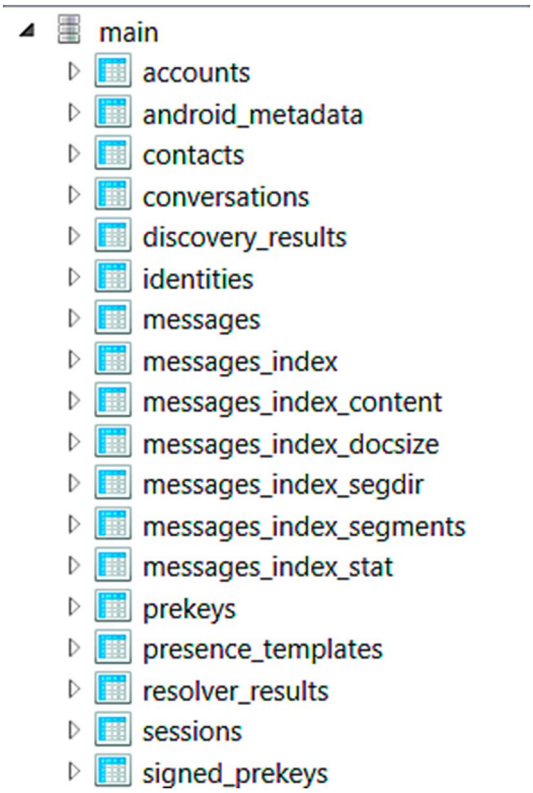

Fig. 2 Structure of the main *history* database

5.2 Recovering account information

The *accounts* table contains information about the local user account or multiple IM accounts set up by the user including passwords for each one and stored in plaintext. However, according to the privacy policy of the app's developers, user passwords are stored as hashes on Conversations' XMPP servers [23]. Each IM account username (XMPP address) is assigned a unique user identifier (primary key) named *"uuid"* in the table. From our investigative scenario, we found both distinct accounts for a local user named *behemoth@conversations.im* and *myotherbehemothaccount@conversations.im* stored in the table. Both accounts were active and used to exchange messages with contacts. The first part of each account is stored in the field "*username*" and domain part *@conversations.im* stored in the field named "*server*", while the avatar field stores a unique file name associated with the user's account image file stored in the subdirectory "*/data/data/eu.siacs.conversations/files/avatars/*" and named *8f44441a2833f9542c14b2a258663a83420aed0b* (see Fig. 3). Avatar images can help investigators reveal the identity of a local user or contacts if the avatar shows the face or feature that can be distinctly associated with the individual.

The *contacts* table stores information associated with active contacts the local user has added and exchanges messages or files with. Information about deleted or blocked contacts is not stored in this table. Each contact is associated with a local user IM account identified by the *uuid* from the *accounts* table and subsequently stored in the "*accountUuid*" field. The contacts XMPP address (usernames) are stored in the *"jid"* field and the last time of message exchanged is stored in the "*last_time*" field. Other information about each contact such as avatar and chat group assigned, are stored in the "*avatar*" and "*groups*" fields respectively.

Information about the first conversation exchanged by the local user with all unique contacts is stored in the *conversations* table. This includes information about active, deleted, or blocked contacts. Each conversation

Table: accounts

New Record

	uuid	username	server	password	avatar
	Filter	Filter	Filter	Filter	Filter
1	cda04d89-6c4a-4f2d-bd16-b2a93f61761b	myotherbehemothaccount	conversations.im	password123	NULL
2	6e0dd0fb-f260-4461-bfff-d461c0af9274	behemoth	conversations.im	v21R1csXAs	8f44441a2833f9542c14b2a258663a83420aed0b

Fig. 3 Fields of *account* table (fields containing irrelevant data are hidden)

is assigned a unique identifier stored in the "*uuid*" field, the associated contact username is stored in the "*name*" field and the full contact XMPP address is stored in the "*contactJid*" field. The "*created*" field contains the Unix epoch time the first message was sent to or received from each contact. The *identities* tables store all only active XMPP addresses which include the multiple local user's IM accounts and associated contacts identified by unique identifiers stored in the "*account*" field.

This table does not store information associated with deleted or blocked contacts. Each unique identifier in the "*account*" field is a foreign key in the *conversations* table stored in the *"accountUuid"* field. This key shows the relationship between a contact that was added by a specific local user IM account. The IM accounts associated with the local user are assigned the integer 1 and 0 for contacts in the "*ownkey*" field. Information on verified and trusted IM accounts are stored in the "*trust*" field. Figure 4 shows the foreign key relationship between the *conversations* and *identities* tables and highlighting the deleted contact IM account information stored in the *conversations* table. The *conversations* table stores information associated with the creation or start of a message with a contact (whether active or deleted) by a local user using a specific IM account.

