9-2010

# iPhone 3GS Forensics: Logical Analysis Using Apple iTunes Backup Utility

Mona Bader
*Zayed University*

Ibrahim Baggili
*University of New Haven*, ibaggili@newhaven.edu



## Publisher Citation

Bader, M., & Baggili, I. (2010). iPhone 3GS forensics: logical analysis using apple itunes backup utility. Small scale digital device forensics journal, 4(1), 1-15.

## Comments

Dr. Baggili was appointed to the University of New Haven's Elder Family Endowed Chair in 2015.
Posted with permission.

# iPhone 3GS Forensics: Logical analysis using Apple iTunes Backup Utility

Mona Bader, Ibrahim Baggili, PhD
Ibrahim.Baggili@zu.ac.ae
Advanced Cyber Forensics Research Laboratory
Zayed University

*Abstract* - **The iPhone mobile is used worldwide due to its enhanced computing capabilities, increased storage capacity as well as its attractive touch interface. These characteristics made the iPhone a popular smart phone device. The increased use of the iPhone lead it to become a potential source of digital evidence in criminal investigations. Therefore, iPhone forensics turned into an essential practice for forensic and security practitioners today. This research aimed at investigating and examining the logical backup acquisition of the iPhone 3GS mobile device using the Apple iTunes backup utility. It was found that significant data of forensic value such as e-mail messages, text and multimedia messages, calendar events, browsing history, GPRS locations, contacts, call history and voicemail recording can be retrieved using this method of iPhone acquisition.**

## Introduction

Mobile phones have become an integral part of peoples' daily lives. Personal as well as professional uses of mobile communication devices have increased in the past few years. According to Ayers, market research illustrates that mobile devices double the number of PCs (Ayers, 2008). The advances and innovations in telecommunication technologies have led to a dazzling revolution in the development of smart mobile devices. This introduced a whole new experience of wireless voice and data communication devices, incorporating large data storage capabilities, web browsing, email messaging, global positioning, and many other services that were only accessible through a typical computer system (Jansen & Ayers, 2007). Today, smart mobile devices have become similar to traditional desktop computers in terms of functionality, yet, they are different in terms of usability, operation and organization (Jansen, Delaitre, & Moenner, 2008). Besides Internet connectivity and increased storage capacity of smart phones, their compact size makes it easy for people to carry them anywhere. This led to the increased usage of such devices in day-to-day activities (Jeroen, Elke den, & Yuan, 2008). The unique characteristics of smart mobile devices provide their users with the ability of a portable computing experience rather than being restricted to home or office desktop computers. Statistical analysis conducted by Gartner estimated that 2.6 billion mobile phones will be in use by the end of 2009 (Nena & Anne, 2009).

Modern smart phones store a vast amount of data including contact details, calendar events, calls history, text and multimedia messages, documents, pictures, emails, GPS locations and web browsing history. Moreover, these devices can maintain sensitive data such as passwords, online banking credentials and transactions, on top of the owner's identity (Punja & Mislan, 2008). The extensive and diverse use of these devices make them a rich source of evidence if they were involved in criminal activity. This poses an essential need to study, investigate and further enhance forensically sound methods to handle the examination and analysis of these devices. Cell phone evidence is as important as the digital evidence acquired from a typical computer system. Mobile phones must be seized as part of the investigation process as they may contain potential evidence or valuable data. The ability to recover data from a smart phone device is a critical investigative demand. For example, in two separate murder crimes, the suspects were convicted of committing the crime based on a mobile phone evidence (Summers, 2003).

One of the emerged smart phone devices that is considered an increased technology trend amongst people is the iPhone. Globally, Apple sold 7.4 million iPhones in the fourth quarter of 2009 ("Apple Reports Fourth Quarter Results," 2009). Moreover, an analytics firm stated that 66.44% of mobile web browsing traffic was generated by iPhone devices in February 2009 (Dredge, 2009). iPhone devices provide large data storage capabilities. The new iPhone 3GS mobile lunched by Apple in June 2009 provides storage up to 16 and 32 GB and allows its users to access and download more than 50,000 applications from the App Store ("Apple Announces the New iPhone 3GS," 2009). These capabilities along with Internet access, e-mail, text messages, GPS, and other advanced features cause the iPhone device to accumulate a sizeable amount of personal and business information, in addition to the activities conducted by the owner of the device. Any of the aforementioned features can be of potential forensic value in a criminal investigation. Hence, iPhone forensic examination and analysis is a vital research area to advance and study forensic methodologies and techniques.

iPhone forensics is an evolving field as is the case with the entire cell phone forensics domain. Nonetheless, some journals and reports have documented aspects of iPhone and iPod forensics (Hoog & Gaffaney, 2009; Husain & Sridhar, 2009; Kiley, Shinbara, & Rogers, 2007; Marsico & Rogers, 2005; Punja & Mislan, 2008; Zdziarski, 2008). These research efforts identified three major methods for data acquisition from an iPhone mobile device, which are physical acquisition, logical acquisition and automated software tools.

This research project explores the logical forensic acquisition of an iPhone 3GS using the iTunes free backup utility provided by Apple, followed by an in-depth examination and analysis of the acquired backup copy. This effort attempts to identify and document what data is stored on the device, and where and how the data is stored. The acquisition and examination was conducted on an iPhone 3GS (Third Generation) mobile device.

## Problem Statement

There is a diversity of smart phone devices in use worldwide. The diversity of such devices is due to the difference in smart phone design and technologies, the broad range of proprietary operating systems and software, and the distinct feature set of each model (Baggili, Mislan, & Rogers, 2007; Jansen & Ayers, 2007). Such unique characteristics make the forensic analysis of smart phones distinct from classical computer forensics. The fundamental principle of any forensic acquisition is to obtain a physical bit-by-bit image of the original write-protected evidence media, with a hash value validating that both, the copy and the original digital media, are identical. In the smart phone realm, the acquisition of these devices is conducted on a live system, thus the forensic procedures applied to such devices are distinct from the well-established computer forensic procedures (Punja & Mislan, 2008). This implies the development and use of forensic methodologies and techniques different from the classical forensic methods applied to extract digital evidence from traditional computers (Jansen et al., 2008; Nena & Anne, 2009). As such, this has led to challenging efforts in identifying proper examination methods and tools for smart phone devices.

At present, the forensic community needs to be aware of the significant forensic methods and tools for the iPhone mobile, being one of the most popular smart phones. Statistical studies conducted by Canalys research revealed that Apple occupies the second rank following Nokia and ahead of RIM (Research In Motion) in the EMAE (Europe, Middle East and Africa) smart phone market. The study showed that Apple had shipped 1.9 million devices to the EMEA region in the second quarter of 2009, with a 1041.6% growth from the same time interval of last year ("Smart phones defy slowdown ", 2009). An iPhone can be a tool or an object in a crime scene, and would provide a rich source of evidence due to its increased storage capabilities and Internet connectivity. Data from the iPhone can by synced and backed-up with a coupled computer, therefore this computer can also be a potential source of evidence if it was seized in a crime or security investigation. As such, it is a critical investigative demand to understand how to perform forensic analysis and examination on the backup files acquired from the phone, and find ways to extract data from these files.

This research focused on the forensic processing of the third generation of Apple iPhone mobile to extract significant data from a logical copy. Acquiring a logical backup was performed using Apple iTunes being the only official utility that can access iPhone memory without jailbreaking the file system. This study examined the logical acquisition from the iPhone 3GS flash memory in a forensically accepted process. The logical backup was thoroughly explored and analyzed to locate where significant data from the iPhone is stored within the backup files using freely available tools. The contribution of this research is twofold 1) This research offers practitioners an option of using a freely available tool (iTunes) for performing iPhone forensics and 2) This research documents the forensically relevant backed up data and the location of the data on both the iPhone 3GS and the PC.

## Literature Review

The domain of cell phone forensics is an evolving area due to the unique characteristics of these devices. To discuss the literature on cell phone forensics is beyond the scope of this paper. The literature review section of this paper focuses on the work performed in the realm of iPhone forensics as it relates to the subject under discussion.

