sustainability

MDPI

Article

Verification of Privacy Protection Reliability through Mobile Forensic Approach Regarding iOS-Based Instant Messenger

Jiho Shin [1] and Jung Taek Seo [2,*]

[1] Police Science Institute, Korea National Police University, Asan 31539, Korea
[2] Department of Computer Engineering, Gachon University, Seongnam 13120, Korea
* Correspondence: seojt@gachon.ac.kr; Tel.: +82-10-6358-1337

Abstract: With the diffusion of mobile devices and Internet hyperconnectivity technology, all daily living records of individuals are being recorded on mobile devices in real time. However, from the user's point of view, the reliability of privacy protection, that is, whether the user's data on the mobile device completely disappears when it is deleted, is critical. This is because, for the sustainability of social growth, it is necessary to control the digitalization and technology that heightens the risks of the future society. Therefore, this study aims to check the traces of the SQLite database to see if instant messenger messages deleted by the user can be recovered. When the SQLite database record is deleted, if the database shrink function or other application-level deletion does not work, it is possible to recover the deleted record. We chose two iOS-based instant messengers, WhatsApp and WeChat, and analyzed the SQLite DB file and Table Schema where messages are stored. As a result of the experiment in this study, it was verified that the area where the deleted message was stored in the SQLite DB file was overwritten with 0 × 00 or updated with a NULL value, making it impossible to recover the deleted message. This process operates at the app level, and user data is safely protected.

Keywords: privacy protection reliability; mobile forensic; SQLite database; message deletion event

Citation: Shin, J.; Seo, J.-T. Verification of Privacy Protection Reliability through Mobile Forensic Approach Regarding iOS-Based Instant Messenger. *Sustainability* **2022**, *14*, 13281. https://doi.org/10.3390/su142013281

Academic Editors: Marc A. Rosen and Aniello Castiglione

Received: 17 August 2022
Accepted: 13 October 2022
Published: 15 October 2022

1. Introduction

With the diffusion of mobile devices and Internet hyperconnectivity technology, all daily living records of individuals are being recorded on mobile devices in real-time. In particular, the wider distribution of smartphones, which have become a necessity in modern times, has enabled individuals to access various online services more easily, and more user data are being accumulated as smartphone specifications, such as the storage capacity, processor, and memory, are expanding. Moreover, the hyperconnectivity that allows access to the Internet from anywhere also provides an environment that enables easy access to various social network services (SNS) and instant messenger (IM) services, so collecting and analyzing the related data has significantly contributed to the solving of criminal cases. Since most smartphone users have access to IM, the IM data is a very important clue that can prove a suspect's guilt or innocence in a criminal investigation. For this reason, mobile forensics, which involves an examination of suspects' smartphones, has become one of the essential investigative procedures during criminal investigations.

However, from the user's point of view, the reliability of privacy protection, that is, whether the user's data on the mobile device completely disappeared when it is deleted, is very important. This is because, for the sustainability of social growth, it is necessary to control the digitalization and technological innovation that heightens the risks of the future society. Law enforcement agencies must conduct investigations according to due process, and control of technology must be carried out within the scope of guaranteeing individual privacy even during forensic work permitted by law enforcement agencies. In other words, we need to consider the reliability of protection for personal privacy due to rapid changes in technology and digitalization. Therefore, this study aims to check the

traces of the SQLite database to see if instant messenger messages deleted by the user can be recovered.

Excluding the industry-wide discussion that the closeness of flash memory itself, the storage device for smartphones, has gradually increased, and code obfuscation to protect data in software [1], this problem starts from acquiring and analyzing data in Apple's iPhone. From the digital forensic perspective, most of the relevant data are in the private storage area of the app, which is only accessible in a rooted smartphone such as an Android device [2]. The iOS provided by Apple sets the user-accessible area quite restrictively by default compared to Android. Data must first be obtained from a smartphone for data analysis, yet it is virtually impossible to obtain the data unless the lock password is obtained from the owner. On 2 December 2015, a mass shooting occurred in San Bernardino in eastern Los Angeles, California [3]. However, the FBI had difficulties with the investigation because it could not unlock the main suspect's iPhone 5C [4]. In Korea, there was also a case in which the iPhone possessed by the main suspect identified by the investigative agency in a criminal case could not be unlocked, and the relevant evidence could not be collected [5].

Therefore, this study aimed to obtain smartphone data by generating backup data from a locked iPhone and then verify the possibility of data recovery by checking the traces of instant messenger messages deleted by the user. In other words, it is intended to verify whether the user's privacy is being protected by analyzing the traces of deleted messages by the user through mobile forensics. Most instant messengers use SQLite for storing and managing exchanged messages. When the SQLite database record is deleted, if the database shrink function or other application-level deletion does not work, it is possible to recover the deleted record. We will choose two popular iOS-based instant messengers to analyze if deleted messages are recovered.

According to the market share surveyed by Statista, the most popular mobile messenger in the global market is WhatsApp, followed by WeChat [6]. The statistics are based on an estimation of the number of monthly messenger apps activated by mobile messenger users, and according to this statistic, WhatsApp has been activated 2 billion times and WeChat 1.263 billion times (see Figure 1). This study has verified the possibility of recovering deleted data through an experiment targeting the top two most widely used apps among various global messenger apps.

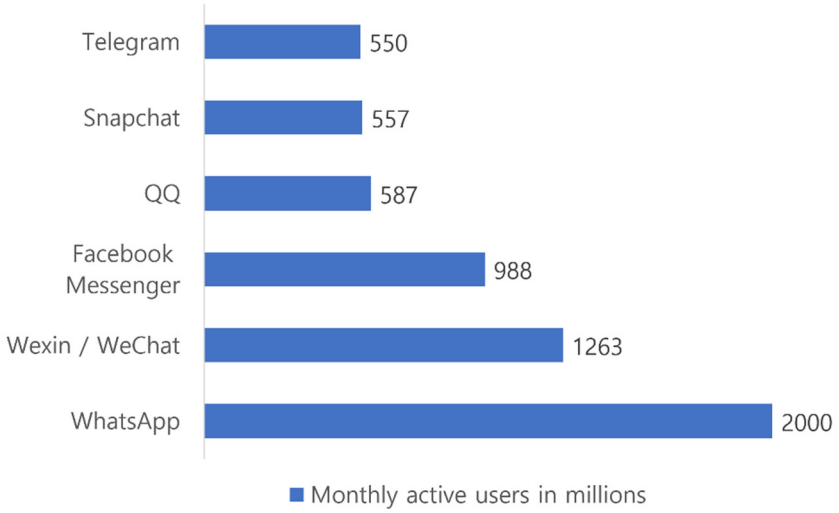


Figure 1. Most popular global mobile messenger apps as of January 2022 from Statista (Based on the number of monthly active users).