5.3 Recovering chronology of chat logs, message contents, and deleted files

Each time a message is sent or received, the Conversations app stores details of both the textual content and associated metadata in the *messages, messages_index,* and *messages_index_content* tables. The *messages_index* and *messages_index_content* tables are similar as they both contain the body of each message stored in fields named "*body*" and "*c1body*" respectively and each associated with unique identifiers. However, the *messages* table is the main table in the database. It contains a detailed record of all textual messages sent and received, the chronological time of when each message was sent or received, and whether each message was sent encrypted or unencrypted. The *messages* table can be joined to the *conversations* table by the field named "*conversationUuid*". Also, the *conversations* table can be joined to the *accounts* table by the field named *"accountUuid"*. These joins and relationships between all three tables can help to identify the correlation of messages exchanged using a specific local user's

Table: identities

	account	name	ownkey
	Filter	Filter	Filter
1	6e0dd0fb-f260-4461-bfff-d461c0af9274	behemoth@conversations...	1
2	6e0dd0fb-f260-4461-bfff-d461c0af9274	behemoth01@conversati...	0
3	cda04d89-6c4a-4f2d-bd16-b2a93f61761b	myotherbehemothaccoun...	1
4	6e0dd0fb-f260-4461-bfff-d461c0af9274	bob_behemoth@conversa...	0

Table: conversations

	uuid	name	accountUuid
	Filter	Filter	Filter
1	78b473ab-8ea3-47cf-b975-53a5e797e98c	conversations.im	6e0dd0fb-f260-4461-bfff-d461c0af9274
2	d52cc163-38b2-4c14-89fe-aa11712a0467	behemoth01	6e0dd0fb-f260-4461-bfff-d461c0af9274
3	fa45043a-4906-4432-bd7b-111feef8ed15	conversations.im	cda04d89-6c4a-4f2d-bd16-b2a93f61761b
4	2ca2c8b2-0c37-4a17-902d-e57d90bf701b	alice_behemoth	6e0dd0fb-f260-4461-bfff-d461c0af9274
5	984d12b1-f316-4640-8fb7-eb333c9f8a20	bob_behemoth	6e0dd0fb-f260-4461-bfff-d461c0af9274