### iPhone Forensics

The field of iPhone forensics is in its early stages. To date, few journals and reports on iPhone forensic techniques and tool testing have been published. iPhone forensic acquisition and analysis presents a challenge for forensic examiners due to the embedded nature of the physical components inside the device. The device uses a solid-state flash memory for persistent data storage, and does not accommodate external memory cards. Thus, iPhone forensics can be performed mainly through logical acquisition, which is the most common method used by mobile forensic applications. Yet, some software applications provide physical memory dump capabilities. A distinct physical acquisition technique was introduced to allow forensic examiners to obtain a raw disk copy of the flash memory. This part of the literature review attempts to acquaint readers with existing methods and techniques for iPhone forensic acquisition and analysis.

A unique approach for data evidence recovery from the iPhone device was developed by Jonathan Zdziarski (Zdziarski, 2008), a research scientist known in the iPhone community for his significant research and software development contributions to the iPhone. Zdziarski (2008) asserted that the amount of data accumulated on the iPhone memory is much more than what is perceived to be stored, or what can be obtained through the user interface. Hence, even if data was deleted, it is the reference to the physical location of the data that is deleted, and the actual data remains live on the file system. Although deleted data becomes invisible, it can be recovered using data recovery mechanisms.

Zdziarski's (2008) technique allows the forensic examiner to obtain a bit-by-bit raw disk image of the user partition on the iPhone flash memory. This method provides a proof of integrity of the acquired image by producing a hash value of both the original user partition before the data dump occurs and the copy after acquisition is completed. Being able to verify that original media and the copy are identical, and no data alteration had occurred is one of the basic rules of forensic data acquisition. Nevertheless, the process is quite complicated, and requires high technical expertise on Mac, Linux and Windows platforms.

Acquiring a raw disk image is achieved through a process known as jailbreaking the phone. This process allows the examiner to access and modify the system partition to install the forensic toolkit that is used to image and validate the integrity of the user partition during the device

acquisition. The flash memory partitioning is the key factor on which this technique was built. The iPhone NAND memory is configured with two distinct partitions, a system or root partition, and a user or media partition. The system partition is a read-only partition containing the operating system and the preloaded applications used with the iPhone, and by default doesn't store user data. This partition was designed to maintain its factory state for the entire life of the iPhone, and can only be modified during the firmware upgrade. The second partition was designed to store and maintain all sorts of user data, which makes this part of the memory significantly valuable to a forensic examiner. This particular design was intended to allow Apple to reformat the system partition and upgrade the iPhone software while maintaining the user media intact. Zdziarski (2008) relied on this specific design where both partitions are completely segregated to install the forensic toolkit on the system partition at which the forensic acquisition of the media partition is conducted.

The forensic acquisition process developed by Zdziarski (2008) consists of various procedures to install the recovery toolkit and obtain the physical image from an iPhone with firmware versions 1.x and 2.x. Initiated by switching the iPhone into recovery mode, the device kernel boots a RAM disk that bypasses passcode protection if activated, and allows customized firmware and software that are unauthorized by Apple to be installed. The forensic recovery toolkit is then installed using the device's communication protocol AFC (Apple File Connection) over the USB cable. The recovery toolkit consists of open source tools such as OpenSSH secure shell, the netcat tool for sending data across networks, the md5 tool for generating the hash value of the media and acquired image, and the dd disk copy/image utility that is used to obtain the raw disk image.

Once installed, the examiner gains direct shell access to the file system, and can perform the traditional acquisition functions starting by calculating the hash value of the entire media partition before transmitting the data, to verify that the partition data hasn't been altered while in transit. The raw disk image is then acquired and transmitted over a preconfigured wireless connection between the iPhone device and the forensic workstation. Once the imaging process is completed, an MD5 hash value of the image is calculated again to satisfy the objective of an identical copy. The acquired bit-by-bit copy of the media partition contains both live and deleted data. It can be imported into commercial forensic tools for further analysis and examination. At this point, free data carving tools can be used to recover and extract files from allocated and unallocated space.

Data carving tools scan the raw disk image for traces of desired file or data types, such as images, voice messages, dynamic dictionaries, property lists, SQLite databases, and other files, and then carve those files out of the image for further analysis. Primarily, the iPhone stores data such as contacts, text messages, email messages, and other personal data in database files. Being able to extract and analyze these database files would potentially reveal some deleted evidence data. The content of these databases files can be accessed and viewed using specific viewer tools such as the SQLite command-line client or SQLite Browser (Zdziarski, 2008).

While this approach recovers a raw disk image and allows examiners to retrieve deleted data, it involves modifying the file system which might affect the forensic reliability of this technique. Zdziarski (2008) affirmed that his approach maintains the user partition untouched during this process, which is primarily what signifies this jailbreak process from the traditional jailbreak methods that make changes on a user partition, allowing the installation of third-party applications on the system. However, Zdziarski's jailbreak method requires the device to be rebooted after the recovery toolkit is installed. This involves minor writes to replace or reset certain configuration files on the media partition upon booting. Some of these writes append a small amount of data to those files (Zdziarski, 2008). Hence, the forensic examiner should assess the implications of such data alteration caused by jailbreaking the device on the forensic acceptance of recovered evidence and the admissibility of this evidence by legal systems.

Today, few proprietary forensic tools can be used to recover data from an iPhone mobile device. These tools vary in their technical implementation, acquisition procedures, amount of recovered data, and how they report results. A report published by Hoog and Gaffaney (2009) on iPhone forensics provides a comprehensive technical review on available forensic software products and techniques used for data recovery from the iPhone device. The authors examined eight different software tools and techniques, compared output results, and ranked the tools. The basic objective of this report was to provide forensic specialists and law enforcement with reliable feedback on available tools for iPhone forensic examination with a reasonable reflection on tool performance and scope of data recovery. The forensic examination was performed on a 3G iPhone device with a non-jailbroken firmware version 2.2, that has been used for about six months. Twenty seven examination scenarios were created. They included all possible sorts of data available on the memory storage such as, call logs, contacts, SMS, emails, calendar, web history, pictures, passwords, etc. The analysis methodology focused on four examination areas, which are installation, acquisition, reporting and accuracy. The accuracy of each tool or technique was determined by comparing the results of the acquisition to the expected results that are available on the device. However, the ranking of these tools and techniques were based on the authors' individual experience during the entire examination process.

The tested tools and techniques utilized three distinct mechanisms for data acquisition. The first method acquires data directly from the iPhone mobile. This technique must be applied carefully and should be performed by an expert forensic specialist. The second mechanism acquires the logical copy of the mobile file system using Apple's proprietary synchronization protocol. This protocol is used to synchronize the phone data and files with the associated PC. The backed up data is stored in SQLite databases and requires a viewer to be able to read data. These database files generally maintain deleted SMS and email messages. The last

mechanism, proved to recover more data than the previous two mechanisms. However, this approach is not an integrated software solution, but instead, a combination of open source tools and procedures used to acquire the physical bit-by-bit copy of the entire user partition of the device memory. It also required modifying the device file system (jailbreaking the phone) to allow the installation of acquisition utilities. This is basically the approach developed by Zdziarski which was discussed earlier (Hoog & Gaffaney, 2009). With relevance to this research paper, by comparing the acquired results of SMS and email messages from the tested tools and techniques, it was observed that most of the tools were able to retrieve undeleted text messages, although Zdziarski's technique was able to recover a significant number of deleted rows in SQLite databases. Comparatively, some tools were able to retrieve some email accounts and folder information, yet Zdziarski's technique was the only method that mostly recovered all of the email messages available on the system.

Despite the significance of physical acquisition that proved to recover more data than other approaches, the logical acquisition of the iPhone provided value in terms of evidence data recovery under forensically accepted conditions. Based on the logical acquisition approach, a forensic analysis of Instant Messaging (IM) conversations on the iPhone was presented by Husain and Sridhar (2009) from the University at Buffalo. The forensic examination focused on retrieving evidence data from Instant Messaging online conversations without altering the device's firmware. Being one of the convenient methods for interpersonal communications, the authors highlighted the importance of forensic analysis of IM conversations on smart phones in anticipation of their involvement in cyber criminal activities.

The tested IM applications included the client-based and the volatile web-based versions of AIM, Yahoo! Messenger, and Google Talk services on an Apple iPhone 3G with firmware version 2.2.1. The authors primarily attempted to compare results of forensic analysis of the traditional client version that requires the download and installation of provider's software, and the web-based Volatile Instant Messaging (VIM) that doesn't require software installation but instead can be accessed through the web browser. The examination approach relied on analyzing the iPhone logical backup acquired through the Apple File Communication Protocol used by iTunes to copy data between the iPhone mobile and a forensic examination machine. The analysis of both versions of IM applications yielded different results in terms of what data could be retrieved. While the client-based messenger conversation retained various valuable data on the device such as a conversation log, screen name, password, account information, and buddy list, the Volatile web-based messenger didn't preserve any evidentiary data on the iPhone (Husain & Sridhar, 2009).

