May 28th, 11:20 AM

LiFE (Logical iOSForensics Examiner): An Open Source iOSBackup Forensics Examination Tool

Ibrahim Baggili
ECECS Department, UNHcFREG, Tagliatela College of Engineering

Shadi Al Awawdeh
ECECS Department, UNHcFREG, Tagliatela College of Engineering, shadi77@hotmail.com

Jason Moore
ECECS Department, UNHcFREG, Tagliatela College of Engineering, jmoor7@unh.newhaven.edu

Follow this and additional works at: https://commons.erau.edu/adfsl

Part of the Aviation Safety and Security Commons, Computer Law Commons, Defense and Security Studies Commons, Forensic Science and Technology Commons, Information Security Commons, National Security Law Commons, OS and Networks Commons, Other Computer Sciences Commons, and the Social Control, Law, Crime, and Deviance Commons

Scholarly Commons Citation

EMBRY-RIDDLE
Aeronautical University™
SCHOLARLY COMMONS

(c)ADFSL

LiFE (LOGICAL iOS FORENSICS EXAMINER): AN OPEN SOURCE iOS BACKUP FORENSICS EXAMINATION TOOL

Ibrahim Baggili, PhD
ibaggili@newhaven.edu

Shadi Al Awawdeh
shadi77@hotmail.com

Jason Moore
jmoor7@unh.newhaven.edu

ECECS Department, UNHcFREG
Tagliatela College of Engineering
Univesity of New Haven
300 Boston Post Rd,
West Haven, CT 06516

ABSTRACT

In this paper, we present LiFE (Logical iOS Forensics Examiner), an open source iOS backup forensics examination tool. This tool helps both researchers and practitioners alike in both understanding the backup structures of iOS devices and forensically examining iOS backups. The tool is currently capable of parsing device information, call history, voice messages, GPS locations, conversations, notes, images, address books, calendar entries, SMS messages, Aux locations, facebook data and e-mails. The tool consists of both a manual interface (where the user is able to manually examine the backup structures) and an automated examination interface (where the tool pulls out evidence from known files). Additionally, LiFE is designed so that the evidence located in files would retain its integrity. It is important to note that most of the evidence examined by LiFE is parsed from SQLite databases that are backed up by iTunes. LiFE also offers an extensibility option to the user, where an examiner can add new evidence SQLite files to the application that can be automatically parsed, and these known files are then automatically populated in the automated GUI's toolbar with an icon added to the investigator's liking.

Keywords: iOS forensics, Small Scale Digital Devices, iPhone forensics, iPad forensics, SQLite, Open source tools, iTunes backup, Extensible forensics software, File identification, LiFE

1. INTRODUCTION

One cannot ignore the importance of forensically examining Small Scale Digital Devices (SSDDs) in today's world, especially smart phones. Smart phones have become mini-computers allowing people to use them very much like a conventional personal computer. The functionality of smartphones has extended to tablet devices, and tablet-laptop hybrids.

iOS devices have played a major role in shaping our understanding of smart phones (iPhones) and tablets (iPads) due to their ubiquity amongst users. As iOS usage continues to grow, the probability of finding critical and relevant digital evidence to a case on iDevices increases as well. Therefore, there is a strong need to build tools that support both practitioners and researchers alike to examine and reconstruct digital evidence on iOS devices.

2. RELATED WORK

2.1 High Level Literature Review

Although there are a number of research projects on iPhone and iPad forensics, the first documented research was on Apple iPods (Kiley, Shinbara, & Rogers, 2007; Marsico & Rogers, 2005). Since these initial research projects, the operating systems on iOS devices have significantly changed.

One of the most seminal works on iOS forensics is by Zdziarski (2008). Zdziarski was unique in that he proposed a method in which the investigators are able to recover deleted data by physically imaging the iOS device. However, jailbraiking the device was necessary to utilize his method, thereby making it more difficult to use in investigations as this can cause alterations on the original device (Barmpatsalou, Damopoulos, Kambourakis, & Katos, 2013).