Fig. 4 *Conversations* table and *identities* tables

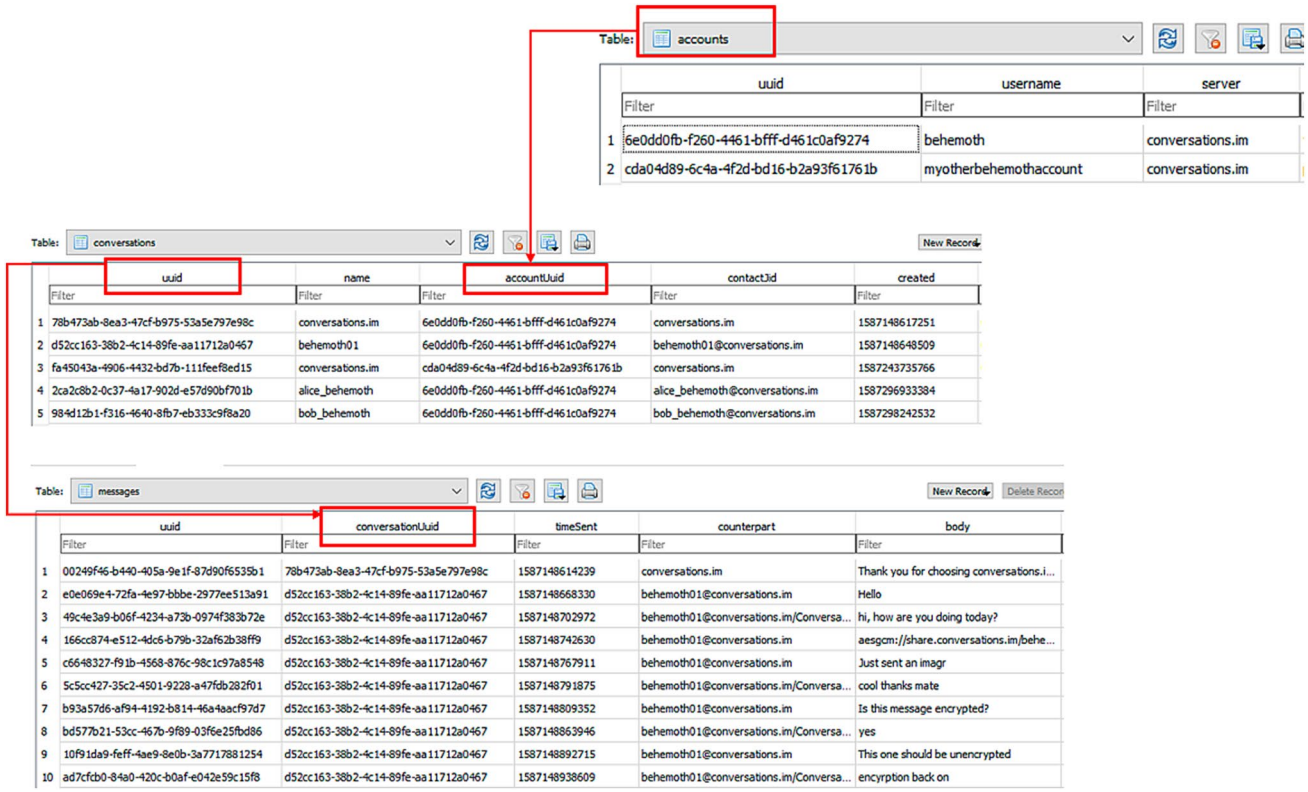

Fig. 5 Reconstruction of user accounts and messages

IM account with a contact. Moreover, it can be used to identify when the first message was created and sent to a contact using a specific user IM account. Figure 5 illustrates the relationships and joins between the *accounts*, *conversations,* and *messages* tables which can help reconstruct distinct Conversations accounts with messages.

In the figure, we see two distinct Conversations accounts, named *behemoth* and *myotherbehemothaccount* (see the records stored in accounts in Fig. 5). Both these accounts have a unique identifier *"uuid"* which is a foreign key in the *conversations* table. The *"uuid"* field (primary key) in the *conversations* table appears as a foreign key in the *messages* table in the field *"conversationUuid"*. Hence we see from the 2nd record in the *conversations* table (uuid = d52cc163…), the *behemoth@conversations.im* user account (accountUuid = 6e0ddfb…) started a conversation (created first message) with the contact *behemoth01* (contactJid = behemoth01@conversations.im) on the 17th of Apr, 2020 at 18:37:28 pm UTC + 1 (created = '1587148648509'). In the 2nd record from the messages table (uuid= e0e069e4… and conversationUuid= d52cc163…), we see details of a message (body= 'Hello') sent on the 17th of Apr, 2020 at 18:37:48 pm UTC+1 (timeSent= '1587148668330').

Other information stored in the *messages* table includes local user and contacts IM accounts that sent or received messages, the relative path of multimedia files exchanged, message status indicating read, edited, or deleted and unique identifiers for each messages which are stored in the "*uuid"* field. Since all the records of messages and files exchanged are stored in the *messages* table, we can easily reconstruct the sequence of events, contents of chat messages, and show evidence of deleted files. In Fig. 6, we see information associated with only active contact IM accounts (*behemoth01@conversations.im* and *bob_behemoth@conversations.im*) stored in the "*counterpart"* field. The "*timeSent"* and "*body"* fields store the time the communication occurred and the textual content of the message respectively. Record of messages and files exchanged with the deleted and blocked contact (*alice_behemoth@conversations.im*) are not stored in the table.

To demonstrate the chronology and sequence of exchanged messages and files from our investigative scenario, a total of 17 messages exchanged is shown in Fig. 6. From this figure, we see that the first record in the field named "*body*" corresponds to an unencrypted (encryption = 0) incoming message (status = 0) from the *conversations.im* server on the 17th Apr. 2020 at 18:36:54 pm UTC + 1 (Unix time stamp = '1587148614239'). The "*type*"

Table: messages

	uuid	conversationUuid	timeSent	counterpart	body	encryption	status	type
	Filter	Filter	Filter	Filter	Filter	Filter	...	
1	00249f46-b440-40...	78b473ab-8ea3-4...	1587148614239	conversations.im	Thank you for cho...	0	0	0
2	e0e069e4-72fa-4e...	d52cc163-38b2-4c...	1587148668330	behemoth01@conversations.im	Hello	5	2	0
3	49c4e3a9-b06f-42...	d52cc163-38b2-4c...	1587148702972	behemoth01@conversations.im/...	hi, how are you d...	5	0	0
4	166cc874-e512-4d...	d52cc163-38b2-4c...	1587148742630	behemoth01@conversations.im	aesgcm://share.c...	5	2	1
5	c6648327-f91b-45...	d52cc163-38b2-4c...	1587148767911	behemoth01@conversations.im	Just sent an imagr	5	2	0
6	5c5cc427-35c2-45...	d52cc163-38b2-4c...	1587148791875	behemoth01@conversations.im/...	cool thanks mate	0	0	0
7	b93a57d6-af94-41...	d52cc163-38b2-4c...	1587148809352	behemoth01@conversations.im	Is this message e...	5	2	0
8	bd577b21-53cc-46...	d52cc163-38b2-4c...	1587148863946	behemoth01@conversations.im/...	yes	0	0	0
9	10f91da9-feff-4ae...	d52cc163-38b2-4c...	1587148892715	behemoth01@conversations.im	This one should b...	0	2	0
10	ad7cfcb0-84a0-42...	d52cc163-38b2-4c...	1587148938609	behemoth01@conversations.im/...	encyrption back on	5	0	0
11	55508273-60d6-4...	fa45043a-4906-44...	1587243730046	conversations.im	Thank you for cho...	0	0	0
12	b2fb74a7-cc54-48...	d52cc163-38b2-4c...	1587243839908	behemoth01@conversations.im	aesgcm://share.c...	5	2	2
13	ba7930b1-5ec7-4c...	d52cc163-38b2-4c...	1587296143750	behemoth01@conversations.im/...	aesgcm://share.c...	5	0	1
14	c0397114-23e6-4...	d52cc163-38b2-4c...	1587296660938	behemoth01@conversations.im	aesgcm://share.c...	5	2	1
15	84ffd310-d23e-4c...	d52cc163-38b2-4c...	1587296686772	behemoth01@conversations.im	geo:53.5069094,-...	5	2	0
16	502462f8-93cd-4f...	984d12b1-f316-46...	1587298337548	bob_behemoth@conversations.i...	what time will you...	5	0	0
17	e4427ef6-4f79-4c...	984d12b1-f316-46...	1587298351430	bob_behemoth@conversations.im	4 pm	5	8	0