## Methodology

This research explored the forensic processing of the new third generation of Apple iPhone 3GS mobile in an attempt to recover a logical backup using the Apple iTunes backup utility. The testing was conducted under forensically accepted conditions, without breaking into the file system, to keep the forensic acquisition legally sound. The fundamental rule in any forensic acquisition is that the process doesn't alter or contaminate the original data.

The examination methodology employed the Computer Forensics Tool Testing program guidelines established by the National Institute of Standards and Technology (NIST) (*General Test Methodology for Computer Forensic Tools Version 1.9*, 2001). NIST's testing approach was developed to provide a quality measure of assurance that the forensic tools used in computer investigations present reliable and valid results. This program aimed to set international standard guidelines that forensic tool developers and investigators can use when developing and testing these tools.

### Logical Acquisition Approach

The logical acquisition approach is based on acquiring a logical bit-by-bit copy of the directories and various types of files found within the iPhone file system. Logical backups are considered a rich source of data files that can help build evidence. They can also provide proof of the pairing relationship between the computer that has been previously synched with the iPhone device if that computer was seized as part of the investigation.

In this research, the logical copy was obtained using the iTunes backup feature that utilizes Apple's synchronization protocol to copy the iPhone live data to a forensic workstation. iTunes is the software application used by Apple to synchronize content on iPhone or iPod Touch with a coupled computer. When the iPhone mobile is synched with the computer, the device's configuration, address book, calendar, images, SMS database, email accounts, web history, and other sorts of personal data is saved on the computer in backup files in a single directory. By default, the iTunes application creates a backup of the iPhone data during the sync process. When iTunes syncs the iPhone with the computer, it copies data from the iPhone to the PC and vice versa to ensure that content is same on both. Consequently, iTunes may copy the computer's address book, calendar, image files, email accounts and other data to the iPhone memory. ***Hence, in a forensic examination it is important to invoke the backup process independently without initiating the synchronization to avoid the risk of data cross-contamination during the forensic logical acquisition***. The acquired backup was parsed and viewed using specific tools such as plist Editor, SQLite Database Browser, and some other tools are discussed in the subsequent sections. Recovered data was examined to extract evidentiary data including the device's serial number, firmware version, phone number, in addition to the known existing and deleted data.

### Examination Process

The examination methodology applied in this research paper is a subset of the general NIST's approach for forensic tool testing. Examination procedures included:

1. Examination Requirements:

The acquisition approach focused on retrieving data from the iPhone 3GS internal flash memory. The examination attempted to locate and trace database and configuration files containing significant data. The testing explored the acquisition process, searching, and data recovery features.

2. Examination Plan and Test Cases:
   The test case scenario involves all data types stored on the iPhone mobile. No predefined data is set, since the examination attempted to retrieve and extract different data files associated with the various applications installed on the phone.

3. Acquisition and Examination Tools
   The logical acquisition required the Apple iTunes synchronization application to create the logical backup of the iPhone file system. The iTunes application was used on both Mac and Windows platforms. Furthermore, some free and open source tools were used for data analysis and recovery including plist Editor, SQLite Database Browser, and iPhone Backup Extractor.

4. Examination Environment Setup and Test Procedures
   A key requirement in forensic testing is the test environment where the acquisition, analysis and recovery procedures are implemented. In this research, two forensic computer workstations were used during the forensic examination. These computer systems were configured with Windows XP and Mac OS X Leopard operating system platforms. The logical acquisition and examinations were conducted on both platforms to explore the capabilities of each platform's tools to acquire and extract data from the iPhone device. Acquisition and analysis tools listed above were installed, configured and validated on the forensic workstations before starting the acquisition process. The forensic workstations were also isolated from the forensic laboratory network. According to the "Best Practices for Mobile Phone Examination" provided by SWGDE ("Best Practices for Mobile Phone Examination," 2009), the equipment the examiner uses to conduct data acquisition and analysis of the evidence should be validated, and the software applications should be installed and validated prior to its use.

   Forensic acquisition and analysis procedures should comply with the rule of evidence policy that emphasize on maintaining the source media intact and that content is not altered or modified. No attempts of altering the firmware was performed during the acquisition and analysis process. The Apple iPhone data cable was used to interface the device with the forensic workstation.

5. Results

   The main goal of the forensic acquisition and analysis is to produce evidence admissible by legal systems. Data acquired was examined and searched to locate and recover evidence data. The output results of the examination highlighted the various data files included in the logical backup, and their respective locations.

**Forensic Acquisition**

The examined Apple iPhone 3GS (Firmware v3.1.2) was physically connected to the preconfigured forensic workstations using the iPhone USB data cable. ***An attempt to connect the iPhone through a write-blocker device failed***. The backup utility needs to mount the iPhone file system to access and retrieve data. The researchers speculate that a write-blocker hinders the backup utility from initiating a connection with the iPhone to mount the file system. This indicates that the iTunes backup utility may need to write to the iPhone file system to mount the device`s storage media on the computer. Yet, with the aforementioned methodology in place, the acquisition was still in a controlled environment. Consequently, without a write blocker, a direct connection was established between the iPhone mobile and the forensic workstation. The acquisition of the device's internal memory was conducted on the iPhone's live system. The iPhone communicated with the computer using the Apple File Connection protocol (AFC) over a USB cable. The logical acquisition was conducted on both Mac OS X Leopard version 10.5 and Windows XP platforms.

**Logical Acquisition**

The logical acquisition of iPhone was performed using the Apple iTunes application. iTunes is the freely available software application provided by Apple for data synchronization and management between the iPhone and the associated host computer. The iTunes application is not designed for forensic acquisition, but is primarily used to sync and copy data back and forth between the iPhone mobile and the associated computer to ensure that content is current and up-to-date on both.

iTunes is configured by default to automatically initiate the synchronization process once the iPhone is connected to the computer. Nonetheless, the main objective of the logical acquisition is to obtain a logical copy of the mobile file system without contaminating the data on the mobile. For this purpose, the synchronization settings in the iTunes application was adjusted to disable the automatic syncing process when the mobile is connected to the computer, so that the backup process can be initiated independently. This was critically important to avoid data exchange back and forth between the mobile phone and the computer. Figure 1 illustrates the option that disables automatic syncing prior initiating the physical connectivity between the iPhone mobile and the computer.

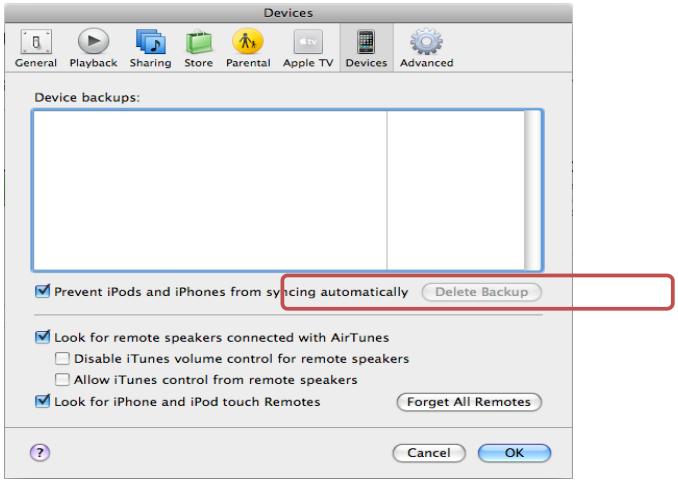


**Figure 1: Disable automatic syncing from iTunes Preferences**

The latest versions of iTunes were installed and configured on the forensic workstations. iTunes version 9.0.2.25 was used on Windows XP, and version 9.0.2 (25) was used on the Mac OS X Leopard forensic workstation. Once the installation and synchronization settings were verified, the iPhone mobile was connected to the forensic workstations via the iPhone USB data cable. iTunes detected and identified the connected device, and this was verified by the phone icon displayed under Devices on the left sidebar of the iTunes interface. A single click on the phone icon displayed summary information of phone's name, capacity, firmware version, serial number, and phone number as shown in Figure 1. The backup process was initiated manually by right-clicking on the device icon and selecting the 'Back Up' option. Once the backup process was completed, the device was disconnected from the forensic workstations to prevent undesired operations from occurring.