This study contributes to digital forensics as follows.

- It presents a method of investigating the structure of the SQLite database, which stores chat messages in most instant messengers, and the verification of privacy protection reliability from a digital forensic point of view.

- Since this study presents a method of creating backup data for the iPhone without unlocking it, even if the mobile phone is locked, the possibility of leakage of personal information inside the mobile was technically confirmed.
- Since this study includes experimental results on data traces targeting the two messengers with the highest usage rates in the global instant messenger market, it is of practical use in mobile forensics.

This paper is organized as follows. Section 2 introduces reviews of prior related studies. Section 3 discusses a method of acquiring data by creating a backup in a locked iPhone and the structure of the SQLite database, and how to detect deleted data. Section 4 configures the experimental environment and a deletion event to check the recoverability of deleted messages. Section 5, following on from the deletion event presented in Section 3, deals with the detection of data, observation of the data change in SQLite, and derivation of the experimental results. Lastly, Section 6 presents the conclusion.

2. Related Works

Until very recently, the greater part of digital forensic analysis of iPhones was largely based on backup files, because it is very difficult to obtain all data in the physical area from the iPhone. Unlike open source-based Android, iOS is very limited in terms of the areas that a user of the leading closed OS platform can access. Since Android users can access the systems area using the ADB (Android Debug Bridge) protocol, previous studies mainly focused on acquiring and analyzing the data in the physical area. On the other hand, studies on iOS mainly focused on extracting the data in the logical area, i.e., the backup file, and analyzing the related artifacts. Anglano (2014) presented a forensic analysis of artifacts left on Android mobile devices by WhatsApp Messenger [7]. The study was notable in that it presented the correlation of various artifacts to infer user behavior and reconstruct the time sequence of messages exchanged by users. However, there is a limitation in that study that cannot be used in this study as it was based on Android OS. Han (2016) discussed the methods of acquiring and analyzing smartphone backup data [8]. It can be considered a prior study on the data acquisition method proposed in this study, as it suggests the use of the lockdown key file (Escrow Keybag) in iOS. As discussed above, the file makes it easy to obtain backup data from a locked iPhone. Shimmi (2020) emphasized that an iOS device's backup file was a potential source of main evidential data and explained the process of changing the SQLite table schema related to the analysis of the iOS backup file obtained through logical collection technology [9]. The study showed a good result that can confirm the direction in which Apple changed the configuration information related to backup files. However, there are some differences from this study as these previous studies mainly focused on the method of acquiring backup data.

SQLite database has been widely used to store and utilize data in smartphones. It is used for data storage and management and input/output in embedded systems such as smartphones because it is light in operation and has a high input/output speed and a simple platform configuration file for running the database. Existing studies related to SQLite have mainly focused on the forensic recovery of deleted data. Joen (2011) proposed a technique for recovering deleted records remaining in an unallocated area of an SQLite database [10]. The study suggested a method of recovering deleted SQLite records, but the corresponding area was not overwritten. Part of the study will be discussed on the SQLite record structure below. Jung (2018) studied the possibility of recovering deleted messages through SQLite Journal analysis in messenger applications [11]. As SQLite manages the journal file separately to manage input/output efficiently, the study proposes a method of restoring messenger conversation by analyzing the deleted data fragments recorded in the journal file. However, since most studies have targeted the SQLite journal in Android OS, there is a difference in the subject of analysis of this study.

3. Background

3.1. Acquisition Method of Locked iPhone

It is necessary to check *Unique Device ID.plist* to acquire logical data from a locked iPhone. Additionally, called the escrow Keybag file, it is generated when connecting an iPhone to a PC on which iTunes is installed and uses the plist format [12,13]. It configures encrypted information related to authentication and creates UDID.plist, of which UDID denotes the iPhone's unique device ID, to prove that it is a computer that has established trust with the iPhone, and saves it in *%SystemDrive%\ProgramData\Apple\LockDown* of the PC (based on Windows OS) [14] (see Figure 2).

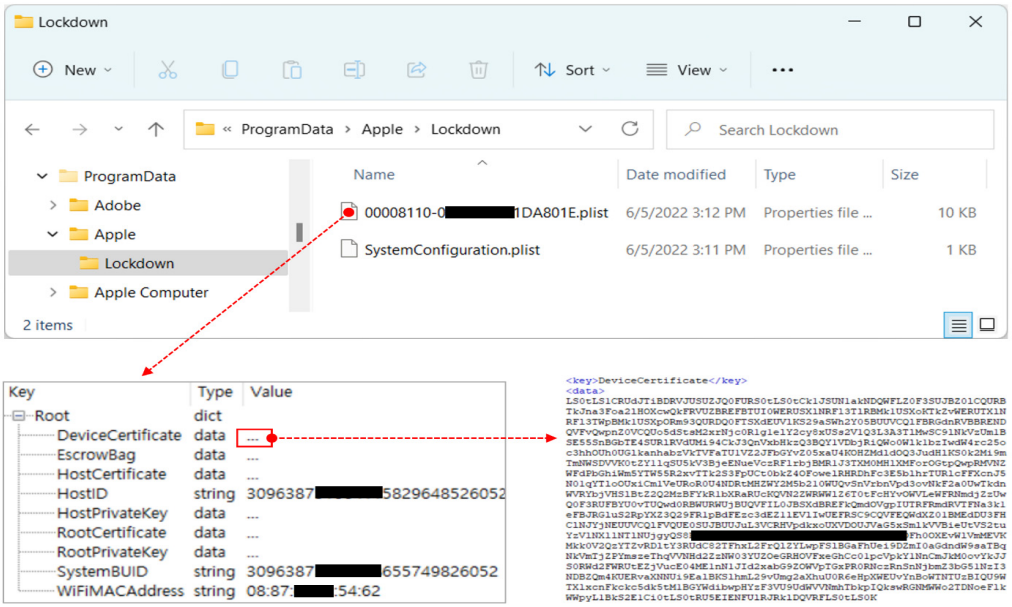


Figure 2. Example of Escrow Keybag (UDID.plist) and its properties in iTunes-installed PC.

Therefore, it is necessary to investigate intensively the computer that has been connected to the analyzed iPhone processed as *Trust This Computer* to generate the locked iPhone's backup file using iTunes, by collecting the UDID.plist file [15]. If it is not feasible to obtain the password from the user who used the iPhone (e.g., the user is dead or does not disclose it on purpose), it will be necessary to collect the UDID.plist file stored in the desktop or notebook PC. This is because securing the UDID.plist file is a way to get a bypass connection without the trust process. After copying the collected UDID.plist file to a forensic workstation and connecting the PC, it can create a backup file from the locked iPhone without an additional authentication process (see Figure 3). In other words, the iPhone and the forensic workstation are recognized as mutually trusted devices, and iTunes can create a backup file for the connected iPhone.