Fig. 6 *Messages* table

field (type = 0), indicates the message is text. The 12th record in the *body* field corresponds to an encrypted (encryption = 5) outgoing message (status = 2) sent on the 18th Apr. 2020 at 21:03:59 pm UTC + 1 (Unix time stamp = '1587243839908') to the contact *behemoth01@conversations.im*. The "*type*" field contains the integer 2 indicating the message exchanged is a video file. In Table 3**,** we presented a detailed interpretation of the relevant fields.

There is no option to delete chat messages from the Conversations app chat window at the time of this writing. However, files exchanged can be deleted, and chat messages can be edited, cleared, or closed. In the scenario, we edited one message and deleted an image from the

Table 3 Structure of the *messages* table

Name	Role	Type	Meaning
uuid	Primary key	Text	Unique identifier of the message
conversationUuid	Foreign key	Text	Unique identifier of the conversation
timeSent	–	Int	The date this message has been sent or received (13-digits Unix epoch format)
Counterpart	–	Text	IM contact message was sent to or received from
Body	–	Text	Body of the message
Encryption	–	Int	Flag indicating whether a message is encrypted (5) or unencrypted (0)
Status	–	Int	Flag indicating whether a message was sent (2 or 8) or received (0)
Type	–	Int	Flag indicating whether the body of a message is text (0), image file (1) or video file (2)
relativeFilePath	–	Text	The relative path of an image or video file sent or received using the AES Galois/Counter Mode of operation
Edited	–	Text	Unique identifier for an edited message
Read	–	Boolean	Flag indicating whether a message is read (1) or unread (0)
Deleted	–	Boolean	Flag indicating whether the body of a message is deleted (1) or kept (0)

Table: messages

	timeSent	counterpart	body	encryption	status	type	edited	deleted
	Filter	Filter	Filter	Filter	F...	...	Filter	Filter
1	1587148614239	conversations.im	Thank you for cho...	0	0	0	[]	0
2	1587148668330	behemoth01@con...	Hello	5	2	0	[]	0
3	1587148702972	behemoth01@con...	hi, how are you d...	5	0	0	[]	0
4	1587148742630	behemoth01@con...	aesgcm://share.c...	5	2	1	[]	0
5	1587148767911	behemoth01@con...	Just sent an imagr	5	2	0	[]	0
6	1587148791875	behemoth01@con...	cool thanks mate	0	0	0	[]	0
7	1587148809352	behemoth01@con...	Is this message e...	5	2	0	[{"edited_id":"...	0
8	1587148863946	behemoth01@con...	yes	0	0	0	[]	0
9	1587148892715	behemoth01@con...	This one should b...	0	2	0	[]	0
10	1587148938609	behemoth01@con...	encyrption back on	5	0	0	[]	0
11	1587243730046	conversations.im	Thank you for cho...	0	0	0	[]	0
12	1587243839908	behemoth01@con...	aesgcm://share.c...	5	2	2	[]	0
13	1587296143750	behemoth01@con...	aesgcm://share.c...	5	0	1	[]	0
14	1587296660938	behemoth01@con...	aesgcm://share.c...	5	2	1	[]	1

Fig. 7 *Messages* table showing record of edited and deleted messages

chat window. The 7th and 14th records correspond to information about this action in the *"edited"* and *"deleted"* (deleted = 1) fields as shown in Fig. 7. Once the deletion occurred, the image was removed from the app's chat window, but a local copy persists on the external SD card in the directory */storage/emulated/0/Conversations/Media/Conversations Images/*. A copy of our deleted image on the local user's phone was recovered in the directory path */storage/emulated/0/Conversations/Media/Conversations Images/c0397114-23e6-4998-9473-3b9e8cea429b.jpg*.