**Figure 2: iPhone summary information on iTunes interface**

By default, iTunes stores backup files in a preconfigured directory for different operating systems. On Mac, backup files are located in *~/Library/Application Support/MobileSync/Backup*. On Windows, backup files are located in *\Documents and Settings\(username)\Application Data\Apple Computer\MobileSync\Backup*.

Older versions of iTunes were used on both Mac and Windows platforms to test the recovery of a logical backup. However, ***iTunes versions prior to 8.2 couldn't recognize and use the iPhone 3Gs device***. Logical backup from a passcode protected iPhone, or iPhone with backup encryption feature enabled can also be obtained ("iPhone 3Gs forensic imaging," 2009; Zdziarski, 2008). Putting the device in recovery mode at start-up loads a custom RAM disk in the memory of the phone where raw commands can be used to bypass active passcode protection. Performing a logical backup with an iPhone passcode however is outside the scope of this research project.

According to Apple, iPhone backup files store personal data and iPhone configuration settings. This includes application settings and preferences, network settings, paired devices, call history, address book, web browser bookmarks, history and cookies, map bookmarks, notes, SMS messages, calendar accounts, mail accounts, voicemails, photos and videos taken by built-in camera, and offline web application cache/database ("iPhone and iPod touch: About backups," 2009). However, by default, iTunes does not backup cached email messages retrieved through the Apple Mail application, as well as photos that have been previously synced with the associated PC. Further investigation on various data recovery was conducted during the examination and analysis phase which is discussed in the following section.

## Examination and Analysis

The examination and analysis of the acquired logical backup was a time consuming process. It involved exploring and examining hundreds of data files that were copied during the logical backup acquisition. Moreover, multiple tools and techniques were explored and tested to parse these backup data files to reverse engineer the data stored in the backed-up files.

### Inside the Logical Backup

Logical backups acquired by the Mac and Windows forensic workstations were examined independently. The examination of the logical copies were purposely performed on both environments to explore the capabilities and available tools for each platform. This was also used as a method to cross-validate the recovered data.

The '*Backup*' folder contained the relevant folder where the backup files were actually stored by iTunes. The name of the backed-up folder is a long combination of forty hexadecimal numbers and characters (0-9 and a-f), and represents a unique identifier for the device from where the backup was obtained. This unique identifier appears to be a hashed value since it was the same unique name given to the backed-up folder by iTunes on both Mac and Windows operating systems. Within this folder resides hundreds of backup files with long hashed filenames consisting of forty numbers and characters. These filenames signify a unique

identifier for each set of data copied from the iPhone memory. Backed-up data is stored in three file formats, plist files which stores data in plaintext format, mddata files which stores data in a raw binary format and mdinfo files which store encoded metadata of the corresponding binary mddata files. Figure 3 shows the Backup folder containing the backed-up files.

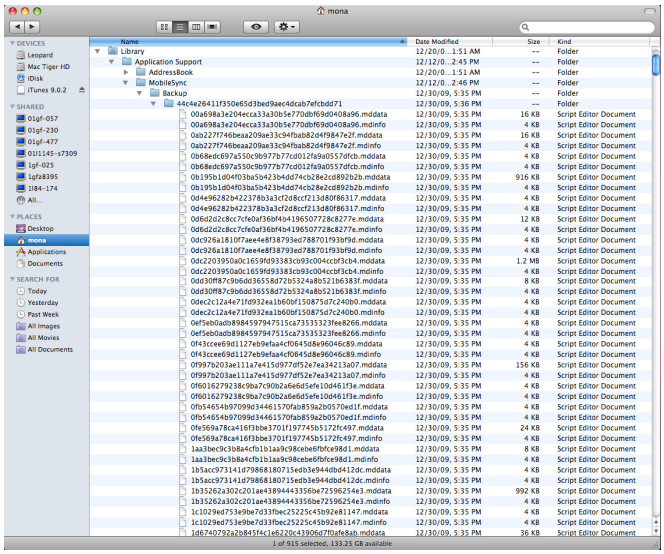

**Figure 3: Backup folder**

Generally, the iPhone file system stores data in binary lists and database files. The device configuration, status, applications settings and preferences are stored in XML format plist files. Such configurations include, current time zone, pairing records with devices and computer, email accounts, network identification, browser history, cookies and bookmarks. Whereas user data such as SMS messages, email messages, contacts, call history, notes, calendar events, and other types of personal data are stored in SQLite database files. Table 1 and 2 below summarize the content available in SQLite database and plist files.

| SQLite Database files | Content |
|---|---|
| Keychain-2.db | Accounts, services associated with the accounts, and encrypted passwords |
| AddressBook.sqlitedb | Address book contact information |
| AddressBookImages.sqlitedb | Images associated with saved contacts |
| call_history.db | Incoming and outgoing call logs |
| Calendar.sqlitedb | Calendar events |
| notes.db | Note files |
| sms.db | Text and multimedia messages |
| 0000000000000001.db | Email messages accessed on Gmail Web interface |
| 0000000000000003.db | Translation terms searched on Google Translate Web interface |
| voicemail.db | Voicemail messages |
| Recordings.db | Voice memos recorded on the device |
| friends.db | Facebook friends list |

**Table 1: Content of SQLite database files**

| plist files | Content |
|---|---|
| com.apple.accountsettings.plist | Email accounts configured on Apple Mail application |
| Directions.plist | Directions to remote locations that have been queried |
| History.plist | Log of searched locations |
| com.apple.Maps.plist | Last viewed latitude and longitude |
| com.apple.mobilephone.speeddial.plist | Speed dial contacts saved in the Favorites list |
| com.apple.mobilephone.plist | Last phone numbers dialed |
| Bookmarks.plist | Bookmarked URLs |
| History.plist | Browsing history |
| Cookies.plist | Information about cookies saved by visited websites |
| com.apple.preferences.datetime.plist | Local date and time zone |
| com.apple.network.identification.plist | Wireless networks accessed by the device |
| com.apple.wifi.plist | Wireless network settings |
| com.apple.preferences.network.plist | Status of wifi and Bluetooth networks |
| com.apple.MobileBluetooth.devices.plist | Log of Bluetooth devices paired with the iPhone |
| com.apple.MobileBluetooth.services.plist | History of Bluetooth pairings |
| com.apple.commcenter.plist | ICCID and IMSI unique identifiers |
| Info.plist | Device information including device name, unique identifier, phone number, serial number, etc. |

**Table 2: Content of plist files**

SQLite is the database system used by Apple to store data on the iPhone. For example, 'sms.db' stores SMS messages, 'Envelope Index' stores email messages, 'notes.db' stores notes items, and 'call_history.db' stores call history, 'Calender.sqlitedb' stores calendar data, and 'AddressBook.sqlitedb' stored address book data (Zdziarski, 2008).

During the backup process, plist and database files on the iPhone file system are encoded into mddata files storing data in XML, ASCII and binary formats, with a corresponding binary metadata (mdinfo) file for each mddata backup file. Each mdinfo file holds the name of its corresponding mddata file and stores metadata of that file in an encrypted binary format. In addition, the backup utility creates three plist files and stores them in the backup directory once the backup process is completed. The first plist file, Status.plist confirms the success of the backup process. The second file is Info.plist contains device information such as device name, IMEI, ICCID, timestamp of last backup, phone number, firmware version, device serial number, and device unique identifier. This data can be used as evidence linking this particular iPhone to the coupled computer where a backup could have been created during a previous synchronizations run by a suspect. The third file, Manifest.plist, contains a list of all backed-up files created during the backup process along with their file size, modification time, and their hash signatures, all in an encoded format. These three plist files can be viewed using a text editor or a plist editor utility.

The ability to view the data in the backup files varies depending on file formats that were saved during the logical copy. The content of the mddata backup files is stored in raw binary data, and hence cannot be directly viewed using iTunes, Text Editor, or any other utility provided by Apple. Instead, they need to be first decoded into a readable format. Once converted back to their native file structure which are either SQLite database files or plist files, they can then be accessed and read using the proper tools. The data stored in SQLite database files can be retrieved and viewed using either 'sqlite3' command-line client ("Command Line Shell For SQLite,") which is built into Mac Leopard OS, or by using a GUI utility such as 'SQLite Database Browser' which is an open source tool designed to browse SQLite databases and is available for both Mac and Windows platforms ("SQLite Database Browser,"). The content of backed-up plist files can be viewed in a text editor if they were saved in ASCII format or in a 'Property List Editor' utility if the data was saved in binary format. This editor utility is capable of converting and displaying the binary data into readable ASCII format and is available for both Mac and Windows platforms ("plist Editor for Windows; PlistEdit Pro 1.5,").