In 2010, the logical backup copy from iTunes was used to investigate instant messaging on an iPhone without jailbreaking the device (Husain & Sridhar, 2010; Morrissey, 2010). The manual methodology used in examining instant messaging artifacts was then explored and expanded in further research on logical iPhone forensics (Bader & Baggili, 2010; Husain, Baggili, & Sridhar, 2011). At a high level, these methodologies focused on parsing iTunes backup files, SQLite databases as well as Plist files. Using the iTunes backup utility was later determined to be the prevailing method for logical acquisition from an iOS device (Tso, Wang, Huang, & Wang, 2012).

The proposed iOS forensic methodologies have since then been used by researchers to examine social networking evidence on mobile devices (Jung, Jeong, Byun, & Lee, 2011; Mutawa, Baggili, & Marrington, 2012) and have been tested on iPads and compared to logical examinations using automated tools (Ali, Alzarooni, & Baggili, 2012).

Since then, a number of forensic vendors have integrated these methodologies into their products, and these various methods have been compared and discussed in the literature (Hoog & Strzempka, 2011; Höne & Creutzburg, 2011).

2.2 iOS Backup Folder Analysis

iPhone backups are stored in different locations based on the computer operating system as shown in Table 1 (Bader & Baggili, 2010; Carpene, 2011).

Table 1 iPhone Backup Folder Location

OS	Backup path
Win XP	C:\Users\(username)\AppData\Roaming\AppleComputer\ MobileSync\Backup
Win 7	\Documents and Settings\(username)\ ApplicationData\ Apple Computer\MobileSync\ Backup
Mac OSX	~/Library/Application Support/MobileSync/Backup.

By using iTunes version 10 or above, files created in the backup folder are of two types: Plist files, and files without extensions. The main Plist files are (a) Info.plist: This file contains the main information about the iDevice, such as name, model, firmware version, and identifiers (b) Manifest.plist: This contains a list on applications from the iDevice and (c) Status.plist: This contains information related to the device's backup history (Bader & Baggili, 2010; Carpene, 2011).

The other kinds of files are files without extensions. The name of each file consists of 40 characters. It is difficult to decide the contents of these files without investigation, but based on the literature and our testing, the backup predominantly includes images, videos, voice recordings and SQLite database files.

Bader and Baggili (2010) explained that when the user creates a backup for the iPhone using iTunes it will create a folder with a name of 40 hexadecimal characters long that represents the unique device ID (UDID), and will copy the device contents to the newly created folder. It was also observed that iTunes can also create a differential backup with a folder name [UDID] + '-' + [Time stamp] in the same backup location.

iTunes makes a copy of almost all the data stored on the device such as contacts, SMS and MMS messages, configuration files, database files, keychain, photos, calendar, music, call logs, network settings, offline web application cache, safari bookmarks, cookies and application data (Bader & Baggili, 2010). Each file has a unique hash value. Some of the most evidence rich files are shown in Table 2. It is important to note that these are SHA-1 hashes of the domain and full path on the iDevice.

3. RESEARCH PROBLEM

The methodologies proposed in the literature for the manual forensics logical examination of the iTunes backup has proven to be useful for both researchers and practitioners alike. However, literature has shown that more evidence can be extracted using the manual examination approach when compared to the commercial tools (Ali et al., 2012). This is due to how the backup structures keep changing from one version of the Apple iOS to the next. Also, most commercial tools only identify digital evidence such as photos, videos, SMS messages and contacts–even though a lot more evidence could be present in the backup files.

A novel tool is needed that allows investigators and researchers to manually examine and identify relevant backup files with potential digital evidence, as well as easily integrate these files into an automated forensic tool. Extensibility is critical to ensure that newly discovered files are easily integrated into the tool's automated Graphical User Interface (GUI).

4. CONTRIBUTION

This research has three major contributions to both the scientific community and practitioners. Primarily, we provide a customizable open source forensics tool (LiFE) that parses the iOS back up files. Second, the tool allows for the quick analysis of the backups using two major interfaces: (1) An interface that helps in the manual examination and discovery of files that could hold potential evidence; and (2) An interface that allows for the automated parsing of the backup structure. Third, our approach allows the user to easily update the automated interface–allowing the tool to be extensible.