The *prekeys, signed_prekeys,* and *sessions* tables contain information of multiple prekeys and verified keys used by the local user's multiple IM accounts in the forward secrecy and secure message communication with contacts as explained in OMEMO cryptographic analysis [24]. Our experiment and analysis of Conversations were limited to the use of OMEMO for encrypted communication because it is considered to have better encryption features than OpenPGP [7].

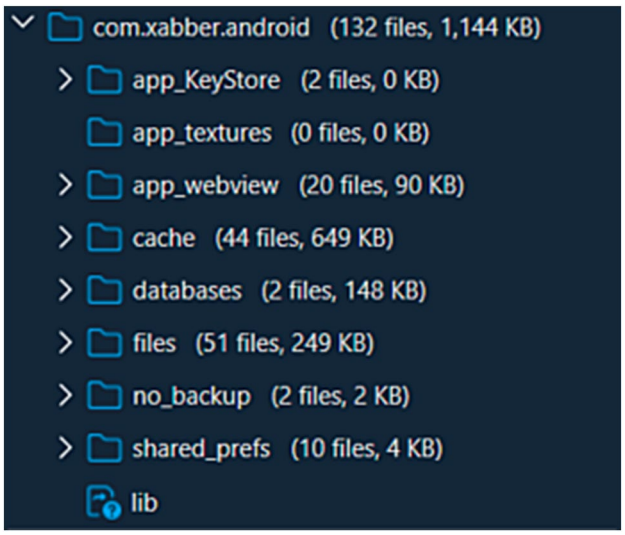

Fig. 8 The folder structure of the Xabber app

6 Forensic analysis of Xabber app

Xabber for Android is a secure, decentralized, and open-source Jabber/XMPP multi-client IM app [25]. On installation, Xabber places the application's path in the main folder *"/data/data/com.xabber.android"* on the Android device as shown in Fig. 8. By default, the app requires the user to set up an existing account by specifying an XMPP address and password or register a new account on the *xabber.org* server. Users can also add multiple accounts to an existing account and use them simultaneously to

communicate with other IM contacts. In our investigative scenario, we created two XMPP IM accounts associated with the local user and two other contact XMPP IM accounts to exchange messages with the local user. One of the contact IM accounts was deleted after a few chat messages were exchanged. Details on these IM accounts are as follows (see Table 4).

The Xabber app supports OTR encryption by default for E2EE message encryption and the user can toggle between sending messages unencrypted or encrypted by switching the OTR plugin mode in the app's security settings. Other setting options include contact, chat and account management, notification, privacy, connection, and debug settings. Forensic artefacts and associated metadata generated by Xabber are stored in 8 subdirectories of the folder *"com.xabber.android"*, they are *"app_KeyStore", "app_textures", "app_web_webview", "cache", "databases", "files", "no_backup"* and *"shared_prefs"* (see Fig. 8).

Like the Conversations app, the Xabber app subdirectories contain several activity files, default preferences settings files, media, and configuration files. The most crucial evidential data of forensic interest are stored in two unencrypted Realm open source object database management system files [26] (*"xabber.realm"* and *"realm_database.realm")* and one unencrypted SQLite database *("xabber.db")*. The path "*/data/data/com.xabber.android/cache/image_manager_disk_cache/*" is used to store multimedia resources such as avatars, sent and received images, audio, and video files. Details for each storage location is described in Table 5.

6.1 Location of Xabber app artefacts

From our findings, the *xabber.realm* database is the main database where the Xabber app stores and maintains the information concerning the accounts used on the app and associated activities. The database stores local copies of the messages in plaintext that have been exchanged and contains 8 different tables. From our findings, only 4 out of these 8 tables contain information of forensic interest namely *class_Attachment, class_ContactRealm, class_MessageItem,* and *pk*. The *realm_database.realm* database stores account information concerning the local user and contact IM accounts. The database consists of 21 tables. From our findings, only 7 out of these 21 tables contain information of forensic interest namely *class_AccountRealm, class_ChatDataRealm, class_UploadServer, class_XabberAccountRealm, class_XMPPUserRealm,class_SyncStateRealm,* and *pk*. Lastly, *xabber.db* database has 30 tables. However, only 4 out of these 30 tables contain information of forensic interest namely *avatars, groups, groups_group, otr,* and *otr_list*.

We now discuss the contents of these databases, their tables, and fields along with their mapping to the user accounts and activities to answer questions from our investigative scenario.