There are two mechanisms for tracing data from backup files, which are either by searching through mddata files manually using the Command-line utility, or by using GUI parsing tools that can decode the binary data and convert it back to its native usable files from which data can be accessed and retrieved in plaintext readable format.

**Manual Examination**

The iPhone operating system, OS X iPhone, is derived from Mac OS X 10.5 (Leopard) ("iPhone OS," 2009), which makes the phone file system natively compatible with the Leopard based Mac. Hence, it was more manageable to utilize the Mac GUI and command-line parsing tools to search, parse and retrieve the data backed-up from the iPhone mobile device. Nevertheless, Windows-based tools were also used to cross-validate the results obtained from the Mac-based examination and analysis.

Each logical backup file is coupled with a specific application on the iPhone, and hence, it contains either the application settings and configurations in plist files, or the user data associated with that application in SQLite databases. There is a wide range of Apple applications that can be installed and used on iPhone mobiles. These applications are either integrated within the iPhone file system or installed by the phone user via the App Store. Such applications include, Mail, Messages, Contacts, Calendar, Call History, Notes, Voice Memos, Voicemail, Safari, Maps, Photos, Facebook, AIM, eBay Mobile, Tweetie, Google Earth, and many more.

By manually searching through backed-up files for key data types, the authors could identify the backup files containing that exact set of data. To locate traces of various types of data in the binary backup files, '*grep*' and '*find*' commands were executed from within the backup directory in the Command-line utility on Mac and Windows respectively. This command returns the name of metadata file or files containing the searched keyword. A metadata file contains the type, path and name of the corresponding backed-up file in an encoded format. Using a Base64 decoding tool, this data can be encoded and retrieved. The file containing the data can then be located in the binary mddata backup file holding the same file name as the resulted metadata file. The file format of the associated mddata files can be verified by reading the first few characters within a text or hex editor utility. The text 'SQLite format 3' indicates that the backup file contains a SQLite database, whereas, the text 'bplist00' indicates that the backup file contains a binary plist data.

Fetching for key data in backup files manually can return more than one matching backup file. Content of resulted backed-up files were then inspected using proper tools such as SQLite Database Browser, plist Editor, and a Base64 decoder. Once content was verified, the name of the backup file saved on the computer was then linked to the actual database or plist file on the iPhone file system. The section below outlines the results obtained from the manual examination process explained.

**Results - Backed-up Databases**

As mentioned earlier, the iPhone file system utilizes SQLite database software to store the vast amount of user and application data in databases with .db and .sqlitedb file extensions. These databases can be found by searching for the known name of the database file as stored on iPhone media. As an example, the command line '*grep "sms.db"* *' located the metadata file associated with the backup file storing the

actual text message data. The SMS database file is then located in the binary mddata backup file holding the same filename as the acknowledged metadata file. By opening the binary file in a text editor, the first few words in the file states 'SQLite format 3' which verifies that it is a SQLite database. Once the backup file containing the database is located, the backup file can be parsed and decoded back to its native readable file format. By accessing the database file using sqlite3 command-line utility on Mac Leopard, further database details can be explored, such as, tables contained in the database, table schemas, in addition to data stored in the tables.

From here, a forensic examiner can view each and every single record, search for particular data, and redirect data into a text file by executing SQLite built-in commands and SQL queries. Using the backup file containing the SMS database 'sms.db' presented earlier as an example, database tables can be displayed using the '*.tables*' command, table schema can be understood using the '*.schema <table_name>*' command, and commands output can be saved into a text file using the '*.output filename*' command. Moreover, SQL queries can be executed to retrieve records from the database tables, for instance, '*select * from message*' would retrieve all records stored in the message table. SQL join queries can be issued as well to obtain records from multiple tables as shown in the following SQL statement,

'*select message.rowid, message.address, message.date, message.text, message.country*
*from message, msg_group*
*where msg_group.newest_message = message.rowid*'

This SQL query obtained the latest message sent to or received by each mobile number stored in the SMS database. Figure 4 illustrates SQL statements querying SMS SQLite Database.

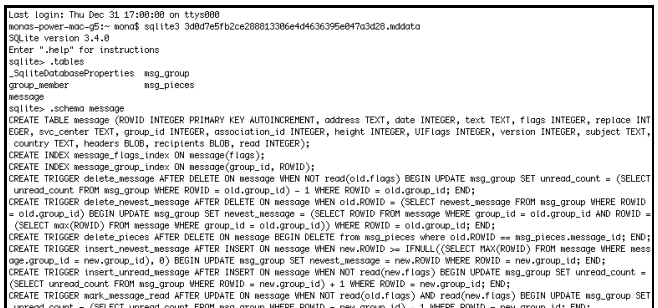


**Figure 4: Querying SMS SQLite Database**

The same data can be displayed and retrieved using the SQLite Database Browser (on Mac or Windows), which provides a GUI interface to examine tables, schemas and stored data. It also provides the capability to export tables directly into CSV text files.

We list below our detailed findings of mddata backed-up files containing known SQLite databases used by the iPhone file system to store user and application data.

***Keychain Database***

| Database name on iPhone | Backed-up database file on PC |
|---|---|
| keychain-2.db | 631e60f16fdb96ac2dea025ba07e858a3095efa6.mddata |

The most important tables in the keychain database are the *genp* and *inet* tables. These tables contain accounts, services that these accounts are associated with, and encrypted passwords. Examples of stored accounts include, email accounts, paired Bluetooth devices accounts, Wi-Fi network accounts, in addition to the device lock password if the passcode lock is enabled on the device.

***Address Book Database***

| Database name on iPhone | Backed-up database file on PC |
|---|---|
| /Library/AddressBook AddressBook.sqlitedb | 31bb7ba8914766d4ba40d6dfb6113c8b614be442.mddata |
| /Library/AddressBook AddressBookImages.sqlitedb | cd6702cea29fe89cf280a76794405adb17f9a0ee.mddata |

The Apple Contacts application stores address book data in two separate databases. The first database *AddressBook.sqlitedb* consists of eighteen tables storing contact information and other related personal data, such as, name, phone numbers, email addresses, birthday, organization, department, job title, nickname, etc. Primary database tables include *ABPerson*, *ABMultiValue*, *ABRecent*. The second database *AddressBookImages.sqlitedb* mainly saves images associated with saved contacts in the table *ABImage*. Image data is stored in an encoded format, and can be dumped and converted back to binary format using Perl scripts.

***Call History Database***

| Database name on iPhone | Backed-up database file on PC |
|---|---|
| /Library/CallHistory/ call_history.db | ff1324e6b949111b2fb449ecddb50c89c3699a78.mddata |

This database stores incoming and outgoing call logs including phone numbers, timestamps, in addition to call duration in the table *call*.

***Calendar Database***

| Database name on iPhone | Backed-up database file on PC |
|---|---|
| /Library/Calendar/Calendar.sqlitedb | 2041457d5fe04d39d0ab481178355df6781e6858.mddata |

This database stores data entered into or synced with the Apple Calendar application. It contains multiple tables, however, the most important table is the *Event* table. This table stores event summaries, descriptions, time intervals, locations, in addition to time zone information. Some of the tables store other calendar related information including other calendars synced with iPhone in the Calendar table, events that have been modified in *EventChanges* table, events that have taken place multiple times in *OccurrenceCache* table, and events that have recurred in the *Recurrence* table.

***Notes Database***

| Database name on iPhone | Backed-up database file on PC |
|---|---|
| /Library/Notes/notes.db | 740b7eaf93d6ea5d305e88bb349c8e9643f48c3b.mddata |

This database is utilized by the Apple Notes application to store note files typed in or synced with the application. Note files data is stored in two tables; *Note* and *note_bodies*. The *Note* table maintains the notes creation date, title, summary, modification date, and the note author. The *note_bodies* table maintains the entire note content.