In this study, an experiment is conducted on the obtained backup file. It detects the tables and columns of the SQLite database, where the messages are stored, from the backup file collected under the constrained environment of the locked iPhone and checks the possibility of recovering the deleted messages through an experiment.

3.2. SQLite Logical Structure

The SQLite DB file is composed of contiguous pages. The pages are largely composed of a header page and a b-tree page, of which the latter is further composed of an internal b-tree page and a leaf b-tree page. Furthermore, a leaf b-tree page is composed of a leaf index b-tree page and a leaf table b-tree page. The database records are stored in the leaf table b-tree pages. The types of pages will be explained again later in the paper.

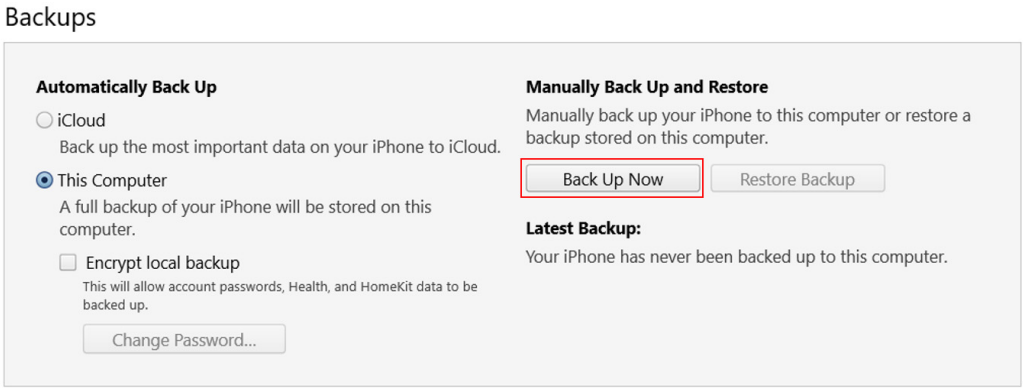


Figure 3. iTunes recognizes locked iPhone using UDID.plist from the user's PC (see the activated button in red box).

The SQLite header page is the start of the DB file [16] and typically begins with a header string that starts with "SQLite format". After that, it specifies the page size using 2 bytes. Various setting values and status information for running the SQLite DB file are managed using headers.

B-tree pages are created with the page size specified in the SQLite's database file header. Pages are composed of a page header, cell offset, free space, and cell data in that order (see Figure 4). The page header consists of information such as the page type, free-block information on the page, the number of cells on the page, and the cell offset [17] (see Table 1).

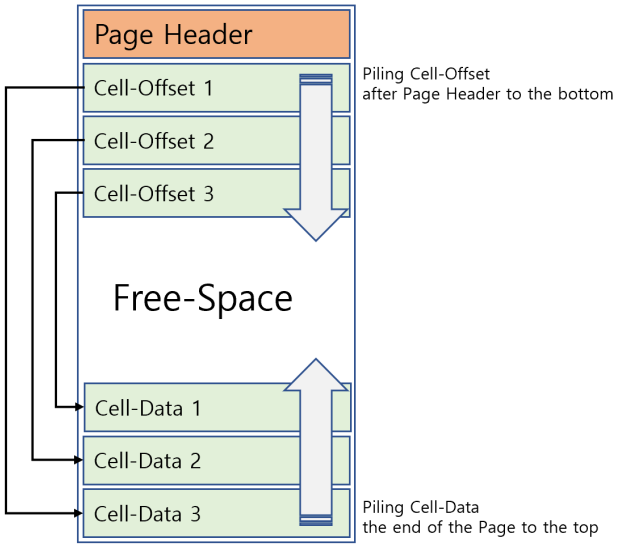


Figure 4. B-tree page structure 16.

The cell offset stores the offset information, which indicates where the cell data in the page starts in a 2-byte array and is stored downward sequentially from the position after the page header. The cell data are stored upward sequentially from the end of the page. The page header contains the page type, which is important information for data detection in SQLite files. There are four types of pages, and the page where the cell data is stored starts with the $0 \times 0D$ value. The page header also contains a value for quickly detecting the first data on the page and the offset value that indicates where the actual data start on the page. Such information is used to quickly detect the last data position for data input/output. Figure 5 below shows an example of analyzing the hexadecimal value of the page containing the actual cell data. It shows that the cell offset array is positioned after the header, which is composed of 8 or 12 bytes.

Table 1. Description of page header 17.

Offset	Size	Description
0	1	A flag indicating the b-tree page type 0 × 02: interior index B-tree page (Internal) 0 × 05: interior table B-tree page (Internal) 0 × 0A: leaf index B-tree page (Leaf) 0 × 0D: leaf table B-tree page (Leaf) Any other value for the b-tree page type is an error.
1	2	Byte offset into the page of the first freeblock
3	2	Number of cells on this page
5	2	Offset to the first byte of the cell content area. A zero value is used to represent an offset of 65,536, which occurs on an empty root page when using a 65,536-byte page size.
7	1	Number of fragmented free bytes within the cell content area
8	4	The right-most pointer (interior b-tree pages only)

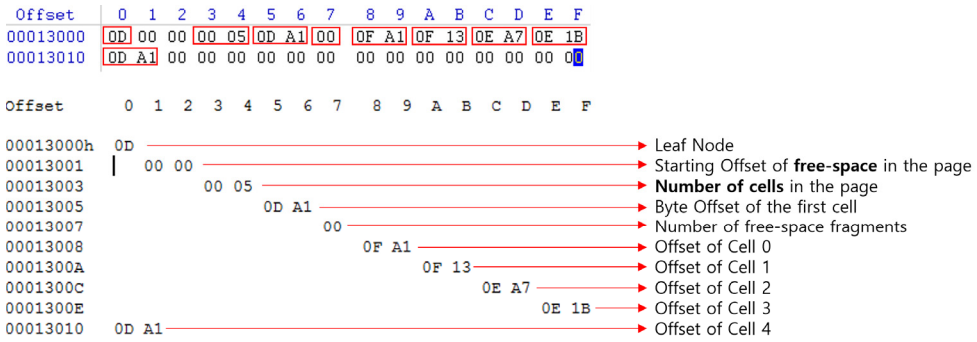


Figure 5. Cell offset example in page header on leaf table B-tree page cell pointer.

3.3. Cell Structure

An SQLite cell is divided into the header and data areas. The header area consists of a payload, which is the size of the cell, and a Row ID, which is a unique cell number. The data area is divided into a cell data header and cell data. The cell data header contains the size information of the column data in an array type and is followed by the actual data (see Figure 6).