6.2 Recovering account information

The *realm_database.realm* database stores information of distinct XMPP IM accounts and passwords configured by the local user on Xabber. The record is stored in the *class_AccountRealm* table in plaintext. Each IM account is assigned a unique identifier (primary key) and stored in the "*id*" field. The account names, associated XMPP

Table 4 Xabber local user and contact IM accounts

Local user XMPP IM accounts on analysed Android smartphone	behemothlabs@xabber.org otherbehemoth@xabber.org
Contacts' XMPP IM accounts on other Android devices	Contact 1: behemothlabs01@xabber.org Contact 2: alice.behemoth@xabber.org
Deleted contact XMPP IM account	alice.behemoth@xabber.org

Table 5 File paths of critical evidence sources of the Xabber app

Directory path	Details
/data/data/com.xabber.android/files/xabber.realm	Unencrypted database containing local user account information and exchanged chat messages
/data/data/com.xabber.android/files/realm_database.realm	Unencrypted database containing local user account information
data/data/com.xabber.android/databases/xabber.db	Unencrypted database containing local user account and contacts' information
/data/data/com.xabber.android/cache/image_manager_disk_cache/	Location where local copies of raw multimedia audio and video files are stored

server names, passwords, and authentication tokens are stored in *the username, serverName password,* and *token* fields respectively. The *avatars* table in the *xabber.db* database, stores information of all IM accounts in the "*user"* field, and the hash value of the local user's avatar in the *"hash"* field. The local user's raw avatar (image file) is stored in the directory "*/data/data/com.xabber.android/files/avatars/<hash value>".* In the *xabber.db* database, the *groups,* and *groups_group* tables jointly store group information on local user IM accounts while the *otr* and *otr_list* tables jointly store information about contacts XMPP IM accounts that have been verified.

Information associated with the contacts of the local user is stored in the *class_ChatDataRealm table of the realm_database.realm* database. The table contains one record of multiple local user IM accounts with the corresponding contact IM account with whom messages have been exchanged. The table also includes records of deleted contacts. From our investigative scenario, we recovered information regarding previous exchanges between one of the local user's IM account *(otherbehemoth@xabber.org)* and the deleted contact's IM account *(alice.behemoth@xabber.org)*. However, the body of the messages is not stored in this table.

6.3 Recovering chronology of chat logs, message contents, and deleted files

The *xabber.realm* database stores record of textual content of both encrypted and unencrypted messages and metadata (e.g., the unique identifier for each message, a status flag which indicates whether a message is sent or received, corresponding contact IM account information, date and time when the exchange occurred). These records are stored in the *class_MessageItem* table. In Table 6**,** we present a detailed interpretation of the relevant fields.

Table 6 Structure of the *class_MessageItem* table

Name	Role	Type	Meaning
uniqueId	Primary key	String	Unique identifier of the message
Account	–	String	Local user IM account
User	–	String	Contact user IM account message is exchanged with
Text	–	String	Body of the message
Incoming	–	Boolean	Flag indicating whether a message was received (true) or sent (false)
Encrypted	–	Boolean	Flag indicating whether a message is encrypted (true) or unencrypted (0)
Offline	–	Boolean	Flag indicating whether the contact IM account was online (true) or offline (false)
Timestamp	–	Int	The date this message has been sent or received (13-digits Unix epoch format)
Error	–	Boolean	Flag indicating message error (true or false)
Delivered	–	Boolean	Flag indicating message delivery (true or false)
Read	–	Boolean	Flag indicating whether a message is read (true) or unread (false)

class_MessageItem (13)

account	user	text	action	incoming	encrypted	offline	timestamp
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org x...	hello		True	False	False	1587132511797
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org	How are things		False	False	False	1587132536077
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org x...	fine not bad		True	False	False	1587132552979
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org	Hello		False	False	False	1587132624171
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org			False	False	False	1587132634521
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org	New contact	available	True	False	False	1587132732426
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org		otr_encryption	True	False	False	1587132753115
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org		otr_smp_verified	True	False	False	1587132829559
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org			False	True	False	1587132846360
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org	That's the picture mats		False	True	False	1587132847075
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org x...	great thanks		True	True	False	1587132861592
behemothlabs@xabber.org/x...	behemothlabs01@xabber.org		otr_unreadable	True	False	False	1587133348053
otherbehemoth@xabber.org/...	alice.behemoth@xabber.org	Hello there		False	False	False	1587328640815

Fig. 9 *class_MessagesItem* table

To demonstrate the chronology of messages exchanged with each one of the contacts from our investigative scenario, we present some of the contents of 13 messages exchanged and stored in this table as shown in Fig. 9.

We see from the figure; both unencrypted (encrypted = False) and encrypted messages (encrypted = True) exchanged between the local user's IM account (*behemothlabs@xabber.org*) and the contact (*behemothlabs01@xabber.org*) are stored in plaintext in the field named *"text"*. Information associated with the local user's actions are stored in the *"action"* field when OTR encryption is enabled (action = otr_encryption) and successful verification of the contact's encryption key is complete (action = otr_smp_verified). All subsequent messages exchanged between the user and contact are encrypted (encrypted = True) until OTR encryption is disabled (action = otr_unreadable).

Messages exchanged with deleted contacts are also stored in the *class_MessageItem* table. In Fig. 9, we see from the last record in the table, an unencrypted message sent (encrypted = False and incoming = False) from the local user's IM account *(otherbehemoth@xabber.org)* to the deleted contact *(alice.behemoth@xabber.org)* on the 19th Apr. 2020 at 20:37:20 pm UTC + 1 (Unix time stamp = '1587328640815') is stored in the field *"text"* (text = 'Hello there').