5. METHODOLOGY

This research uses a constructive research methodology. A constructive methodology aims at producing novel solutions to practical and theoretical problems and is widely used by software engineers (Casper, Soininen, & Vanhanen, 2001). In this research, the following phases were used as discussed by Casper et al. (2001):

1. Find a practically relevant problem
2. Obtain understanding of the topic and problem
3. Innovate: construct a solution or idea
4. Demonstrate that the solution works
5. Examine the scope of applicability

Table 2 Popular Backup Files and Their Hash Names

SHA1 value of the DomainName-FilePath
SMS database
3d0d7e5fb2ce288813306e4d4636395e047a3d28
HomeDomain-Library/SMS/sms.db
/private/var/mobile/Library/SMS/sms.db
Call History database
ff1324e6b949111b2fb449ecddb50c89c3699a78
HomeDomain-Library/CallHistory/call_history.db
/private/var/wireless/Library/CallHistory/call_history.db
Safari Bookmarks
d1f062e2da26192a6625d968274bfda8d07821e4
HomeDomain-Library/Safari/Bookmarks.db
/private/var/mobile/Library/Safari/Bookmarks.db
Address Book
31bb7ba8914766d4ba40d6dfb6113c8b614be442
HomeDomain-Library/AddressBook/AddressBook.sqlitedb
/private/var/root/Library/AddressBook/AddressBook.sqlitedb
Notes
ca3bc056d4da0bbf88b5fb3be254f3b7147e639c
HomeDomain-Library/Notes/notes.sqlite
/private/var/mobile/Library/Notes/notes.sqlite
Photos
12b144c0bd44f2b3dffd9186d3f9c05b917cee25
/private/var/mobile/Media/PhotoData/Photos.sqlite
iTunes Store
9143d986a77ab8cf5878e4e9ac80627477eb6674
HomeDomain-Library/com.Apple.itunesstored/itunesstored2.sqlitedb
/private/var/mobile/Library/com.Apple.itunesstored/itunesstored2.sqlitedb
Voice Mail
992df473bbb9e132f4b3b6e4d33f72171e97bc7a
HomeDomain-Library/Voicemail/voicemail.db
/private/var/mobile/Library/Voicemail/voicemail.db
Calendar
2041457d5fe04d39d0ab481178355df6781e6858
HomeDomain-Library/Calendar/Calendar.sqlitedb
/private/var/mobile/Library/Calendar/Calendar.sqlitedb
Email Account (Plist)
5fd03a33c2a31106503589573045150c740721dd
HomeDomain-Library/Preferences/com.Apple.accountsettings.plist
/private/var/mobile/Library/Preferences/com.Apple.accountsettings.plist
Safari History (Plist)
1d6740792a2b845f4c1e6220c43906d7f0afe8ab
HomeDomain-Library/Safari/History.plist
/private/var/mobile/Library/Safari/History.plist

6. LiFE

The backup structures were documented by the researchers after a comprehensive literature examination was conducted on the iOS backup structure from iTunes. The literature pointed out that some of the files residing in the backup folder are SQLite databases, while others are .Plist files and images. Once the file structures were understood, the tool (LiFE) was built.

LiFE was designed to extract evidence from iOS devices with iOS 6.0.1 backups, because the earlier iOS versions output different backup structures. To build the tool, we used C# .NET 2010 as a development platform.

The tool itself is available for use and can be found at http://www.unhcfreg.com <Datasets & Tools>.