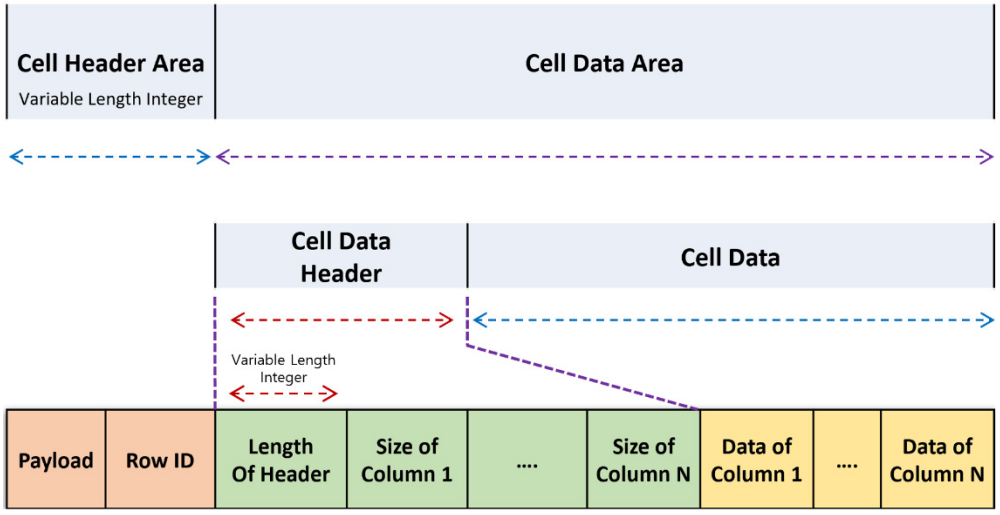


Figure 6. SQLite cell structure.

The data in each cell column can be carved by carving the actual data by as much as the data size of each column stored in the cell data header (see Figure 7).

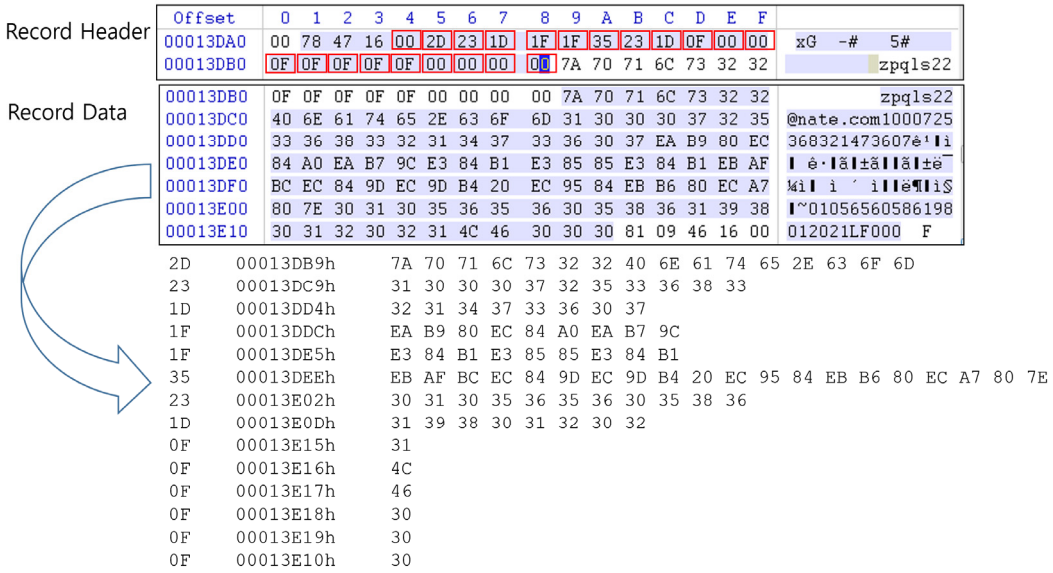
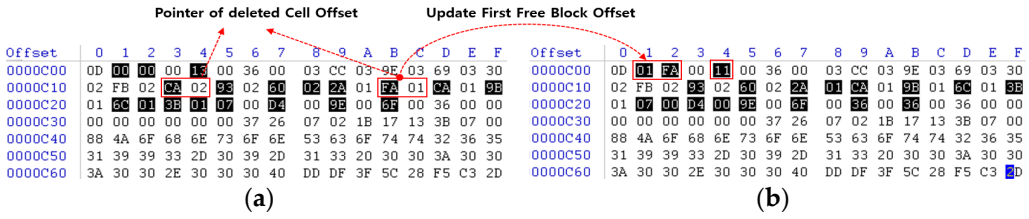


Figure 7. Cell data carving example by each column size.

3.4. Existing Method Detection and Recovery of Deleted Cell

Jeon et al. (2011) explained that it is necessary to check the page header first to detect deleted data because the first freeblock information is stored in 2 bytes from the header byte offset 0 × 01 in the page header [10]. If the offset information is stored, it means that the deleted cell is present on the page [18]. Another way to detect deleted data is to perform a keyword search in the database file. It directly searches and checks whether the data to be found exists in the database file. If the keyword search result is found on the data page (0 × D), the data can be recovered through additional detailed investigation and data carving. Figure 8 shows the header of the page containing the deleted cell. It shows that two cells have been deleted, and the pointer information of the first cell on the page is stored in the page header.



(**a**) (**b**)

Figure 8. Comparison of page header according to cell deletion. (**a**) is normal and (**b**) is deleted.

For example, 2 bytes from page offset 0 × C01 in Figure 8a are first freeblocks and empty (0 × 0000). However, as shown in Figure 8b, when a record (cell) on the page is deleted, the SQLite engine stores the page offset value where the deleted record starts in the first freeblock. In this way we can analyze whether there is any deleted data on that page and where the deleted data is.

After analyzing the start offset of the deleted cell using the page header, the next step is to analyze the cell. According to the cell structure discussed in Section 3.2, the size of each column can be calculated and carved sequentially (see Figure 8). However, in this study, we adopted *Keyword Search* for detecting deleted cells instead of existed method. Although *Keyword Search* takes a long time, it is the most sophisticated way to find the data you want to find with the raw level.

3.5. Summary

In Section 3, we discussed how easy it is to get backup data from a locked iPhone. Without app-level software data protection, unencrypted data could easily be leaked, leading to privacy breaches. We also discussed the basic structure of SQLite and how to recover deleted cells. As discussed above, it was found that generally deleted data remains as it is unless it is completely deleted. Additionally, after finding the data page (It starts with $0 \times 0D$) and checking the deleted offset, it was confirmed that the deleted cell can be recovered. In the next chapter, we will analyze how deleted messages are managed by two popular mobile messengers through an experiment.