***SMS Database***

| Database name on iPhone | Backed-up database file on PC |
|---|---|
| /Library/SMS/sms.db | 3d0d7e5fb2ce288813306e4d4636395e047a3d28.mddata |

SMS database is one of the most important databases on the iPhone device. It preserves text and multimedia messages sent and received through the Apple SMS application. Actual text messages along with their timestamps, phone number of communicating party in addition to other data are stored in the *message* table. The *group_member* table assigns a unique group number for each phone address that has communicated with this iPhone through the SMS application. The *msg_group* table keeps track of recent messages sent to and received by each phone number stored in the *group_member* table. In addition, records of MMS messages are stored in both the *message* and *msg_pieces* tables. MMS message content-related data such as text, content type, content location, and message id are logged in *msg_pieces* table. Other message data such as the associated phone number, message timestamp, group id, as well as the message id are logged in the *message* table. Both tables are linked by the message id, which is a unique number assigned to each sent or received message. Hence a full MMS message record can be combined and retrieved from the two tables based on the message id.

***Email Database***

| Database name on iPhone | Backed-up database file on PC |
|---|---|
| /Library/WebKit/Databases/Databases.db | 970922f2258c5a5a6d449f85b186315a1b9614e9.mddata |
| /Library/WebKit/Databases/http_mail.google.com_0/0000000000000001.db | 420714fa93c5f7d9b78c3d4c50ce0e48a91af641.mddata |
| /Library/WebKit/Databases/http_help.apple.com_0/0000000000000002.db | 89718d723b510d9f4b02de11ec1f59e789198aaf.mddata |
| /Library/WebKit/Databases/http_www.google.com_0/0000000000000003.db | e748422c92368b1481ff400476358b3cc564d39a.mddata |

The iPhone can be configured to access multiple email accounts including Microsoft Exchange, Gmail, yahoo Mail, MobileMe, AOL and many other email services through the Apple Mail application. This application mainly stores cached email messages on the local device in a SQLite database named 'Envelope Index'. As mentioned earlier, the Apple iTunes utility does not backup this database during the backup acquisition process. Yet, searching through the backed-up SQLite databases revealed a collection of four database files saved under the directory */Library/WebKit/Databases*. These database files are utilized by the Apple Safari application to store http based services that are accessed through the Safari web browser on the iPhone.

- *Databases.db* file mainly keeps a record of the three web-based services of Gmail, Google Translate and Apple User Guide. Gmail web interface.
- *0000000000000001*.db file stores Gmail messages that have been accessed through the Gmail web interface on the Safari web browser which looks very similar to the Apple Mail application. Primary tables in this database include *cached_messages, cached_contacts, cached_conversation_headers* tables. *cached_messages* table keeps a log of sent and received email messages stored locally on the device. Key data includes message id, conversation id, sender email address, recipients' email address, carbon copy and blind carbon copy email addresses, message content, and receiving timestamp. *cached_conversation_headers* table contains data about cached conversations that have taken place between the user and the communicating person.

This data involves conversation id, fragment of the cached conversation, sender's name in addition to conversation timestamp. *cached_contacts* table keeps track of stored email addresses and names.

- *0000000000000002.db* file is mainly where the Apple iPhone User Guide information is stored, and the information is mainly accessible through the Safari web interface. However this database doesn't include data of forensic significance.
- *0000000000000003.db* file keeps a log of translation phrases that have been searched on the Google Translate web interface. This is a simple database and mainly stores the phrases, translations, timestamp of translation, source and destination languages in the *Translations* table.

***Voicemail Database***

| Database name on iPhone | Backed-up database file on PC |
|---|---|
| /Library/Voicemail/voicemail.db | 992df473bbb9e132f4b3b6e4d33f72171e97bc7a.mddata |

It is a simple database used by the iPhone file system to store data about voicemail messages received on the device. The *voicemail* table keeps track of sending phone number, message timestamp, message duration, callback number, message expiration date, and message deletion date.

***Recordings Database***

| Database name on iPhone | Backed-up database file on PC |
|---|---|
| /Media/Recordings/Recordings.db | 303e04f2a5b473c5ca2127d65365db4c3e055c05.mddata |

Recordings database contains data about voice memos recorded on the iPhone device using the Apple Voice Memos application. The primary table used to store this data is the *ZRECORDING* table, which contains the recorded file name, storage path where the file is saved on the file system, recording timestamp, as well as the recording duration. Actual voice memos recordings are saved within the same directory where the Recordings database is stored.

***Facebook Application Database***

| Database name on iPhone | Backed-up database file on PC |
|---|---|
| /com.facebook.Facebook/Documents/friends.db | 6639cb6a02f32e0203851f25465ffb89ca8ae3fa.mddata |

This database is maintained by the Facebook application on the iPhone and mainly contains a log of the friends on the Facebook profile of the device user. The *friend* table within the database stores the full name, first name, last name, unique id and phone numbers for each friend in the list. This table also contains the URL address pointing to the friend's profile picture on Facebook.

Each database associated with an Apple native application maintains a table of properties called *_SqliteDatabaseProperties* mainly storing the database and application related attributes. Primary attributes include the database unique identifier and the application client version.

**Results - Backed-up plist Files**

Property List files store a wealth of data that can be of forensic significance. Such data includes email accounts, maps history, cookies, bookmark, browsing history, pairing records, and many other forensic significant data. Binary plist files are identified by the following characters 'bplist00' at the very beginning of the file when it is opened with a text editor. The content of the plist file can then be viewed using the Property List Editor tool. The listing below looks at the various plist files included within the iPhone logical backup copy and their matching mddata backup filenames.

***Mail Accounts***

| plist file name on iPhone | Backed-up plist file on PC |
|---|---|
| /Library/Preferences/com.apple.accountsettings.plist | 5fd03a33c2a31106503589573045150c740721dd.mddata |

This plist file maintains data about email accounts that are configured on the Apple Mail application to access email services. Multiple email accounts from different email services may be configured on the device, hence, significant information may be gathered from this plist file. Such information includes, email address, full username, display name, outgoing mail server with the associated protocol and port number, incoming mail server with associated protocol and port number, SSL configuration, hashed account password, in addition to the account path which seems to be the location where mailboxes for that particular account are stored.

***Maps Directions and History***

| plist file name on iPhone | Backed-up plist file on PC |
|---|---|
| /Library/Maps/Directions.plist | b88b75bddaa69139b66d948b7cbd4f41d9dd416d.mddata |
| /Library/Maps/History.plist | b60c382887dfa562166f099f24797e55c12a94e4.mddata |
| /Library/Preferences/com.apple.Maps.plist | a30335a2c0f0316c9610d868a527b2ade1911542.mddata |

The *Directions* plist mainly stores remote locations that have been queried for directions on Apple Maps along with the starting point from where the search should provide directions. The *History* plist keeps track of searched queries including: locations searched, latitude, latitude span, longitude and longitude span. The *Maps* plist contains the

last viewed latitude and longitude, search strings, in addition to route start and end points.

***Call Favorites***

| plist file name on iPhone | Backed-up plist file on PC |
|---|---|
| /Library/Preferences/ com.apple.mobilephone .speeddial.plist | b64e73540b6221bffc16b18f2205e 1335e31d7d8.mddata |
| /Library/Preferences/ com.apple.mobilephone. plist | fb7786ced1add24313fa258c8e1ed 041e24d52a4.mddata |

The *speeddial* plist maintains contact numbers linked from the Address Book and saved in the favorites list on the Apple Phone application for speed dialing. The list contains the associated User ID saved in the Address Book database as well as the database unique identifier. The *mobilephone* plist file provides significant information on the last phone number dialed by keying the number into the phone keypad, the last phone number dialed from the Address Book, and the timestamp of last time the Recents list have been accessed.

***Safari Bookmarks, History and Cookies***

| plist file name on iPhone | Backed-up plist file on PC |
|---|---|
| /Library/Safari/Book marks.plist | 04cc352fd9943f7c0c3f0781d4834fa1 37775761.mddata |
| /Library/Safari/Histo ry.plist | 1d6740792a2b845f4c1e6220c43906d 7f0afe8ab.mddata |
| /Library/Cookies/Co okies.plist | 1dd07f2fbb1169bed93c21047ca5616 371ea4a04.mddata |

The *Bookmarks* plist simply saves the URL address of bookmarked websites. These websites have been either manually created within the Safari web browser or copied from the default browser on the PC with which the mobile has been synchronized with using iTunes. The *History* plist keeps a record of the user's browsing history on Safari. History data includes the visited URL, visit count, last visit timestamp, and the redirecting URL. *Cookies* plist, as well, maintains key data about cookies saved by websites accessed on the Safari web browser. This file contains the date cookies were created, domain names the cookies belong to, and cookies expiration date.