6.1. Manual User Interface

To allow the researchers to better understand the backup structures, as well as to improve the efficiency of the manual examination process described in the literature, a manual user interface was created as shown in Figure 1.
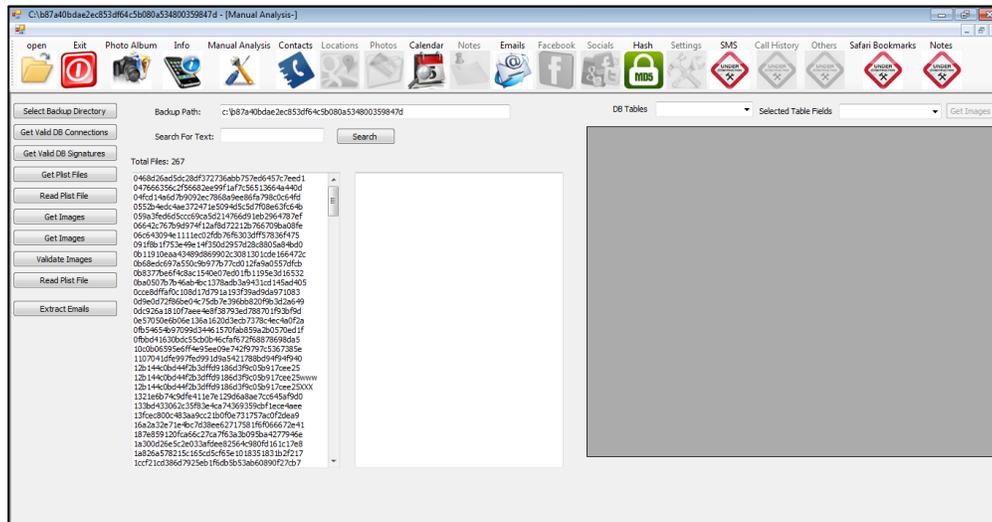

Figure 1 LiFE Manual Examination GUI

In the manual examination interface, the user can search for string data throughout all of the files. Additionally, files are identified based on their header information and connectivity (if the file is a SQLite database). If a file is a database, its contents are rendered in a grid view displayed on the right hand of Figure 1, and all its available tables are populated in the Dropdown Box above it.

6.2 Automated user interface

This part of the GUI consists of multiple Document Interface forms (MDI) as well as child forms for the different modules. Figure 2 shows the interface that automatically extracts evidence from the backup folder. One can easily examine the evidence of interest by clicking on the respective icon on the menu bar.

LiFE was divided into modules. Currently, the prototype GUI consists of the following menu items:

1. Device Information
2. Call History
3. Voice Messages
4. GPS Locations
5. Conversations

6. Notes
7. Images
8. Address Book
9. Calendar
10.SMS
11.Locations
12.Aux Locations
13.Facebook
14.Emails

It is critical to note that these menu items are optional. The user has the full authority to change the menu names, icons, and orders as preferred. In the prototype we created some fixed menu items that are necessary for the function of the tool.
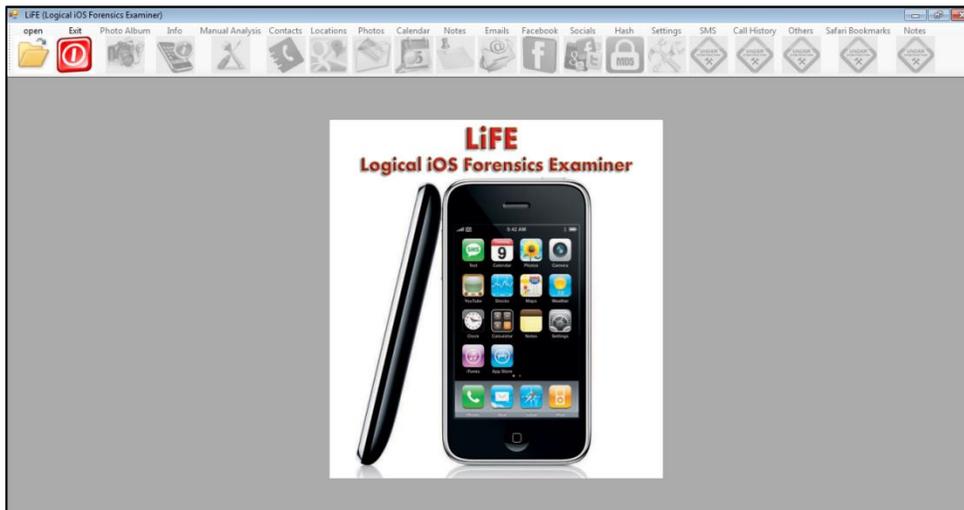