4. Experimental Environments

4.1. Message Deletion Events of Target Messengers

There are three main ways to delete an instant message from a mobile device. The first method consists of deleting a specific chat log from the chat room. Both WhatsApp and WeChat offer the ability to delete specific conversations from chat rooms. The second is to delete the chat room. When a chat room is deleted, all users' messages in the chat room are also deleted, and the chat room's users must be searched to begin messaging again. The third is to delete the instant messaging app. When the instant messages app is deleted, all chatting records using the app are also deleted. This study experimented with deleting the chat log and deleting the chatroom. The related detailed deletion method will be discussed again in Experimental Environments.

4.2. Tools and Experiment Environment

For this study, an experiment was conducted to detect deleted data in each mobile messenger based on the SQLite logical structure and the deleted data detection method, the results of which are summarized below. The backup data from Apple iPhone XS of the sender was created using iTunes, and the experiment was conducted on messenger app data acquired from it. The reason for using the backup data is that there is virtually no way of physically acquiring data (acquiring the entire storage area as a bitstream) for a digital forensic investigation performed on Apple iPhones. Therefore, it is reasonable to acquire data in the most similar way to the environment performed in actual practice and conduct digital forensic experiments on it. This study used the latest version of the mobile messenger that could be downloaded from the Apps Store at the time of the experiment. The detailed experimental environment is presented in the table below (see Table 2).

The scenario for experimenting is as follows. First, it is assumed that the acquisition target device is an iPhone turned on but cannot be unlocked because the password is unknown. As previously discussed, if the iPhone is locked and the password is unknown, the backup file should be analyzed as the next best solution. A laptop computer used by the user was found, and UDID.plist generated from the suspect's iPhone information was collected from the laptop computer. The analyst copied and moved UDID.plist to the analyst's forensic workstation, connected the user's iPhone, acquired backup data using iTunes, and analyzed it.

4.3. Experimental Method

For this study, the keyword *ajVVicFO09* was used to experiment with deleting part of the chat log in the chatroom, and *qWMLuPMX6h* was used to experiment with deleting the chatroom (see Figures 9 and 10).

Table 2. Properties of the experimental environment.

Category	Property	Value
Forensic Workstation	OS	Microsoft Windows 11
	Platform	INTEL X64
	File system	NTFS
Target Device (Evidence)	Model	Apple iPhone XS
	Capacity	64 GB
	Software Version	iOS 15.4.1
	File system	APFS
Instant Messenger	WhatsApp Version	v2.22.6.74
	WeChat Version	v8.0.18
Data Acquisition	Method	Logical (Backup File)
	Tool	iTunes v12.12.3.5
Experiment Keyword	For deleting part of the chat Message	*ajVVicFO09*
	For deleting chatroom	*qWMLuPMX6h*
Forensics Tools for Analysis	SQLite DB Browser	DB Browser for SQLite v3.12.2
	Backup Reconstructor	iMazing v2.14.8
	Hexadecimal Comparison	Beyond Compare4 v4.4.2
	Keyword Search	dnGrep v3.0.42.0

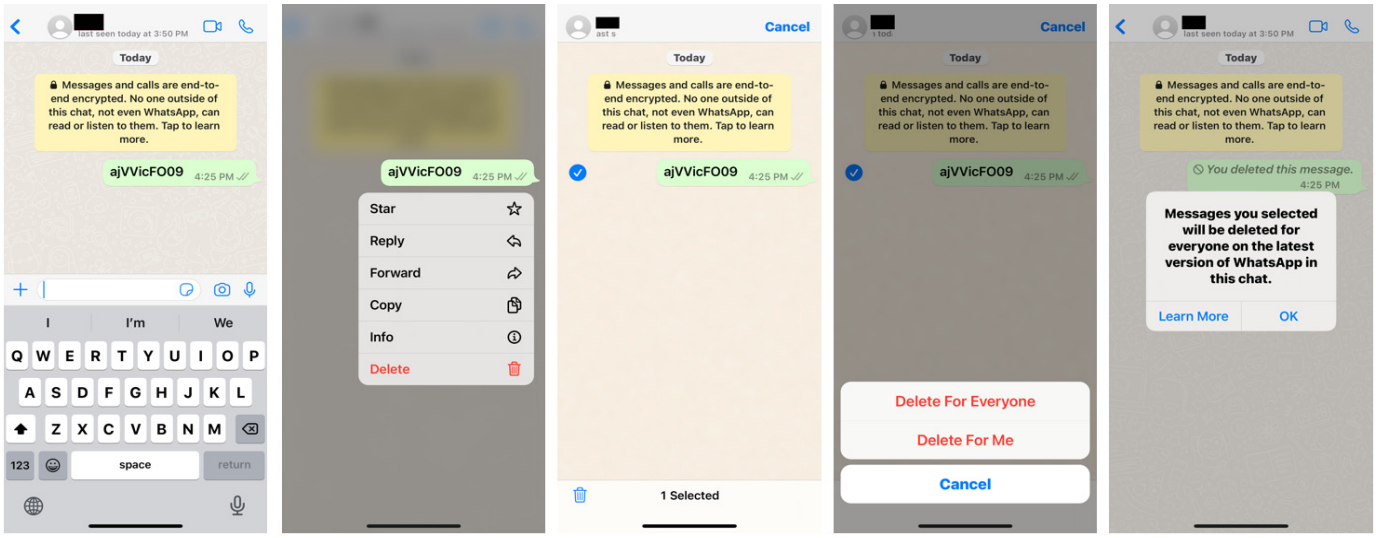


Figure 9. Deleting part of the WhatsApp chat history used in the experiment (Same in WeChat).

4.3.1. Delete Part of Chat Message

The experiment was conducted in the following order to check the change of SQLite data for the deleted message.

1. After launching the instant messenger app (WhatsApp or WeChat) on the iPhone, send a message composed of *ajVVicFO09*.
2. When the receiver confirms the message, delete the *ajVVicFO09* message, close the app, and lock the phone.
3. After copying and saving *UDID.plist* to the forensic workstation, create a backup file using iTunes.

4. Extract the instant messenger app (WhatsApp or WeChat) to be analyzed using iMazing to reconstruct the created backup file.
5. Check the SQLite database file where the conversation contents are saved from the extracted app data, and search and analyze the traces of the *ajVVicFO09* message.