***Time Zone***

| plist file name on iPhone | Backed-up plist file on PC |
|---|---|
| /Library/Preferences/ com.apple.preferences. datetime.plist | 3c54cb1e89c54d3c09664c5b8311c 0a00f9ea06e.mddata |

This is a very simple file that saves the local date and time zone configured on the device.

***Wifi Network Preferences***

| plist file name on iPhone | Backed-up plist file on PC |
|---|---|
| /Preferences/SystemCo nfiguration/ com.apple.network.ide ntification.plist | 64852404d8347fafdc95c5d68f762 9995baef161.mddata |
| /Preferences/SystemCo nfiguration/ com.apple.wifi.plist | 34f7f8423d8f77bc812dd8d70f84c3 3a5caacbe8.mddata |
| /Library/Preferences/ com.apple.preferences. network.plist | 2b70b844834321b9f4e9760531ce7 6db7819afd3.mddata |

The *network.identification* plist file keeps track of wireless networks the iPhone has been configured to access. Saved data includes the acquired IP address at time of connection, wireless router IP address and hardware address, DNS address, connection timestamp, in addition to a unique ID assigned to each network service provider. The *wifi* plist on the other hand mainly stores wireless network settings such as the disassociation interval, join mode, BSSID, and authentication mode. Other significant data available in this file includes the SSID, the date of last time the network was joined, and the path where the wifi.log file is saved. The *network* plist, is a very simple file that records the On or Off status of both wifi and Bluetooth networks.

***Bluetooth Pairing***

| plist file name on iPhone | Backed-up plist file on PC |
|---|---|
| /Library/Preferences/ com.apple.MobileBlueto oth.devices.plist | 93b8fe568d0130901706c942bef2 fc8c274ca275.mddata |
| /Library/Preferences/ .apple.MobileBluetooth.s ervices.plist | ca8bff963333eecdb5063359d068 b94daf67ac88.mddata |

*MobileBluetooth.devices* plist keeps a log of Bluetooth devices paired with the iPhone mobile. The data includes the remote device's name and hardware address. Similarly, the *MobileBluetooth.services* plist file maintains a history of pairings with each Bluetooth device. Paired device history preserves the device's hardware address and the date and time when pairing occurred. Moreover, this plist maintains a record of unauthorized Bluetooth devices that the iPhone declined pairing with.

***Device Information***

| plist file name on iPhone | Backed-up plist file on PC |
|---|---|
| Info.plist | Info.plist |
| /Library/Preference s/ | b80af3611da846772e32024c3abace0de 6072e73.mddata |

com.apple.commce
nter.plist

The *Info* plist is created and saved in the backup directory, and primarily stores the device specific information, such as, the device name, device unique identifier, ICCID, IMEI, phone number, firmware version, serial number, last backup date, installed applications IDs, in addition to path and name of photos folder on the paired PC. The commcenter plist also stores the ICCID as well as the IMSI unique identifiers.

### Photo and Video Data

The iPhone file system stores media files in different formats. However, the Apple iTunes utility backs up only photos, videos and screenshots taken by the built-in camera, as well as photos sent or received by MMS messages. When media files are backed up, they are encoded into mddata binary format. The associated metadata file contains the filename and path where the actual file is stored on the device memory. Media file formats can be distinguished by the first few characters in the file header when it is opened within a text or hex editor utility. Photos taken by the device's camera are stored in JPG format, and are characterized by the text 'ExifMM' at the beginning of the binary file. Screenshot images taken by the camera by holding down both the Home and Power buttons are saved in PNG format and can be identified by the text 'PNG' at the beginning of the binary file. Videos taken by the built-in camera are saved in MOV format, and the first few characters in the binary file read 'ftypqt'. Images received via MMS messages are possibly stored in jpg or gif formats. They are identified by the text 'JFIF' and 'GIF89' respectively at the beginning of the binary files. Finding photos and video files in the backup directory primarily depended on searching for these particular formats.

### Results - Automated Parsing Tools

Few parsing tools such as, 'MobileSyncBrowser' ("MobileSyncBrowser,") or 'iPhone Backup Extractor' ("iPhone / iPod Touch Backup Extractor,") can be used to translate iPhone binary backed-up files into their original readable file formats. In this study, the 'iPhone Backup Extractor' utility for Mac was used to parse the mddata, mdinfo backed-up files and extract them back into a format that can be accessible directly within their associated utility or application. The obtained SQLite database and plist files were created under a directory structure parallel to that on the iPhone file system as shown in Figure 5.

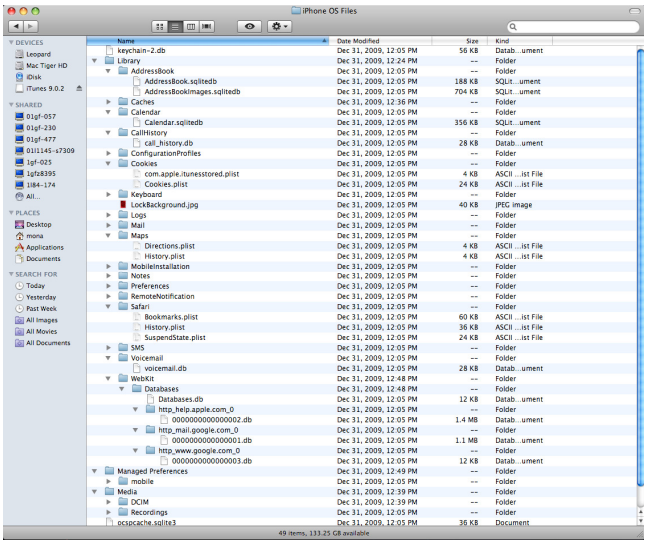


**Figure 5: Extracted iPhone files using 'iPhone Backup Extractor'**

A Windows-based version of this parsing tool was tested and explored as well. It yielded comparable results in terms of retrieved files. Both the windows and Mac tools are capable of converting the binary backup data into their native readable files, however, they don't provide forensic processing and reporting capabilities.

### Results - iPhone Timestamps

The iPhone file system uses a mix of Unix timestamp and Absolute time formats to store date and time values. Unix timestamp format defines date and time stamp as an integer value representing the seconds elapsed since midnight GMT on January 1st , 1970 known as Unix Epoch. For example the timestamp value '1260522215' corresponds to 'December 11, 2009 at 1:03 pm (GMT+4)'. Online Unix timestamp conversion tools are available on the Web. Alternatively, formulas have been created to convert Unix timestamp values to readable date and time using spreadsheet applications such as Excel as shown in Figure 6 ("Converting UNIX Date/Time Stamps," 2009; Converting Unix Timestamp; Sintay, 2009; Unix time," 2009).

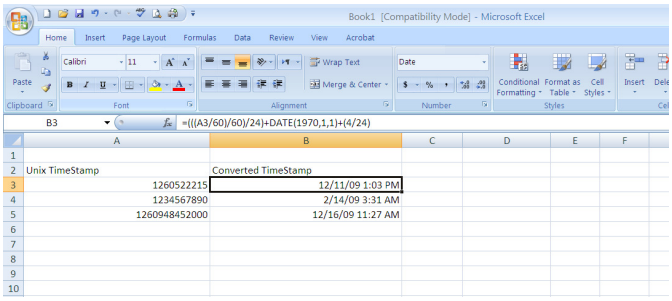


**Figure 6: Using Excel to convert Unix Timestamp to readable format**

Absolute time format, on the other hand, measures time as the number of seconds between a specific date and the

absolute reference date of January 1 2001 00:00:00 GMT ("CFDate Reference," 2005; Time Utilities Reference," 2007). Absolute times can be interpreted into a readable format using special conversion tools such as the open source tool CFAbsoluteTimeConverter ("Hsoi's Shop: Software," 2007) as shown in Figure 7.