The *class_Attachment* table in the *xabber.realm* database, stores record file attachments sent or received by the local user accounts (see Fig. 10). Each file is assigned a unique identifier and stored in the "*uniqueId"* field. The "*fileUrl"* field contains the direct link to file attachment content on XMPP servers. By entering the URL link stored in the field into a web browser, for example (https://upload02.xabber.org/4f9edc.../QG.../title), the file uploaded by the Xabber client can be accessed directly.

Deleted media files such as images can also be recovered from the */data/data/com.xabber.android/cache/image_manager_disk_cache/* directory path within the Xabber app's Android filesystem folder. In the directory, each media file is stored as a unique file name (hash) with the "0.0" extension. In our investigative scenario, we deleted an image that was received by the local user in the app chat window. However, a raw copy of the same image was recovered from this directory.

7 Discussion

Lack of privacy and anonymity on encrypted instant messaging platforms are huge concerns for privacy advocates and many users. At this moment, apps like WhatsApp have been criticised for their new privacy policy which allows the vendor to collect user data and account information for marketing purposes. Rival platforms such as Signal and Telegram apps require users to provide a valid phone number that is tied to their account during registration. These concerns have made many users switch to open-source XMPP multi-client instant messaging apps which provide E2EE and anonymity for communication. We believe these reasons make our work even more relevant as many users utilize these applications for private legitimate communication but also illicit ones. Therefore, to the best of our knowledge, this is the primary forensic analysis of Conversations and Xabber, two popular XMPP multi-client apps that support two distinct protocols, OMEMO and OTR respectively for E2EE communication. The goal of this research was to analyse databases maintained by both apps and the internal device storage locations on Android devices for digital forensic artefacts and metadata. The study also aimed to show the importance of correlation and interpretation of the artefacts generated by each app and present findings that would be beneficial for forensic investigators.

Our findings show both Conversations and Xabber apps store local copies of user data in unencrypted databases, internal and external device storage locations, that can be extracted from an Android mobile device during mobile forensic analysis. These include user account information, user contact information, chat messages, files exchanged and evidence of deleted messages or contacts. These findings are similar to the results shown in the forensic analysis of *Telegram* [17] and *IMO* [18] apps on Android devices. However, there are major differences due to the features and E2EE protocols used. Both Conversations and Xabber apps support the use of multiple XMPP IM accounts by a

class_Attachment (2)

uniqueId	title	fileUrl
bb50db1f-86a6-4445-8c4b-8016cb3b34ed	1571653125988.jpg	https://upload02.xabber.org/4f9edc4e9
05e96792-18f9-494c-850a-81b263fa4fc8	c0397114-23e6-4998-9473-3b9e8cea429b.jpg	https://upload02.xabber.org/a610736c2

Fig. 10 *class_Attachment* table

local user to send messages, support the exchange of messages with other decentralized XMPP clients irrespective of the E2EE protocol implemented and allow a user to enable or disable encryption for communication.

These features are indicative of how forensic artefacts that persist in the apps' databases are stored compared to *Telegram* and *IMO,* which are centralized IM apps, implement alternative protocols for E2EE and are not interoperable with other messaging apps. Results presented in the forensic analysis of ChatSecure on Android devices [3], which uses OTR for E2EE communication, show local copies of both exchanged messages and files can be recovered from two distinct databases maintained by the app. The findings are consistent with results shown from our analysis of the Xabber app which also supports OTR. Although ChatSecure uses encrypted SQLite databases, Xabber uses a combination of unencrypted Realm open-source object databases and an SQLite database to store user data locally. It is also worth noting that all data and forensic artefacts associated with both Conversations and Xabber apps are deleted from the mobile device and cannot be recovered once the apps are uninstalled.

Therefore, the discussion of our findings and interpretation of artefacts from this study can be valuable for forensic investigators that come across these apps during mobile forensic investigations.

8 Conclusion

In this paper, we identify all the artefacts left by Conversations and Xabber apps. In our analysis of the Conversations app which uses OMEMO for E2EE, we access the main database maintained by the app, analyse, and recover all critical user's information stored in plaintext. These include information associated with the local user's multiple IM accounts, information associated with contact IM accounts (active and deleted), the textual content of messages (encrypted and unencrypted) exchanged, files including deleted ones, and chronology in the order of their occurrence. Although we showed information associated with a deleted contact's IM account, information associated with messages exchanged with such contact is not stored in the app's main database.

In our analysis of the Xabber app which uses OTR for E2EE, we were able to access the three databases maintained by the app. We identified, analysed, and showed information associated with the local user's multiple IM accounts, information associated with all contact IM accounts (active and deleted), deleted files and textual content of all encrypted and unencrypted messages exchanged.