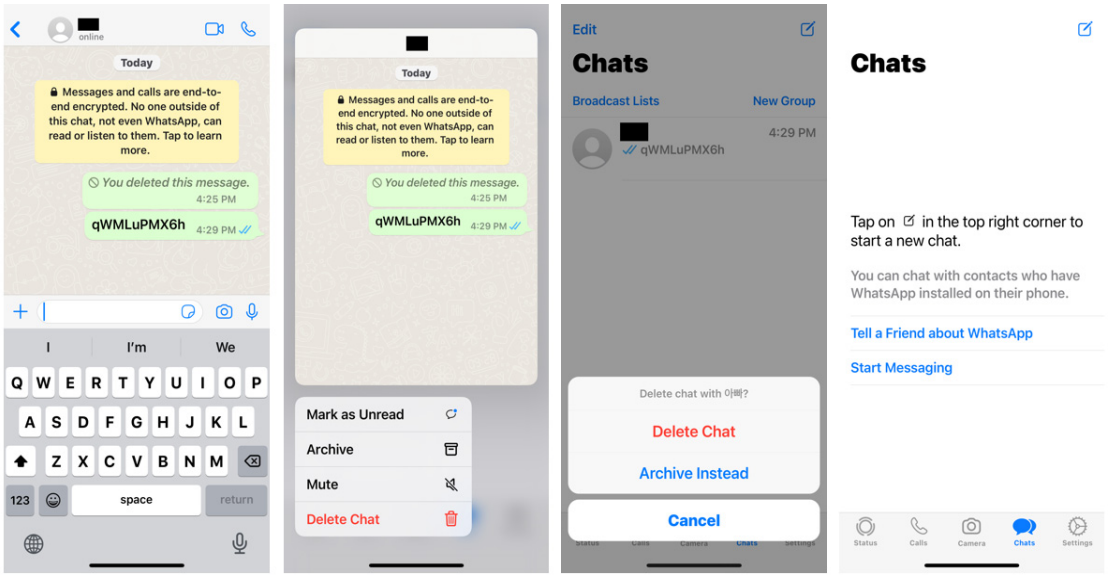


Figure 10. Deleting chat room of the WhatsApp used in the experiment (Same in WeChat).

4.3.2. Delete Chatroom

The experiment was conducted in the following order to check the change of SQLite data for deleted chatroom.

1. After launching the instant messenger app (WhatsApp or WeChat) on the iPhone, send a message composed of *qWMLuPMX6h*.
2. When the receiver confirms the message, delete the chat room, close the app, and lock the phone.
3. After copying and saving *UDID.plist* to the forensic workstation, create a backup file using iTunes.
4. Extract the instant messenger app (WhatsApp or WeChat) to be analyzed using iMazing to reconstruct the created backup file.
5. Check the SQLite database file where the conversation contents are saved from the extracted app data, and search to analyze the traces of the *qWMLuPMX6h* message.

5. Verification Results

5.1. WhatsApp

5.1.1. Delete Chat Message

As shown in Figure 9, after sending the unique keyword *ajVVicFO09* used in the experiment as a message, it was searched using dnGrep to specify the SQLite database file in which the data were stored. The result confirmed that the data stored in *ChatStorage.sqlite* in the *AppDomainGroup-group.net.whatsapp.WhatsApp.shared*/directory. The content of messages exchanged between users is stored in the *ZTEXT* column of the *ZWAMESSAGE* table (see Table 3) [19].

Checking the hexadecimal code in *ChatStorage.sqlite* confirmed that the experiment's keyword *ajVVicFO09* was stored normally (see Figure 11).

Table 3. Table scheme of *ZWAMESSAGE* in *ChatStorage.sqlite*.

No	Column Name	Declared Type	Default Value	Not Null
1	Z_PK	INTEGER	0	FALSE
2	Z_ENT	INTEGER	0	FALSE
3	Z_OPT	INTEGER	0	FALSE
4	ZCHILDMESSAGESDELIVEREDCOUNT	INTEGER	0	FALSE
	(*skip 26 columns*)			
31	ZPUSHNAME	VARCHAR	0	FALSE
32	ZSTANZAID	VARCHAR	0	FALSE
33	ZTEXT	VARCHAR	0	FALSE
34	ZTOJID	VARCHAR	0	FALSE

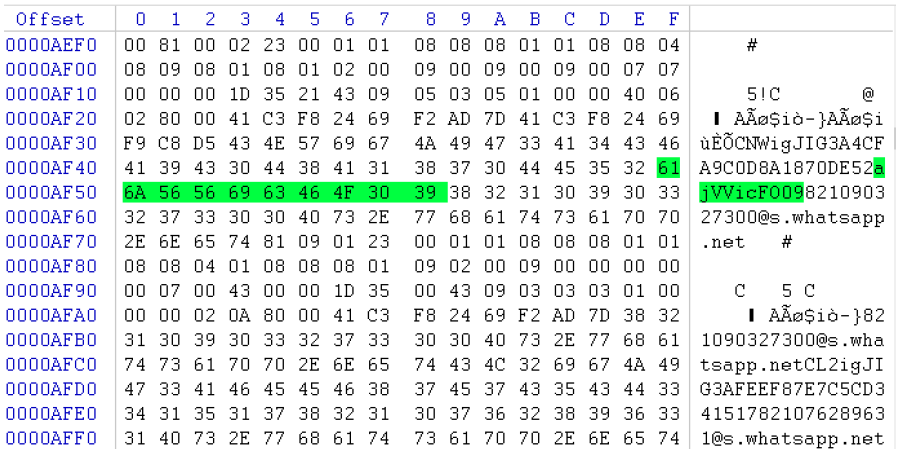


Figure 11. Message keyword *ajVVicFO09* in *ChatStorage.sqlite* of WhatsApp messenger.

Checking the data after the deletion showed that the record (cell) in the *ZWAMESSAGE* table where the message was stored had not been deleted but maintained. However, it was confirmed that the *ZTEXT* column of the corresponding cell was updated to NULL, and the corresponding area was reduced (see Figure 12).

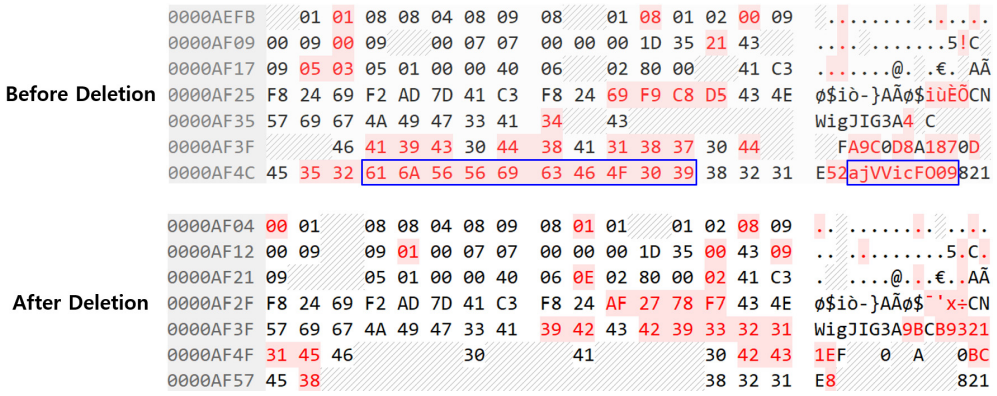


Figure 12. Comparison of before and after according to deletion of *ZTEXT* column data.