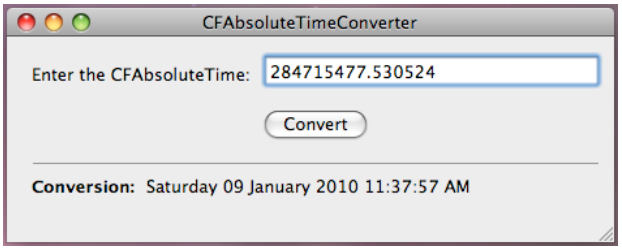


**Figure 7: CFAbsolute Time Converter**

## Conclusion

Few studies have addressed the forensic recovery of data from an iPhone mobile device. This study explored the forensic acquisition, analysis and examination of the logical backup copy of the iPhone 3GS mobile. The examination process attempted to identify what significant data is stored on the device, where it is stored on the memory, as well as where data is located within the acquired backed-up files. The acquisition was conducted in a controlled manner using the iTunes backup utility available freely from Apple to synchronize data between the iPhone and a paired computer. All the tools used to perform the forensic analysis are freely available, and the examination was conducted on both Windows and Mac operating system platforms.

Acquiring a logical copy from the iPhone results in hundreds of backup files containing user data, device settings, application preferences and status, all encoded into XML, ASCII or binary formats. This large quantity of backed-up files are generated from natively installed Apple applications in addition to various applications installed by the user. Each backup file is paired with a metadata file encrypted in a binary format containing the filename of the backed-up data along with the location where it is saved on the iPhone file system. Each backed-up file corresponds to plist, database, photo, video file or other data types. Exploring the logical copy acquired from the iPhone file system affirmed that the backup files contain a wealth of data that can be of a potential evidentiary value, such as text messages, multimedia messages, email messages, call history, contacts, GPS locations, calendar events, images, and device pairing information. When backed-up on the paired computer, data files are assigned unique hashed filenames. The unique filenames were identified and categorized as either plist or SQLite database files. The analysis and examination section documented the filenames for these well-known backed-up data files.

iPhone forensics is an evolving field, and requires further attention and exploration of the recovery of evidentiary data in a forensically accepted manner.

## Future Work

Future work can build on the outcome of this study. An open source iPhone forensics tool can be designed to read and report stored data from the SQLite database files contained in distinct mddata files that have been extracted from the logical backup of the iPhone device. These defined backup files can be fed into the tool which in turn can detail stored data in a structured format. This tool can then be used by law enforcement to generate iPhone evidence reports that can be submitted to court. Moreover, further investigation on acquiring logical backups from iPhone devices with an activated passcode or backup encryption should be pursued. Enabling backup encryption is a new feature available on the new 3GS generation of iPhone mobiles that deserves exploration.

## References

Amorebieta, M. (2007). Cell Phone Forensics. *Inside Dateline* Retrieved October 20, 2009, from http://insidedateline.msnbc.msn.com/archive/2007/01/23/39096.aspx

Apple Announces the New iPhone 3GS. (2009). Retrieved October 26, 2009, from http://www.apple.com/pr/library/2009/06/22iphone.html

Apple Reports Fourth Quarter Results. (2009). Retrieved October 26, 2009, from http://www.apple.com/pr/library/2009/10/19results.html

Ayers, R. (2008). *Mobile Device Forensics - Tool Testing*.

Baggili, I. M., Mislan, R., & Rogers, M. (2007). Mobile Phone Forensics Tool Testing: A Database Driven Approach. *International Journal of Digital Evidence, 6*(2).

Best Practices for Mobile Phone Examination. (2009). *Scientific Working Group on Digital Evidence* Retrieved November 23, 2009, from http://www.swgde.org/documents/swgde2009/Best%20Practices%20for%20Mobile%20Phone%20Examinations%20v1.0.pdf

CFDate Reference. (2005). *Mac OS X Reference Library* Retrieved January 7, 2010, from http://developer.apple.com/iPhone/library/documentation/CoreFoundation/Reference/CFDateRef/Reference/reference.html

Command Line Shell For SQLite. *SQLite* Retrieved December 2, 2009, from http://www.sqlite.org/sqlite.html

Converting UNIX Date/Time Stamps. (2009). *Excel.Tips.Net* Retrieved December 16, 2009, from http://excel.tips.net/Pages/T002051_Converting_UNIX_DateTime_Stamps.html

Converting Unix Timestamp. (2009). *The Spreadsheet Page* Retrieved December 13, 2009, from http://spreadsheetpage.com/index.php/tip/converting_unix_timestamps/

Dredge, S. (2009). iPhone dominating mobile web surfing stats. *PG.BIZ News* Retrieved October 25, 2009, from

http://www.pocketgamer.biz/r/PG.Biz/iPhone+news/news.asp?c=11906

*General Test Methodology for Computer Forensic Tools Version 1.9*. (2001). National Institute of Standards and Technology.

Hoog, A., & Gaffaney, K. (2009). iPhone Forensics. *ViaForensics* Retrieved October 12, 2009, from http://chicago-ediscovery.com/wpinstall/wp-content/uploads/2009/03/iPhone-Forensics-2009.pdf

Hsoi's Shop: Software. (2007). *Hsoi's Shop* Retrieved January 8, 2010, from http://www.hsoi.com/hsoishop/software/

Husain, M. I., & Sridhar, R. (2009). *iForensics: Forensic Analysis of Instant Messaging on Smart Phones.* Paper presented at the International Conference on Digital Forensics and Cyber Crime, Albany, NY.

iPhone 3Gs forensic imaging. (2009). *Mobile Device Forensics* Retrieved December 14, 2009, from http://mobileforensics.wordpress.com/2009/07/25/iphone-3gs-forensic-imaging/

iPhone / iPod Touch Backup Extractor. Retrieved December 2, 2009, from http://supercrazyawesome.com/

iPhone and iPod touch: About backups. (2009). Retrieved December 12, 2009, from http://support.apple.com/kb/HT1766

iPhone OS. (2009). *Wikipedia, the free encyclopedia* Retrieved November, 24, 2009, from http://en.wikipedia.org/wiki/IPhone_OS

Jansen, W., & Ayers, R. (2007). *Guidelines on Cell Phone Forensics*: National Institute of Standards and Technology.

Jansen, W., Delaitre, A., & Moenner, L. (2008). Overcoming Impediments to Cell Phone Forensics.

Jeroen, K., Elke den, O., & Yuan, L. (2008). *Usability benchmark study of commercially available smart phones: cell phone type platform, PDA type platform and PC type platform*. Paper presented at the Proceedings of the 10th international conference on Human computer interaction with mobile devices and services.

Kiley, M., Shinbara, T., & Rogers, M. (2007). iPod Forensics Update. *International Journal of Digital Evidence, 6*(1).

Marsico, C. V., & Rogers, M. K. (2005). iPod Forensics. *International Journal of Digital Evidence, 4*(2).

MobileSyncBrowser. Retrieved November 28, 2009, from http://homepage.mac.com/vaughn/msync/

Nena, L. I. M., & Anne, K. (2009). Forensics of Computers and Handheld Devices Identical or Fraternal Twins? *Communications of the ACM, 52*(6), 132-135.

plist Editor for Windows. *iPodRobot.com* Retrieved December 4, 2009, from http://www.ipodrobot.com/download.htm

PlistEdit Pro 1.5. *Apple* Retrieved December 4, 2009, from http://www.apple.com/downloads/macosx/development_tools/plisteditpro.html

Punja, S. G., & Mislan, R. P. (2008). Mobile Device Analysis. *Small Scale Digital Devices Forensics Journal, 2*(1).

Sintay, B. (2009). Unix Time Stamp . com. Retrieved December 16, 2009, from http://www.unixtimestamp.com/index.php

Smart phones defy slowdown (2009). *Canalys* Retrieved October 19, 2009, from http://www.ifra.net/system/files/Smart%20phones%20defy%20slowdown.pdf

SQLite Database Browser. Retrieved December 2, 2009, from http://sqlitebrowser.sourceforge.net

Summers, C. (2003). Mobile phones - the new fingerprints. *BBC News* Retrieved October 15, 2009, from http://news.bbc.co.uk/2/hi/uk_news/3303637.stm

Time Utilities Reference. (2007). *Mac OS X Reference Library* Retrieved January 7, 2010, from http://developer.apple.com/Mac/library/documentation/CoreFoundation/Reference/CFTimeUtils/Reference/reference.html#//apple_ref/doc/uid/20001452-CH203-DontLinkElementID_3

Unix time. (2009). *Wikipedia, the free encyclopedia* Retrieved December 16, 2009, from http://en.wikipedia.org/wiki/Unix_time

Zdziarski, J. (2008). *iPhone Forensics* (First ed.). Sebastopol: O'Reilly Media, Inc.