It is worth noting that the recovery of critical information as demonstrated in this study cannot be generalized to other E2EE decentralized open-source XMPP/Jabber multi-client instant messaging apps. However, for forensic investigators, this study can aid forensic investigations for both Conversations and Xabber apps and our methodology can be adopted in the forensic analysis of similar XMPP/Jabber apps. Future directions include analyses of XMPP private servers for traces of evidential forensic artefacts.

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

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Express VPN (2019) The most secure messaging apps in 2019, Express VPN. https://www.expressvpn.com/blog/best-messaging-apps/. Accessed April 18, 2020
2. Conference Support Section (2013) Organized crime branch, division for treaty affairs, unodc, comprehensive study on cybercrime, United Nations off. drugs crime. http://www.unodc.org/documents/organized-crime/UNODC_CCPCJ_EG.4_2013/CYBERCRIME_STUDY_210213.pdf. Accessed April 18, 2020
3. 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
4. Express VPN (2019) How to keep your messages private and anonymous. https://www.expressvpn.com/blog/anonymous-chat-services/. Accessed April 18, 2020
5. XMPP Standards Foundation (XSF) (2020) An overview of XMPP, about XMPP. https://xmpp.org/about/technology-overview.html. Accessed April 18, 2020
6. Borisov N, Goldberg I, Brewer E (2004) Off-the-record communication, or, why not to use PGP. In: Proceedings of the 2004 ACM workshop on privacy in the electronic society—WPES '04. ACM Press, New York, p 77. doi: https://doi.org/10.1145/1029179.1029200

7. XMPP Standards Foundation (XSF) (2020) XEP-0384: OMEMO encryption. https://xmpp.org/extensions/xep-0384.html. Accessed April 18, 2020
8. Statista (2020) Share of global smartphone shipments by operating system from 2014 to 2023, Statista.Com. https://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/. Accessed April 18, 2020
9. Al Mutawa N, Baggili I, Marrington A (2012) Forensic analysis of social networking applications on mobile devices. Digit Investig 9:S24–S33. https://doi.org/10.1016/j.diin.2012.05.007
10. Quick D, Choo K-KR (2017) Pervasive social networking forensics: intelligence and evidence from mobile device extracts. J Netw Comput Appl 86:24–33. https://doi.org/10.1016/j.jnca.2016.11.018
11. Norouzizadeh-Dezfouli F, Dehghantanha A, Eterovic-Soric B, Choo K-KR (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
12. Akinbi A, Ojie E (2021) Forensic analysis of open-source XMPP multi-client social networking apps on iOS devices. Forensic Sci Int Digit Investig 36:301122. https://doi.org/10.1016/j.fsidi.2021.301122
13. van Dongen WS (2007) Forensic artefacts left by pidgin messenger 2.0. Digit Investig 4:138–145. https://doi.org/10.1016/j.diin.2008.01.002
14. Ballinger C (2016) The end of ChatSecure Android, Chatsecure.Org. https://chatsecure.org/blog/chatsecure-android-deprecated/. Accessed April 18, 2020
15. Mehrotra T, Mehtre BM (2013) Forensic analysis of Wickr application on android devices. In: 2013 IEEE international conference on computational intelligence and computing research. IEEE, pp 1–6. doi: https://doi.org/10.1109/ICCIC.2013.6724230
16. 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
17. 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
18. Sudozai MAK, Saleem S, Buchanan WJ, 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
19. Computer Fraud & Security (2020) Hundreds of alleged criminals arrested after European authorities infiltrate encrypted chat service. Comput Fraud Secur 2020(7):1–3. https://doi.org/10.1016/S1361-3723(20)30067-1
20. Lin X, Chen T, Zhu T, Yang K, Wei F (2018) Automated forensic analysis of mobile applications on Android devices. Digit Investig 26:S59–S66. https://doi.org/10.1016/j.diin.2018.04.012
21. Scrivens N, Lin X (2017) Android digital forensics. In: Proceedings of the ACM turing 50th celebration conference-China—ACM TUR-C '17. ACM Press, New York, pp 1–10. doi: https://doi.org/10.1145/3063955.3063981
22. Cellebrite (2020) Cellebrite UFED 4PC and Physical Analyzer. https://www.cellebrite.com/en/home/. Accessed April 18, 2020
23. Conversations, Privacy Policy (2020) Acc. Privacy Policy. https://account.conversations.im/privacy/ Accessed April 18, 2020
24. Verschoor SR (2016) OMEMO: Cryptographic analysis report. http://alexandria.tue.nl/extra1/afstversl/wsk-i/Verschoor_2016.pdf%5Cn https://conversations.im/omemo/audit.pdf. Accessed April 22, 2020
25. Xabber.com (2020) Introducing Xabber, Xabber. https://www.xabber.com/. Accessed April 22, 2020
26. Realm (2020) Realm database. Realm database https://realm.io/products/realm-database. Accessed April 22, 2020