5.1.2. Delete Chatroom

The unique keyword *qWMLuPMX6h* was sent via a message, and the chatroom was deleted to observe changes in data following its deletion. Afterward, the experimental keyword was searched in the backup files but was not found.

5.2. WeChat

5.2.1. Delete Chat-Log

Experiments with WeChat were conducted in the same way as WhatsApp. Like WhatsApp, the unique keyword *ajVVicFO09* was sent as a message, and it was searched using dnGrep to specify the SQLite database file containing the data. The result confirmed that the data were stored in *message_2.sqlite* in the *AppDomain-com.tencent.xin/Documents/8750670c8c1f06f1d8b1f62c47637fa1/DB/*directory. The *8750670c8c1f06f1d8b1f62c47637fa1* value in the directory path is estimated to be a unique value generated by the app, and the content of messages exchanged by users is stored in the *Message* column of the *Chat_bf2456a533c56bf9488ab126ae200ea2* table (see Table 4). The *bf2456a533c56bf9488ab126ae200ea2* value included in the table name is estimated to be a unique value composed of information about the chat partner.

Table 4. Table Scheme of Chat_bf2456a533c56bf9488ab126ae200ea2 in message_2.sqlite.

No	Column Name	Declared Type	Default Value	Not Null
1	CreateTime	INTEGER	0	FALSE
2	Des	INTEGER		FALSE
3	ImgStatus	INTEGER	0	FALSE
4	MesLocalID	INTEGER		FALSE
5	Message	TEXT		FALSE
6	MesSvrID	INTEGER	0	FALSE
7	Status	INTEGER	0	FALSE
8	TableVer	INTEGER	1	FALSE
9	Type	INTEGER		FALSE

Checking the hexadecimal code in *message_2.sqlite* confirmed that the experimental keyword *ajVVicFO09* was stored normally (see Figure 13).

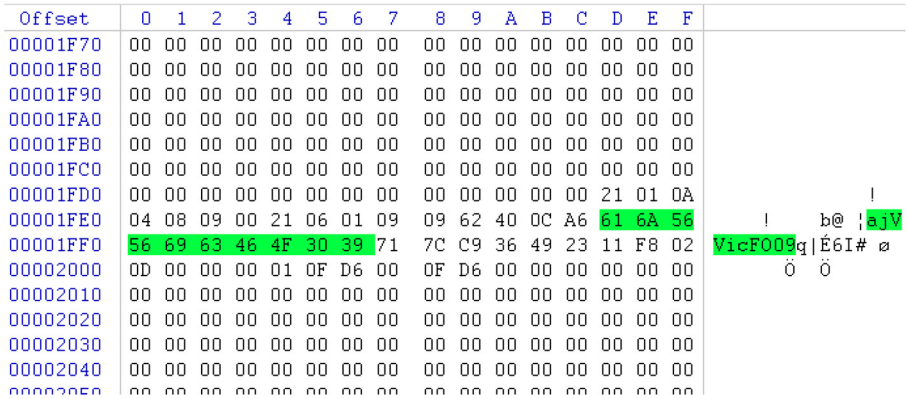


Figure 13. Message Keyword *ajVVicFO09* in *message_2.sqlite* of WeChat Messenger.

A check of the data after deletion showed that the record (cell) stored in the *Chat_bf2456a533c56bf9488ab126ae200ea2* table, which stored the message, had been deleted. Moreover, it was confirmed that the record (cell) area in which the data were stored existed but had been updated to 0 × 00 (see Figure 14).

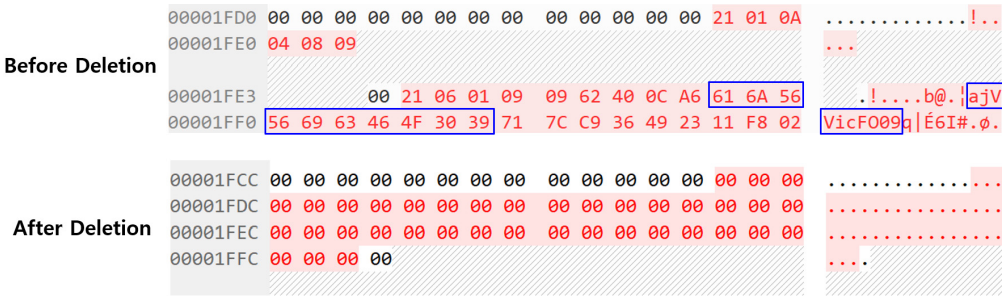


Figure 14. Comparison of before and after according to deletion of message column data.

5.2.2. Delete Chatroom

As with WhatsApp, a unique keyword *qWMLuPMX6h* was sent as a message, and the experimental keyword was searched in the backup data to observe data changes after deleting the chatroom, but it was not found.

5.3. Discussion

When the SQLite database record is deleted, if the database shrink function or other application-level deletion does not work, it is possible to recover the deleted record. However, the experimental results confirmed that the deleted column data or cell data could not be recovered from iPhone backup data for both instant messengers examined in this study (see Table 5). This is because the deletion process of both instant messengers was operated not only at the SQLite database level but at the app level. The results of this experiment were conducted on the backup data acquired from the sender's mobile device, but the receiver's mobile phone has the same result.

Table 5. Verification result of privacy protection reliability regarding iOS-based instant messenger.

Category	SQLite File	Deletion Subject	The Deleted Area on Page		Message Privacy
			Column Data	Cell	
WhatsApp	ChatStorage.sqlite	Part of message	Update with NULL (shrink)	Exists	Protected
		Chat room	Message not found		
WeChat	Message_2.sqlite	Part of message	Updated whole cell area with 0 × 00		Protected
		Chat room	Message not found		

In other hand, we must continue to consider many further discussions on various ways to protect privacy in the future. For example, Qu (2020) proposed a customizable reliable differential privacy model (CRDP), which provides customizable protection on each individual while being attack-proof for optimizing the tradeoff between customizable privacy preservation and data utility [20]. Additionally, Feng (2020) proposed PMF, a privacy-preserving mobility prediction framework via federated learning, to solve this problem without significantly sacrificing the prediction performance [21]. As such, various methods of protecting privacy have been researched and presented, so it should be highly considered to use the most suitable method according to the privacy surrounding environment.