Figure 2 Automated LiFE MDI Window

6.3 Ensuring the Integrity of the Backup Files

LiFE was designed in a manner that protects the integrity of the digital evidence. Protecting the integrity of the evidence is critical to ensure its admissibility (Wayne & Ayers, 2007). When the code was written, no methods that wrote to the backup folder or modified the files were utilized, only methods and functions that read the backup files and searched within the files were used.

Once the backup folder is selected for analysis, all the files are hashed and the pre-analysis values are stored in a SQLite database file. Once the user closes the application, the application will compute the hash values of all the files post-analysis, and compare them to the pre-analysis hash values. An integrity report is then presented to the user.

To ensure the accuracy of our implemented hashing function, we purposefully modified some files after the pre-analysis hashing and observed that our hashing function detected the changes in the modified files.

6.4 File Identification

We used different methods to identify files in the manual and automated GUIs to identify both SQLite and Plist files.

To identify the SQLite database files we used two techniques. The first was to search the files for the SQLite file signature. In this method we used the file stream command File.ReadAllText() to read the

contents of every file as a text file, we then searched the returned text file for the signature "SQLite format 3" by simply searching the first two lines in each file in the backup folder.

The second method used was to loop through all of the files while trying to perform a database connection to each file. By default, SQLite database files will succeed in creating the connection while other non-SQLite files will return error messages. We observed that this method is slower than the signature based searching method, however, it can be used to validate the files after the signature is found.

In order to connect to SQLite database file we needed a database engine. We first used a DLL library called SQLite.NET and referenced it by typing "using Finisar.SQLite." This DLL was not able to establish connectivity to all the database files. The solution was to upgrade to the latest version of SQLite which supported WAL (Write Ahead Logging). The latest version of the SQLite DLL was downloaded from http://system.data.sqlite.org and referenced by typing "using System.Data.SQLite." This helped ensure the tools compatibility in connecting to various types and versions of SQLite databases.

We also used the keyword or signature searching method to identify the Plist files in the iOS backup folder. We used the Plist file-signature matching technique and searched all of the backup file headers for the Plist file signature "bplist00Ø."

Some of the visible Plist files in the backup folder are Status.plist, Manifest.plist and Info.plist. These files were clearly identified because they have the file extension .Plist, but only the file signature could identify the other Plist files as they did not have an extension. We were able to read the first level of tags in Info.plist, but we could not properly read the other Plist files. The Plist parser in our tool requires more coding and enhancement to translate the Plist files to a readable format.

An important aspect of the iOS backup are the images. Two techniques were used to identify image files. In the first technique we applied a file signature searching technique searching for the existence of the "Exif" file signature. We also searched the files for known image file formats: gif, png, jpg, jpeg, bmp and ico.

In the other technique, we attempted to validate the image files by looping through all of the files in the backup folder, reading them in as binary files, then loading them one by one into a PictureBox control. The valid images were loaded successfully into the PictureBox, while the non-valid image files generated error messages. We then displayed all of the valid images in a photo album as shown in Figure 3.
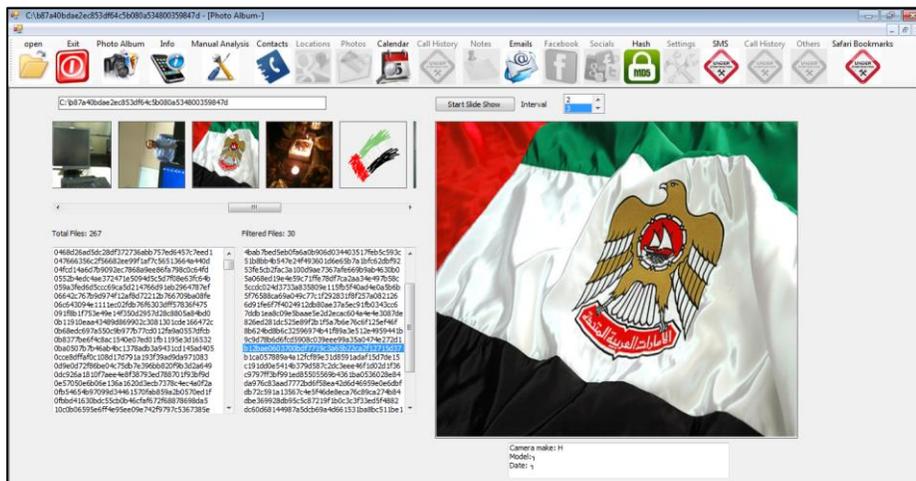