6. Conclusions and Future Works

With the diffusion of mobile devices and Internet hyperconnectivity technology, all daily living records of individuals are being recorded on mobile devices in real-time. Especially smartphones are widely distributed, and a considerable amount of user data is being stored in them. This study verified the reliability of the user's privacy protection by acquiring smartphone data through the creation of backup data from a locked iPhone in a

restricted environment and checking the traces after deleting the instant messenger message sent by the user in the collected data. When the SQLite database record is deleted, if the database shrink function or other application-level deletion does not work, it is possible to recover the deleted record. However, the analysis of the iPhone backup data confirmed that the SQLite data for deleted messages could not be recovered in this study. The deleted message was stored in the SQLite file of the iPhone backup data was overwritten with 0×00 or updated with a NULL value, making it impossible to recover the deleted message, so it confirmed that the user privacy is safely protected. The deletion process was operated at the app level. This has a positive effect in terms of privacy security. This study confirmed the data storage structure, such as the SQLite database file containing the chat message and the related table, making it practically useful. In the future, it is necessary to verify that the same experimental results are obtained when directly stored in the mobile disk space, such as audio and video, as well as text stored in the SQLite database. In addition, further studies on the verification of privacy protection reliability are needed by confirming the recoverability for global messengers not covered in this study.

Author Contributions: Conceptualization, J.S.; Data curation, J.S.; Formal analysis, J.S.; Funding acquisition, J.-T.S.; Investigation, J.S.; Methodology, J.S.; Project administration, J.-T.S.; Resources, J.S.; Software, J.S.; Supervision, J.-T.S.; Validation, J.S. and J.-T.S.; Visualization, J.S.; Writing—original draft, J.S.; Writing—review and editing, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Nuclear Safety and Security Commission (NSSC) of the Republic of Korea (No. 2106058, 40%), the Korea government (MSIT) (No.2021-0-01806, Development of security by design and security management technology in smart factory, 40%), and the Gachon University research fund of 2021(GCU-202106330001, 20%).

Data Availability Statement: Not applicable.

Acknowledgments: This research was supported by the Nuclear Safety Research Program through the Korea Foundation of Nuclear Safety (KoFONS) and Institute of Information & communications Technology Planning & Evaluation (IITP), and Gachon University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. You, G.; Kim, G.; Cho, S.J.; Han, H. A comparative study on Optimization, Obfuscation, and Deobfuscation tools in Android. *J. Internet Serv. Inf. Secur.* **2021**, *11*, 2–15.
2. Domingues, P.; Nogueira, R.; Francisco, J.C.; Frade, M. Analyzing TikTok from a digital forensics perspective. *J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl.* **2021**, *12*, 87–115.
3. Nakashima, E.; Albergotti, R. The FBI Wanted to Unlock the San Bernardino Shooter's iPhone. It Turned to a Little-Known Australian Firm. Available online: https://www.washingtonpost.com/technology/2021/04/14/azimuth-san-bernardino-apple-iphone-fbi/ (accessed on 21 April 2022).
4. Mahalik, H.; Edwards, S.; Murphy, C. iPhone Forensics-Separating the Facts from Fiction. A Technical Autopsy of the Apple/FBI Debate. Available online: https://www.sans.org/webcasts/iphone-forensics-separating-facts-fiction-technical-autopsy-apple-fbi-debate-101890/ (accessed on 25 March 2022).
5. Hong, H. The Number of Password Cases Alone Is 56 Billion. iPhone Password That Even the FBI Can't Solve. Available online: https://www.mk.co.kr/news/society/view/2022/04/321122/ (accessed on 11 May 2022).
6. Dixon, S. Most Popular Global Mobile Messenger Apps as of January 2022, Based on Number of Monthly Active Users. Statista. Available online: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/ (accessed on 8 May 2022).
7. Anglano, C. Forensic analysis of WhatsApp messenger on Android smartphones. *Forensic Sci. Int. Digit. Investig.* **2014**, *11*, 201–221. [CrossRef]
8. Han, J.; Lee, S. A Practical Approach to Analyze Smartphone Backup Data as a Digital Evidence. *DFRWS, USA*. Available online: https://dfrws.org/presentation/a-practical-approach-to-analyze-smartphone-backup-data-as-a-digital-evidence/ (accessed on 22 May 2022).
9. Shimmi, S.S.; Dorai, G.; Karabiyik, U.; Aggarwal, S. Analysis of iOS SQLite schema evolution for updating forensic data extraction tools. In Proceedings of the 8th International Symposium on Digital Forensics and Security (ISDFS), Beirut, Lebanon, 1–2 June 2020; pp. 1–7.

10. Jeon, S.; Bang, J.; Byun, K.; Lee, G.; Lee, S. The method of recovery for deleted record in the unallocated space of SQLite database. *J. Korea Inst. Inf. Secur. Cryptol.* **2011**, *21*, 143–154.
11. Jung, B.; Han, J.; Choi, H.; Lee, S. A study on the possibility of recovering deleted data through analysis of SQLite Journal in messenger application. *J. Digit. Forensic* **2018**, *12*, 11–20.
12. Property List. Available online: https://en.wikipedia.org/wiki/Property_list (accessed on 17 May 2022).
13. Satish, B. Forensic Analysis of iPhone Backups. Available online: https://www.exploit-db.com/docs/english/19767-forensic-analysis-of-ios5-iphone-backups.pdf (accessed on 2 June 2022).
14. Apple Platform Security. Available online: https://support.apple.com/ko-kr/guide/security/welcome/web (accessed on 2 June 2022).
15. About the 'Trust This Computer' Alert on Your iPhone, iPad, or iPod Touch. Available online: https://support.apple.com/en-us/HT202778 (accessed on 2 June 2022).
16. The Database Header. Available online: https://sqlite.org/fileformat.html#the_database_header (accessed on 18 May 2022).
17. B-Tree Pages. Available online: https://sqlite.org/fileformat.html#b_tree_pages (accessed on 18 May 2022).
18. Jeon, S.; Bang, J.; Byun, K.; Lee, S. A recovery method of deleted record for SQLite database. *Pers. Ubiquitous Comput.* **2011**, *16*, 707–715. [CrossRef]
19. Khalid, Y. Extracting WhatsApp Messages from an iOS Backup. Available online: https://yasoob.me/posts/extracting-whatsapp-messages-from-ios-backup/ (accessed on 6 March 2022).
20. Qu, Y.; Yu, S.; Zhou, W.; Chen, S.; Wu, J. Customizable Reliable Privacy-Preserving Data Sharing in Cyber-Physical Social Networks. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 269–281. [CrossRef]
21. Feng, J.; Rong, C.; Sun, F.; Guo, D.; Li, Y. PMF: A Privacy-preserving Human Mobility Prediction Framework via Federated Learning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2020**, *4*, 1–21. [CrossRef]