Figure 3 Images Identified in the PhotoAlbum

At this point, we were able to identify and display the SQLite database files, image files and partially view some of the Plist file contents.

We successfully connected to the following databases: Call History, Voice Messages, GPS Locations, Conversations, Notes, Images, Address Book, Calendar, SMS, Locations, Aux Locations, Facebook friends, and we successfully retrieved all the valid email combinations in the backup files using regular expressions.

6.5 Retrieving Data from the Databases

For each database, we select one table to be the initially selected table for binding to the DataGridView control. For example, once the call history tab is clicked the grid is populated with the data from the table named "call" using this SQL statement "Select rowid, address, datetime(date ,'unixepoch') as MyDate, duration from call". The other tables in the selected database are populated in the DropDown list where the user can select any table to view its contents. Figure 4 shows the call history list populated from the call table.

ROWID	address	MyDate	duration
13	0552123259	2010-05-28 16:0...	0
14	+971506980885	2010-12-29 12:0...	255
15	+971502022300	2010-12-29 12:5...	0
16	+971502402251	2010-12-30 05:4...	56
17	026674782	2010-12-30 07:0...	96
18	+971502022300	2010-12-30 08:0...	0
19	+971502022300	2010-12-30 08:0...	0
20	+971502022300	2010-12-30 08:0...	0
21	+971502022300	2010-12-30 08:0...	0
22	+971502022300	2010-12-30 08:0...	9
23	0506980885	2010-12-30 08:0...	22

Figure 4 Call History List, From the Call Table

The Unix-epoch modifier in the previously mentioned SQL select statement expects a value in seconds and it is used to convert the Unix time format into a readable human format. For example, select datetime ('1289325613', 'unixepoch') returns the value 2010-11-09 18:00:13 if executed in SQLite Maestro.

Maps and location tracking is one of the new features in smart phones allowing users to view maps, select locations, etc. One of the tables called AuxPhoto, exists in the database with the hashed name "1CCF21CD386D7925EB1F6DB5B53AB60890F27CB7," and contains longitude and latitude columns. We allow the user to navigate to the real location of that longitude and latitude and preview a map using the Google maps website by embedding it into a webBrowser control in our tool. Figure 5 shows the values of the AuxPhoto table.

primaryKey	latitude	longitude	ShowMap
18	24.466	54.37966666666...	ShowMap
19	24.466	54.37966666666...	ShowMap
20	24.466	54.37966666666...	ShowMap
21	24.466	54.37966666666...	ShowMap
22	24.466	54.37966666666...	ShowMap
23	24.466	54.37966666666...	ShowMap
24	24.466	54.37966666666...	ShowMap
25	24.466	54.37966666666...	ShowMap
26	24.466	54.37966666666...	ShowMap

Figure 5 AuxPhoto Table Contents

For the longitudes and latitudes shown in Figure 5 we included a ShowMap column in the results of the AuxPhoto table to allow the user to view the map for a certain latitude and longitude. When clicking on the ShowMap corresponding to a specific longitude and latitude, we pass the values to google maps and display the map result in our tool as shown in Figure 6.
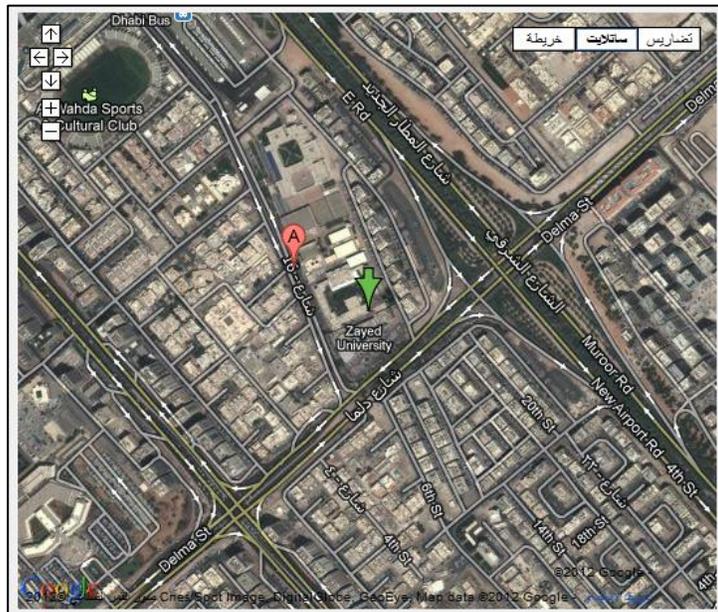

Figure 6 A Map Retrieved by LiFE

6.6 LiFE Extensibility

To make LiFE more extensible and dynamic, and to minimize the amount of required coding for the Automated GUI, we created a database table for the known SQLite database files. In that table we added a readable name for the database, as well as its hashed name and the name of an icon file that reflects the database contents. If these items are properly added to this table, these new items are dynamically populated in the toolbar and the required evidence is populated from the configured database once the program is started. This extensible database table is shown in Figure 7, and its respective data entry form is shown in Figure 8.

1	1	open	img_open.jpg	NULL	1	NULL
2	2	Info	img_info.jpg	Info.plist	4	NULL
3	3	Contacts	img_contacts.jpg	31bb7ba8914766d4ba40d6dfb6113c8b614be442	6	NULL
4	4	Locations	img_locations.jpg	b88b75bddaa69139b66d948b7cbd4f41d9dd416d	7	NULL
5	5	Photos	img_photos.jpg	12b144c0bd44f2b3dffd9186d3f9c05b917cee25w	8	NULL
6	6	Calendar	img_calendar.jpg	2041457d5fe04d39d0ab481178355df6781e6858	9	NULL
7	7	Notes	img_notes.jpg	740b7eaf93d6ea5d305e88bb349c8e9643f48c3b	10	NULL
8	8	Emails	img_emails.jpg	970922f2258c5a5a6d449f85b186315a1b9614e9	11	NULL
9	9	Facebook	img_facebook.jpg	6639cb6a02f32e0203851f25465ffb89ca8ae3fa	12	NULL
10	55	Call History	CallHistory.jpg	ff1324e6b949111b2fb449ecddb50c89c3699a78	9	NULL
11 ▶	10	Socials	img_socials.jpg	NULL	13	NULL
12	11	Hash	img_hashing.jpg	NULL	14	NULL
13	12	Settings	img_Settings.jpg	NULL	15	NULL
14	13	SMS	img_sms.jpg	3d0d7e5fb2ce288813306e4d4636395e047a3d28	16	NULL
15	14	Call History	img_calls.jpg	ff1324e6b949111b2fb449ecddb50c89c3699a78	17	NULL
16	15	Others	Others.jpg	ff1324e6b949111b2fb449ecddb50c89c3699a78	18	NULL
17	16	Safari Bookmarks	NULL	d1f062e2da26192a6625d968274bfda8d07821e4	19	NULL
18	17	Notes	NULL	ca3bc056d4da0bbf88b5fb3be254f3b7147e639c	20	NULL

Figure 7 Toolbar Icons and Related File Names

Figure 8 Toolbar Icon Details Entry Form

7. FUTURE WORK

In the future the authors hope to find better mechanisms for parsing Plist files located in the iTunes backup. Additionally, the authors are currently working on developing a triage screen that could be integrated into LiFE so that investigators can quickly and reliably understand the contents of the iOS device, aiding in profiling the device and its user.

8. CONCLUSION

In this paper we have taken a manual iOS backup forensics methodology and have integrated that methodology into a tool called LiFE. We illustrated the vast amount of evidence that could be retrieved from an iOS logical backup that could potentially be relevant to an investigation. LiFE can potentially speed up research and investigations related to iOS backup forensics.

REFERENCES

Ali, S., Alzarooni, F., & Baggili, I. (2012). iPad2 logical acquisition: Automated or manual examination? ADFSL Conference on Digital Forensics Security and Law, Richmond, VA, May 30-31, 113-128.

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.

Barmpatsalou, K., Damopoulos, D., Kambourakis, G., & Katos, V. (2013). A critical review of 7 years of Mobile Device Forensics. *Digital Investigation*, *10*(4), 323-349.

Carpene, C. (2011). Looking to iPhone backup files for evidence extraction, Australian Digital Forensics Conference. Retrieved from http://ro.ecu.edu.au/adf/92

Casper, L., Soininen, T., & Vanhanen, J. (2001). Constructive research. SoberIT. Retrieved from http://www.soberit.hut.fi/~mmantyla/work/Research_Methods/Constructive_Research/constructive_research.ppt

Hoog, A., & Strzempka, K. (2011). *iPhone and iOS forensics: Investigation, analysis and mobile security for Apple iPhone, iPad and iOS devices.* Elsevier.

Husain, M. I., Baggili, I., & Sridhar, R. (2011). A simple cost-effective framework for iPhone forensic analysis. In *Digital Forensics and Cyber Crime*, 27-37. Springer Berlin Heidelberg.

Husain, M. I., & Sridhar, R. (2010). iForensics: Forensic analysis of instant messaging on smart phones. In S. Goel, O. Akan, P., Bellavista, J., Cao, F., Dressler, D., Ferrari, M., Gerla, et al. (Eds.), *31*, 9–18. Springer Berlin Heidelberg. doi:10.1007/978-3-642-11534-9_2

Höne, T., & Creutzburg, R. (2011). iPhone forensics: A practical overview with certain commercial software. In S. S. Agaian, S. A. Jassim, & Y. Du (Eds.), *SPIE Defense, Security, and Sensing* (p. 80630M–80630M–12). doi:10.1117/12.884589

Jung, J., Jeong, C., Byun, K., & Lee, S. (2011). Sensitive privacy data acquisition in the iPhone for digital forensic analysis. *Communications in Computer and Information Science*, *186*, 172-186.

Kiley, M., Shinbara, T., & Rogers, M. (2007). iPod forensics update. *International Journal of Digital Evidence*, *6*(1), 1–9. Retrieved from http://cryptome.org/isp-spy/ipod-spy.pdf

Marsico, C., & Rogers, M. (2005). iPod forensics. *International Journal of Digital Evidence*, *4*(2). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.160.1925&rep=rep1&type=pdf

Morrissey, S. (2010). OS forensic analysis: For iPhone, iPad, and iPod touch. Berkely, CA: Apress.

Mutawa, N. Al, Baggili, I., & Marrington, A. (2012). Forensic analysis of social networking applications on mobile devices, *9*, 24–33. doi:10.1016/j.diin.2012.05.007

Tso, Y.C., Wang, S.J., Huang, C.T., & Wang, W.J. (2012). iPhone social networking for evidence investigations using iTunes forensics. 6th International Conference on Ubiquitous Information Management and Communication, 1-7. New York, NY: ACM.

Wayne, J., & Ayers, R. (2007). Guidelines on cell phone forensics. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-101/SP800-101.pdf

Zdziarski, J. (2008). *Iphone Forensics: Recovering Evidence, Personal Data, and Corporate Assets*. O'Reilly Media.